Introduction to Statistical Finance

Coursework 1 - part A

CID 01945214, Robin Mathelier Imperial College London, Spring 2020

We consider the trinomial market model with parameter values:

$$\mu = 0.005, \quad h = 0.06, \quad r = 0.01, \quad S_0 = 100, \quad T = 8, \quad p_+ = 0.8, \quad p_- = 0.1$$

We recall the basic definitions of the trinomial market model. The market where we are trading offers :

- a risk-free asset, whose price is non-random and given by the process $B = (B_t)_{t=0}^T$. We suppose its initial price is normalised to one so that we have $B_t = R^t$ with $R := e^r$, $t = 0, 1, ..., T$.
- a risky asset, whose prices are given by an adapted stochastic process $S = (S_t)_{t=0}^T$ on the filtered probability space $(\Omega, \mathcal{F}, \mathbb{P}, \mathbb{F}^S)$ with $\Omega := \{1, 0, 1\}^T$, $\mathcal{F} = \mathcal{P}(\Omega)$ and :

$$\mathbb{P}[\{\omega\}] = \prod_{t=1}^{T} \left[p_+ \mathbb{1}_{(\omega_t=1)} + p_- \mathbb{1}_{(\omega_t=-1)} + (1 - (p_+ + p_-)\mathbb{1}_{(\omega_t=0)}\right] \qquad \omega \in \Omega \tag{0.1}$$

We define random variable $\xi_t(\omega) := \omega_t$, $\omega \in \Omega$, for any $t = 1, ..., T$. The price process $S = (S - t)_{t=0}^T$ of the risky asset is defined for $t = 1, ..., T$ recursively :

$$S_t = S_{t-1} \begin{cases} U := e^{\mu+h} & \xi_t = 1 \\ M := e^{\mu}, & \xi_t = 0 \\ D := e^{\mu-h} & \xi_t = 1 \end{cases} \tag{0.2}$$

Moreover, for any price process $(Z_t)_{t=0}^T$, we define the discounted price process $\tilde{Z}_t = (\frac{Z_t}{B_t})_{t=0}^T$

1 Question 1

Recall that an an equivalent martingale measure (EMM) is defined by :

(i) $\mathbb{Q}(A) > 0$ if and only if $\mathbb{P}(A) > 0$ for any $A \in \mathcal{F}$,

(ii) $\tilde{S}$ is a martingale with respect to $\mathbb{F}^S$ under $\mathbb{Q}$.

We seek an EMM $\mathbb{Q}$ of a similar form than $\mathbb{P}$:

$$\mathbb{Q}[\{\omega\}] = \prod_{t=1}^{T} \left[q_+ \mathbb{1}_{(\omega_t=1)} + q_- \mathbb{1}_{(\omega_t=-1)} + (1 - (q_+ + q_-)\mathbb{1}_{(\omega_t=0)}\right] \qquad \omega \in \Omega \tag{1.1}$$

We want to specify suitable $q_+ := \mathbb{Q}[\xi_1 = 1]$ and $q_- := \mathbb{Q}[\xi_1 = 1]$ to satisfy the conditions of an EMM.

- To satisfy condition (i), we require $q_+ > 0$, $q_- > 0$ and $q_+ + q_- < 1$.
- To satisfy condition (ii), we know from lecture notes (P.3.7) that we require :

$$\frac{U}{R}q_+ + \frac{M}{R}(1 - (q_+ + q_-)) + \frac{D}{R}q_- = 1 \tag{1.2}$$

We can write the previous equation :

$$(U - M)q_+ + (D - M)q_- = R - M \tag{1.3}$$

We naturally look for solutions of the form :

$$\begin{cases} q_+ = \alpha \frac{R-M}{U-M} \\ q_- = (1 - \alpha)\frac{R-M}{D-M} \end{cases} \tag{1.4}$$

- To ensure that $q_+ > 0$ and $q_- > 0$, we notice that $U > R > M > D$ so we must have $\alpha > 1$.
- To ensure that $q_+ < 1$ and $q_- < 1$, we require $\alpha < \frac{U-M}{R-M} \simeq 12.34$ and $\alpha - 1 < \frac{D-M}{M-R} \simeq 11.62$.
- To ensure that $q_+ + q_- < 1$, a straightforward calculation shows that we require

$$\alpha < \frac{D-R}{D-M} \times \left(\frac{R-M}{U-M} + \frac{M-R}{D-M} \right)^{-1} \simeq 6.50$$

To get an EMM $\mathbb{Q}$, we just have to select a value $\alpha \in (1, 6.50)$ and write solutions according to (1.4). For example, with $\alpha = 2$, we obtain the following solution :

$$\begin{cases} q_+^* = 2\frac{R-M}{U-M} \simeq 0.162 \\ q_-^* = \frac{M-R}{D-M} \simeq 0.086 \end{cases} \tag{1.5}$$

The probability measure $\mathbb{Q}$ defined like in (1.1) with the values of (1.5) is an EMM for our trinomial model.

2 Question 2

In question 1, we showed that we can find an EMM $\mathbb{Q}$ by selecting a value $\alpha \in (1, 6.50)$ and write solutions according to (1.4). Therefore, there exists an infinity of EMMs in this trinomial market model. The admissible values to obtain an EMM are plotted on figure 1.

