CMSC 5718 INTRODUCTION TO COMPUTATIONAL FINANCE

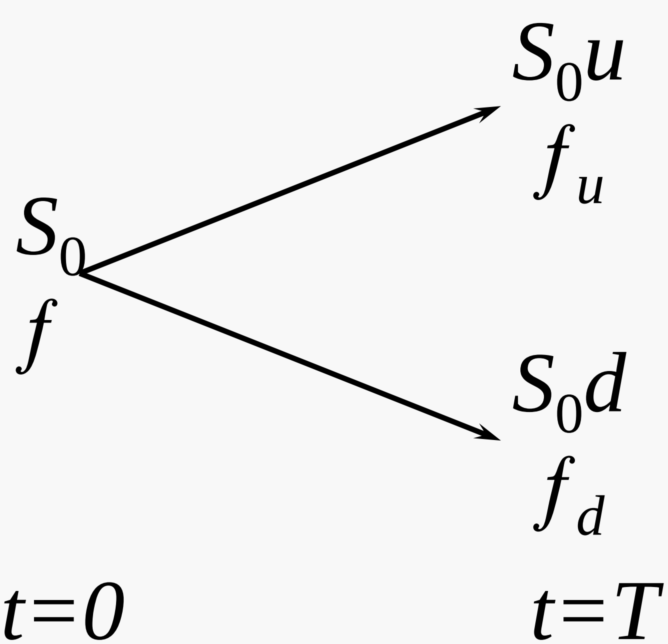
Lecture 11

Outline

- Using the pricing framework that we learned in lecture 8, how do we compute the price of a derivative contract?
- Two general pricing algorithms are introduced in this lecture:
  - Binomial pricing model
  - Monte Carlo methods
- Reference: Hull, 7th edition, Chapters 11, 19, 26; or 8th edition, Chapters 12, 20, 26
- **The contents in this session will not appear in the final examination**

Binomial pricing model

- In this two-state example, a derivative lasts for time T and its value is dependent on the stock price



- S_0 is the initial stock price
- f is the value of the derivative today;
- u and d are the ratios of the upward and downward movements
- f_u and f_d are the values of the derivative after the u and d movements

Binomial pricing model

- Consider the portfolio that is long Δ shares and short 1 derivative

$$\begin{array}{l} \nearrow \; S_0u\Delta - f_u \\ \\ \searrow \; S_0d\Delta - f_d \end{array}$$

- The portfolio is riskless when $S_0u\Delta - f_u = S_0d\Delta - f_d$
  - i.e. no matter how the stock moves, the value of the portfolio is the same

$$\Delta = \frac{f_u - f_d}{S_0u - S_0d}$$

Binomial pricing model

- Value of the portfolio at time T is $S_0 u\Delta - f_u$
- Value of the portfolio today is $(S_0 u\Delta - f_u)e^{-rT}$
  - e^{-rT} is the discounting factor to today
- Another expression for the portfolio value today is $S_0\Delta - f$
- Hence $f = S_0\Delta - (S_0 u\Delta - f_u)e^{-rT}$
- Substituting for Δ we obtain

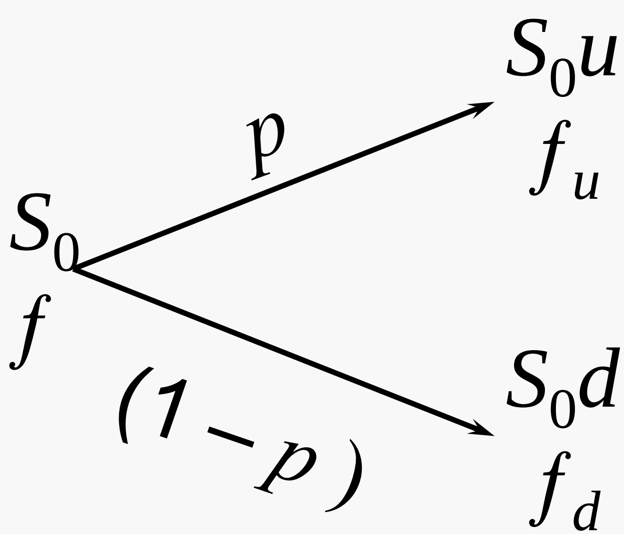
$$f = [\, p f_u + (1-p) f_d \,]e^{-rT}$$

where

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

Binomial pricing model

- The value of a derivative is then its expected payoff in a risk-neutral world discounted at the risk-free rate



