Quantitative Finance

Lab - Bond Management"

Fiona Martin, Romain Pepin, Hedi Sagar, Nathan Sanglier

Version: 03 Apr 2024

I/ Bond Liability Hedging

The aim of this part is to build a bond management model which combines the Cash Flow Matching and Immunization methods.

The Cash Flow Matching method is suitable for short-term maturities, as it avoids doing too many transactions. However, it lacks from flexibility and is an expensive solution. For longer maturities, we thus prefer Immunization method.

Data

Data is simulated for the needs of the lab.

Bonds

```
df.o <- read.csv("../GP/data/obligations.csv",
                           colClasses=c("character", "character", NA))
df.o$dtE <- as.Date(df.o$dtE, format("%m/%d/%Y"))
df.o$dtM <- as.Date(df.o$dtM, format("%m/%d/%Y"))
df.o$Nom <- sapply(seq_along(df.o$dtE), function(x) paste('Bond-',x, sep=''))
nb.bonds <- nrow(df.o)
```

Actuarial Yield Curve

```
dt.mat <- seq(ymd('2021-01-01'), ymd('2040-01-01'), by="year")
tx <- 1 + log(seq_along(dt.mat))
df.cdt <- data.frame(mat=dt.mat, tx=tx)
plot(dt.mat, tx, main='Actuarial Yield Curve', xlab="Maturity", ylab="Rate (in %)")
lines(dt.mat, tx, type='l', lwd=2, col='red')
```

Table 1: Bonds Data

Issue Date	Maturity Date	Coupon (%)	Name
2018-06-01	2021-06-01	1.7	Bond-1
2016-12-01	2021-12-01	3.5	Bond-2
2019-04-01	2022-04-01	1.7	Bond-3
2017-10-01	2022-10-01	2.6	Bond-4
2020-02-01	2023-02-01	2.7	Bond-5
2016-04-01	2023-04-01	1.2	Bond-6
2018-08-01	2023-08-01	5.0	Bond-7
2020-12-01	2023-12-01	2.2	Bond-8
2017-02-01	2024-02-01	3.5	Bond-9
2019-06-01	2024-06-01	2.7	Bond-10

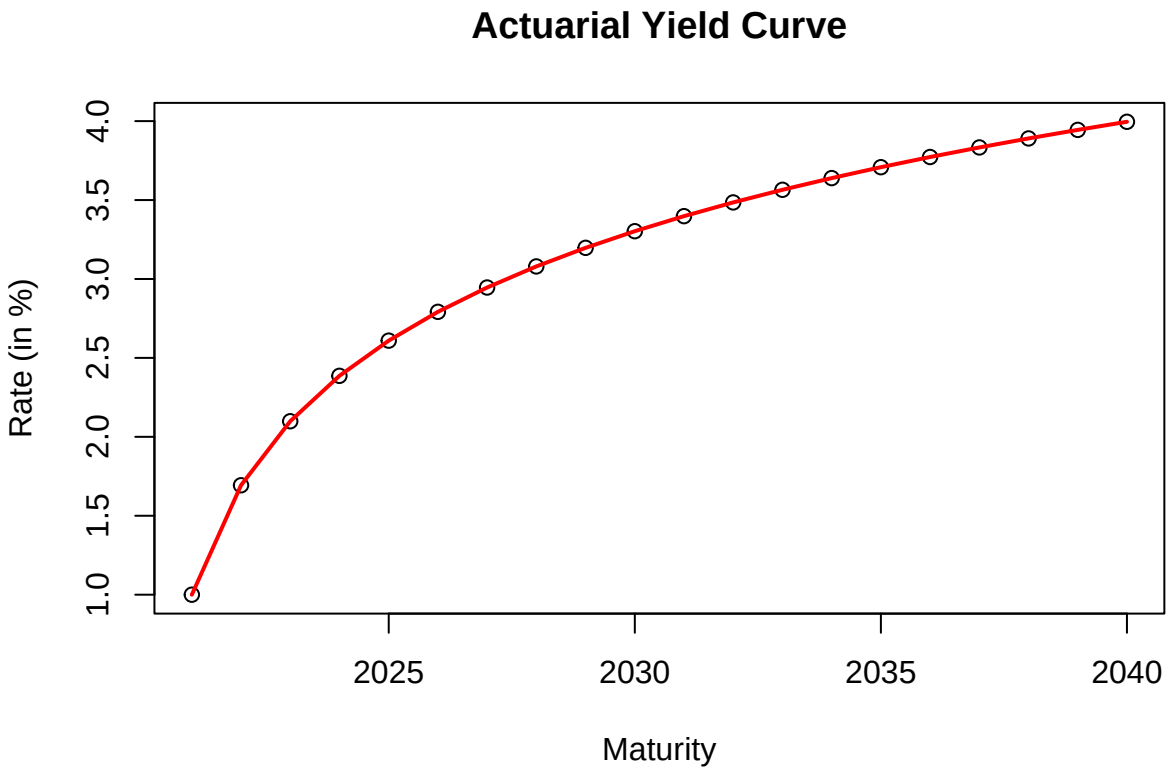

Figure 1: Actuarial Yield Curve, at any date

Preliminary calculations

1. Write a function that interpolates the yield curve for a given maturity date.
2. Choose a bond from the list, interpolate the actuarial yield and calculate the clean price, accrued interest, dirty price, and risk indicators. Use the "BondValuation" package and use the AFB ACT/ACT convention for day counts.