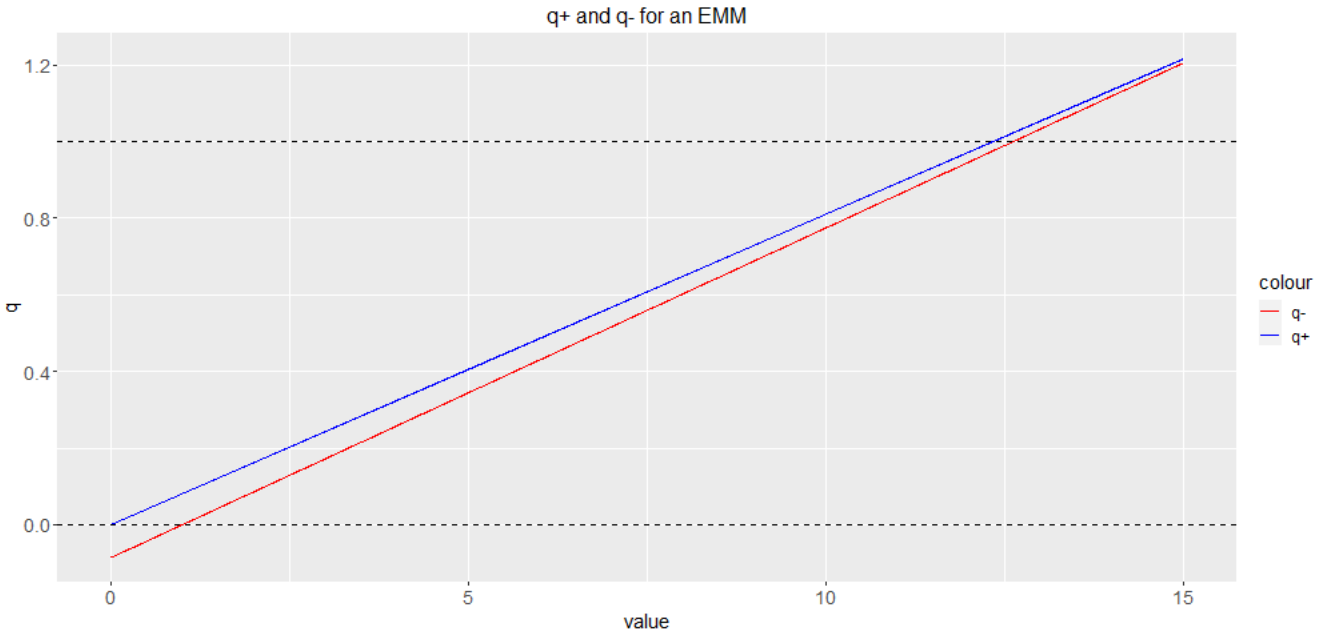

Figure 1: Admissible values for q_+ and q_- to obtain an EMM (both q_+ and q_- must be between the two horizontal dotted lines)

3 Question 3

Let f be the function defined in figure 2. We aim to obtain an analytical expression for $f(S_T)$, $S_T > 0$. To do so, we notice that f is a continuous piecewise linear function. The slope of f changes in $S_T = 100$, $S_T = 150$ and $S_T = 200$. For each interval where the slope of f is constant, we then seek an expression of f of the form $f(S_T) = aS_T + b$. For example, for $S_T \in (0, 100]$, we read on the figure 2 that $f(0) = 250$ and $f(100) = 50$, so we determine :

$$a = \frac{50 - 250}{100 - 0} = \frac{-200}{100} = -2 \tag{3.1}$$

Then, $b = f(0) = 250$ such that $f(S_T) = -2S_T + 250$. We use the same method to obtain the analytical expression of f for all $S_T > 0$. Results are reported in table 1.

