Coursework - Introduction to Statistical Finance

CID: 02091191

February 16, 2022

Abstract

In this document we are presenting the work conducted during the first Introduction to Statistical Finance Coursework of the year.

A Derivatives Pricing

(i) Find an equivalent martingale measure

Consider the trinomial market model: $\Omega = \{-1, 0, 1\}^T$, $\mathcal{F}$ as the power set $\mathcal{P}(\Omega)$ and

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

where $T = 8$, $p_+ = 0.1 > 0$ and $p_- = 0.3 > 0$ such that $p_+ + p_- < 1$. We may moreover define the random variable $\xi_t(\omega) = \omega_t$ for $\omega \in \Omega$ and for any $t = 1, \ldots, T$ such that $\xi_1, \ldots, \xi_T$ are *iid* and

$$\mathbb{P}\left[\xi_t = y\right] = \begin{cases} p_+ & \text{if } y = 1 \\ 1 - (p_+ + p_-) & \text{if } y = 0 \\ p_- & \text{if } y = -1 \end{cases}$$

The price process $S = (S_t)_{t=0}^T$ of the risky asset is constructed by letting $S_0 = 10 > 0$ to be non-random and defining for $t = 1, \ldots, T$

$$S_t = S_{t-1} \cdot \begin{cases} e^{\mu+h} & \text{if } \xi_t = 1 \\ e^{\mu} & \text{if } \xi_t = 0 \\ e^{\mu-h} & \text{if } \xi_t = -1 \end{cases}$$

where $\mu = 0.02$ and $h = 0.1$. For convenience, we introduce the variables

$$U = e^{\mu+h}, \quad M = e^{\mu}, \quad D = e^{\mu-h}$$

Although the price process of the risky asset S is stochastic, we consider the bond price process B to be deterministic such that

$$B_t = B_{t-1} \cdot e^r$$

where $r = 0.02$ and we define $R = e^r$.

Note that since $p_+ + p_- = 0.4 < 1$, all three transitions are possible and we obtain a proper trinomial model. The tree of possible price paths for the underlying S in this defined trinomial market model can therefore be represented graphically on Figure A.1.

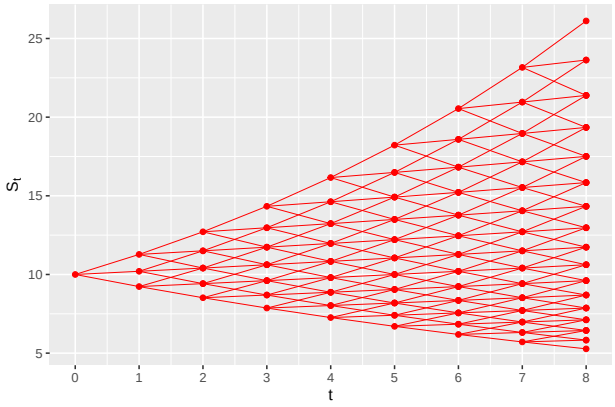

Figure A.1: The tree of possible price paths of S in the trinomial market as defined.

To determine when this model is arbitrage-free, one may use the first fundamental theorem of asset pricing (Theorem P.27 in the notes [1]): we seek an equivalent martingale measure (EMM) $\mathbb{Q}$ of the form

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

And as the probability measure $\mathbb{P}$, the random variables $\xi_1, \ldots, \xi_T$ are *iid* with

$$\mathbb{Q}\left[\xi_t = y\right] = \begin{cases} q_+ & \text{if } y = 1 \\ 1-(q_+ + q_-) & \text{if } y = 0 \\ q_- & \text{if } y = -1 \end{cases}$$