- Note that the subjective probability about the upward or downward of the asset price does not appear in the formula for calculating the value of f

Binomial pricing model

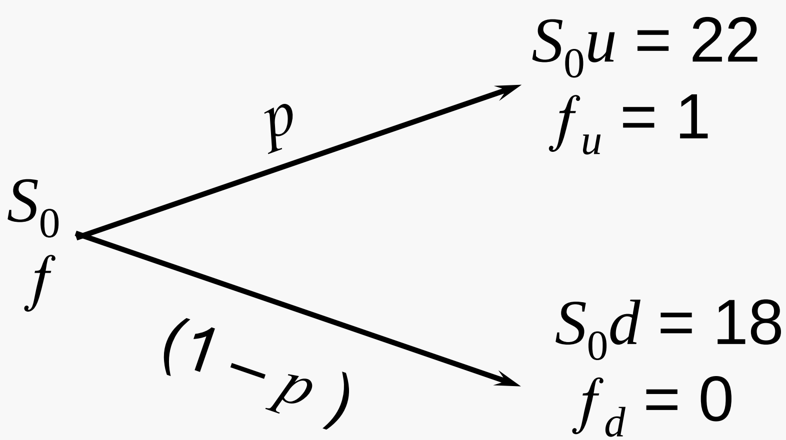
- It is natural to interpret p and $1-p$ as probabilities of up and down movements
  - If we select u and d such that $u > e^{rT} > d$, it can easily be shown that $0 < p < 1$

- We can also demonstrate that the expected rate of return of the asset price equals the risk free rate

$$[\, pS_0u + (1-p)\, S_0d \,] = S_0e^{rT}$$

- This is known as using risk-neutral valuation
  - p is a pseudo-probability, not a real assessment

Numerical example



$S_0 = 20$
$r = 12\%$
$T = 0.25$

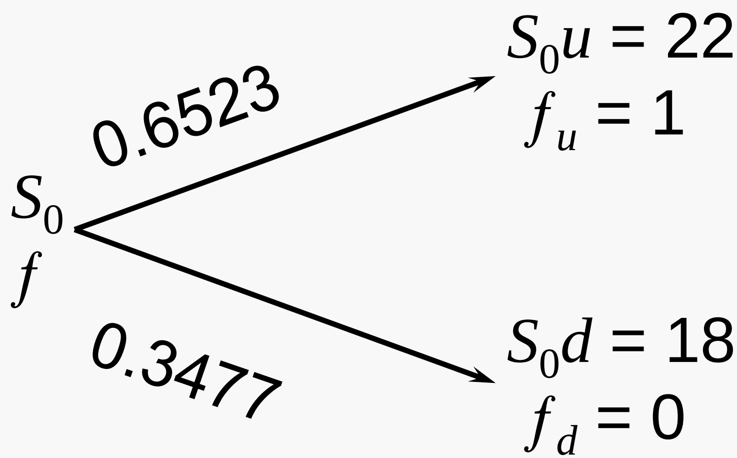
- Since p is the probability that gives a return on the stock equal to the risk-free rate, we can find it from

  $20e^{0.12 \times 0.25} = 22p + 18(1-p)$

  which gives $p = 0.6523$
- Alternatively, we can use the formula

$$p = \frac{e^{rT} - d}{u - d} = \frac{e^{0.12 \times 0.25} - 0.9}{1.1 - 0.9} = 0.6523$$

