Numerical Methods of Option Pricing

Jia-Ping Huang

This material is based on
Hull, C. (2018), *Options, Futures, and Other Derivatives*,
10th Edition, Pearson.

Options

- Option
  the *right* to sell/buy the underlying asset by a certain date for a certain price.

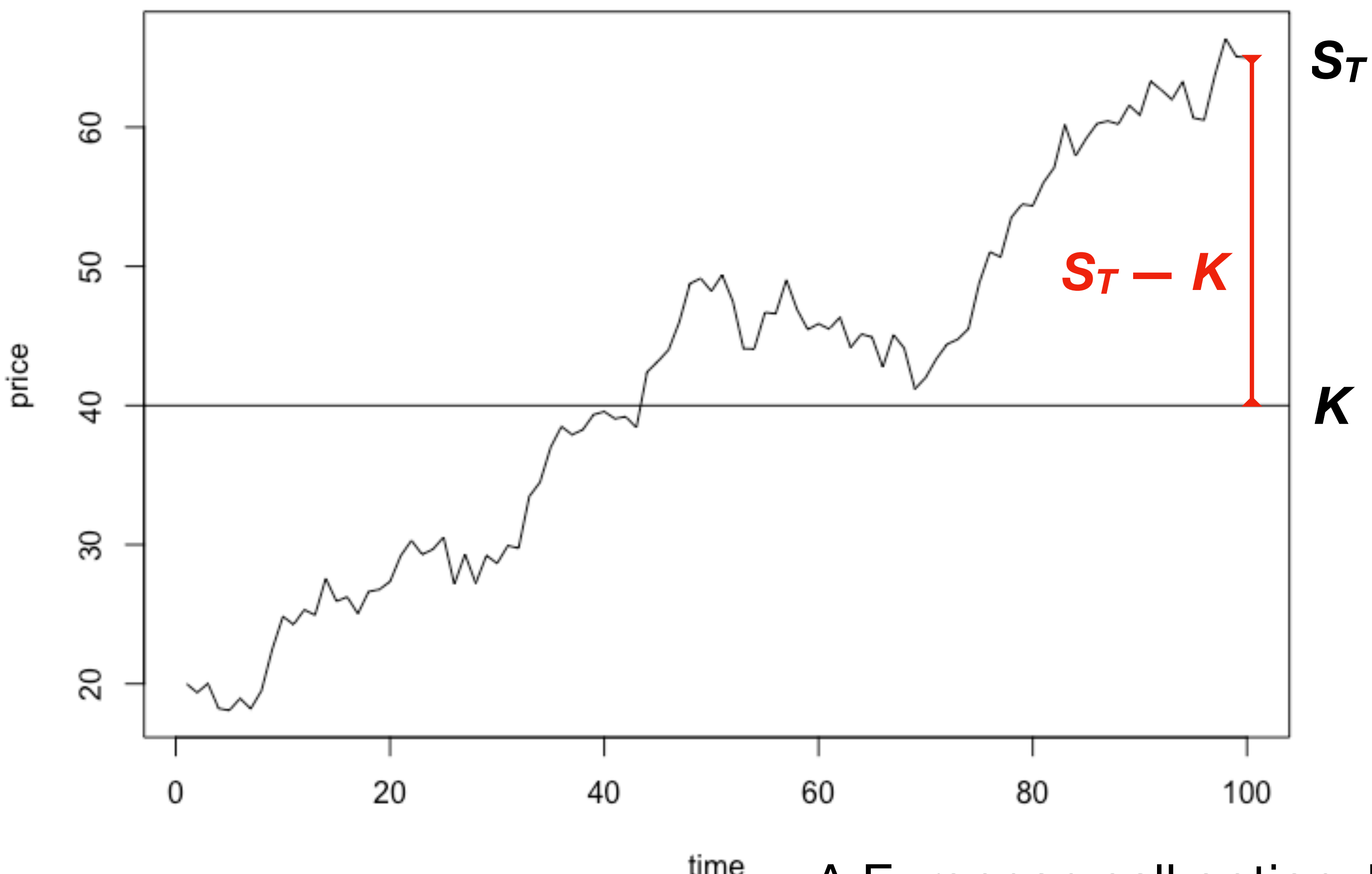
- Expiration (maturity) date, strike price.

- Call option — the right to buy;
  Put option — the right to sell.

- American option — can be exercised at any time up to the expiration date;
  European option — can be exercised only on the expiration date.