Recall the definition of an EMM (Definition P.24): we require $q_+ > 0$, $q_- > 0$ and $q_+ + q_- < 1$. Now define the discounted price process $\tilde{S}_t = \frac{S_t}{B_t}$ for $t = 0, \ldots, T$ which could be a martingale with respect to $\mathbb{F}^S$ under $\mathbb{Q}$. The adaptedness requirement is satisfied since B is non-random and the integrability condition follows since $0 < \tilde{S}_t \leq S_0(\frac{U}{R})^t$ with $\mathbb{Q}$-probability one. Finally, we note that $\mathcal{F}_0^S = \{\Omega, \emptyset\}$ and $\mathcal{F}_t^S = \sigma\{S_1, \ldots, S_t\} = \sigma\{\xi_1, \ldots, \xi_t\}$ for any $t = 1 \ldots, T$ with ξ_t independent of $\mathcal{F}_{t-1}^S$ for any $t = 1, \ldots, T$. Then,

$$\begin{aligned} \mathbb{E}^{\mathbb{Q}}\left[\widetilde{S}_t \mid \mathcal{F}_{t-1}^S\right] = \mathbb{E}^{\mathbb{Q}}\left[\frac{S_t}{B_t} \mid \mathcal{F}_{t-1}^S\right] &= \frac{S_{t-1}}{B_{t-1}} \mathbb{E}^{\mathbb{Q}}\left[\frac{U\mathbb{1}_{\{\xi_t=1\}} + M\mathbb{1}_{\{\xi_t=0\}} + D\mathbb{1}_{\{\xi_t=-1\}}}{R} \mid \mathcal{F}_{t-1}^S\right] \\ &= \widetilde{S}_{t-1}\left(\frac{U}{R}\mathbb{Q}\left[\xi_1 = 1\right] + \frac{M}{R}\mathbb{Q}\left[\xi_1 = 0\right] + \frac{D}{R}\mathbb{Q}\left[\xi_1 = -1\right]\right) \end{aligned}$$

so that $\mathbb{E}^{\mathbb{Q}}\left[\widetilde{S}_t \mid \mathcal{F}_{t-1}^S\right] = \tilde{S}_{t-1}$ whenever

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

The theory of convexity ensures that q_+ and q_- exist if and only if

$$U > R > D$$

which is the case here (arbitrage-free condition). Therefore, to find a particular EMM $\mathbb{Q}$, one may choose arbitrarily a value for q_+ and infer q_- such that the equality (1) is verified.

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

We will choose here $q_+ = 0.25$ which gives $q_- = 0.2763$, and verifies $q_+ + q_- < 1$.

(ii) Existence of other EMMs for this model

Equation (1) has an infinite number of solutions q_+ and q_-: the ones chosen in this report are therefore not unique. One could have chosen a different value for q_+ that would have given a different value for q_-. However, note that not all values for q_+ are suitable: for example, $q_+ = 0.5$ would give $q_- = 0.5526$ which do not verify $q_+ + q_- < 1$.

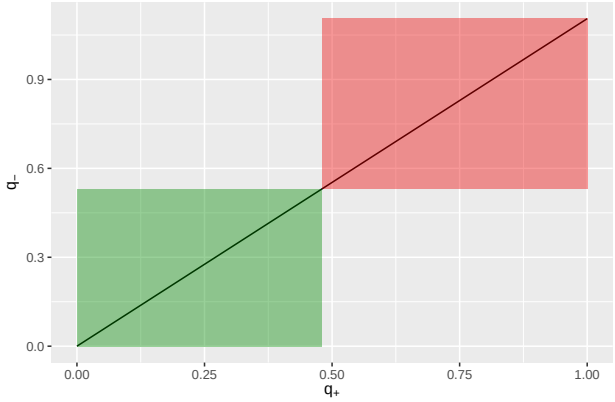

Figure A.2: Values for q_- for given values of q_+ in black line according to Equation (1). Values such that $q_+ + q_- < 1$ are in the green rectangle. Values such that $q_+ + q_- > 1$ are in the red rectangle.

Figure A.2 shows solutions to Equation (1) as the black line. The suitable solutions which verify $q_+ + q_- < 1$ are within the green rectangle, whereas the non suitable solutions are within the red rectangle. Note that the values we chose for $q_+ = 0.25$ and $q_- = 0.2763$ are located in the green area. This figure hence shows that Equation (1) has an infinite number of solutions. The edge point located between the red and the green rectangles corresponds to values of q_+ and q_- such that $q_+ + q_- = 1$, which is the binomial market model.