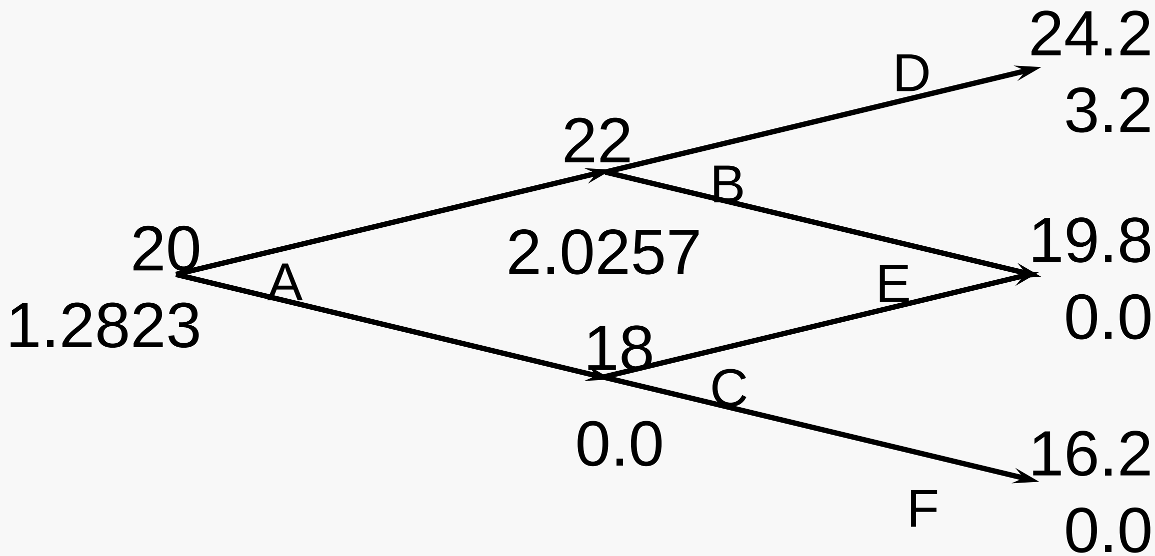
Valuing the Option Using Risk-Neutral Valuation