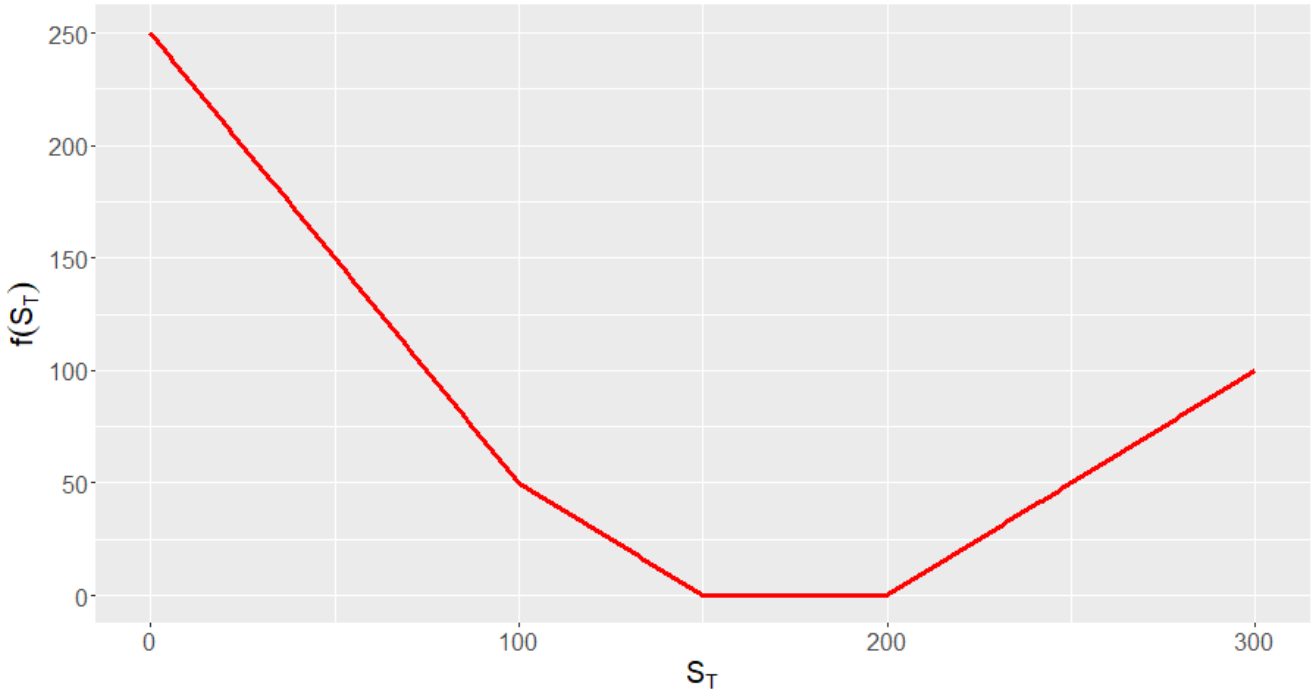

Figure 2: Payoff $f(S_T)$

S_T	$f(S_T)$
(0,100]	-2 $S_T + 250$
(100, 150]	-$S_T + 150$
(150, 200]	0
> 200	$S_T - 200$

Table 1: analytical expression of $f(S_T)$, $S_T > 0$

4 Question 4

We want to find an arbitrage-free price process $(\Pi_t)_{t=0}^T$ for the payoff $f(S_T)$ in the trinomial market model.

We choose to use risk-neutral pricing. Suppose that S is arbitrage-free with EMM $\mathbb{Q}$ defined by (1.1) with values given by (1.5). We can see that the payoff satisfies $\mathbb{E}_{\mathbb{Q}}[|f(S_T)|] < +\infty$. Indeed, we have $64.4 \simeq S_0 D^T < S_T < S_0 U^T \simeq 168.2$ and then from figure 2 we get $|f(S_T)| < 250$.

Therefore, theorem P.45 (P.5.1) tells us that an arbitrage-free price process of the derivative with payoff $f(S_T)$ is :

$$\Pi_t = B_t \mathbb{E}_{\mathbb{Q}} \left[\frac{f(S_T)}{B_T} | \mathcal{F}_t^S \right] \qquad t = 0, .., T \tag{4.1}$$

We showed in the lecture notes (P.5.2) that the arbitrage-free price $(\Pi_t)_{t=0}^T$ satisfies, for $t = 0, 1, ..., T-1$:

$$\tilde{\Pi}_T(S_T) = \frac{f(S_T)}{B_T} \tag{4.2}$$

$$\tilde{\Pi}_t(S_t) = \tilde{\Pi}_{t+1}(S_t U) q_+ + \tilde{\Pi}_{t+1}(S_t M)(1 - q_+ - q_-) + \tilde{\Pi}_{t+1}(S_t D) q_- \tag{4.3}$$

First, thanks to the equation (0.2), we create a tree of the spot prices (figure 3). From the tree of spot prices, we can implement from equation (4.2) and (4.3) an algorithm that computes recursively $\tilde{\Pi}_t$ and then $\Pi_t = B_t \tilde{\Pi}_t$ from $t = T$ to $t = 0$. We present the tree of the arbitrage-free price process $(\Pi_t)_{t=0}^T$ for a derivative with payoff $f(S_T)$ in figure (4).