(iii) Analytical expression for a given payoff

Assume we are now interested in pricing a payoff $f(S_T)$ given graphically for $x > 0$ by

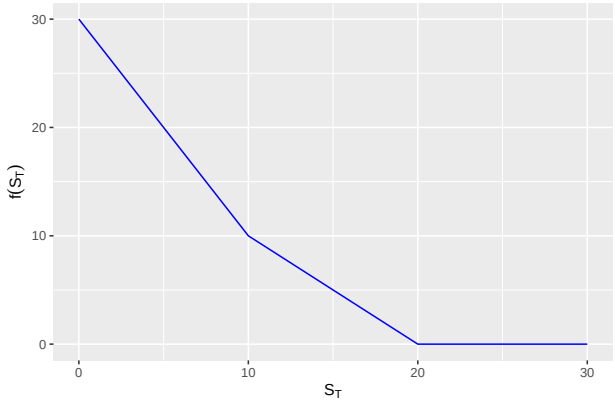

Figure A.3: Payoff $f(S_T)$

Figure A.3 allows to find an analytical expression for f such that

$$f(S_T) = \begin{cases} 30 - 2S_T & \text{if} \quad S_T \in (0, 10] \\ 20 - S_T & \text{if} \quad S_T \in (10, 20] \\ 0 & \text{if} \quad S_T \in (20, \infty) \end{cases} = (30 - 2S_T)\mathbb{1}_{S_T \in (0,10]} + (20 - S_T)\mathbb{1}_{S_T \in (10,20]}$$

(iv) Arbitrage-free price process for the payoff

Using the EMM $\mathbb{Q}$ derived previously with $q_+ = 0.25$ and $q_- = 0.2763$, we now want to find an arbitrage-free price process $(\Pi_t)_{t=0}^T$ for the payoff $f(S_T)$ in the trinomial market model. To do so, one may use the risk-neutral pricing theorem (Theorem P.45 in the notes): S being arbitrage-free with EMM $\mathbb{Q}$, if the payoff X satisfies $\mathbb{E}^{\mathbb{Q}}[|X|] < \infty$, then $\Pi_t = B_t \mathbb{E}^{\mathbb{Q}}\left[\frac{X}{B_T} \mid \mathcal{F}_t^S\right]$, $t = 0, \dots, T$ defines an arbitrage-free price process of X.

In this case, there is no arbitrage since $U > R > D$ and we found an appropriate EMM $\mathbb{Q}$ as defined previously. Since $S = (S_t)_{t=0}^T$ is a Markov process, the discounted price process satisfies:

$$\tilde{\Pi}_t = \mathbb{E}^{\mathbb{Q}}\left[\frac{X}{B_T} \mid \mathcal{F}_t^S\right] = \mathbb{E}^{\mathbb{Q}}\left[\frac{X}{B_T} \mid S_t\right] = \tilde{\Pi}_t(S_t) \quad \text{and} \quad \tilde{\Pi}_T = \frac{f(S_T)}{B_T}$$

Then, proceeding recursively, we may express $\tilde{\Pi}_t(S_t)$ in terms of $\tilde{\Pi}_{t+1}(S_{t+1})$

$$\tilde{\Pi}_t(S_t) = \mathbb{E}^{\mathbb{Q}}\left[\frac{X}{B_T} \mid S_t\right] = \mathbb{E}^{\mathbb{Q}}\left[\mathbb{E}^{\mathbb{Q}}\left[\frac{X}{B_T} \mid \mathcal{F}_{t+1}^S\right] \mid S_t\right] = \mathbb{E}^{\mathbb{Q}}\left[\tilde{\Pi}_{t+1}(S_{t+1}) \mid S_t\right]$$