Using the parameters from the previous slide, the value of the option is

$$e^{-0.12\times0.25}(0.6523\times1 + 0.3477\times0) = 0.633$$

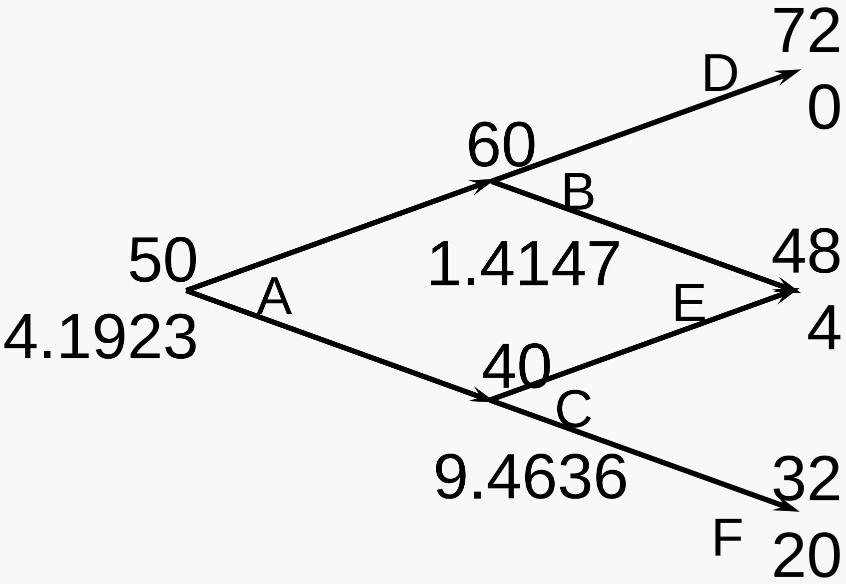
A two-step call option example

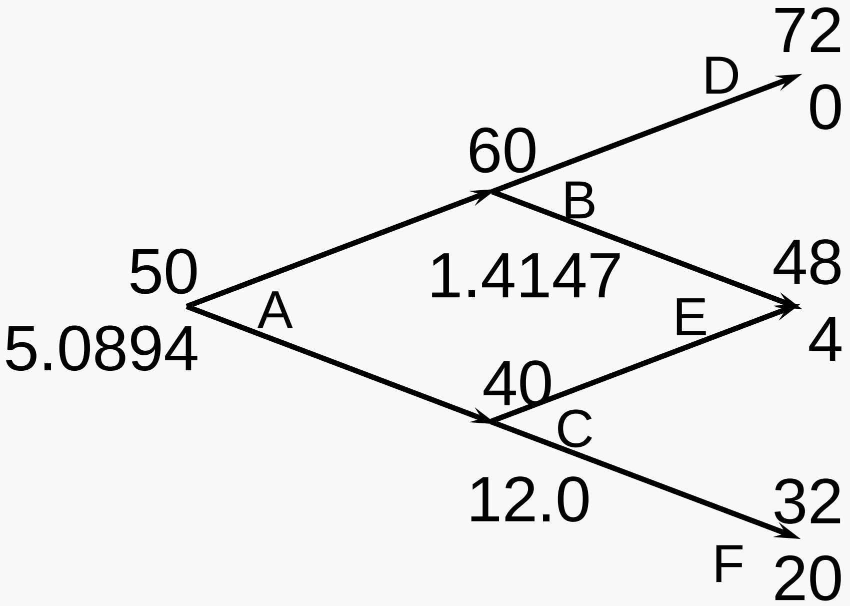


- We start at the last time step; each time step is 3 months
- Call option, K=21, r=12%
- Value at node B is
  $e^{-0.12 \times 0.25}(0.6523 \times 3.2 + 0.3477 \times 0) = 2.0257$
- Value at node A is
  $e^{-0.12 \times 0.25}(0.6523 \times 2.0257 + 0.3477 \times 0) = 1.2823$

Put option example

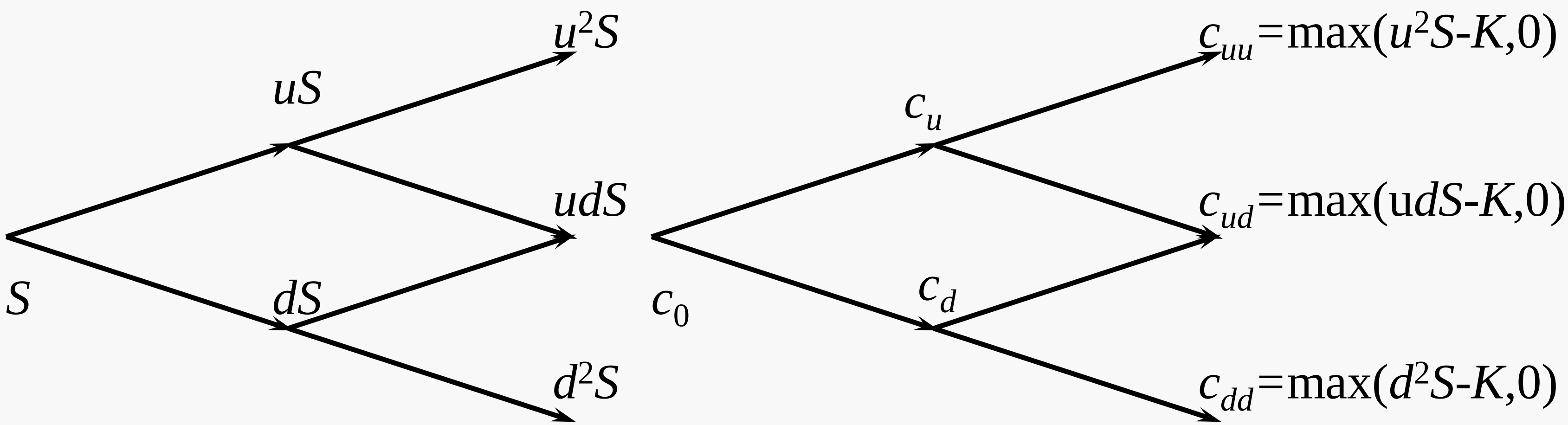
K = 52, time step =1yr, r = 5





- Left diagram represents the European put, right diagram is an American put
- Note that in node C, the early exercise value replaces the continuation value