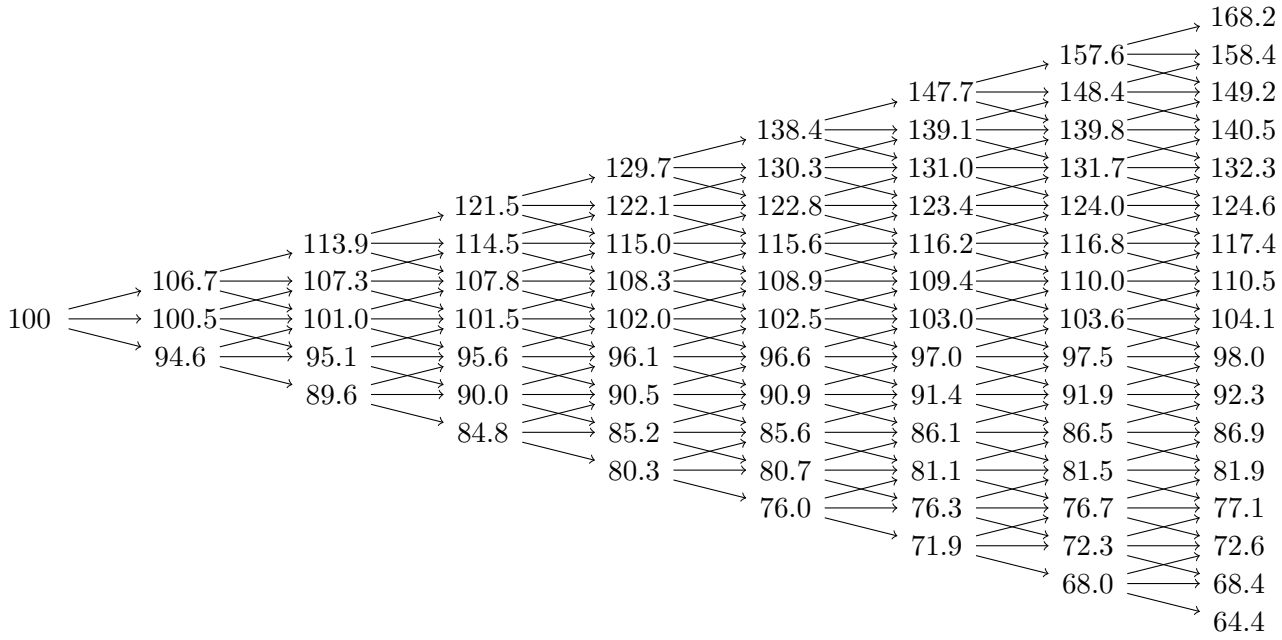

Figure 3: Tree of possible price paths $(S_t)_{t=0}^T$ in the trinomial market model

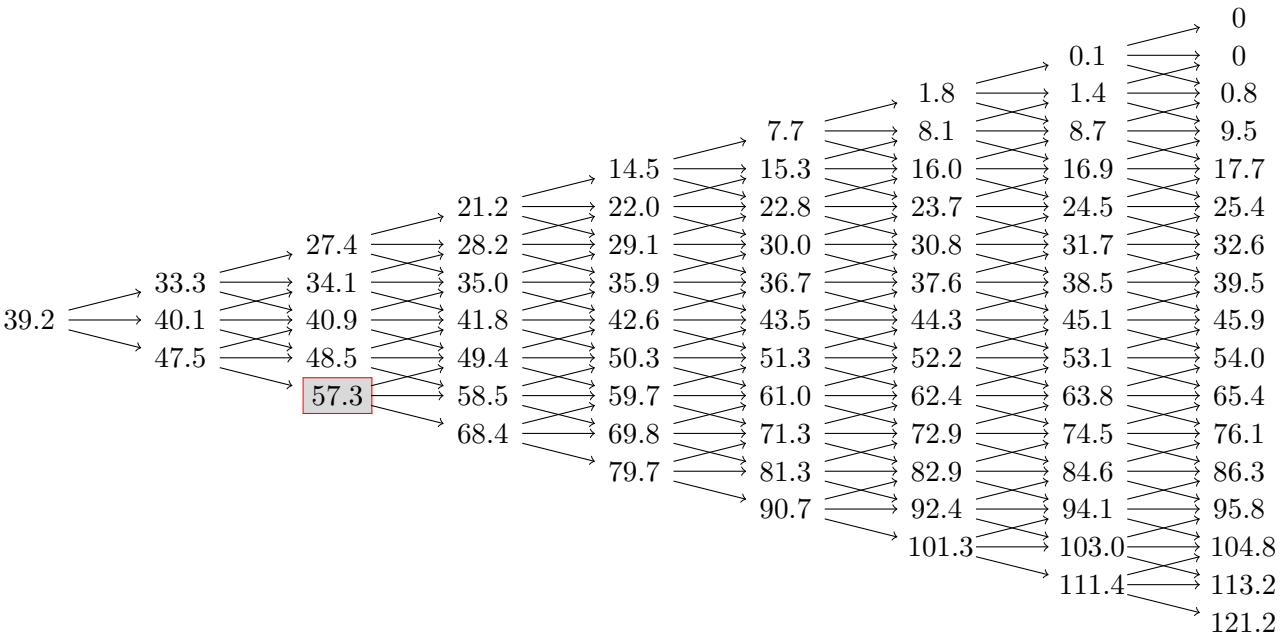

Figure 4: Tree of the arbitrage-free prices $(\Pi_t)_{t=0}^T$ of a derivative with payoff $f(S_T)$

5 Question 5

It is noticed that we can write $f(S_T)$ as :

$$f(S_T) = (S_T - 200)^+ + (150 - S_T)^+ + (100 - S_T)^+ \tag{5.1}$$

Then the payoff $f(S_T)$ can be written as a combination of :

- a European call option with maturity T and strike price $K = 200$
- a European put option with maturity T and strike price $K = 150$
- a European put option with maturity T and strike price $K = 100$

We can plot the payoffs of these options to verify that their sum corresponds with $f(S_T)$ (figure 5)

