# Options and Mathematics: Lecture 14

November 25, 2020

## Computation of the binomial price of European/American derivatives

### Computation of the binomial stock price

In this section it is shown how to compute the binomial stock price with Matlab.

Our goal is to construct a binomial tree for the stock price in some interval $[0,T]$, with $T>0$ measured in fraction of years.

Let us start by dividing the interval $[0,T]$ into $N$ subintervals of length $h=T/N$, i.e.,

$$0=t_1<t_2<\cdots<t_{N+1}=T, \quad t_{i+1}=t_i+h, \quad i=1,\dots,N.$$

Let $S(i)=S(t_i)$. We define the binomial stock price on the given partition of $[0,T]$ as

$$S(i+1)=\left\{\begin{array}{ll} S(i)e^u, & \text{with probability } p \\ S(i)e^d, & \text{with probability } 1-p \end{array}\right., \quad i\in\mathcal{I}.$$

The following code defines the Matlab function `BinomialStock` which generates the binomial tree for the stock price on the partition $Q = \{t_1, \dots, t_{N+1}\}$ of the interval $[0, T]$:

```
function [Q,S]=BinomialStock(p,alpha,sigma,s,T,N)
h=T/N;
u=alpha*h+sigma*sqrt(h)*sqrt((1-p)/p);
d=alpha*h-sigma*sqrt(h)*sqrt(p/(1-p));
Q=zeros(N+1,1);
S=zeros(N+1);
Q(1)=0;
S(1,1)=s;
for j=1:N
Q(j+1)=j*h;
S(1,j+1)=S(1,j)*exp(u);
for i=1:j
S(i+1,j+1)=S(i,j)*exp(d);
end
end
```

For example, by running the command

```
[Q,S]=BinomialStock(0.5,0.01,0.2,10,1/12,5)
```

we get the output

$$Q = \begin{matrix} 0 \\ 0.0167 \\ 0.0333 \\ 0.0500 \\ 0.0667 \\ 0.0833 \end{matrix}$$

$$S = \begin{matrix} 10.0000 & 10.2633 & 10.5335 & 10.8108 & 11.0954 & 11.3875 \\ 0 & 9.7467 & 10.0033 & 10.2667 & 10.5370 & 10.8144 \\ 0 & 0 & 9.4999 & 9.7500 & 10.0067 & 10.2701 \\ 0 & 0 & 0 & 9.2593 & 9.5030 & 9.7532 \\ 0 & 0 & 0 & 0 & 9.0248 & 9.2624 \\ 0 & 0 & 0 & 0 & 0 & 8.7962 \end{matrix}$$

## Random paths of the binomial stock price

Recall that in the applications to real-word problems the number $N$ should be chosen sufficiently large (i.e., $h$ should be small compared to $T$), which makes it practically impossible to generate all possible $2^N$ paths of the stock price.

The following code computes a set of $M$ random paths of the stock price, where $M \leq 2^N$

```
function [Rp,Nu]=RandomPathsBinomial(S,M)
N=length(S)-1;
rng shuffle;
r = randi(2,M,N) - 1;
Nu=sum(r==0,2);
Rp=zeros(M,N+1);
rows=zeros(M,N+1);
rows(:,1)=1;
columns=transpose(ones(1,M))*[1:N+1];
for j=1:N
rows(:,j+1)=rows(:,j)+r(:,j);
end
idx = sub2ind(size(S), rows, columns);
Rp=S(idx);
```

$$[\mathtt{Q},\mathtt{S}] = \mathtt{BinomialStock}(1/2, 0, 0.5, 10, 1, 100);$$
$$[\mathtt{Rp},\mathtt{Nu}] = \mathtt{RandomPathsBinomial}(\mathtt{S}, 100);$$

we generate 100 random paths of the binomial stock price with $\alpha = 0$, $\sigma = 50\%$, $S(0) = 10$ in the time interval $[0, 1]$ divided in 100 points. A graphical representation of these paths is shown in Figure 1 .