Multi-period extension



- c_{uu} represents the value of a call option (with strike K) with two consecutive upward moves of the asset price
- Let R = $\exp(r\Delta t)$; option price at time 0 is given by

$$c_0 = \frac{p^2 c_{uu} + 2p(1-p)c_{ud} + (1-p)^2 c_{dd}}{R^2}$$

Multi-period extension

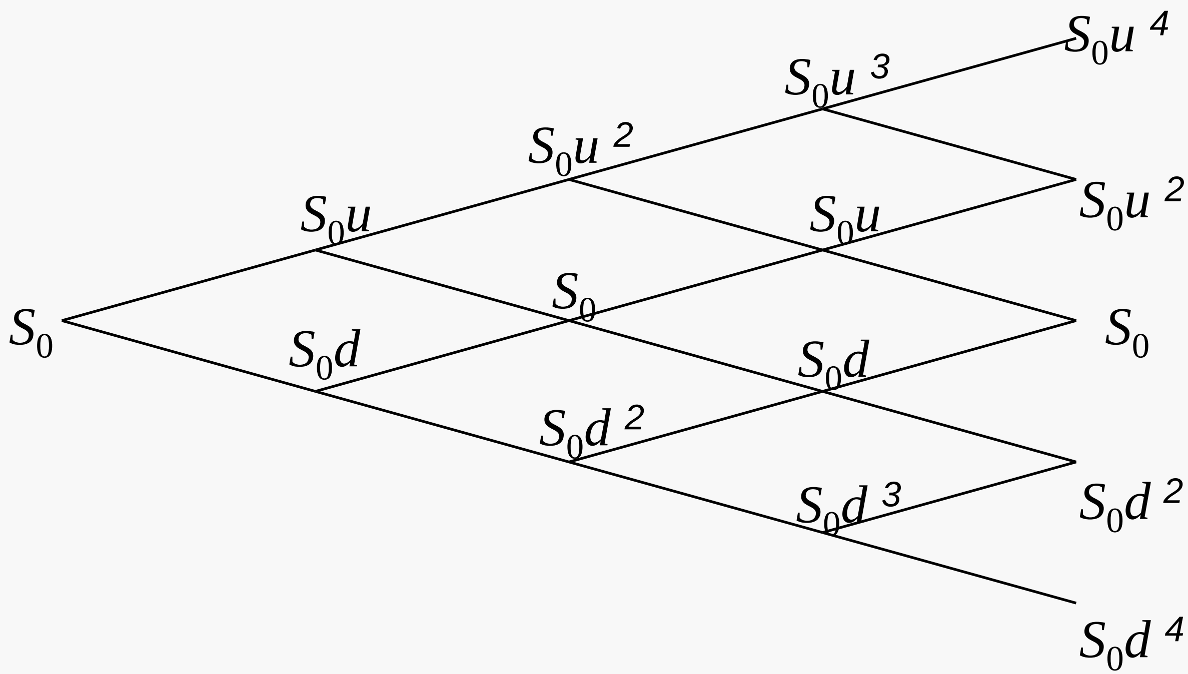
- For a tree with n binomial steps, the probability of having j upward moves and $n-j$ downward moves is:

$$probability = C_j^n p^j (1-p)^{n-j}, C_j^n = \frac{n!}{j!(n-j)!}$$

- The call value is thus:

$$c_0 = \frac{\sum_{j=0}^{n} C_j^n p^j (1-p)^{n-j} \max(u^j d^{n-j} S - K, 0)}{R^n}$$