using the tower property and the definition of $\tilde{\Pi}_{t+1}$. Hence,

$$\begin{aligned}
\tilde{\Pi}_t(S_t) &= \mathbb{E}^{\mathbb{Q}}\left[\tilde{\Pi}_{t+1}(S_{t+1}) \mid S_t\right] \\
&= \mathbb{E}^{\mathbb{Q}}\left[\tilde{\Pi}_{t+1}(S_tU)\mathbb{1}_{\xi_{t+1}=1} + \tilde{\Pi}_{t+1}(S_tM)\mathbb{1}_{\xi_{t+1}=0} + \tilde{\Pi}_{t+1}(S_tD)\mathbb{1}_{\xi_{t+1}=-1} \mid S_t\right] \\
&= \tilde{\Pi}_{t+1}(S_tU)\mathbb{Q}[\xi_{t+1}=1] + \tilde{\Pi}_{t+1}(S_tM)\mathbb{Q}[\xi_{t+1}=0] + \tilde{\Pi}_{t+1}(S_tD)\mathbb{Q}[\xi_{t+1}=-1] \\
\tilde{\Pi}_t(S_t) &= \tilde{\Pi}_{t+1}(S_tU)q_+ + \tilde{\Pi}_{t+1}(S_tM)(1-(q_+ + q_-)) + \tilde{\Pi}_{t+1}(S_tD)q_-
\end{aligned}$$

and finally get the price process from the discounted price process using $\Pi_t = B_t\tilde{\Pi}_t = B_t\tilde{\Pi}_t(S_t)$.

Therefore, this recursive algorithm can easily be implemented numerically by first working out the values of S in all nodes of the trinomial tree and then populating the analogous tree for the values of $\tilde{\Pi}$ and then descending the tree.

$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$
								0.00
							0.00	0.00
						0.35	0.18	0.00
					1.29	1.16	0.98	0.65
				2.55	2.52	2.48	2.44	2.49
			3.85	3.89	3.94	4.00	4.08	4.16
		5.07	5.15	5.24	5.34	5.44	5.55	5.67
	6.20	6.30	6.40	6.51	6.63	6.76	6.89	7.03
7.29	7.40	7.51	7.62	7.73	7.85	7.97	8.10	8.26
	8.62	8.74	8.87	8.99	9.10	9.21	9.30	9.38
		10.03	10.18	10.33	10.47	10.61	10.72	10.78
			11.56	11.76	11.96	12.16	12.36	12.61
				13.19	13.45	13.71	13.98	14.27
					14.85	15.15	15.45	15.76
						16.45	16.78	17.12
							17.98	18.35
								19.45

Table 1: Values of the price process Π in the trinomial tree from the recursive algorithm.

Values shown in Table 1 shows that the price at $t = 0$ of the derivative is 7.29 and as the price of the underlying stock S increases with time, the price of the derivative decreases and vice versa. This is consistent since the payoff we are pricing is a combination of put options, *ie* we are betting on a decline of the underlying stock.

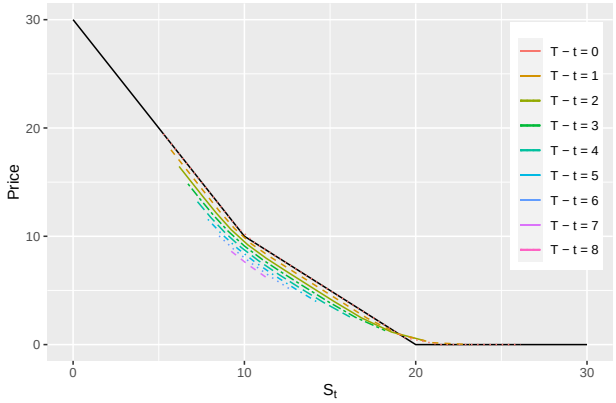

Figure A.4: Prices of the derivative at different times t

Figure A.4 shows the prices of the derivative at different times t as function of the underlying stock price S. The closer the time t to the expiration time T, the closer the price of the derivative is to the payoff.

(v) The payoff seen as a combination of European put and call options

One may easily notice that the payoff $f(S_T)$ can be decomposed as two distinct European put options with strike prices $K = 10$ and $K = 20$:

$$f(S_T) = (20 - S_T)^+ + (10 - S_T)^+$$

where $x^+ = \max\{x, 0\}$. These two put options are represented on Figure A.5.