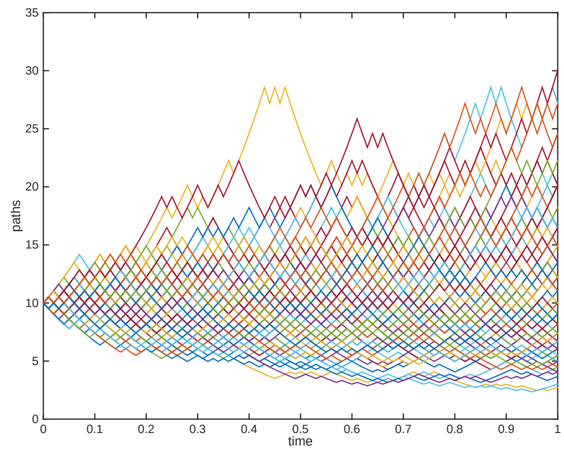



Figure 1: 100 random paths of a binomial stock price

## Computation of the binomial price of standard European derivatives

In this section we discuss the numerical implementation of the binomial options pricing model with Matlab.

Consider a partition $0 = t_1 < t_2 < \cdots < t_{N+1} = T$ of the interval $[0, T]$ and the binomial stock price

$$S(i+1) = \begin{cases} S(i)e^u, & \text{with probability } p \\ S(i)e^d, & \text{with probability } 1-p \end{cases}, \quad i \in \mathcal{I} = \{1, \ldots, N\},$$

where $S(i) = S(t_i)$ and

$$u = \alpha h + \sigma \sqrt{\frac{1-p}{p}} \sqrt{h}, \quad d = \alpha h - \sigma \sqrt{\frac{p}{1-p}} \sqrt{h}, \quad h = \frac{T}{N}.$$

The value of the risk-free asset at time $t_i$ is given by

$$B(t_i) = B_0 e^{rt_i} = B_0 e^{(rh)i} := B(i).$$

Hence the pair $(S(i), B(i))$ defines a binomial marke with parameters $u, d$ given as above and risk-free rate $rh$.

Since

$$\alpha h - \sigma \sqrt{\frac{p}{1-p}} \sqrt{h} < rh < \alpha h + \sigma \sqrt{\frac{1-p}{p}} \sqrt{h}$$

holds for $h$ small, hence the condition for the non-existence of self-financing arbitrage portfolios in the market is satisfied provided we take our time partition to be sufficiently fine.

The recurrence formula for the price of the European option with pay-off $Y$ at maturity $T$ becomes

$$\Pi_Y(N+1) = Y, \quad \text{and} \quad \Pi_Y(i) = e^{-rh}[q_u \Pi_Y^u(i+1) + q_d \Pi_Y^d(i+1)], \quad \text{for } i \in \mathcal{I},$$

where $\Pi_Y(i) = \Pi_Y(t_i)$ and

$$q_u = \frac{e^{rh} - e^{\alpha h - \sigma \sqrt{\frac{p}{1-p}} \sqrt{h}}}{e^{\alpha h + \sigma \sqrt{\frac{1-p}{p}} \sqrt{h}} - e^{\alpha h - \sigma \sqrt{\frac{p}{1-p}} \sqrt{h}}}, \quad q_d = 1 - q_u.$$

The recurrence formula for pricing standard European options is implemented by the following Matlab function

```
function Price=BinomialEuropean(Q,S,r,g)
h=Q(2)-Q(1);
N=length(Q)-1;
expu=S(1,2)/S(1,1);
expd=S(2,2)/S(1,1);
qu=(exp(r*h)-expd)/(expu-expd);
qd=(expu-exp(r*h))/(expu-expd);
if (qu<0 || qd<0)
display('Error:  the market is not arbitrage free.');
Price=0;
return
end
Price=zeros(N+1);
Price(:,N+1)=g(S(:,N+1));
for j=N:-1:1
for i=1:j
Price(i,j)=exp(-r*h)*(qu*Price(i,j+1)+qd*Price(i+1,j+1));
end
end
```

For example, let $S$ be the binomial tree constructed before and run the command

```
Price=BinomialEuropean(Q,S,0.01,@(x)max(x-10,0))
```

which computes the binomial price of a European call with strike $K = 10$. The result is

$$Price = \begin{matrix} 0.2461 & 0.3817 & 0.5705 & 0.8141 & 1.0971 & 1.3875 \\ 0 & 0.1140 & 0.1978 & 0.3333 & 0.5387 & 0.8144 \\ 0 & 0 & 0.0325 & 0.0658 & 0.1333 & 0.2701 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