Option profits

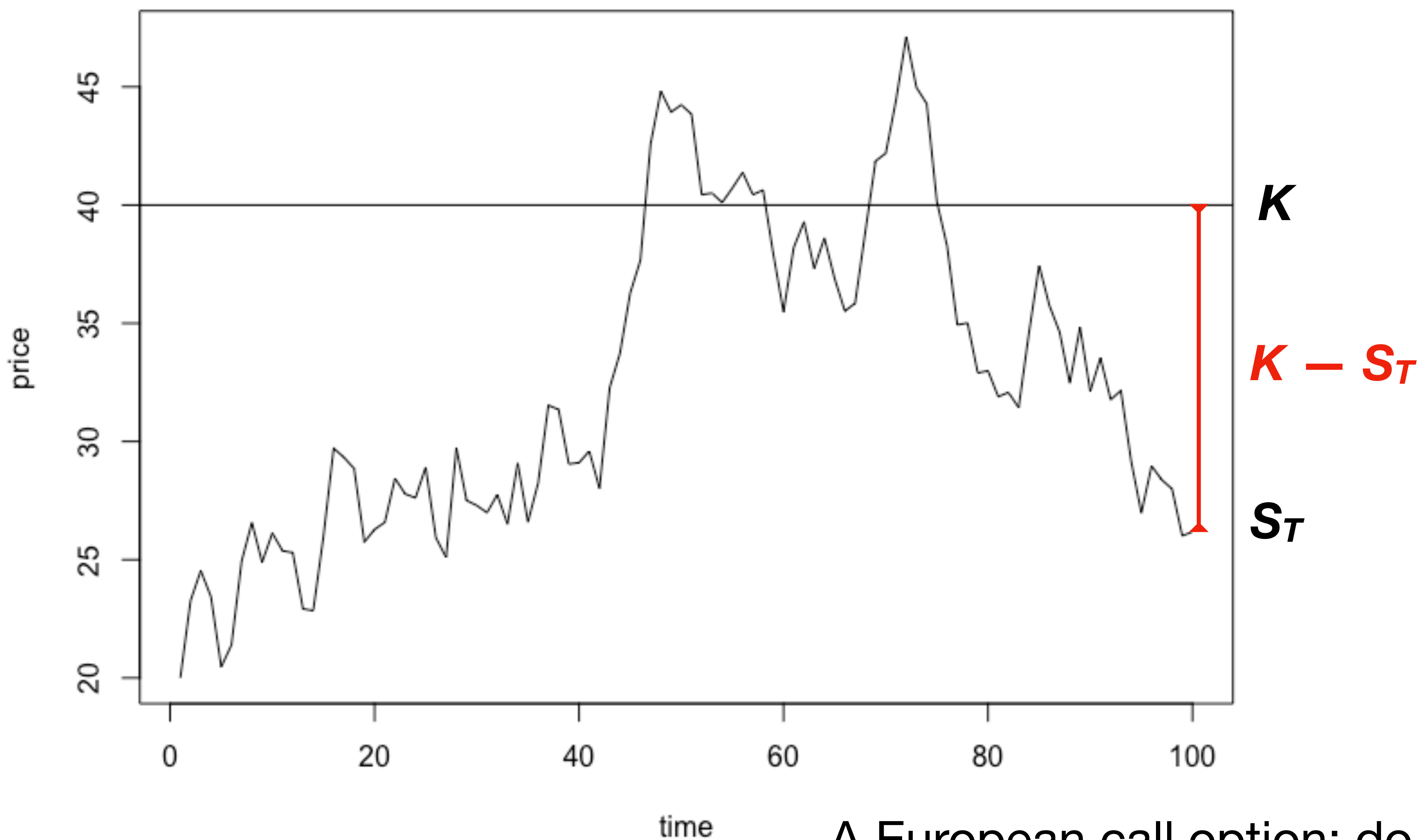
- Underlying asset price at maturity S_T; strike price K.



A European call option: buy at K and and sell at S_T.
The profit is then (S_T - K) minus the option price.

Option profits

- Underlying asset price at maturity S_T; strike price K.