Tree Configuration



- The tree is known as "recombining", meaning that $S_0u_id_{i+1}$ and $S_0d_iu_{i+1}$ will lead to the same node value
- Number of nodes in an n-step tree is $(n+1)(n+2)/2$
  - The corresponding figure for a non-recombining tree is $2^{n+1}-1$

Different types of underlyings

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

- $R = e^{r\Delta t}$ for a nondividend paying stock
- $R = e^{(r-q)\Delta t}$ for a stock index where q is the dividend yield on the index
- $R = e^{(r-r_f)\Delta t}$ for a currency where r_f is the foreign risk-free rate
- $R = 1$ for a futures contract

Tree Parameters for asset paying a dividend yield of q

Parameters p, u, and d are chosen so that the tree gives correct values for the mean & variance of the stock price changes in a risk-neutral world

Mean: $R = pu + (1-p)d$

Variance:

$$R^2(e^{\sigma^2\Delta t} - 1) = pu^2 + (1-p)d^2 - R^2$$

A further condition often imposed is $u = 1/d$

Tree Parameters for asset paying a dividend yield of q

$$u = \frac{1}{d} = \frac{R^2 e^{\sigma^2 \Delta t} + 1 + \sqrt{(R^2 e^{\sigma^2 \Delta t} + 1)^2 - 4R^2}}{2R}$$

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

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

- When Δt is small, we can approximately u by a simpler expression without sacrificing the order of accuracy, which is of the order of $O(\Delta t)$

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

Dynamic programming procedure

- Backward induction
  - We know the value of the option at the final nodes
  - We work back through the tree using risk-neutral valuation to calculate the value of the option at each node, testing for early exercise when appropriate

- Formally, at each tree node, compute

  $V = \max(V_{cont}, h(S))$, where
  - V_{cont} is the continuation value, being the value obtained from discounting through the risk neutral probability
  - $h(S)$ is the value obtained through immediate exercise