Question 1. As the yield curve is relatively simple (observation points sampled from logarithmic function), there is no need for spline interpolation, and a linear interpolation between two points is sufficient here.

```
yield_func <- function(maturityDate) {

  approx(df.cdt$mat, df.cdt$tx, maturityDate, method="linear")$y
}
```

Question 2. Let us recall the expressions mentioned in the question. The clean price of a bond refers to the price of a bond that does not include any accrued interest. Indeed, when a bond is traded between interest payment dates, the buyer typically pays the seller the clean price of the bond plus any accrued interest (interest accumulated since the last interest payment date). This total price is called the dirty price. Notice that as we do not want the bond quoted price to be influenced by the amount of accrued interest (i.e. we do not want mechanical rise or fall in bond quoted price), then the clean price is also the quoted price.

Risk indicators (w.r.t. interest rates) for a bond typically refer to durations (or PV01) and convexity. To get an understanding of these notions, let us write the Taylor expansion to order 2 of the percentage change in bond's dirty price (DP) due to a change of its yield to maturity (ytm) by Δy :

$$\begin{aligned}\frac{\Delta DP}{DP} &\approx \frac{\Delta y}{DP}\frac{\partial DP}{\partial y} + \frac{1}{2}\frac{(\Delta y)^2}{DP}\frac{\partial^2 DP}{\partial y^2} \\ &= -\Delta y \cdot D_{mod} + (\Delta y)^2 \cdot C\end{aligned}$$

D_{mod} is the modified duration and C is the convexity. Notice we put a minus in the modified duration formula, as we want for simplicity a positive quantity when interest rates increase (i.e. bond price will diminish). Basically, D_{mod} and convexity represent risk w.r.t. respectively small and large parallel changes in interest rates.

PV01 is also related to parallel changes in yield, since it is defined as the bond's clean price (CP) change due to a 1 bp (basis point, 0.01%) negative change in yield. Notice we can alternatively compute it with dirty price as it will give the same result.

$$\begin{aligned}PV01 &= DP\left(y - 10^{-4}\right) - DP\left(y\right) \\ &\approx -\frac{\partial DP}{\partial y} \cdot 10^{-4}\end{aligned}$$

We won't calculate PV01 here (as we already have duration and convexity), but it will be calculated later using the calculation by difference.

Here, we'll also retrieve Macaulay duration, which is the weighted average maturity of the cash flows (CFs):

$$D_{mac} = \frac{\sum_t^T t\frac{CF_t}{(1+y)^t}}{DP}$$

One can easily check with the formulas given that the relation between D_{mod} and D_{mac} is :

$$D_{mac} = (1+y)\, D_{mod}$$

Finally, the AFB ACT/ACT convention for day counts divides the actual number of days by 366 if a leap day is contained, or by 365 if not, with additional rules for periods over one year.

Let us select the first bond to verify our calculations. We assume yield curve given above is the one on 2025/03/17, current date.

Table 2: Clean Price, Accrued Interest, Dirty Price, Risk Indicators of Bond n°1

Clean Price	Accr. Int.	Dirty Price	Mod. Duration	Mac. Duration	Convexity
100.0822	1.346027	101.4282	0.2076628	0.2082192	0.0431238

```
# Current date
date_set = ymd('2021-03-17')

bond = df.o[1,]
yield = yield_func(bond$dtM)

res = BondVal.Price(
  YtM = yield, # yield to maturity of the bond on settlement date in %
  SETT = date_set, # settlement date
  Em = bond$dtE, # emission date
  Mat = bond$dtM, # maturity date
  CpY = 1, # nb of coupons per year
  DCC = 3, # to select AFB ACT/ACT convention
  Coup = bond$Coupon # coupon value in %
  )

df_res = data.frame(t(c(res$CP, res$AccrInt, res$DP, res$ModDUR.inYears,
                        res$MacDUR.inYears, res$Conv.inYears)))
```

I.a) Immunization

Assume we have a €10,000,000 liability due for 2025/01/02. Define a portfolio of two bonds with the same value and PV01 as the liability on 2021/03/17. Optimize the average return on this portfolio.

First, let us calculate the present values (i.e. dirty prices) and PV01 of all bonds. Moreover, we notice there is no bond with a maturity date sooner than current date (2021/03/17), so there is no need to remove some bonds from the list.

```
nb_bonds_before = sum(df.o$dtM < date_set)
print(paste("There are ", nb_bonds_before, " bonds with maturity before current date."))
```

```
## [1] "There are  0  bonds with maturity before current date."
```

```
dp_bonds = rep(0, times = nb.bonds) # Bonds dirty prices
pv01_bonds = rep(0, times = nb.bonds) # Bonds PV01
yield_bonds = rep(0, times = nb.bonds) # Bonds yields

for (i in 1:nb.bonds) {

  bond = df.o[i, ]

  yield = yield_func(bond$dtM)
  res = BondVal.Price(YtM = yield, SETT = date_set, Em = bond$dtE, Mat = bond$dtM,
                      CpY = 1, DCC = 3, Coup = bond$Coupon)
```

```
  # recall yield is already in % and 1bp = 0.01%
  res_below = BondVal.Price(YtM = yield - 0.01, SETT = date_set, Em = bond$dtE,
                            Mat = bond$dtM, CpY = 1, DCC = 3, Coup = bond$Coupon)

  dp_bonds[i] = res$DP
  pv01_bonds[i] = res_below$DP - res$DP
  yield_bonds[i] = yield
}
```

Then, let us do the same for the liability, noticing the liability is a zero-coupon bond (thus, we do not care about emission date, we can set it to 2021/03/17).

```
FV = 10^7 # Liability face value
liab_mat_date = ymd('2025-01-02') # Liability due date

yield_liability = yield_func(liab_mat_date)
res = BondVal.Price(YtM = yield_liability, SETT = date_set, Em = date_set,
                    Mat = liab_mat_date, CpY = 0, DCC = 3, Coup = 0)
res_below = BondVal.Price(YtM = yield_liability - 0.01, SETT = date_set, Em = date_set,
                          Mat = liab_mat_date, CpY = 0, DCC = 3, Coup = 0)

# We need to divide by 100 as in "BondValuation", default face value is 100
dp_liability = FV * res$DP / 100 # Liability dirty price
pv01_liability = FV * (res_below$DP - res$DP) / 100 # Liability PV01
```

Notice we did use the same yield curve for pricing bonds and liability. We thus assumed each financial instrument presented here share same levels on risks other than interest rate risks (credit risk, liquidity risk, etc), which may not be true in reality.

