Finance Quantitative

TP: Modèle de Black-Derman-Toy

Patrick Hénaff

Version: 30 mars 2023

1 Le modèle de Black-Derman-Toy

On considère le modèle de Black, Derman et Toy décrit dans la note de cours.

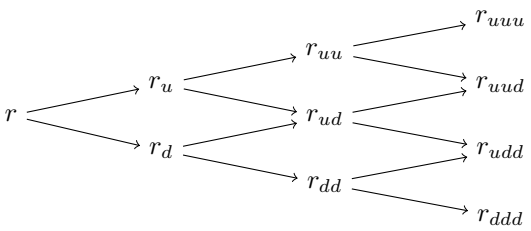


Figure 1: Black-Derman-Toy short rate tree

On doit calibrer le modèle à une courbe zero-coupon et une courbe de volatilité du taux zero-coupon.

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

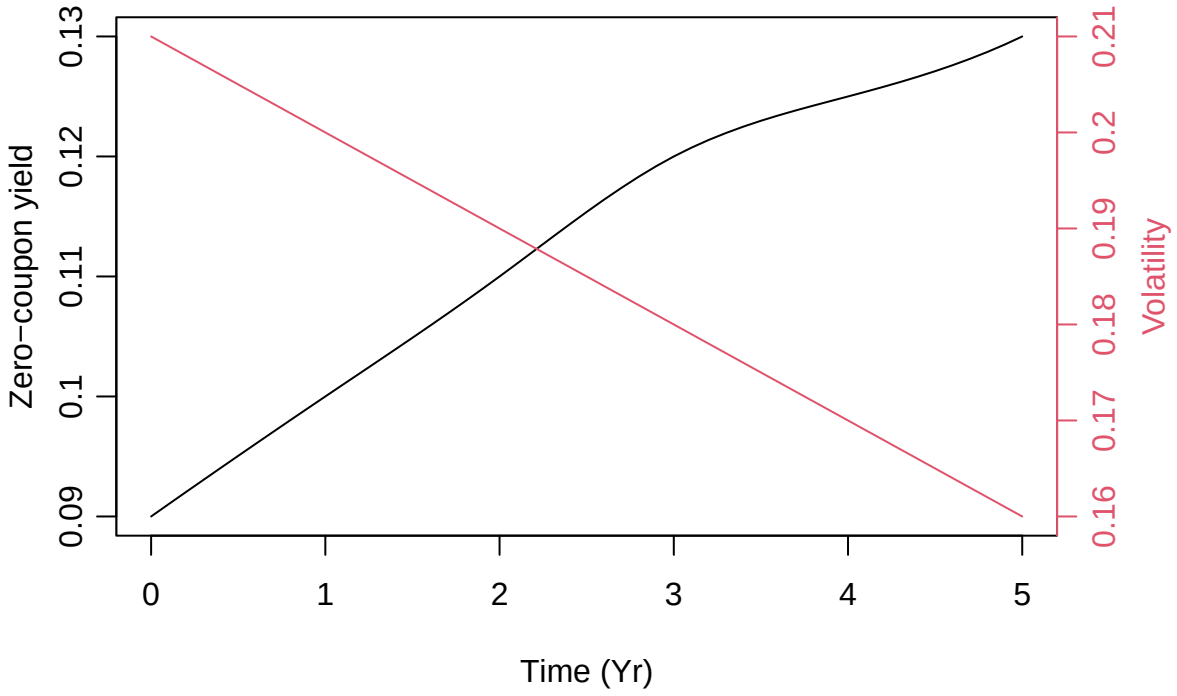
1.1 Construction d'un arbre BDT

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

Fonctions d'interpolation pour la courbe zero-coupon et la courbe de volatilité. On ajoute un taux court à la courbe zero-coupon pour permettre une interpolation robuste.

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

df <- function(r) {
  1/(1+r)
}
```



2 Questions

1. Calibrage de l'arbre: généraliser la méthode de calibration vue en cours pour pouvoir construire un arbre de n pas, et d'incrément Δt.
2. A partir de l'article de Boyle (2000), utiliser les prix d'Arrow-Debreu pour optimiser les calculs.
3. Construire un arbre de maturité 5 ans, par pas de temps de 1 mois.
4. Utiliser cet arbre pour valoriser un call de strike 79, de maturité 1 an, sur une obligation de maturité 5 ans et de coupon 5%.

Définissons une fonction qui calcule le prix et volatilité du rendement d'une obligation zéro-coupon de maturité $n\Delta t$, On suppose que les variables $\hat{r}(k), \alpha(k), k = 1, \ldots, n$ sont connues. On note que $\hat{r}(1)$ est obtenu directement à partir de la courbe zero-coupon $z(t)$ et que $\alpha(1) = 1$.