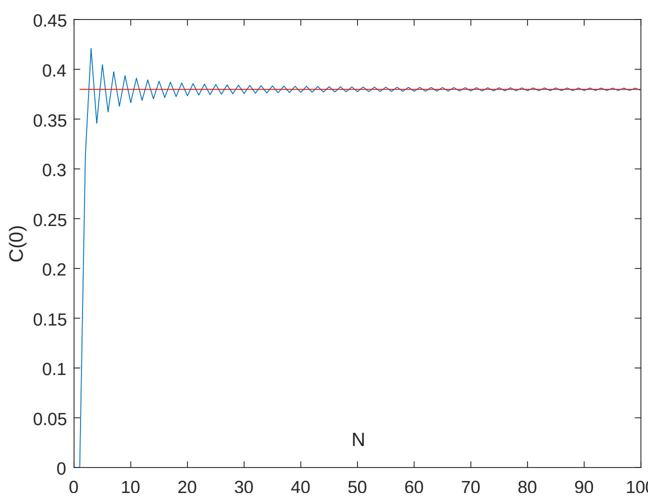



Figure 2: Initial binomial price $C(0) = Price(1,1)$ of the call computed for increasing values of $N$ ($S_0 = 10, K = 10.5, T = 1/2, \alpha = 0.1, \sigma = 0.2, r = 0.01, p = 1/2$). The red line indicates the Black-Scholes price of the call. The binomial price stabilizes around the Black-Scholes price for $N \gtrsim 100$.

## Parameters sensitivity analysis

An important application of the binomial model, as of any other options pricing model, is the study of how the value of an option depends on the market parameters.

In this section we perform this **parameters sensitivity analysis** for the call option using the Matlab code given above.

In the subsequent discussion the numbers of periods $N$ is fixed to $N = 10000$. With such a large number of steps, and for realistic values of the market parameters, the binomial price and the Black-Scholes price of the call option are practically the same.

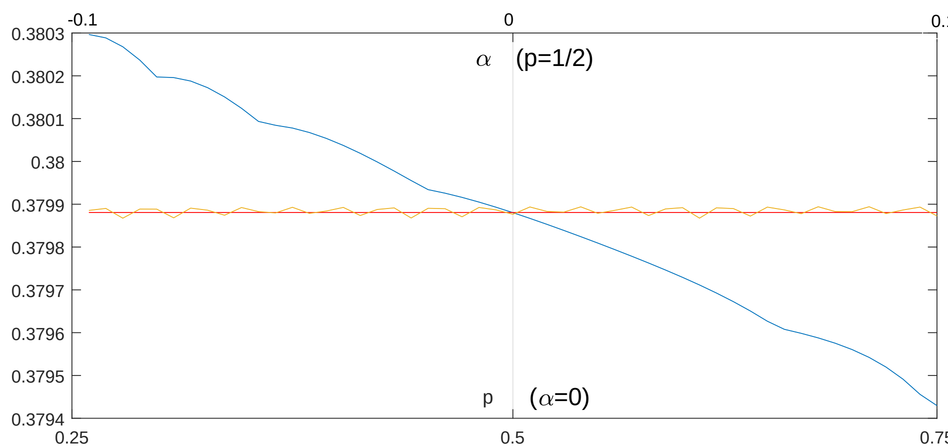
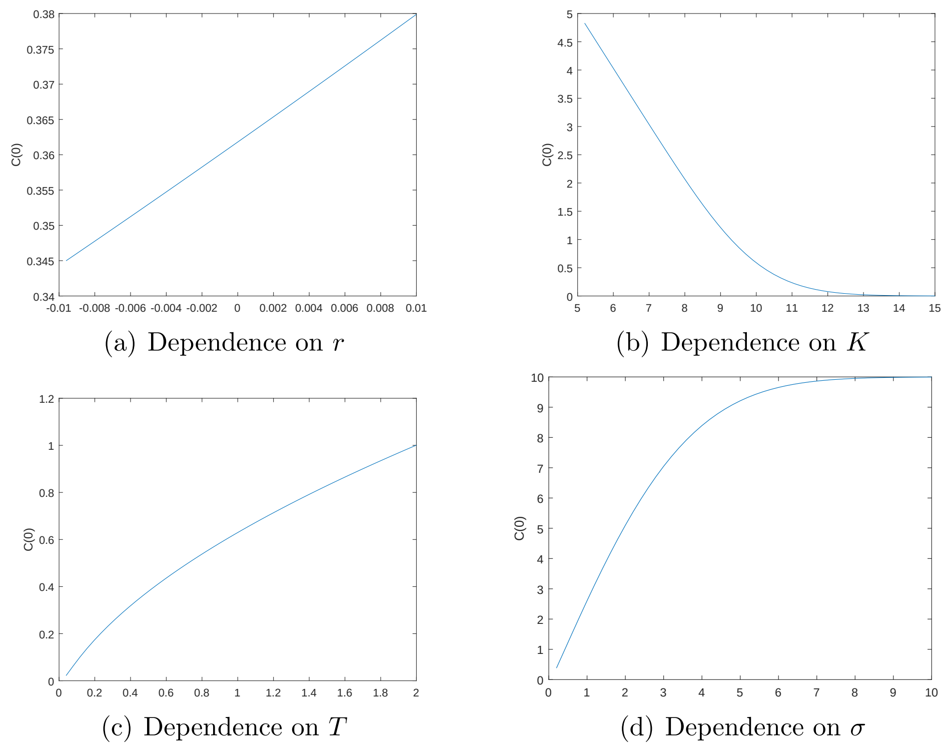