Now, recall we want to build a portfolio such that we have immunization at 2021/03/17, while maximizing the average return on this portfolio. This (actuarial) average return r_p is such that the present value of the portfolio considering yield for each bond as this average return should be equal to the present value of the portfolio considering original yield for each bond :

$$\begin{aligned}
&\sum_{i=1}^{N} q_i \cdot DP_i(y_i) = \sum_{i=1}^{N} q_i \cdot DP_i(r_p) \\
\Leftrightarrow &\sum_{i=1}^{N} q_i \cdot (DP_i(y_i) - DP_i(r_p)) = 0 \\
\Leftrightarrow &\sum_{i=1}^{N} q_i \cdot \frac{\partial DP_i}{\partial y}(y_i) \cdot (y_i - r_p) = 0 \\
\Leftrightarrow &\, r_p = \frac{\sum_{i=1}^{N} q_i \cdot \frac{\partial DP_i}{\partial y}(y_i) \cdot y_i}{\sum_{i=1}^{N} q_i \cdot \frac{\partial DP_i}{\partial y}(y_i)}
\end{aligned}$$

where q_i, DP_i and y_i are respectively quantity (to be determined), dirty price and yield to maturity of bond i. Notice that to get this expression of r_p, we have used an order 1 Taylor expansion on each bond's dirty price :

$$DP_i(r_p) = DP_i(y_i) + \frac{\partial DP_i}{\partial y}(y_i) \cdot (r_p - y_i)$$

Now, as the denominator of r_p can be seen as a constant, then maximizing r_p is equivalent to maximizing $\sum_{i=1}^{N} q_i \cdot \frac{\partial DP_i}{\partial y}(y_i) \cdot y_i$. Moreover, notice that taking the PV01 of each bond instead of $\frac{\partial DP_i}{\partial y}(y_i)$ leads to the same result.

Finally, adding equality constraints between assets and liabilities on present value and PV01, and quantities greater than 0 leads to :

$$\begin{aligned} &\max_q \sum_{i=1}^{N} q_i \cdot PV01_i(y_i) \cdot y_i \\ s.t. \ &\sum_{i=1}^{N} q_i \cdot PV01_i(y_i) = PV01_L(y_L) \\ &\sum_{i=1}^{N} q_i \cdot DP_i(y_i) = DP_L(y_L) \\ &\forall i \in 1:N, \, q_i \geq 0 \end{aligned}$$

where $PV01_L$, DP_L and y_L are respectively PV01, dirty price and yield to maturity of liability.

Indeed, having present value and PV01 of assets and liabilities equal enable to hedge against small parallel shifts in the yield curve. However, in reality, the quantities of assets should be rebalanced frequently with the yield curve evolution.

To solve this linear optimization problem, we'll use the function "lp" of the linear programming library "lpSolve". In matrix formulation, our problem is :

$$\begin{aligned} &\max_q \left[PV01_1(y_1) \cdot y_1, \ldots, PV01_N(y_N) \cdot y_N\right] \cdot q \\ s.t. \ &\begin{pmatrix} PV01_1(y_1) & \ldots & PV01_N(y_N) \\ DP_1(y_1) & \ldots & DP_N(y_N) \end{pmatrix} \cdot q = \begin{pmatrix} PV01_L(y_L) \\ DP_L(y_L) \end{pmatrix} \\ &\forall i \in 1:N, \, q_i \geq 0 \end{aligned}$$

where $q = [q_1, \ldots, q_N]$. Notice that in "lp" function, there is 1 constraint per row and for each constraint we need to defined the constraint direction. The constraints matrix will be :

$$A = \begin{pmatrix} PV01_1(y_1) & \ldots & PV01_N(y_N) \\ DP_1(y_1) & \ldots & DP_N(y_N) \\ & I_N & \end{pmatrix}$$

The right-hand side array will be :

$$b = \begin{pmatrix} PV01_L(y_L) \\ DP_L(y_L) \\ 0_{N \times 1} \end{pmatrix}$$

```
res = lp(
  direction = "max",
  objective.in = pv01_bonds * yield_bonds,
  const.mat = rbind(pv01_bonds, dp_bonds, diag(nb.bonds)),
  const.dir = c(rep("=", 2), rep(">=", nb.bonds)),
  const.rhs = c(pv01_liability, dp_liability, rep(0, nb.bonds))
  )
q = res$solution
```

We can see our portfolio is only invested in two bonds : bond n°1 and bond n°28.

Table 3: Immunization at 2021/03/17 of €10,000,000 liability due for 2025/01/02

Bond N°	Quantity
1	49017.09
28	33062.07

I.b) Cash Flow Matching & Immunization

We now consider a liability made of several cash flows, as summarized below :

Table 4: Liability Cash Outflows

Date	Amount
2021-10-01	1,000,000
2022-04-01	1,000,000
2022-10-01	1,000,000
2023-04-01	1,000,000
2023-10-01	1,000,000
2024-04-01	1,000,000
2024-10-01	10,000,000

Build a portfolio with maximal return such that :

- we do cash flow matching for the first 4 liability cash flows
- on 2023/04/01 (immunization date), the present value and PV01 of the asset and liability are equal.