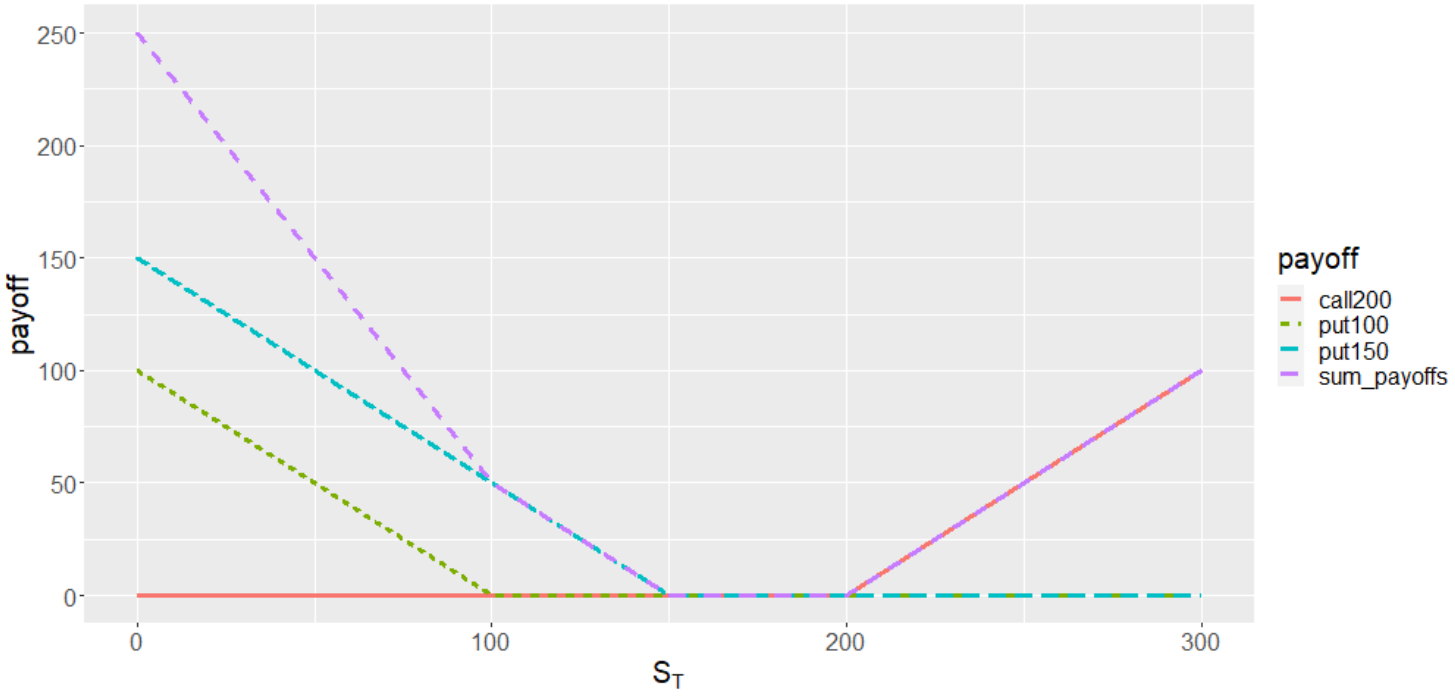

Figure 5: The payoffs of a European call option with K = 200 (red), a European put option with K = 150 (blue), a European put option with K = 100 (green), and the sum of them (purple)

Then, with the same algorithm than in question 4, we price those instruments separately. We present the tree of each price process (figure 6, 7 and 8). It is important to precise that we rounded the results with 1 digit for more readability. This round explains why we observe price moves from 0 to 0.1 for example, which could be seen as an arbitrage opportunity but is actually not.

It can be verified that the value in each node of the tree in the figure 4 is obtained by summing the values in the same node for the trees of options prices. For example, we circled the price in each tree after 2 time periods where we observed UP movements. It is well observed that $57.3 = 0 + 51.7 + 5.6$.

To conclude, our methods in question 4 and question 5 give exactly the same arbitrage free price process.

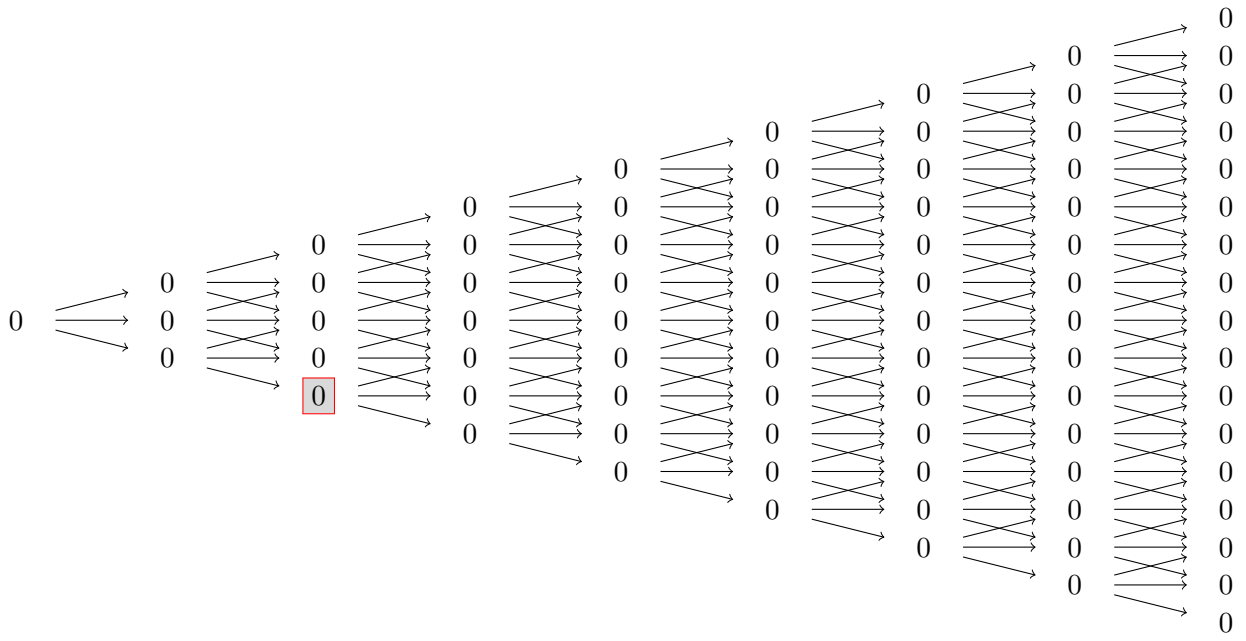

Figure 6: Tree of the arbitrage-free prices of a European call option with maturity $T = 8$ and strike $K = 200$