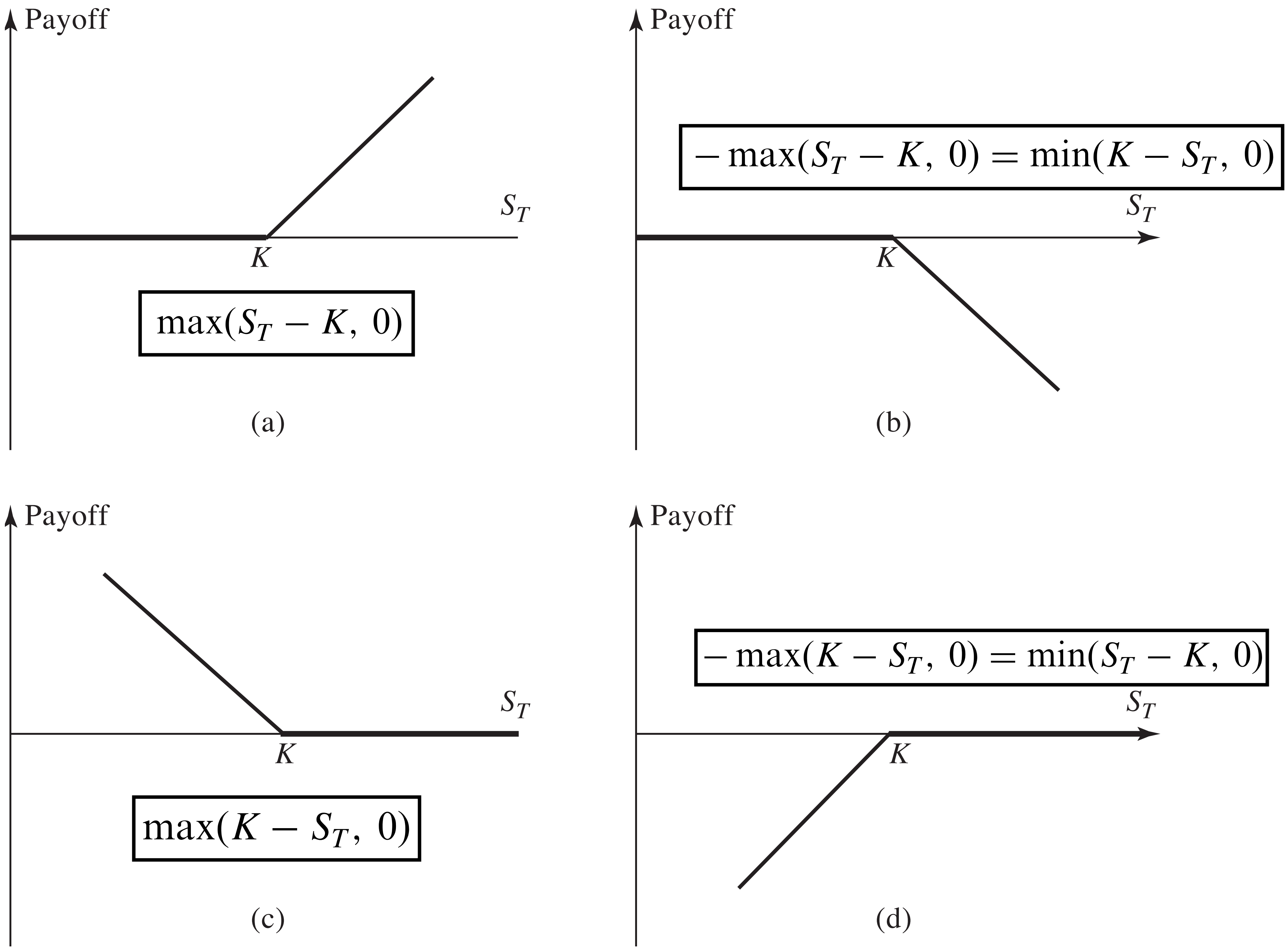


A European call option: do not exercise.
The profit is 0 minus the option price.

Option positions

- Long position — buy the option;
  Short position — sell the option.

- Four positions:
  1. a long position in a call option;
  2. a long position in a put option;
  3. a short position in a call option;
  4. a short position in a put option.



a) a long call
b) a short call
c) a long put
d) a short put

Stock option pricing

- Assumptions

  1. There are no transaction costs.
  2. All trading profits (net of trading losses) are subject to the same tax rate.
  3. Borrowing and lending are possible at the risk-free interest rate.

- Notation

S_0: Current stock price
K: Strike price of option
T: Time to expiration of option
S_T: Stock price on the expiration date
r: Continuously compounded risk-free rate of interest for an investment maturing in time T