We assume yield curve on 2023/04/01 is the same as on 2021/03/17.

Moreover, the current date is 2025/03/17.

First, let us recall what is Clash Flow Matching. In this method, the goal is to balance cash flows between assets and liabilities on the period desired, according to the flow equation :

$$C(t) = \left(1 + \frac{r}{2}\right) C(t-1) + \sum_{i=1}^{N} q_i F_i(t) - L(t)$$

where $C(t)$ is cash holdings at t, r is the annual return on cash, q_i and $F_i(t)$ are respectively quantity and cash flow at t of asset i, and $L(t)$ is liability cash flow at t. For simplicity we take $C(0) = 0$. Be careful, our liability cash flows are semi-annual hence the division by 2 of r.

Thus combining the equations of previous part about Immunization and this Cash Flow Matching method and noticing we still want to maximize the average return, we have the following optimization problem (here $T = 4$, immunization date) :

$$\max_{q} \sum_{i=1}^{N} q_i \cdot PV01_i(y_i) \cdot y_i$$

$$
\begin{aligned}
s.t.\ \forall t \in 1:T,\ & \left(1+\frac{r}{2}\right) C(t-1) + \sum_{i=1}^{N} q_i F_i(t) - C(t) = L(t) \\
& \sum_{i=1}^{N} q_i \cdot PV01_i(y_i, T) = PV01_L(y_L, T) \\
& \sum_{i=1}^{N} q_i \cdot DP_i(y_i, T) = DP_L(y_L, T) \\
& \forall i \in 1:N,\ q_i \geq 0 \\
& C(0) = 0 \\
& \forall t \in 1:T,\ C(t) \geq 0
\end{aligned}
$$

Then, let us recalculate PV01 and dirty prices of our selected bonds as now the immunization date is 2023/04/01.

```
immun_date = ymd('2023-04-01') # Immunization date
dp_bonds = rep(0, times = nb.bonds) # Bonds dirty prices
pv01_bonds = rep(0, times = nb.bonds) # Bonds PV01
yield_bonds = rep(0, times = nb.bonds) # Bonds yields

for (i in 1:nb.bonds) {

  bond = df.o[i, ]

  # we focus only on bonds with maturity above immunization date
  # if a bond has maturity below immunization date, we simply set its dirty price,
  # PV01 and yield at 0, so it won't be taken into account for immunization
  # but notice it can be taken into account in cash flow matching method, so we
  # cannot just remove it
  if (bond$dtM > immun_date) {

    yield = yield_func(bond$dtM)
    res = BondVal.Price(YtM = yield, SETT = immun_date, Em = bond$dtE, Mat = bond$dtM,
                        CpY = 1, DCC = 3, Coup = bond$Coupon)
    res_below = BondVal.Price(YtM = yield - 0.01, SETT = immun_date, Em = bond$dtE,
                              Mat = bond$dtM, CpY = 1, DCC = 3, Coup = bond$Coupon)

    dp_bonds[i] = res$DP
    pv01_bonds[i] = res_below$DP - res$DP
    yield_bonds[i] = yield
  }
}
```

Let us also calculate the PV01 and present value of our liability as of immunization date (notice that we now have several cash flows).

```
 # Liability cash flows above our immunization date
df_flow_selec = df.flow[df.flow$dt > immun_date, ]
dp_liability = 0 # Liability value as of immunization date
pv01_liability = 0 # Liability PV01 as of immunization date

for (i in 1:nrow(df_flow_selec)) {
```

```
  # Liability cash flow on a given date can be seen as a zero-coupon bond
  bond = df_flow_selec[i, ]
  yield = yield_func(bond$dt)
  res = BondVal.Price(YtM = yield, SETT = immun_date, Em = immun_date,
                      Mat = bond$dt, CpY = 0, DCC = 3, Coup = 0)
  res_below = BondVal.Price(YtM = yield - 0.01, SETT = immun_date,
                            Em = immun_date, Mat = bond$dt, CpY = 0, DCC = 3, Coup = 0)

  # Total value of the liability is the sum of values of each zero-coupon bond
  # that compose our liability (weighted by cash flow value)
  # As before, we need to divide by 100
  dp_liability = dp_liability + res$DP / 100 * bond$vx
  pv01_liability = pv01_liability + (res_below$DP - res$DP) / 100 * bond$vx
}
```

Finally, let us retrieve our assets cash flows (i.e. coupons) in a matrix $F \in \mathbb{R}_+^{T \times N}$, using the function "AnnivDates" of package "BondValuation". We must retrieve them at liability cash flow dates, so if the assets cash flows are received before the next liability cash flow, we assume we reinvest them at our rate of return on cash r. We'll fix $r = 1\%$.

```
r = 0.01 # yearly return on cash
T_ = 4 # Number of periods on which we do cash flow matching
CF_mat = matrix(0, nrow = T_, ncol = nb.bonds)

for (i in 1:T_) {

  # Be careful, all bonds are eligible for cash flow matching (as no bonds have
  # maturity below current date), contrary to immunization
  for (j in 1:nb.bonds) {

    bond = df.o[j, ]
    # gives the dates and values of coupons of the bond (notice it only represent
    # coupons; face value is not added at the end)
    res = AnnivDates(Em=bond$dtE, Mat=bond$dtM, CpY=1, Coup=bond$Coupon, DCC=3)$PaySched
    list_pay_dates = res$CoupDates # dates of coupons
    list_pay_amounts = res$CoupPayments # values of coupons
    nb_flows = nrow(res) # number of coupons
    # we add bond face value to last coupon, sych that we have the bond cash flows
    list_pay_amounts[nb_flows] = list_pay_amounts[nb_flows] + 100

    for (k in 1:nb_flows) {

      pay_date = list_pay_dates[k] # date of bond's cash flow
      # if the bond cash flow date is above previous liability cash flow date and
      # lower than next liability cash flow date or above current date if we're
      # considering first liability outflow (as we'll buy the bonds at current date)
      if (i==1) {
        prev_liab_date = date_set
      } else {
        prev_liab_date = df.flow[i-1, ]$dt
      }
      next_liab_date = df.flow[i, ]$dt

      if ((pay_date > prev_liab_date) & (pay_date <= next_liab_date)) {
```

```
        # number of years between bond cash flow date and date of liability cash flow
        nb_years_btw = as.numeric(next_liab_date - pay_date) / 365
        # We reinvest the bond cash inflows received before next date of liability
        # cash outflow at cash rate of return until this date
        CF_mat[i, j] = CF_mat[i, j] + list_pay_amounts[k] * (1 + r)^nb_years_btw
      }
    }
  }
}
```