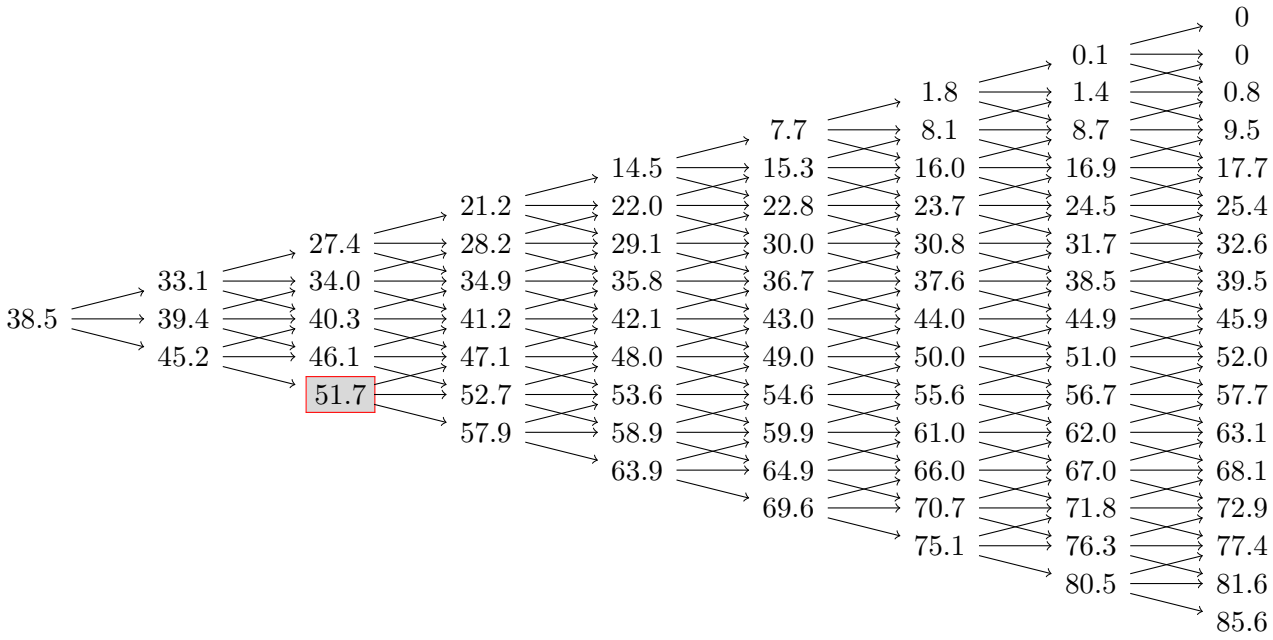

Figure 7: Tree of the arbitrage-free prices of a European put option with maturity $T = 8$ and strike $K = 1500$

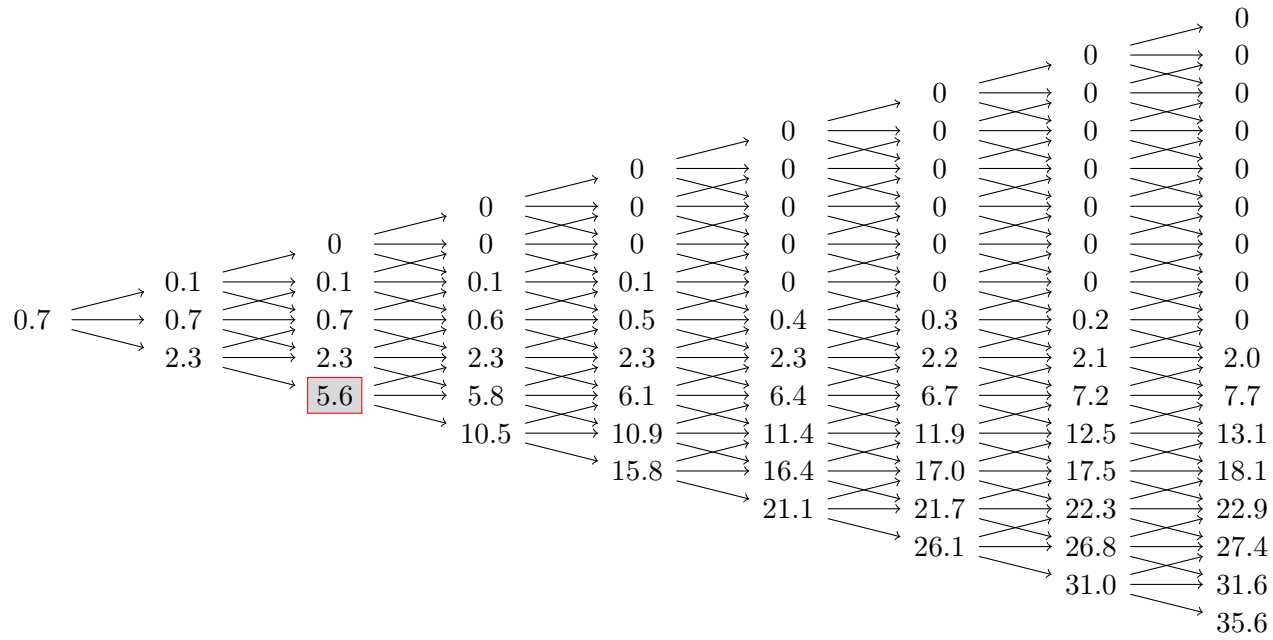

Figure 8: Tree of the arbitrage-free prices of a European put option with maturity $T = 8$ and strike $K = 100$