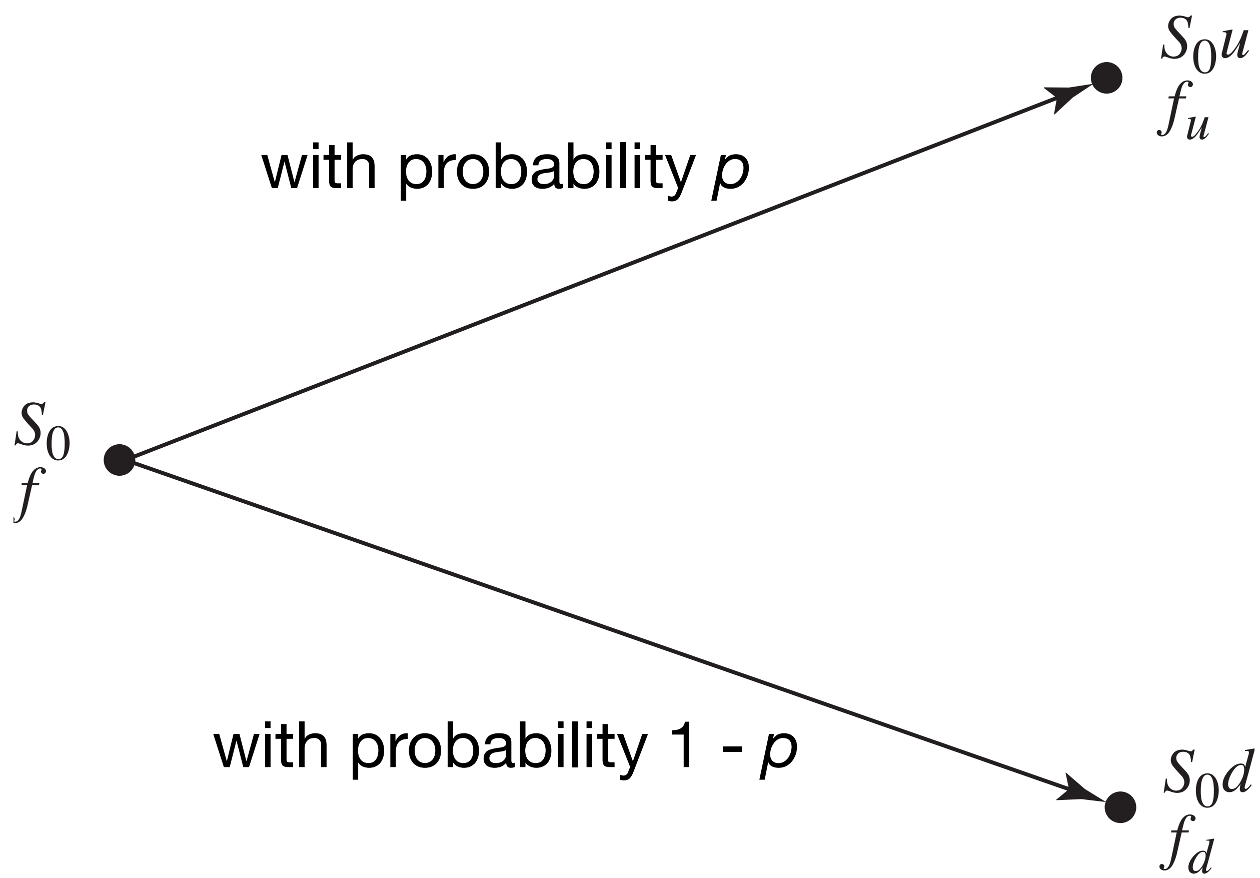
Risk-neutral evaluation

- We assume a risk-neutral world such that

  1. The expected return on a stock (or any other investment) is the risk-free rate.

  2. The discount rate used for the expected payoff on an option (or any other instrument) is the risk-free rate.

Models of stock price
Binomial trees

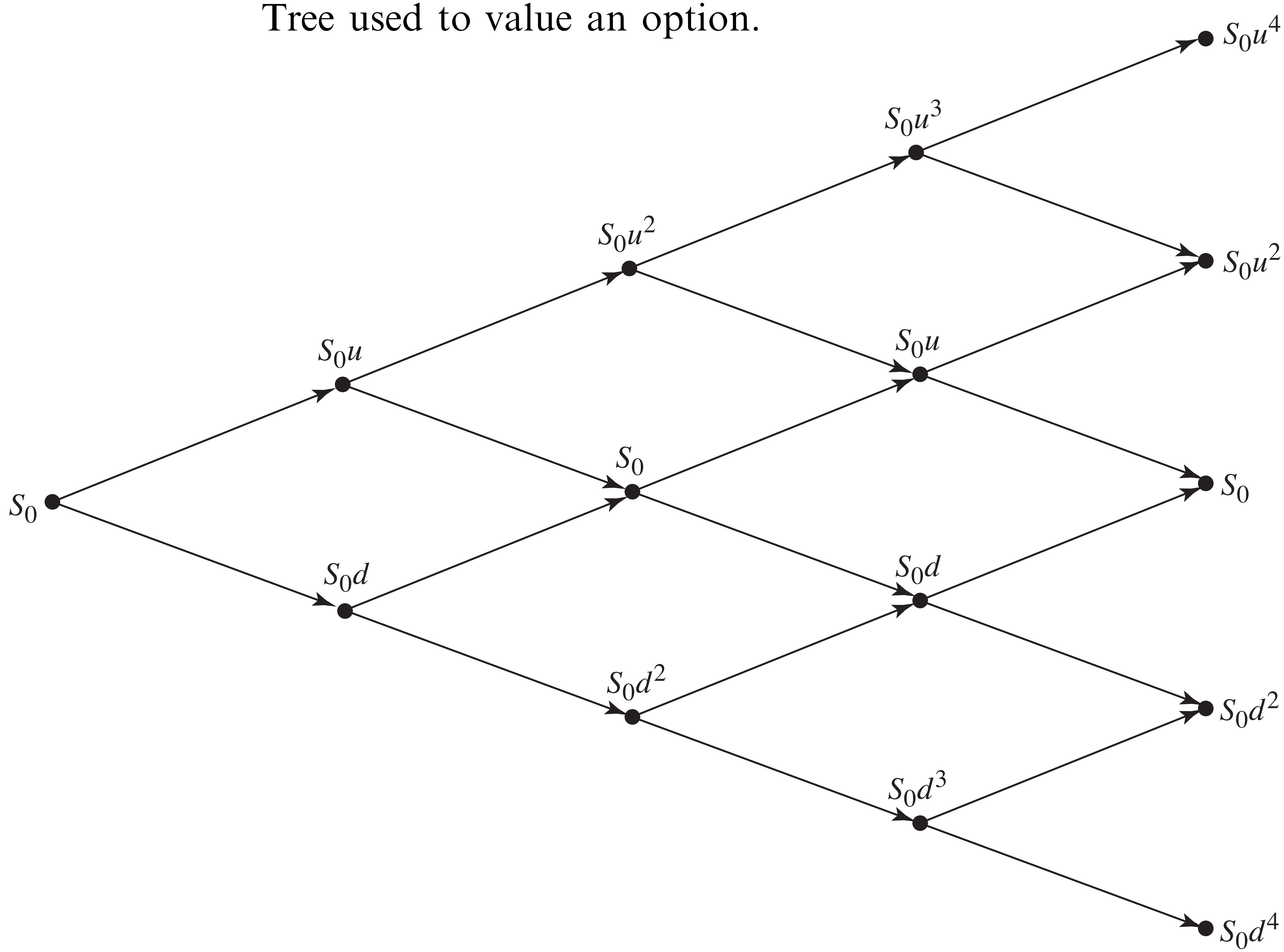


$$E(S_T) = pS_0u + (1 - p)S_0d$$

$$E(S_T) = S_0e^{rT}$$

$$\Rightarrow \quad p = \frac{e^{rT} - d}{u - d}$$

Tree used to value an option.