Now, to solve this linear optimization problem, we'll also use the function "lp" of the linear programming library "lpSolve". In matrix formulation, our problem is :

$$\max_u \left[PV01_1(y_1)\cdot y_1,\ \ldots,\ PV01_N(y_N)\cdot y_N,\ 0,\ 0,\ 0,\ 0\right]\cdot u$$

$$s.t. \begin{pmatrix} F_1(1) & \ldots & F_N(1) & -1 & 0 & 0 & 0 \\ F_1(2) & \ldots & F_N(2) & 1+\frac{r}{2} & -1 & 0 & 0 \\ F_1(3) & \ldots & F_N(3) & 0 & 1+\frac{r}{2} & -1 & 0 \\ F_1(4) & \ldots & F_N(4) & 0 & 0 & 1+\frac{r}{2} & -1 \\ PV01_1(y_1,T) & \ldots & PV01_N(y_N,T) & 0 & 0 & 0 & 0 \\ DP_1(y_1,T) & \ldots & DP_N(y_N,T) & 0 & 0 & 0 & 0 \end{pmatrix} \cdot u = \begin{pmatrix} L(1) \\ L(2) \\ L(3) \\ L(4) \\ PV01_L(y_L) \\ DP_L(y_L) \end{pmatrix}$$

$$\forall i \in 1:(N+T),\ u_i \geq 0$$

where $u = \begin{pmatrix} q \\ C(1) \\ | \\ C(T) \end{pmatrix}$ and recall that $T = 4$. Let us denote this system $R \cdot u = d$.

Now, the constraints matrix for "lp" function will be :

$$A = \begin{pmatrix} R \\ I_{N+T} \end{pmatrix}$$

The right-hand side array will be :

$$b = \begin{pmatrix} d \\ 0_{(N+T)\times 1} \end{pmatrix}$$

```
match_mat = diag(-1, nrow=T_)
for (i in 2:T_) {

  match_mat[i, i-1] = 1 + r / 2
}
# Constraint matrix corresponding to cash flow matching part
match_mat = cbind(CF_mat, match_mat)

immun_mat = rbind(pv01_bonds, dp_bonds)
# Constraint matrix corresponding to immunization part
immun_mat = cbind(immun_mat, matrix(0, nrow=2, ncol=T_))

R = rbind(match_mat, immun_mat)
```

Table 5: Cash Flow Matching and Immunization on liability with several cash flows

Bond N°	Quantity
1	3935.228
3	9716.247
4	4300.632
6	9881.423
7	93544.106
28	15297.666

```
res = lp(
  direction = "max",
  objective.in = c(pv01_bonds * yield_bonds, rep(0, T_)),
  const.mat = rbind(R, diag(nb.bonds + T_)),
  const.dir = c(rep("=", T_+2), rep(">=", nb.bonds+T_)),
  const.rhs = c(df.flow[1:T_, "vx"], pv01_liability, dp_liability, rep(0, nb.bonds+T_))
  )
u = res$solution
q = u[1:nb.bonds] # quantities invested in bonds
```

II/ Bond Option Pricing with Black-Derman-Toy Model

We consider the Black-Derman-Toy model, described in the course notes.

The questions are the following :

1. Tree calibration: generalize the calibration method seen in class to be able to build a tree of N steps, and increment Δt.
2. From Boyle paper (2000), use the Arrow-Debreu prices to optimize the calculations.
3. Build a tree with maturity of 5 years and timestep of 1 month
4. Use this tree to price a call with strike 79, maturity 1 year, on a bond with maturity 5 years and coupon 5%.

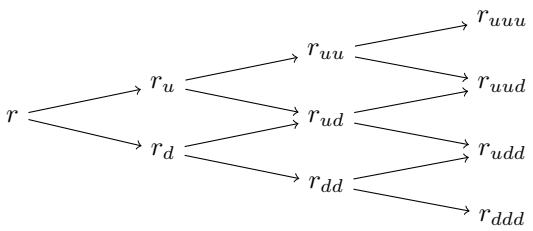

Figure 2: Black-Derman-Toy short rate tree

The objective of this section is to calibrate the model to a zero-coupon curve and a zero-coupon rate volatility curve in order to price a bond option.