A R code

```
### Introduction to statistical finance - Coursework 1 - Part A ###

setwd("C:/Users/robin/Dropbox/Applications/Overleaf/Coursework_finance_part_A")

library("ggplot2")
library("tidyverse")

my_theme <-
  theme(plot.title = element_text(hjust = 0.5, size = 16),
        axis.title = element_text(size = 15),
        axis.text = element_text(size = 14),
        legend.title = element_text(size = 15),
        legend.text = element_text(size = 12),
        strip.text = element_text(size = 14))

mu = 0.005
h = 0.06
r = 0.01
S0 = 100
T_final = 8
p_plus = 0.8
p_minus = 0.1

R = exp(r)
U = exp(mu+h)
M = exp(mu)
D = exp(mu-h)

(U-M)/(R-M)
(D-M)/(M-R)
(D-R)/(D-M) * 1/( (R-M)/(U-M) + (M-R)/(D-M))

q_plus = 2*(R-M)/(U-M)
q_minus = (M-R)/(D-M)
q_plus
q_plus; q_minus; 1 - q_plus - q_minus

gen_q_plus = function(alpha) alpha*(R-M)/(U-M)
gen_q_minus = function(alpha) (1-alpha)*(R-M)/(D-M)
abs = seq(0,15,by = 0.01)
ord_plus = gen_q_plus(abs)
ord_minus = gen_q_minus(abs)

png(filename = "q_+_q_-.png", width = 1000)

df_q = data.frame(abs, ord_plus, ord_minus)
q = ggplot(df_q)
q = q + geom_line(aes(x = abs, y = ord_plus, color = "q+"))
q = q + geom_line(aes(x = abs, y = ord_minus, color = "q-"))
q = q + my_theme
q = q + scale_color_manual(values = c("q+" = 'blue',
                                      "q-" = 'red'))
q = q + geom_hline(yintercept = 0, lty = 2)
q = q + geom_hline(yintercept = 1, lty = 2)
q = q + labs(title = "q+ and q- for an EMM", x = "value", y = "q")
q

dev.off()

######### creation payoff function ##########
```

```
B = function(t) return(exp(r*t))

f = function(s){
  if (s <= 100) return(-2*s + 250)
  if (s > 100 & s <= 150) return(-s + 150)
  if (s > 150 & s <= 200) return(0)
  if (s > 200 & s <= 300) return(s - 200)
  else print("error : spot price is not atteignable")
}

f = Vectorize(f)

abs = seq(0,300,by=1)
ord = f(abs)
df_f = data.frame(abs,ord)

png(filename="f_S_T.png", width = 900)
g = ggplot(df_f, aes(x = abs, y = ord))
g = g + geom_line(color = 'red', lwd = 1.2)
g = g + theme(legend.position="right", text = element_text(size=20))
g = g + xlab(expression(S[T])) + ylab(expression(f(S[T])))
g
dev.off()

######### create tree stock prices ############


tree_St = list()
tree_St[[1]] = S0

for (t in 1:T_final){
  prices_t = c()
  grid = 0:t
  set = expand.grid(grid, grid, grid)
  set = set[sapply(1:dim(set)[1], function(i) sum(set[i,]) == t), ]
  prices = sapply(1:dim(set)[1], function(i) S0*U^(set[i,1])*M^(set[i,2])*D^(set[i
      ,3]))
  prices = unique(sort(prices))
  tree_St[[t+1]] = unique(round(prices,digits=5))
}

sapply(1:length(tree_St), function(i) round(tree_St[[i]],digits = 1))


########## create tree option prices ##########


PI_disc_T = function(s) f(s)/B(T_final)

tree_prices_disc = list()
tree_prices_disc[[T_final+1]] = PI_disc_T(tree_St[[T_final+1]])

for (t in rev(1:T_final)){
  PI_next = tree_prices_disc[[t+1]]
  PI_cur = c()
  for (i in 1:(length(PI_next)-2)){
    price = PI_next[i]*q_minus +
      PI_next[i+1]*(1 - q_plus - q_minus) +
      PI_next[i+2]*q_plus
    PI_cur = c(PI_cur,price)
  }
  tree_prices_disc[[t]] = PI_cur
}
```

```
tree_prices = list()
for (t in 1:length(tree_prices_disc)){
  tree_prices[[t]] = tree_prices_disc[[t]]*B(t-1)
}

tree_prices[[9]]       # need to be equal
f(tree_St[[9]])

sapply(1:length(tree_St), function(i) round(tree_prices[[i]], digits = 1))

############   tree price European call option (K = 200)  #########

B = function(t) return(exp(r*t))

f_call_K_200 = function(s) max(0,s-200)
f_call_K_200 = Vectorize(f_call_K_200)

abs = seq(0,300,by=0.01)
ord = f_call_K_200(abs)
plot(abs,ord, col = 'red')

PI_disc_T_K_200 = function(s) f_call_K_200(s)/B(T_final+1)

tree_prices_disc_call_K_200 = list()