$$u = 1/d$$

$$\Downarrow$$

$$p = \frac{a - d}{u - d}$$

$$u = e^{\sigma\sqrt{\Delta t}}$$

$$d = e^{-\sigma\sqrt{\Delta t}}$$

$$a = e^{(r-q)\Delta t}$$

An American put option

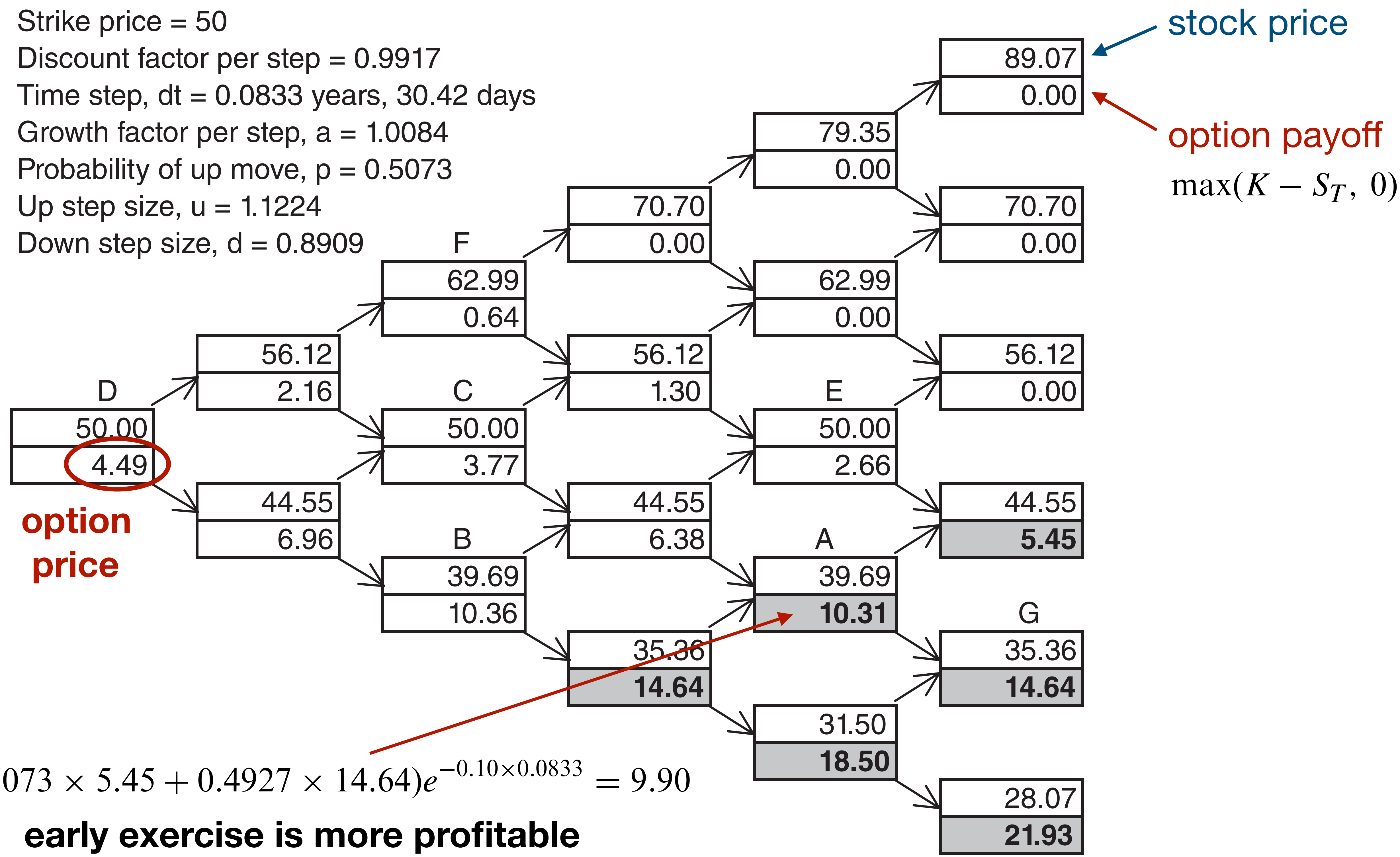
Consider a 5-month American put option on a non-dividend-paying stock when the stock price is \$50, the strike price is \$50, the risk-free interest rate is 10% per annum, and the volatility is 40% per annum. With our usual notation, this means that $S_0 = 50$, $K = 50$, $r = 0.10$, $\sigma = 0.40$, $T = 0.4167$, and $q = 0$. Suppose that we divide the life of the option into five intervals of length 1 month (= 0.0833 year) for the purposes