```
PV.Y <- function(n, r.hat, alpha, delta.t) {
  vol = NA
  # value=1 at time n \Delta t
  pv <- as.vector(rep(1,n+1))
  for(i in seq(n, 1, -1)) {
    iExp <- seq(from=0, to=(i-1), by=1)
    r <- r.hat[i] * exp(2*alpha[i] * iExp)
    discount.fac <- 1/(1+r)^delta.t
    pv <- .5 * discount.fac * pv[2:(i+1)] + .5 * discount.fac * pv[1:i]
    if(i==2) {
      Y.up <- (1/pv[2])^(1/((n-1)*delta.t)) - 1
      Y.down <- (1/pv[1])^(1/((n-1)*delta.t)) - 1
      vol <- (1/2) * log(Y.up / Y.down)
    }
```

```
  }
  list(pv=pv[1], vol=vol)
}
```

Définissons le système d'équations à résoudre pour un pas de temps n, sachant que les pas de temps précédents ont été résolus:

```
obj <- function(x, n, delta.t) {
  r.hat[n] <- x[1]
  alpha[n] <- x[2]
  tmp <- PV.Y(n, r.hat, alpha, delta.t)
  # browser()
  res <- numeric(2)
  # z(n) is the zero-coupon yield of a bond maturing at n \Delta t
  res[1] <- tmp$pv - df(zc.curve(n*delta.t))^(n*delta.t)
  res[2] <- tmp$vol - vol.curve(n*delta.t)
  res
  }
```

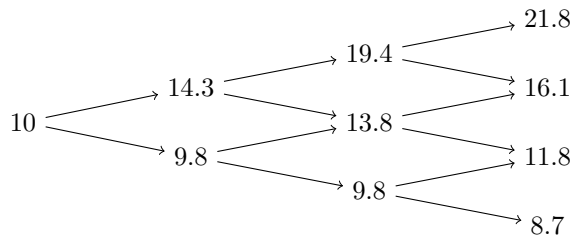
On peut maintenant calibrer l'arbre un pas de temps à la fois, en commencant par la maturité $2\Delta t$. La maturité Δt est obtenue par interpolation sur la courbe zero-coupon.

```
r.hat <- numeric(5)
alpha <- numeric(5)
r.hat[1] <- z[1]
alpha[1] <- 1

for(n in seq(2,5)) {
  # valeurs initiales de r.hat et alpha
  x.0 <- as.vector(c(.1, .2))
  sol <- nleqslv(x.0, obj, n=n, delta.t=1)
  r.hat[n] <- sol$x[1]
  alpha[n] <- sol$x[2]
  print(sol$x)
}
```

```
## [1] 0.0979156 0.1900000
## [1] 0.09759998 0.17198636
## [1] 0.08717235 0.15268201
## [1] 0.08653436 0.13521070
```

Les solutions pour les trois premiers pas de temps sont représentés ci-dessous.



Vérification: on valorise des obligations zero-coupon dans l'arbre, et on doit retrouver les prix calculés directement avec la courbe de taux.

```
delta.t <- 1
error <- 0
for(n in seq(1,5)) {
  P.ZC <- 100 * df(zc.curve(n*delta.t))^(n*delta.t)
  P.BDT <- 100 * PV.Y(n, r.hat, alpha, delta.t)$pv
  error <- max(error, abs(P.ZC-P.BDT))
}
```

L'erreur maximale est: 1.2688162×10^{-7}.

2.1 Optimisation des calculs

La fonction PV.Y répète inutilement des calculs d'actualisation, alors que le seul élément variable du calcul est l'actualisation du pas de temps $n+1$ vers n à l'aide des variables $\hat{r}(n)$ et $\alpha(n)$. On peut grandement simplifier les calculs en calculant l'actualisation du pas de temps 2 à n avec les prix d'Arrow-Debreu associés aux états de l'étape n.