Data

Table 6: Annual ZC Yields and Annual ZC Yield Volatilities

Maturity	$z(t)$	$\gamma(t)$
1	10.0	
2	11.0	19
3	12.0	18
4	12.5	17
5	13.0	16

```
z <- data.bdt$z/100
vol <- data.bdt$b/100
```

Let us define interpolation function for ZC curve and volatility curve. We add a short-term rate to the ZC curve to enable a robust interpolation.

```
zc.curve <- splinefun(seq(0,5), c(.09, z))
vol[1] <- .2
vol.curve <- splinefun(seq(0,5), c(.21, vol))
```

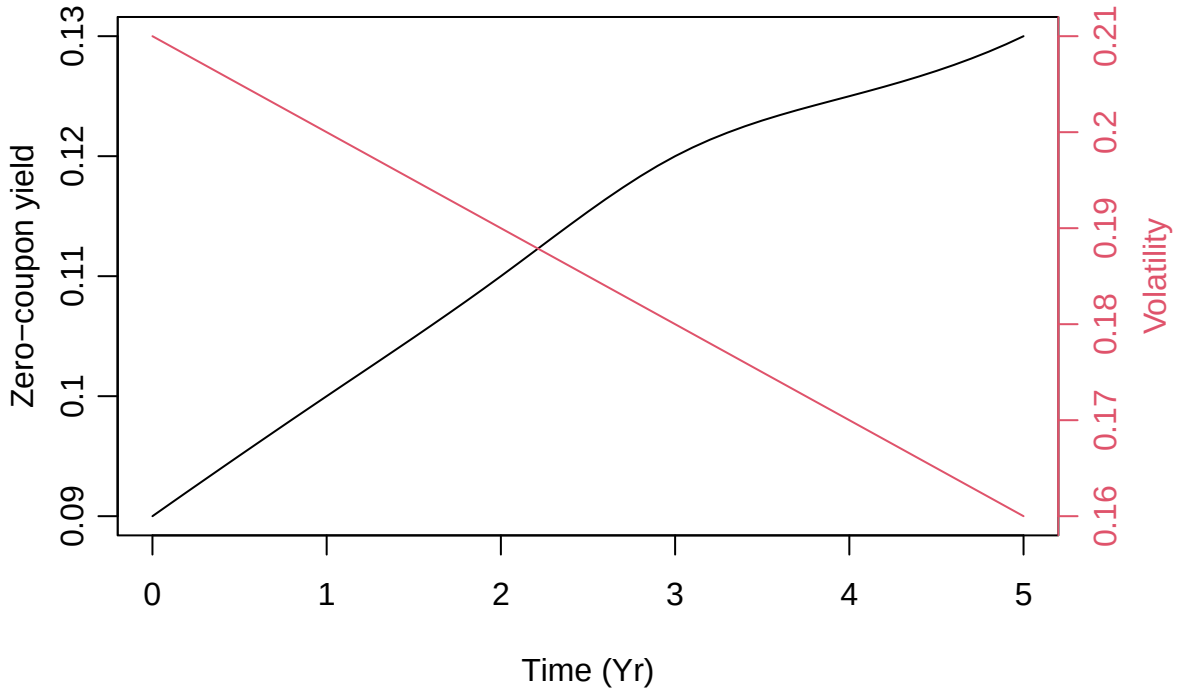

Figure 3: Interpolated ZC Yield Curve and Volatility Curve

Question 1.

The Black-Derman-Toy model is a model of interest rates that can be used to value any interest-rate sensitive security. The only factor is the short rate (i.e. annualized one-period interest rate), and the model takes as inputs zero-coupon yields, as well as yield volatilities for the same bonds. We also assume short rates at any time are lognormally distributed (i.e. logarithm of short rate follows a gaussian distribution).

The idea is to first build the short-rates tree with forward induction, and then construct the option prices tree from this short-rates tree using backward induction. Let us focus on the first step (named calibration step, as we calibrate our model short rates to the data).

Assume we have a binomial tree, with N periods and a timestep Δt. All ZC bonds here are considered with face value of 1. We denote by :

$\hat{Y}(n)$ the annualized 1-period market yield on a ZC bond with maturity $n\Delta t$

$\hat{\sigma_Y}(n)$ the market volatility of annualized 1-period yield on a ZC bond with maturity $n\Delta t$

$\hat{P}(n) = \frac{1}{\left(1+\hat{Y}(n)\Delta t\right)^n}$ the market price of a ZC bond with maturity $n\Delta t$

We also denote by :

$Y(n)$ the annualized 1-period model yield on a ZC bond with maturity $n\Delta t$

$\sigma_Y(n)$ the model implied volatility of annualized 1-period yield on a ZC bond with maturity $n\Delta t$

$P(n) = \frac{1}{(1+Y(n)\Delta t)^n}$ the model implied price (at node $(0,0)$) of a ZC bond with maturity $n\Delta t$

And finally, $\forall n \in 0 : N-1 \ \forall i_n \in 0 : n$, $r(n,i)$ is the annualized model 1-period short rate at period n and state i.

Let us also denote by $Y_u(n)$ and $Y_d(n)$ the annualized 1-period model yield on a ZC bond with maturity $n\Delta t$ at period 1, respectively on node $(1,1)$ and node $(1,0)$. Similarly, we denote $P_u(n)$ and $P_d(n)$ the model price of a ZC bond with maturity $n\Delta t$ at period 1, respectively on node $(1,1)$ and node $(1,0)$.

Thus, assuming equiprobability of an up or down movement in short rates :

$$P(n) = \frac{1}{1+r(0,0)\Delta t} \cdot \left[\frac{1}{2}P_d(n) + \frac{1}{2}P_u(n)\right]$$

Notice that if we know all previous short rates: $\forall k \in 0 : n-2 \ \forall i_k \in 0 : k \ r(k,i_k)$, then both $P_d(n)$ and $P_u(n)$ can be computed only as functions of parts of short rates to be determined at period $n-1$. For instance, for $n = 3$:

$$P_u(3) = \frac{1}{1+r(1,1)\Delta t} \cdot \left[\frac{1}{2} \cdot \frac{1}{1+r(2,1)\Delta t} + \frac{1}{2} \cdot \frac{1}{1+r(2,2)\Delta t}\right]$$

$$P_d(3) = \frac{1}{1+r(1,0)\Delta t} \cdot \left[\frac{1}{2} \cdot \frac{1}{1+r(2,0)\Delta t} + \frac{1}{2} \cdot \frac{1}{1+r(2,1)\Delta t}\right]$$