of constructing a binomial tree. Then $\Delta t = 0.0833$ and using equations (21.4) to (21.7) gives

$$u = e^{\sigma\sqrt{\Delta t}} = 1.1224, \qquad d = e^{-\sigma\sqrt{\Delta t}} = 0.8909, \qquad a = e^{r\Delta t} = 1.0084$$

$$p = \frac{a-d}{u-d} = 0.5073, \qquad 1 - p = 0.4927$$

At each node:
Upper value = Underlying Asset Price
Lower value = Option Price
Shading indicates where option is exercised

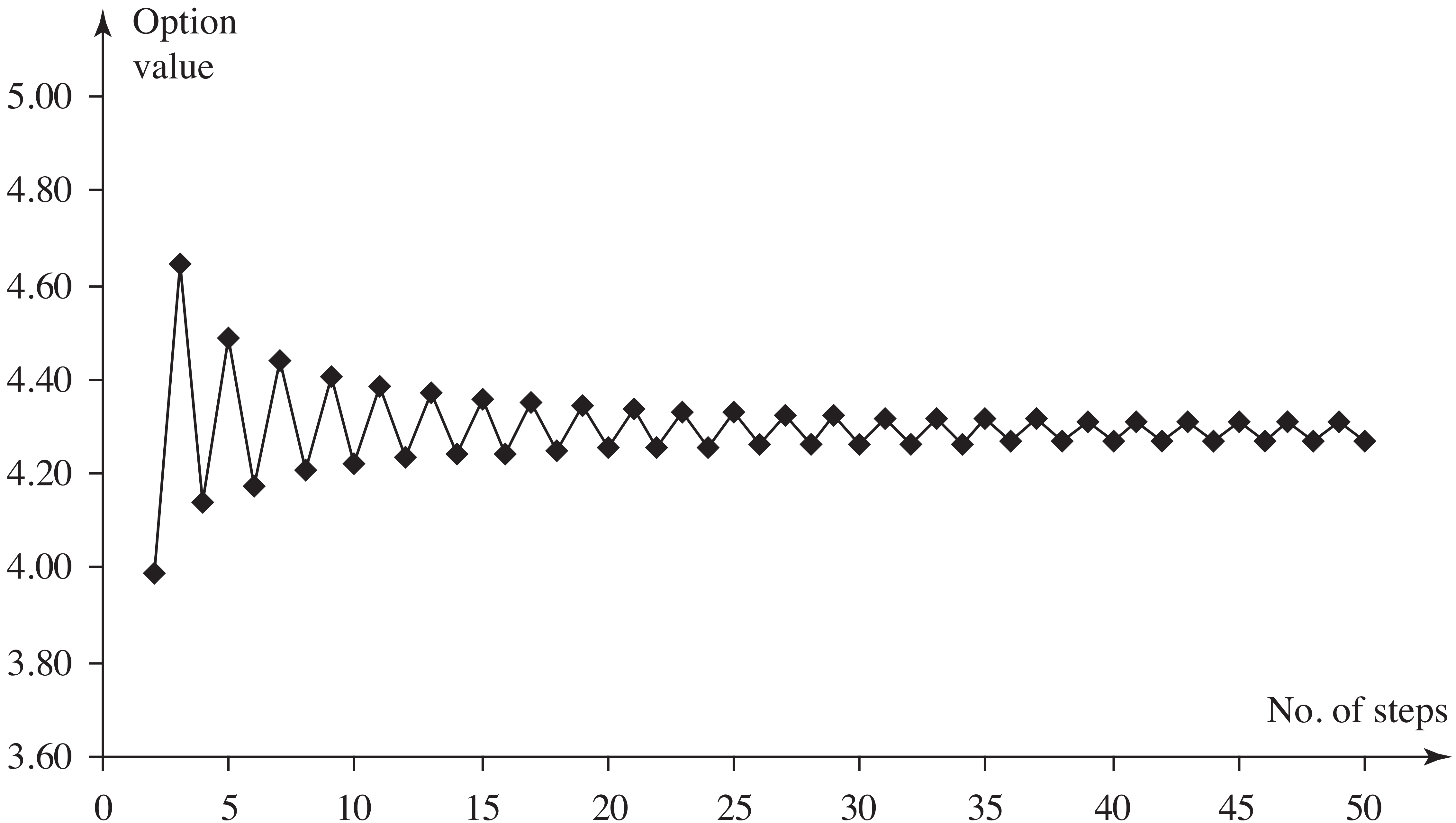
Strike price = 50
Discount factor per step = 0.9917
Time step, dt = 0.0833 years, 30.42 days
Growth factor per step, a = 1.0084
Probability of up move, p = 0.5073
Up step size, u = 1.1224
Down step size, d = 0.8909