American put option example

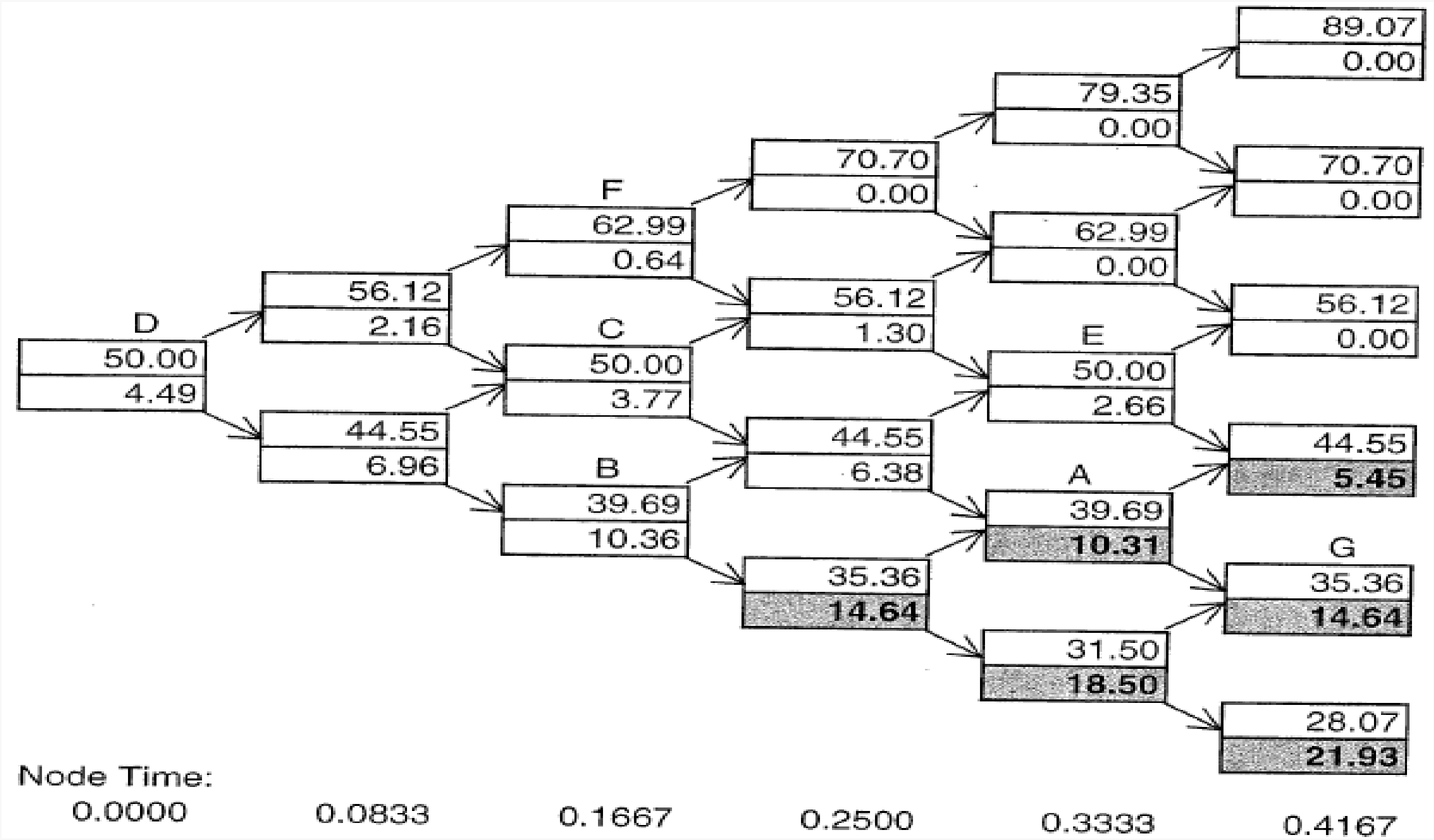
- S_0 = 50; K = 50; r =10%; σ = 40%;
  T = 5 months = 0.4167;
  Δt = 1 month = 0.0833

- The parameters imply

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

$$p = \frac{R-d}{u-d} = 0.5073, \quad 1-p = 0.4927.$$

American put option example



- At each node, the upper value is the underlying asset price, the lower value is the option price, and the shaded areas represent the cases where the option is exercised