Figure 3: Initial binomial price $C(0) = Price(1,1)$ of the call computed for different values of $p \in (0,1)$ (blue line) and $\alpha \in [-0.1, 0.1]$ (yellow line). The red line indicates the Black-Scholes price ($S_0 = 10, K = 10.5, T = 1, \sigma = 0.1, r = 0.01$). This picture clearlt indicates that the binomial price is weakly dependent on the parameter $\alpha, p$ and that the best approximation to the Black-Scholes price is obtained for $p = 1/2$.

*From now on we assume* $p = 1/2$ *and* $\alpha = 0$. This choice is justifed if $N$ is sufficiently large




(a) Dependence on $r$

(b) Dependence on $K$

(c) Dependence on $T$

(d) Dependence on $\sigma$

Figure 4: Sensitivity of the call option binomial price with respect to the parameters $(r, K, T, \sigma)$.

## Computation of the binomial price of non-standard European derivatives

The algorithm used for standard European derivatives cannot be applied to non-standard derivatives, because in the latter case one has to compute the pay-off long each of the $2^N$ paths of the stock price, which is possible by today's computers only for $N \lessapprox 20$.

We use a different numerical method, called **Monte Carlo**.

Recall that the binomial price at time $t = 0$ of the European derivative with pay-off $Y$ and maturity $T = Nh$ is given by

$$\Pi_Y(0) = e^{-rhN} \sum_{x \in \{u,d\}^N} (q_u)^{N_u(x)} (1-q_u)^{N_d(x)} Y(x),$$

Now, let $\mathcal{O}$ be a set of $M$ *randomly* chosen paths of the binomial stock price, where $M \leq 2^N$.

Our approximation for $\Pi_Y(0)$ is

$$\Pi_Y(0) \approx \frac{2^N}{M} e^{-rhN} \sum_{x \in \mathcal{O}} (q_u)^{N_u(x)} (q_d)^{N_d(x)} Y(x),$$

that is to say, we restrict the sum to the paths in the set $\mathcal{O}$ and multiply further by the factor $2^N/M$, which is the total number of paths divided by the number of sample paths.