$$(0.5073 \times 5.45 + 0.4927 \times 14.64)e^{-0.10\times 0.0833} = 9.90$$

early exercise is more profitable

Number of periods



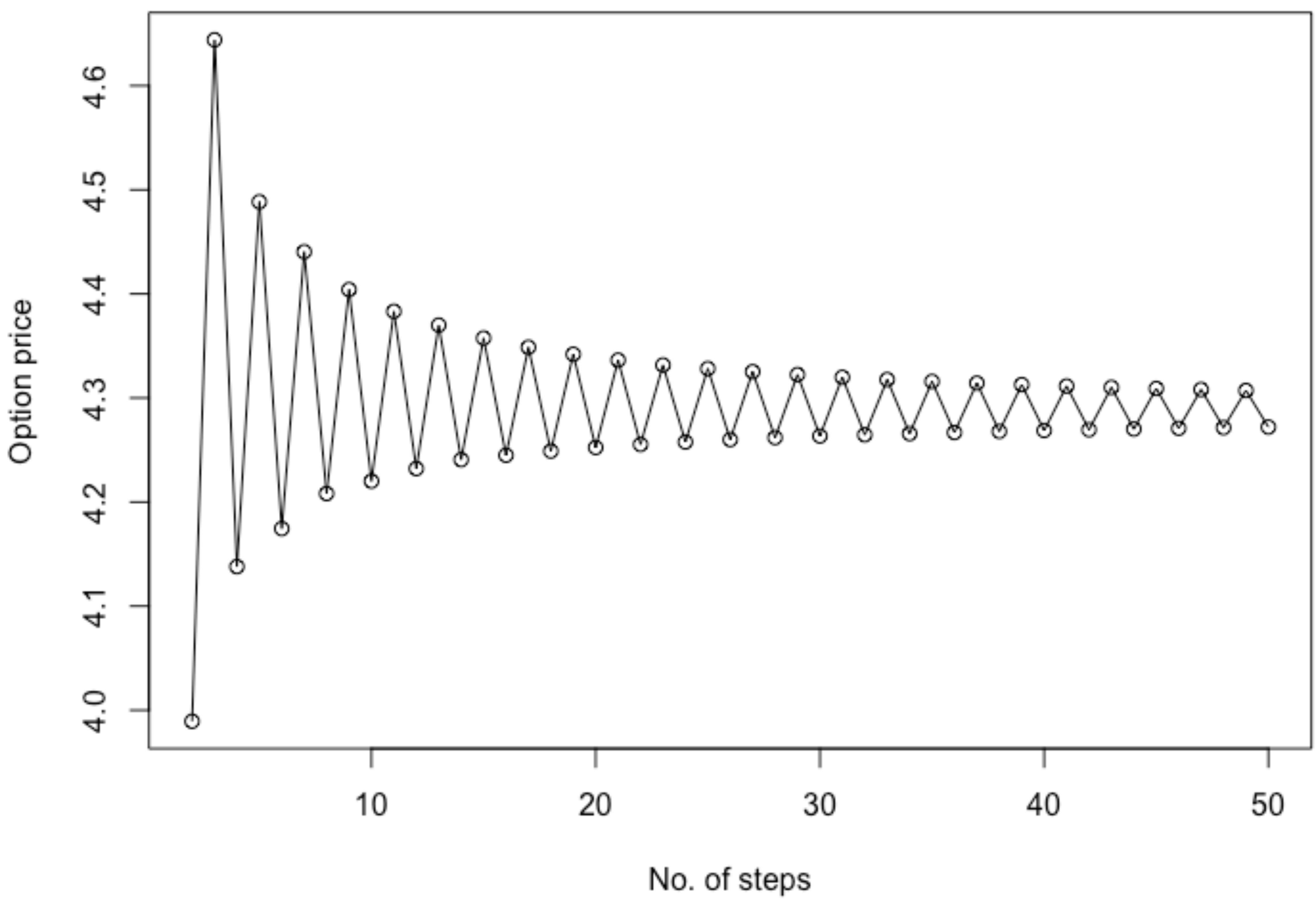
```r
### Binomial Tree of an American Put Option ###
rm(list = ls())   # remove (almost) everything in the working environment

## Parameters
n <- 5   # number of periods
S0 <- 50  # stock price at period 0
K <- 50   # strike price
r <- 0.1   # risk-free interest rate (annual)
q <- 0   # yield of the underlying asset (annual)
sigma <- 0.4   # volatility (annual)
M <- 5/12   # maturity (in years)

## Variables
dt <- M/n   # duration of each period (in years)
u <- exp(sigma * sqrt(dt))   # up step size
d <- 1/u   # down step size
a <- exp((r-q) * dt)   # growth factor per step
p <- (a-d) / (u-d)   # probability of up move

S <- matrix(rep(0, (n+1)^2), n+1, n+1)   # stock prices
V <- matrix(rep(0, (n+1)^2), n+1, n+1)   # option values
S[1,1] <- S0
```

```r
## Calculation of stock prices
# # Method 1
# for (j in 2:(n+1)) {
#   for (i in 1:j) {
#     nd <- i - 1   # number of down moves
#     nu <- j - 1 - nd   # number of up moves
#     S[i,j] <- S0 * u^nu * d^nd
#   }
# }

# Method 2
for (j in 2:(n+1)) {
  for (i in 1:j-1) {
    S[i,j] <- S[i,j-1] * u
  }
  S[j,j] <- S[j-1,j-1] * d
}
```

```r
## Calculation of option values
# Final nodes
for (i in 1:n+1) {
  V[i,n+1] <- max(K-S[i,n+1], 0)
}

# Earlier nodes
for (j in n:1) {
  for (i in 1:j) {
    dp <- (p * V[i,j+1] + (1-p) * V[i+1,j+1]) * exp(-r * dt)
        # discounted price
    V[i,j] <- max(K-S[i,j], dp)
        # comparing the early exercise and discounted price
  }
}

# the option value is V[1,1]
```

```
> S
     [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
[1,]   50 56.12005 62.98919 70.69912 79.35276 89.06561
[2,]    0 44.54736 50.00000 56.12005 62.98919 70.69912
[3,]    0  0.00000 39.68935 44.54736 50.00000 56.12005
[4,]    0  0.00000  0.00000 35.36112 39.68935 44.54736
[5,]    0  0.00000  0.00000  0.00000 31.50489 35.36112
[6,]    0  0.00000  0.00000  0.00000  0.00000 28.06920
> V
         [,1]     [,2]       [,3]      [,4]      [,5]      [,6]
[1,] 4.488459 2.162519  0.6359836  0.000000  0.000000  0.000000
[2,] 0.000000 6.959743  3.7711415  1.301666  0.000000  0.000000
[3,] 0.000000 0.000000 10.3612944  6.378043  2.664116  0.000000
[4,] 0.000000 0.000000  0.0000000 14.638882 10.310650  5.452637
[5,] 0.000000 0.000000  0.0000000  0.000000 18.495109 14.638882
[6,] 0.000000 0.000000  0.0000000  0.000000  0.000000 21.930804
```

```r
## Demonstration of the convergence of option value
optionPrice <- function(n, S0, K, r, q, sigma, M) {
  ## Variables
  ……

  ## Calculation of stock prices
  ……

  ## Calculation of option values
  # Final nodes
  ……
  # Earlier nodes
  ……

  return(V[1,1])
}

seqn <- 2:50
seqV <- rep(0, length(seqn))

for (k in 1:length(seqn)) {
  seqV[k] <- optionPrice(seqn[k], S0, K, r, q, sigma, M)
}

plot(seqn, seqV, type = "o", xlab = "No. of steps", ylab = "Option price")
```



Option price
4.0
4.1
4.2
4.3
4.4
4.5
4.6
10
20
30
40
50
No. of steps

Models of stock price
Black-Sholes-Merton model

- Assume that the percentage changes in stock price in a very short period of time are normally distributed.

- Denote

  μ: Expected return in a short period of time (annualized)

  σ: Volatility of the stock price.