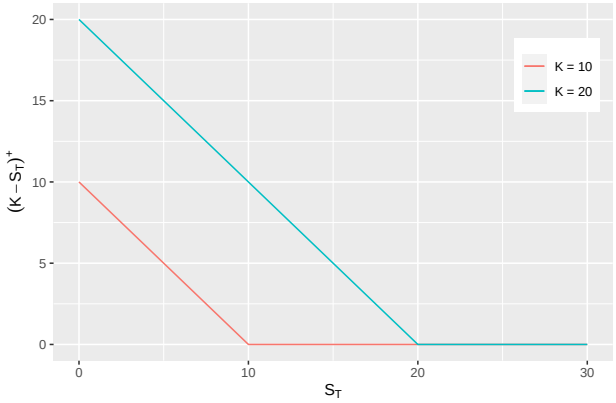

Figure A.5: Decomposition of the payoff $f(S_T)$ in two distinct put options

Hence, one may price these two derivatives distinctly and verify that the total value is equal to the first derivative pricing. To price the put options individually, one shall use the previous risk-neutral pricing recursive definition twice, using first a put option with strike price at $K = 10$ as the payoff, and second a put option with strike price at $K = 20$ as the payoff.

$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$
								0.00
							0.00	0.00
						0.00	0.00	0.00
					0.00	0.00	0.00	0.00
				0.00	0.00	0.00	0.00	0.00
			0.00	0.00	0.00	0.00	0.00	0.00
		0.01	0.01	0.00	0.00	0.00	0.00	0.00
	0.08	0.06	0.04	0.02	0.01	0.00	0.00	0.00
0.24	0.21	0.18	0.14	0.10	0.07	0.03	0.00	0.00
	0.46	0.42	0.38	0.33	0.27	0.19	0.11	0.00
		0.82	0.78	0.74	0.69	0.62	0.54	0.39
			1.33	1.32	1.31	1.29	1.28	1.31
				1.99	2.02	2.05	2.09	2.13
					2.71	2.77	2.83	2.88
						3.42	3.49	3.56
							4.09	4.17
								4.73

Table 2: Values of the price process in the trinomial market for a put option with strike price $K = 10$

$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$
								0.00
							0.00	0.00
						0.35	0.18	0.00
					1.29	1.16	0.98	0.65
				2.55	2.52	2.48	2.44	2.49
			3.85	3.89	3.94	4.00	4.08	4.16
		5.05	5.14	5.24	5.34	5.44	5.55	5.67
	6.12	6.24	6.36	6.49	6.62	6.76	6.89	7.03
7.05	7.19	7.33	7.48	7.63	7.78	7.94	8.10	8.26
	8.16	8.32	8.49	8.66	8.84	9.01	9.20	9.38
		9.22	9.40	9.59	9.79	9.98	10.19	10.39
			10.23	10.44	10.65	10.86	11.08	11.31
				11.20	11.43	11.66	11.89	12.13
					12.13	12.38	12.63	12.88
						13.03	13.29	13.56
							13.89	14.17
								14.73

Table 3: Values of the price process in the trinomial market for a put option with strike price $K = 20$

Values given in Tables 2 and 3 shows the price process separately for each put option with different strike prices. Adding all the prices together gives exactly the same values as obtained in the tree in Table 1 (the only differences are the result of rounding errors, but running the code in `R` shows no difference with more than 5 decimals). To make sure of that, we have added in appendix the table of the sum of the price values of the two instruments. Hence, this shows that pricing these two instruments separately and adding their prices is equivalent to pricing the initial payoff directly, which is consistent.

References

[1] Ciara Pike-Burke, Mikko Pakkanen, Axel Gandy (January 2022) *MATH70079 - Introduction to Statistical Finance lecture notes*, Imperial College London MSc Statistics resources

Table of values of the sum of the two instruments prices

$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$
								0.00
							0.00	0.00
						0.35	0.18	0.00
					1.29	1.16	0.98	0.65
				2.55	2.52	2.48	2.44	2.49
			3.85	3.89	3.94	4.00	4.08	4.16
		5.07	5.15	5.24	5.34	5.44	5.55	5.67
	6.20	6.30	6.40	6.51	6.63	6.76	6.89	7.03
7.29	7.40	7.51	7.62	7.73	7.85	7.97	8.10	8.26
	8.62	8.74	8.87	8.99	9.10	9.21	9.30	9.38
		10.03	10.18	10.33	10.47	10.61	10.72	10.78
			11.56	11.76	11.96	12.16	12.36	12.61
				13.19	13.45	13.71	13.98	14.27
					14.85	15.15	15.45	15.76
						16.45	16.78	17.12
							17.98	18.35
								19.45