```
PV.Y <- function(n, r.hat, alpha, delta.t) {
  vol = NA
  # value=1 at time n \Delta t
  pv <- as.vector(rep(1,n+1))
    iExp <- seq(from=0, to=(n-1), by=1)
    r <- r.hat[n] * exp(2*alpha[n] * iExp)
    discount.fac <- 1/(1+r)^delta.t
    pv <- .5 * discount.fac * pv[2:(n+1)] + .5 * discount.fac * pv[1:n]
    pv.up <- sum(pv[2:n]*AD.up)
    pv.down <- sum(pv[1:(n-1)]*AD.down)
      Y.up <- (1/pv.up)^(1/((n-1)*delta.t)) - 1
      Y.down <- (1/pv.down)^(1/((n-1)*delta.t)) - 1
      vol <- (1/2) * log(Y.up / Y.down) / sqrt(delta.t)
    pv.0 <- (1/2)*(pv.up+pv.down)/(1+zc.curve(delta.t))^delta.t

  list(pv=pv.0, vol=vol)
}
```

La fonction objectif ne change pas, mais il faut mettre à jour les prix d'Arrow-Debreu à chaque itération.

```
r.hat <- numeric(5)
alpha <- numeric(5)
r.hat[1] <- z[1]
alpha[1] <- 1
AD.up <- 1
AD.down <- 1
delta.t <- 1

for(n in seq(2,5)) {
  # valeurs initiales de r.hat et alpha
  x.0 <- as.vector(c(.1, .2))
  sol <- nleqslv(x.0, obj, n=n, delta.t=1)
  r.hat[n] <- sol$x[1]
  alpha[n] <- sol$x[2]
  tmp <- numeric(length=n)
```

```
  iExp <- seq(0, (n-1))
  r <- r.hat[n] * exp(2*alpha[n]*iExp)
  # AD prices to up state
  tmp[-1] <- (1/2)*AD.up/(1+r[-1])^delta.t
  tmp[-n] <- tmp[-n] + (1/2) *AD.up/(1+r[-1])^delta.t
  # browser()
  AD.up <- tmp
  tmp <- numeric(length=n)
  tmp[-1] <- (1/2)*AD.down/(1+r[-n])^delta.t
  tmp[-n] <- tmp[-n] + (1/2)* AD.down/(1+r[-n])^delta.t
  AD.down <- tmp
  # browser()
  }
```

Construction d'un arbre de maturité 5 ans avec des pas de temps de 1 mois.

```
delta.t = 1/12
horizon = 5
nb.steps <- round(horizon/delta.t)
r.hat <- numeric(nb.steps)
alpha <- numeric(nb.steps)
r.hat[1] <- zc.curve(delta.t)
alpha[1] <- .1
AD.up <- 1
AD.down <- 1
for (n in seq(2, nb.steps)) {
  x.0 <- as.vector(c(r.hat[n-1], alpha[n-1]))
  sol <- nleqslv(x.0, obj, n=n, delta.t=delta.t)
  if(sol$termcd != 1) {
    print(paste("nleqslv error for n:", n))
    print(sol)
    break
  }
  r.hat[n] <- sol$x[1]
  alpha[n] <- sol$x[2]
  tmp <- numeric(length=n)
  iExp <- seq(0, (n-1))
  r <- r.hat[n] * exp(2*alpha[n]*iExp)
  # AD prices to up state
  tmp[-1] <- (1/2)*AD.up/(1+r[-1])^delta.t
  tmp[-n] <- tmp[-n] + (1/2) *AD.up/(1+r[-1])^delta.t
  # browser()
  AD.up <- tmp
  tmp <- numeric(length=n)
  tmp[-1] <- (1/2)*AD.down/(1+r[-n])^delta.t
  tmp[-n] <- tmp[-n] + (1/2)* AD.down/(1+r[-n])^delta.t
  AD.down <- tmp
  }
```

On commence par écrire une fonction qui valorise une obligation à une date future définie $t_0 = n_0 \times \Delta t$. L'obligation est définie par une échéancier de flux payés à des dates $t_i = n_i \times \Delta t, i = 1, \ldots, N$.

```
CF.discount <- function(r.hat, alpha, delta.t, n.pv, cf=NULL, pv=NULL) {
  # noeud associé au dernier cash-flow
  if(!is.null(cf)) {
    n.max <- max(cf$dt)+1
    idx = which((cf$dt+1) == n.max)
    pv <- as.vector(rep(cf$flow[idx],n.max))
  } else {
    n.max <- length(pv)
  }
  for(i in seq(n.max-1, n.pv, -1)) {
    iExp <- seq(from=0, to=(i-1), by=1)
    r <- r.hat[i] * exp(2*alpha[i] * iExp)
    discount.fac <- 1/(1+r)^delta.t
    pv <- (discount.fac * pv[2:(i+1)] + discount.fac * pv[1:i])/2
    # on ajoute la valeur du coupon payé à cette date.
    idx <- which((cf$dt+1) == i)
    if(length(idx) == 1) {
      pv <- pv + cf$flow[idx]
    }
  }
  pv
}
```

Vérifications:

Facteur d'actualisation d'un cash-flow payé dans n pas de temps:

```
df.n <- function(n) {
  df(zc.curve(n*delta.t))^(n*delta.t)
}
```

1. Calcul du prix d'une obligation zéro-coupon de maturité $T = n \times \Delta t$ dans l'arbre. En T, il y a $(n+1)$ états dans l'arbre.