Consider for instance the Asian option. The pay-off is

$$Y(x) = \left( \frac{1}{N+1} \sum_{t=0}^{N} S(t) - K \right)_+,$$

The following code computes the Monte Carlo approximation of the price at time $t = 0$

```
function P=AsianCallBinomialMC(Q,S,r,K,M)
h=Q(2)-Q(1);
N=length(Q)-1; expu=S(1,2)/S(1,1);
expd=S(2,2)/S(1,1);
qu=(exp(r*h)-expd)/(expu-expd);
qd=(expu-exp(r*h))/(expu-expd);
if (qu<0 || qd<0)
display('Error:  the market is not arbitrage free');
P=0;
return
end
[R,Nu]=RandomPathsBinomial(S,M);
payoff=max((1/(N+1)*sum(R,2))-K,0);
terms=(qu.^Nu).*(qd.^(N-Nu)).*payoff;
P=exp(-r*h*N)*sum(terms)*2^N/M;
```

For example, upon writing

$$\begin{aligned}
&[\mathtt{Q},\mathtt{S}] = \mathtt{BinomialStock}(1/2, 0, 0.3, 10, 1/2, 100);\\
&\mathtt{P} = \mathtt{AsianCallBinomialMC}(\mathtt{Q}, \mathtt{S}, 0.01, 11, 1000000)
\end{aligned}$$

we compute the price at time $t = 0$ of the Asian call with strike $K = 11$ and maturity $T = 1/2$ on a stock with initial price $S(0) = 10$ and volatility $\sigma = 0.2$; the risk-free rate is $r = 0.01$ and the number of sample paths is $M = 10^6$ of the $2^N = 2^{100}$ possible paths of the binomial stock price.

The computation gives the result $P = 0.1677$.

Now, the crucial question is: how reliable is this result? That is to say, how close is this value to the exact binomial price of the Asian call?

In the following we present an experimental analysis of this problem.

We begin by repeating the above calculation $n$ times to produce the values $P_1, \ldots, P_n$ for the price of the Asian call and pick, as our best estimate for the exact value of the price, the sample average

$$P = \frac{1}{n}\sum_{i=1}^{n} P_i.$$

To measure how reliable is the approximation $P$, we compute the so called **standard error of the mean**

$$\mathrm{Err} = \frac{s}{\sqrt{n}}, \quad \text{where} \quad s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(P_i - P)^2}$$

is the standard deviation of the sample $P_1, \ldots, P_n$.

In the figure below the prices $P_1, \dots P_n$ computed for $n = 50$ trials are depicted using $M = 100, 1000, 10000$ sample paths. The average $P$ and the error Err are given in the following table.

| $M$ (number of paths) | $P$ (average price) | Err (standard error of the mean) |
|---|---|---|
| 100 | 0.1725 | 0.0067 |
| 1000 | 0.1659 | 0.0020 |
| 10000 | 0.1678 | 0.0005 |

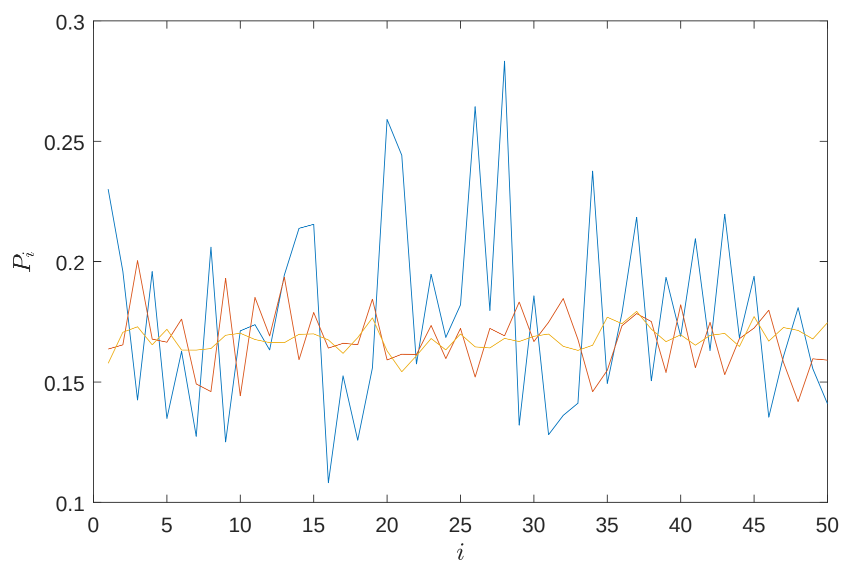



Figure 5: Monte Carlo approximation of the Asian call price for $n = 50$ different trials using 100 paths (blue line), 1000 paths (orange line) and 10000 paths (yellow line). Market parameters: $T = 1/2$, $K = 11$, $S(0) = 10$, $\sigma = 0.3$, $r = 0.01$.