R code

```
### Model parameters

mu <- 0.02
h <- 0.1
r <- 0.02
S0 <- 10
T <- 8
p_plus <- 0.1
p_minus <- 0.3
U <- exp(mu+h)
M <- exp(mu)
D <- exp(mu-h)
R <- exp(r)

### Represent the tree of possible price paths

# Initialize the tree
nrow <- 2*T+1
ncol <- T+1
S = matrix(NA, nrow=nrow, ncol=ncol, dimnames=list(
  NULL, paste("t", 0:T, sep="=")))
S[ncol, 1] <- S0      # At t=0, S=S0

library(ggplot2)

tree <- ggplot() +
  geom_point(aes_(x=0, y=S0), colour="red") +
  labs(x="t", y=expression(S[t])) +
```

```
  scale_x_continuous(n.breaks = 9)

# Compute the possible paths
for (j in 1:T) {
  for(i in (ncol-(j-1)):(ncol+(j-1))) {
    S[i-1, j+1] = S[i, j] * U
    tree <- tree +
      geom_segment(aes_(x=j-1, y=S[i, j], xend=j, yend=S[i-1, j+1]),
                   colour="red", size=.3) +
      geom_point(aes_(x=j, y=S[i-1, j+1]), shape=21, colour="red", fill="red")
    S[i, j+1] = S[i, j] * M
    tree <- tree +
      geom_segment(aes_(x=j-1, y=S[i, j], xend=j, yend=S[i, j+1]),
                   colour="red", size=.3) +
      geom_point(aes_(x=j, y=S[i, j+1]), shape=21, colour="red", fill="red")
    S[i+1, j+1] = S[i, j] * D
    tree <- tree + geom_segment(aes_(x=j-1, y=S[i, j], xend=j, yend=S[i+1, j+1]),
                           colour="red", size=.3) +
      geom_point(aes_(x=j, y=S[i+1, j+1]), shape=21, colour="red", fill="red")
  }
}

tree

# Define an appropriate EMM
q_plus <- 0.25
q_minus <- (R-M)/(D-M) + (M-U)/(D-M) * q_plus

qq_plus <- seq(0, 1, by=0.01)
qq_minus <- (R-M)/(D-M) + (M-U)/(D-M) * qq_plus

limit <- qq_plus + qq_minus

q <- ggplot() +
  geom_line(aes_(x=qq_plus, qq_minus), colour="black") +
  labs(x=expression(q["+"]), y=expression(q["-"])) +
  geom_rect(aes(xmin=0, xmax=qq_plus[which(limit>1)[1]],
                ymin=0, ymax=qq_minus[which(limit>1)[1]], colour="green4"),
            color=NA, alpha=0.4, fill="green4") +
  geom_rect(aes(xmin=qq_plus[which(limit>1)[1]], xmax=1,
                ymin=qq_minus[which(limit>1)[1]], ymax=qq_minus[length(qq_minus)],
                colour="red2"),
            color=NA, alpha=0.4, fill="red2")

q

### Payoff analytical expression

f <- function(x){
  if(x>=0 & x<=10) return(30-2*x)
  else if(x>10 & x<=20) return(20-x)
  else return(0)
}
```

```
payoff <- ggplot() +
  geom_line(aes(x=seq(0, 30, by=0.01), y=sapply(seq(0, 30, by=0.01), f)),
            colour="blue") +
  labs(x=expression(S[T]), y=expression(f(S[T])))

payoff

### Arbitrage-free pricing of the payoff

# First the bond price process
B0 <- 1
B <- matrix(data=NA, nrow=1 , ncol=T+1, dimnames=list("B",
                                                      paste("t", 0:T, sep="=")))
B[1] <- B0
for (i in 1:T){
  B[i+1] <- B[i] * R
}

# Discounted price process
Pi_tilde <- matrix(NA, nrow=nrow, ncol=ncol, dimnames=
                     list(NULL, paste("t", 0:T, sep="=")))

# The last column is equal to the discounted payoff
Pi_tilde[,T+1] <- sapply(S[,T+1], f) / B[T+1]

# Recursively, computes Pi_tilde
for (j in (ncol-1):1) {