```
delta.t <- 1/12
n <- 50
Principal <- rep(100, n+1)
PV.BDT <- CF.discount(r.hat, alpha, delta.t, n.pv=1, pv=Principal)
PV.ZC <- 100 * df.n(n)
```

On obtient bien des prix cohérents. Le prix selon la courbe ZC est: 61.07, et le prix selon l'arbre BDT: 61.07.

2. Prix d'une obligation de maturité 3 ans et de coupon 5.

```
cf <- list(dt= c(12,24,36), flow=c(5,5,105))
```

Calcul avec la courbe zéro-coupon et actualisation dans l'arbre:

```
disc.factor <- unlist(lapply(cf$dt, df.n))
PV.ZC <- sum(cf$flow*disc.factor)
PV.BDT <- CF.discount(r.hat, alpha, delta.t, 1, cf=cf)
```

On obtient bien des prix cohérents. Le prix selon la courbe ZC est: 83.34, et le prix selon l'arbre BDT: 83.34.

2.2 Option sur obligation

Calculer le prix d'une option (call) strike = 79, maturité 1 an, sur une obligation de maturité 5 ans et de coupon 5.

A maturité de l'option, l'obligation a une maturité résiduelle 4 ans. On calcule la valeur à terme en $T_1 = 12 \times \Delta t$ de l'obligation.

```
cf <- list(dt= c(12,24,36, 48, 60), flow=c(5,5,5,5,105))
# tranche de temps 13 = 12 mois.
FV <- CF.discount(r.hat, alpha, delta.t, 13, cf=cf)
Strike <- 79
pv <- pmax(FV-Strike, 0)
pv <- CF.discount(r.hat, alpha, delta.t, 1, pv=pv)
```

Prix de l'option sur obligation: 2.1.

A titre informatif, on peut calculer le prix de cette même option à l'aide de la formule de Black (1976).

Calculons en premier lieu le rendement actuariel de l'obligation:

```
disc.factor <- unlist(lapply(cf$dt, df.n))
PV.ZC <- sum(cf$flow*disc.factor)

Y.to.P.error <- function(y) {
  df <- 1/(1+y)^(cf$dt/12)
  sum(df*cf$flow) - PV.ZC
}

ytm <- uniroot(Y.to.P.error, c(.12, .13))$root
```

Soit un rendement actuariel de 12.85 %.

La volatilité du prix de l'obligation est reliée à la volatilité du rendement par la formule:

$$\sigma_y = \frac{\sigma_P}{y \frac{D}{1+y}}$$

avec D: duration de Macaulay. On suppose que la volatilité du rendement de l'obligation est la volatilité du taux zero-coupon de même duration.

```
disc.factor <- unlist(lapply(cf$dt, df.n))
Dur <- sum(seq(1,5)*cf$flow*disc.factor) / sum(cf$flow*disc.factor)
sigma.Y <- vol.curve(Dur)
sigma.P <- sigma.Y * ytm * Dur / (1+ytm)
```

soit une volatilité de prix de 8.37%. Le prix à terme est ensuite calculé à partir de la courbe de taux zero-coupon:

```
cf.terme <- list(dt= c(24,36, 48, 60), flow=c(5,5,5,105))
disc.factor <- unlist(lapply(cf.terme$dt, df.n)) / df.n(12)
P.Fwd <- sum(cf.terme$flow*disc.factor)
```

La formule de Black donne le prix de l'option:

```
Ttm <- 1
d1 <- (log(P.Fwd/Strike) + Ttm*(sigma.Y^2/2))/(sigma.Y*sqrt(Ttm))
d2 <- d1 - sigma.Y*sqrt(Ttm)
r <- zc.curve(Ttm)
c <- exp(-r*Ttm) *(P.Fwd*pnorm(d1) - Strike*pnorm(d2))
```

The Black estimate of the call value is 2.83, ce qui représente une différence assez significative par rapport au résultat du modèle BDT. Le formule de Black utilise une hypothèse très approximative pour la dynamique du prix d'une obligation, en particulier ne prend pas en compte la convergence du prix vers le pair.