## Computation of the binomial price of standard American derivatives

We work under the same set-up as before. Namely we consider a partition $0 = t_1 < t_2 < \cdots < t_{N+1} = T$ of the interval $[0, T]$ and the binomial stock price

$$S(i+1) = \begin{cases} S(i)e^u, & \text{with probability } p \\ S(i)e^d, & \text{with probability } 1-p \end{cases}, \quad i \in \mathcal{I} = \{1, \ldots, N\},$$

where $S(i) = S(t_i)$ and

$$u = \alpha h + \sigma \sqrt{\frac{1-p}{p}} \sqrt{h}, \quad d = \alpha h - \sigma \sqrt{\frac{p}{1-p}} \sqrt{h}.$$

The value of the risk-free asset at time $t_i$ is given by $B(t_i) = B_0 e^{r t_i} = B_0 e^{(rh)i} := B(i)$.

Hence the pair $(S(i), B(i))$ defines a binomial market with parameters $u, d$ given and interest rate $rh$.

The definition of binomial price of American derivative becomes

$$\widehat{\Pi}_Y(N+1) = Y(N+1) \quad \widehat{\Pi}_Y(i) = \max(Y(i), e^{-rh}(q_u \widehat{\Pi}_Y^u(i+1)) + q_d \widehat{\Pi}_Y^d(i+1)), \quad i \in \mathcal{I},$$

where $\widehat{\Pi}_Y(i) = \widehat{\Pi}_Y(t_i)$ and

$$q_u = \frac{e^{rh} - e^{\alpha h - \sigma \sqrt{\frac{p}{1-p}} \sqrt{h}}}{e^{\alpha h + \sigma \sqrt{\frac{1-p}{p}} \sqrt{h}} - e^{\alpha h - \sigma \sqrt{\frac{p}{1-p}} \sqrt{h}}}, \quad q_d = 1 - q_u.$$

Moreover $Y(i) = g(S(i))$, $i = 1, \ldots, N+1$, is the intrinsic value of the American derivative.

```
function [Price,C]=BinomialAmerican(Q,S,r,g)
h=Q(2)-Q(1);
N=length(Q)-1;
expu=S(1,2)/S(1,1);
expd=S(2,2)/S(1,1);
qu=(exp(r*h)-expd)/(expu-expd);
qd=(expu-exp(r*h))/(expu-expd);
if (qu<0 || qd<0)
display('Error:  the market is not arbitrage free.');
P=0;
return
end
Price=zeros(N+1);
Price(:,N+1)=g(S(:,N+1));
C(:,N+1)=0;
Y=g(S);
for j=N:-1:1
for i=1:j
Price(i,j)=max(Y(i,j),exp(-r*h)*(qu*Price(i,j+1)+qd*Price(i+1,j+1)));
C(i,j)=Price(i,j)-exp(-r*h)*(qu*Price(i,j+1)+qd*Price(i+1,j+1));
end
end
```

For example, using as inputs the binomial stock price computed at the beginning of the lecture, the interest rate $r = 0.01$ and the pay-off function of the put with strike 10, i.e., $g(x) = (10 - x)_+$, we obtain the output

$$Price = \begin{matrix} 0.2385 & 0.1120 & 0.0320 & 0 & 0 & 0 \\ 0 & 0.3619 & 0.1899 & 0.0633 & 0 & 0 \\ 0 & 0 & 0.5297 & 0.3133 & 0.1250 & 0 \\ 0 & 0 & 0 & 0.7407 & 0.4970 & 0.2468 \\ 0 & 0 & 0 & 0 & 0.9752 & 0.7376 \\ 0 & 0 & 0 & 0 & 0 & 1.2038 \end{matrix} \qquad C = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0017 & 0.0017 & 0 \\ 0 & 0 & 0 & 0 & 0.0017 & 0 \end{matrix}$$

**Remarks**

- Using the pay-off of the call, $g(x) = (x-10)_+$, we obtain that the price of the American call is exactly the same as the corresponding European call, while $C \equiv 0$. This of course is consistent with the proven fact that, in the absence of dividends, it is never optimal to exercise an American call prior to expire.
- The non-zero entries of $C$ identify the optimal exercise times