  for(i in (ncol-j+1):(ncol+j-1)) {
    Pi_tilde[i, j] = (q_plus * Pi_tilde[i-1, j+1] +
                        (1 - (q_plus + q_minus)) * Pi_tilde[i, j+1] +
                        q_minus * Pi_tilde[i+1, j+1])}
}

# Final price process
Pi <- Pi_tilde
for (j in 1:ncol) {
  Pi[, j] = Pi_tilde[, j] * B[j]
}

colors <- rev(c("T-t=0" = "T - t = 0",
            "T-t=1" = "T - t = 1",
            "T-t=2" = "T - t = 2",
            "T-t=3" = "T - t = 3",
            "T-t=4" = "T - t = 4",
            "T-t=5" = "T - t = 5",
            "T-t=6" = "T - t = 6",
            "T-t=7" = "T - t = 7",
            "T-t=8" = "T - t = 8"))

price_plot <- ggplot() +
  geom_line(aes(x=seq(0, 30, by=0.01), y=sapply(seq(0, 30, by=0.01), f)),
            colour="black") +
```

```
  labs(x=expression(S[T]), y=expression(f(S[T])))

for(k in 1:(T+1)){
  price_plot <- price_plot +
    geom_line(aes_(x=S[,k], y=Pi[,k], colour=colors[k]), linetype=k)
}

price_plot <- price_plot +
  labs(x = expression(S[t]),
                  y = "Price",
                  color = NULL) +
  theme(legend.position=c(.9,.6))

price_plot

library(xtable)
print(xtable(Pi), include.rownames=FALSE)


### Combination of put and call options

# Definition of a put with strike k
put <- function(x, k) return(max((k-x), 0))

colors <- c(
  "1"="K = 10",
  "2"="K = 20"
)

decomposition <- ggplot() +
  geom_line(aes(x=seq(0, 30, by=0.01), y=sapply(seq(0, 30, by=0.01), put, k=10),
                colour=colors[1])) +
  geom_line(aes(x=seq(0, 30, by=0.01), y=sapply(seq(0, 30, by=0.01), put, k=20),
                colour=colors[2])) +
  labs(x=expression(S[T]), y=expression((K-S[T])^{"+"}), color=NULL) +
  theme(legend.position=c(.9,.8))

decomposition

# Price the K=10 put option
put_k_10_tilde <- matrix(NA, nrow=nrow, ncol=ncol, dimnames=
                     list(NULL, paste("t", 0:T, sep="=")))
put_k_10_tilde[,T+1] <- sapply(S[,T+1], put, k=10) / B[T+1]

for (j in (ncol-1):1) {
  for(i in (ncol-j+1):(ncol+j-1)) {
    put_k_10_tilde[i, j] = (q_plus * put_k_10_tilde[i-1, j+1] +
                        (1 - (q_plus + q_minus)) * put_k_10_tilde[i, j+1] +
                        q_minus * put_k_10_tilde[i+1, j+1])}
}

put_k_10 <- put_k_10_tilde
for (j in 1:ncol) {
```

```
  put_k_10[, j] = put_k_10_tilde[, j] * B[j]
}

# Price the K=20 put option
put_k_20_tilde <- matrix(NA, nrow=nrow, ncol=ncol, dimnames=
                           list(NULL, paste("t", 0:T, sep="")))
put_k_20_tilde[,T+1] <- sapply(S[,T+1], put, k=20) / B[T+1]

for (j in (ncol-1):1) {
  for(i in (ncol-j+1):(ncol+j-1)) {
    put_k_20_tilde[i, j] = (q_plus * put_k_20_tilde[i-1, j+1] +
                              (1 - (q_plus + q_minus)) * put_k_20_tilde[i, j+1] +
                              q_minus * put_k_20_tilde[i+1, j+1])}
}

put_k_20 <- put_k_20_tilde
for (j in 1:ncol) {
  put_k_20[, j] = put_k_20_tilde[, j] * B[j]
}

# Sum of prices
total_price <- put_k_10 + put_k_20
round(total_price, 5) == round(Pi, 5)
print(xtable(total_price), include.rownames = FALSE)
```