  Then,

$$\frac{\Delta S}{S} \sim \phi(\mu\,\Delta t,\ \sigma^2 \Delta t)$$

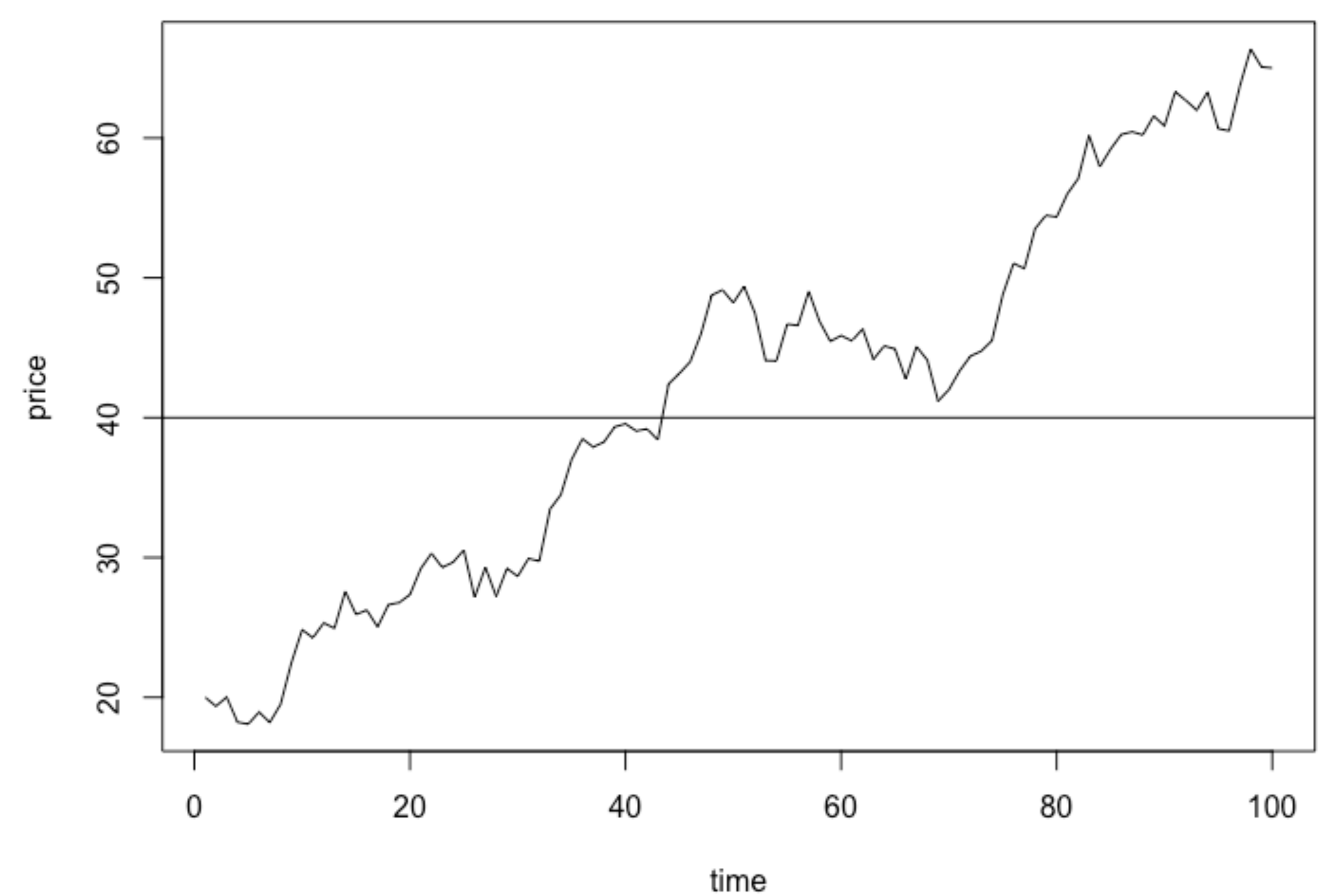
- The stock price process (in continuous time)

$$dS = \mu S\, dt + \sigma S\, dz$$

- The discrete version

$$\Delta S = \mu S\, \Delta t + \sigma S \epsilon \sqrt{\Delta t}$$

where ϵ has a standard normal distribution.



BSM pricing formulas

- Price of European call options:

$$c = S_0 N(d_1) - K e^{-rT} N(d_2)$$

- Price of European put options:

$$p = K e^{-rT} N(-d_2) - S_0 N(-d_1)$$

where

$$d_1 = \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = \frac{\ln(S_0/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

Monte Carlo simulation

- According to the discrete version of BSM model, we can generate sample paths of stock price using a pseudo random number generator.

- Assume that every sample path of stock price occurs with equal probability. Then the option payoff at maturity is the sample mean of payoffs derived from each sample path.

	A	B	C	D	E	F	G
1	45.95	0	S_0	K	r	σ	T
2	54.49	4.38	50	50	0.05	0.3	0.5
3	50.09	0.09		d_1	d_2	BSM price	
4	47.46	0		0.2239	0.0118	4.817	
5	44.93	0					
⋮	⋮	⋮					
1000	68.27	17.82					
1001							
1002	Mean:	4.98					
1003	SD:	7.68					

```r
### Monte Carlo Simulation of Black-Sholes-Merton Model
rm(list = ls())  # remove (almost) everything in the working environment

## Parameters of a European call option
S0 <- 50
K <- 50
r <- 0.05
sigma <- 0.3
M <- 0.5
n <- 100   # number of periods
nn <- 1000   # number of sample paths


## Variables
dt <- M/n
S <- matrix(rep(0, n*nn), nn, n)   # sample paths of stock price
V <- rep(0, nn)   # option values at maturity
```

```r
## Generation of sample paths
RN <- matrix(rnorm(n*nn), nn, n)   # random noise matrix
for (k in 1:nn) {
  # period 1
  S[k,1] <- S0 * (1 + r * dt + sigma * sqrt(dt) * RN[k,1])
  # later periods
  for (t in 2:n) {
    S[k,t] <- S[k,t-1] * (1 + r * dt + sigma * sqrt(dt) * RN[k,t])
  }
}


## Calculation of price
for (k in 1:nn) {
  V[k] <- max(S[k,n] - K, 0)
}

MCprice <- mean(V)   # price obtained from MC simulation
```

```r
## Plots
# first 100 sample paths (gray)
plot(1:n, S[1,], type = "l", ylim = c(20, 80), col = "gray", xlab =
"Periods", ylab = "Stock price")
for (k in 2:100) {
  lines(1:n, S[k,], col = "gray")
}
# add averaged price of all sample paths (black)
lines(1:n, colMeans(S,2), col = "black")


## BSM formula
d1 <- (log(S0/K) + (r + sigma^2/2)*M) / (sigma * sqrt(M))
d2 <- d1 - sigma * sqrt(M)
BSMprice <- S0 * pnorm(d1) - K * exp(-r * M) * pnorm(d2)   # price
obtained from BSM formula
```

MC price = 4.7666,   BSM price = 4.8174