Now, the objective of the BDT model is to ensure that $\hat{P}(n) = P(n)$, and $\hat{\sigma_Y}(n) = \sigma_Y(n)$ (calibration), thus we have :

$$\frac{1}{\left(1+\hat{Y}(n)\Delta t\right)^n} = \frac{1}{1+r(0,0)\Delta t} \cdot \left[\frac{1}{2}P_d(n) + \frac{1}{2}P_u(n)\right]$$

Moreover, notice that :

$$P_u(n)\left(1+Y_u(n)\Delta t\right)^{n-1} = 1 \Leftrightarrow Y_u(n) = \frac{1}{\Delta t}\left(\frac{1}{{P_u}^{n-1}} - 1\right)$$

$$P_d(n)\left(1+Y_d(n)\Delta t\right)^{n-1} = 1 \Leftrightarrow Y_d(n) = \frac{1}{\Delta t}\left(\frac{1}{{P_d}^{n-1}} - 1\right)$$

Then, at period 1, the implied volatility of annualized 1-period yield on a ZC bond with maturity $n\Delta t$ is :

$$\begin{aligned}\sqrt{\mathbb{V}\left[\ln\left(Y(n)\right)\right]} &= \sqrt{\mathbb{E}\left[\ln\left(Y(n)\right)^2\right] - \mathbb{E}\left[\ln\left(Y(n)\right)\right]^2} \\ &= \sqrt{\frac{1}{2}\ln\left(Y_d(n)\right)^2 + \frac{1}{2}\ln\left(Y_u(n)\right)^2 - \left(\frac{1}{2}\ln\left(Y_d(n)\right) + \frac{1}{2}\ln\left(Y_u(n)\right)\right)^2} \\ &= \frac{1}{2}\ln\left(\frac{Y_u(n)}{Y_d(n)}\right)\end{aligned}$$

Now, as we consider such yield to have log-normal distribution (i.e. $\ln\left(Y(n)\right) \sim \mathcal{N}\left(\mu, \sigma_Y(n)\Delta t\right)$) and recalling we should have $\hat{\sigma_Y}(n) = \sigma_Y(n)$, then :

$$\frac{1}{2}\ln\left(\frac{Y_u(n)}{Y_d(n)}\right) = \hat{\sigma_Y}(n)\sqrt{\Delta t}$$

Finally, we also want local volatility of our model short rates at period $n-1$ to be constant (and recall short rates are following log-normal distribution) :

$$\begin{aligned}&\forall i \in 0 : n-1,\ \frac{1}{2}\ln\left(\frac{r(n-1,i+1)}{r(n-1,i)}\right) = cste \\ &\Leftrightarrow \forall i \in 0 : n-1,\ r(n-1,i+1) = r(n-1,i)\cdot\beta(n) \\ &\Leftrightarrow \forall i \in 0 : n,\ r(n-1,i) = r(n-1,0)\cdot\beta(n-1)^i\end{aligned}$$

Thus, at period $n-1$ (i.e. for pricing ZC bonds with maturity $n\Delta t$), knowing all short rates at previous periods, we only have 2 unknown variables: $r(n-1,0)$ and $\beta(n-1)$, since other rates can at this period can be deduced from above relation.

Then, from what we've said before, we can write $P_d(n) = f\left(r(n-1,0), \beta(n-1)\right)$ and $P_u(n) = g\left(r(n-1,0), \beta(n-1)\right)$, f and g being non-linear functions. Finally, putting everything together, we have 2 unknowns for 2 equations :

$$\frac{1}{\left(1+\hat{Y}(n)\Delta t\right)^n} = \frac{1}{1+r(0,0)\Delta t}\cdot\left[\frac{1}{2}f\left(r(n-1,0),\beta(n-1)\right) + \frac{1}{2}g\left(r(n-1,0),\beta(n-1)\right)\right]$$

$$\frac{1}{2}\ln\left(\frac{\frac{1}{g(r(n-1,0),\beta(n-1))^{n-1}} - 1}{\frac{1}{f(r(n-1,0),\beta(n-1))^{n-1}} - 1}\right) = \hat{\sigma_Y}(n)\sqrt{\Delta t}$$

The tree can thus be constructed forward, period by period. The second step in the BDT model is simply to price the option backward, as in a CRR (Cox, Ross, Rubinstein) model, but using the short rates calculated previously with the first tree, to compute the bond price and then the option price at each node.

Although this procedure might be straightforward using a nonlinear solver, the tricky part here is to define $P_d(n)$ and $P_u(n)$ (i.e. functions f and g) as we have to work backward in the tree, from period $n-1$ to period 1.

Question 2.

Fortunately, there is a way to simplify the tricky part mentioned above. As pointed out in Boyle's article, the idea is to make the calibration procedure by the use of a forward induction technique that involves using Arrow-Debreu securities. With this technique, we won't have to work backward through the tree one period at a time to obtain the expressions of $P_d(n)$ and $P_u(n)$.