tree_prices_disc_call_K_200[[T_final+1]] = PI_disc_T_K_200(tree_St[[T_final+1]])

for (t in rev(1:T_final)){
  PI_next = tree_prices_disc_call_K_200[[t+1]]
  PI_cur = c()
  for (i in 1:(length(PI_next)-2)){
    price = PI_next[i]*q_minus +
      PI_next[i+1]*(1 - q_plus - q_minus) +
      PI_next[i+2]*q_plus
    PI_cur = c(PI_cur,price)
  }
  tree_prices_disc_call_K_200[[t]] = PI_cur
}

tree_prices_call_K_200 = list()
for (t in 1:length(tree_prices_disc_call_K_200)){
  tree_prices_call_K_200[[t]] = tree_prices_disc_call_K_200[[t]]*B(t)
}

tree_prices_call_K_200[[9]]       # need to be equal
f_call_K_200(tree_St[[9]])

sapply(1:length(tree_St), function(i) round(tree_prices_call_K_200[[i]], digits = 1))

############   tree price European put option (K = 150)  #########


B = function(t) return(exp(r*t))

f_put_K_150 = function(s) max(0,150-s)
f_put_K_150 = Vectorize(f_put_K_150)

abs = seq(0,300,by=0.01)
ord = f_put_K_150(abs)
plot(abs,ord, col = 'red')

PI_disc_put_K_150 = function(s) f_put_K_150(s)/B(T_final+1)

tree_prices_disc_put_K_150 = list()
```

```
tree_prices_disc_put_K_150[[T_final+1]] = PI_disc_put_K_150(tree_St[[T_final+1]])

for (t in rev(1:T_final)){
  PI_next = tree_prices_disc_put_K_150[[t+1]]
  PI_cur = c()
  for (i in 1:(length(PI_next)-2)){
    price = PI_next[i]*q_minus +
      PI_next[i+1]*(1 - q_plus - q_minus) +
      PI_next[i+2]*q_plus
    PI_cur = c(PI_cur, price)
  }
  tree_prices_disc_put_K_150[[t]] = PI_cur
}

tree_prices_put_K_150 = list()
for (t in 1:length(tree_prices_disc_put_K_150)){
  tree_prices_put_K_150[[t]] = tree_prices_disc_put_K_150[[t]]*B(t)
}

tree_prices_put_K_150[[9]]       # need to be equal
f_put_K_150(tree_St[[9]])

sapply(1:length(tree_St), function(i) round(tree_prices_put_K_150[[i]], digits = 1))

############   tree price European put option (K = 100)  #########

B = function(t) return(exp(r*t))

f_put_K_100 = function(s) max(0,100-s)
f_put_K_100 = Vectorize(f_put_K_100)

abs = seq(0,300,by=0.01)
ord = f_put_K_100(abs)
plot(abs,ord, col = 'red')

PI_disc_put_K_100 = function(s) f_put_K_100(s)/B(T_final+1)

tree_prices_disc_put_K_100 = list()
tree_prices_disc_put_K_100[[T_final+1]] = PI_disc_put_K_100(tree_St[[T_final+1]])

for (t in rev(1:T_final)){
  PI_next = tree_prices_disc_put_K_100[[t+1]]
  PI_cur = c()
  for (i in 1:(length(PI_next)-2)){
    price = PI_next[i]*q_minus +
      PI_next[i+1]*(1 - q_plus - q_minus) +
      PI_next[i+2]*q_plus
    PI_cur = c(PI_cur, price)
  }
  tree_prices_disc_put_K_100[[t]] = PI_cur
}

tree_prices_put_K_100 = list()
for (t in 1:length(tree_prices_disc_put_K_100)){
  tree_prices_put_K_100[[t]] = tree_prices_disc_put_K_100[[t]]*B(t)
}

tree_prices_put_K_100[[9]]       # need to be equal
f_put_K_100(tree_St[[9]])

sapply(1:length(tree_St), function(i) round(tree_prices_put_K_100[[i]], digits = 1))

############ total value options ###############
```

```
sum_tree_prices_options = list()
for (t in 1:length(tree_prices_put_K_100)){
  sum_tree_prices_options[[t]] = tree_prices_call_K_200[[t]] + tree_prices_put_K_
      150[[t]] + tree_prices_put_K_100[[t]]
}

sum_tree_prices_options
tree_prices

data_gg_options = data.frame(S = abs)
data_gg_options$call200 = f_call_K_200(abs)
data_gg_options$put150 = f_put_K_150(abs)
data_gg_options$put100 = f_put_K_100(abs)
data_gg_options$sum_payoffs = f(abs)
data_gg_options = data_gg_options %>%  gather(key = "payoff", value = "value", -S)


png(filename = "options.png", width = 1000)
g = ggplot(data = data_gg_options, aes(x = S, y = value))
g = g + geom_line(aes(color = payoff, lty = payoff), lwd = 1.2)
g = g + theme(legend.position="right", text = element_text(size=20))
g = g + xlab(expression(S[T])) + ylab("payoff")
g
dev.off()
```