We denote by $\forall n \in 0 : N-1 \ \forall i_n \in 0 : n \ A(n,i)$ the price of the Arrow-Debreu security which pays 1 unit at time $(n+1)\Delta t$ (i.e. period n) in state i, and 0 elsewhere. We can deduce the following recursive relation in our binomial tree :

$$A(n,i) = \begin{cases} \frac{A(n-1,i-1)}{2[1+r(n-1,i-1)\Delta t]} & i = n \\ \frac{A(n-1,i-1)}{2[1+r(n-1,i-1)\Delta t]} + \frac{A(n-1,i)}{2[1+r(n-1,i)\Delta t]} & i \in 1 : n-1 \\ \frac{A(n-1,i)}{2[1+r(n-1,i)\Delta t]} & i = 0 \end{cases}$$

Moreover, we have :

$$P(n) = \sum_{i=0}^{n-1} A(n-1,i)$$

Let us also denote by $A_u(n,i)$ and $A_d(n,i)$ the model price of an Arrow-Debreu security which pays 1 unit at time $(n+1)\Delta t$ (i.e. period n) in state i, and 0 elsewhere, at period 1, respectively on node $(1,1)$ and node $(1,0)$. Then we have :

$$P_u(n) = \sum_{i=0}^{n-1} A_u(n-1,i)$$

$$P_d(n) = \sum_{i=0}^{n-1} A_d(n-1,i)$$

Notice that $\forall n \in 0 : N-1$, $A_d(n,n) = A_u(n,0) = 0$, and that the recursive relation is also verified for A_d and A_u.

Taking back our notations of Question 1., and using the our Arrow-Deubreu securities, we have :

$$\begin{aligned} f\left(r(n-1,0), \beta(n-1)\right) &= \sum_{i=0}^{n-1} A_d(n-1,i) \\ &= \sum_{i=0}^{n-2} \frac{A_d(n-2,i)}{1+r(n,i)\Delta t} \\ &= \sum_{i=0}^{n-2} \frac{A_d(n-2,i)}{1+r(n,0)\beta(n-1)^i\Delta t} \end{aligned}$$

$$\begin{aligned} g\left(r(n-1,0), \beta(n-1)\right) &= \sum_{i=0}^{n-1} A_u(n-1,i) \\ &= \sum_{i=0}^{n-2} \frac{A_u(n-2,i)}{1+r(n,i)\Delta t} \\ &= \sum_{i=0}^{n-2} \frac{A_u(n-2,i)}{1+r(n,0)\beta(n-1)^i\Delta t} \end{aligned}$$

where $A_d(n-2,i)$ and $A_u(n-2,i)$ have been fully determined at previous step, from previous recursive relation.

Thus, we can see that f and g can be easily computed (since we just need at each step to calculate all Arrow-Debreu prices corresponding to this step), and we can price the option the same way as explained in Question 1.

Question 3.

The first step is to recover data for our ZC yields and volatilities for each month of our 5 years, using the interpolation functions defines previously.

```
maturity = 5
dt = 1/12
t = seq(0, maturity, dt)

Y_hat = zc.curve(t) # Yields
# do not take first value as it is for a maturitity of 0
Y_hat = Y_hat[2:length(Y_hat)]

sigma_hat = vol.curve(t) # Volatilities
# do not take first value as it is for a maturitity of 0
sigma_hat = sigma_hat[2:length(sigma_hat)]
# volatility should not be defined for first value
sigma_hat[1] = NA
```

Unfortunately, we did not have time to implement by ourselves the method explained previously to build the short rates tree. Instead, we wanted to directly use a package in R. But it seems there exist no up-to-date package to compute the Black-Derman-Toy model.

Question 4.

As we did not have time to compute the short rates tree, we also did not have time to price the option considered. However the steps are the following.

The first step is to price the bond at each node of the tree, by going through the tree backwards, using the short rate at this node as discount rate and noticing that we must take into account the coupons of the bond when pricing it.

Once the tree of bond (underlying) prices has been generated, then we would do as in a CRR (Cox, Ross, Rubinstein) model, i.e. going through the tree backwards by computing payoff of the option at a node as :

$$f = \frac{1}{1 + r\Delta t}\left(\frac{1}{2}f_u + \frac{1}{2}f_d\right)$$

where r is the short rate at the corresponding node, f_u and f_d are option payoff at respectively up and down scenario from the node considered. Notice that we dot not price the option on the entire tree but only on the section corresponding to first year, as the option maturity is 1 year, whereas the bond maturity is 5 years. Moreover, recall that at option maturity, the payoff is given (as we have a call of strike $K = 79$) :

$$f = \max(B_T - K, 0)$$

where B_T is the computed bond price at 1 year.

Conclusion :

This lab gave us the opportunity to first implement immunization on a single cash flow liability by investing in a portfolio of bonds, while maximizing the average return of this portfolio. The immunization has been made by balancing PV01 and Present Value of portfolio and liability as of immunization date. However, in reality, the portfolio should be rebalanced as the yield curve changes. We have then implemented cash flow matching and immunization on a liability with several cash outflows by investing again in a portfolio of bonds, while maximizing the average return of this portfolio. We matched cash flows for the first 4 periods, and then immunized the portfolio for the next periods (assuming yield curve does not change). Cash flow

matching is a method adapted for short term maturities as it avoids doing many transactions, but lacks of flexibility at longer maturities, if for instance we have new liability cash outflows to be settled.

The second part of this lab was about implementing the Black-Derman-Toy model, which focuses on pricing options on interest-rate sensitive underlying using a model of short rates. We first defined the principles and assumptions of this model, noticing that the goal is to match observed market zero-coupon yields and their associated volatilities for different maturities, to the ones implied by the model. We have seen that the implementation is not trivial as it involves computing some quantities by going through all the tree backward. However, calculations can be simplified by using Arrow-Debreu securities, as described in Boyle's article. Finally, we did not have time to implement ourselves this model, but we shared the guidelines and the steps to build such model.

Note : Concerning LLM, they were not as proficient in R as they can be in other languages such as Python. Moreover, the .Rmd version of exercices solutions and of the slides available to us were great sources of information. As a result, we ended up using LLMs on really rare occasions and mainly focused on the files in IMT-GP-2024 and on the rdocumentation site.