American put option example

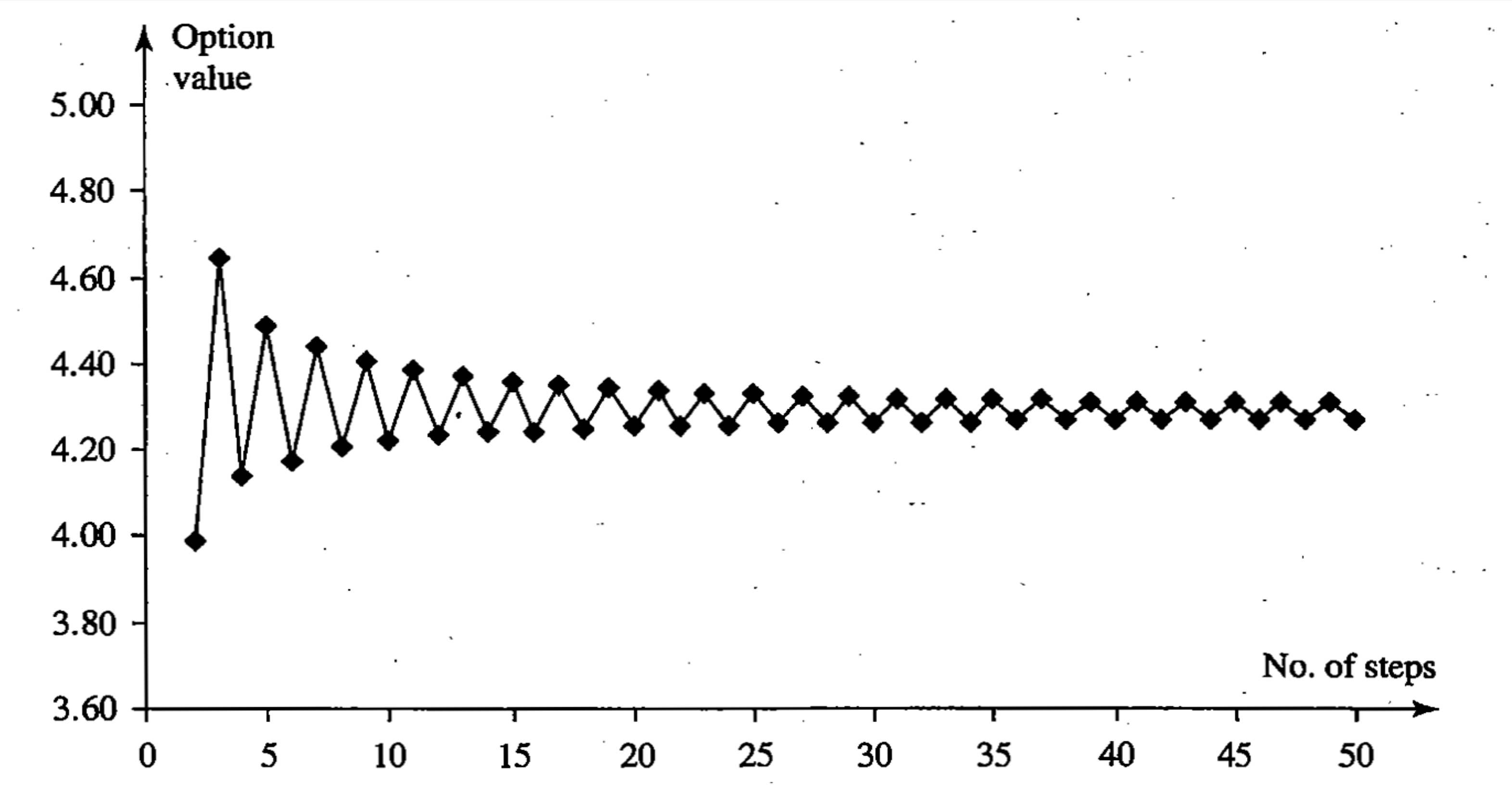
- Calculation of stock price at each node $S_0 u^j d^{n-j}$
  - E.g. at node A (n=4, j=1), stock price = 50 x 1.1224 x 0.8909^3 = \$39.69

- Calculation of the option prices at the final nodes: max ($K - S_T$, 0)
  - E.g. at node G, the option price is 50.00 – 35.36 = \$14.64

American put option example

- Backward induction: calculate the continuation value and the exercise value at each node; sometimes it should be exercised and sometimes not
  - At node E, continuation value is (0.5073 x 0 + 0.4927 x 5.45) $e^{-0.10\times0.0833}$ = 2.66, whereas the exercise value is 0 (strike 50, asset price 50); therefore the option value is 2.66
  - At node A, continuation value is (0.5073 x 5.45 + 0.4927 x 14.64) $e^{-0.10\times0.0833}$ = 9.90, exercise value is max(50 – 39.69, 0) = 10.31; therefore the option value is 10.31 (higher than 9.90)
  - At node B, continuation value is (0.5073 x 6.38 + 0.4927 x 14.64) $e^{-0.10\times0.0833}$ = 10.36, exercise value is max(50 – 39.69, 0) = 10.31; therefore the option value is 10.36
- The price of the option is given by the value at the initial node, i.e. $4.49

Convergence of option value in a binomial tree



- In practice, using more time steps will lead to better accuracy; it is common to use at least 100 steps in a binomial tree

Binomial tree for stock paying known dividends

- Procedure:
  - Draw the tree for the stock price less the present value of the dividends
  - Create a new tree by adding the present value of the dividends at each node

- This ensures that the tree recombines and makes assumptions similar to those when the Black-Scholes model is used

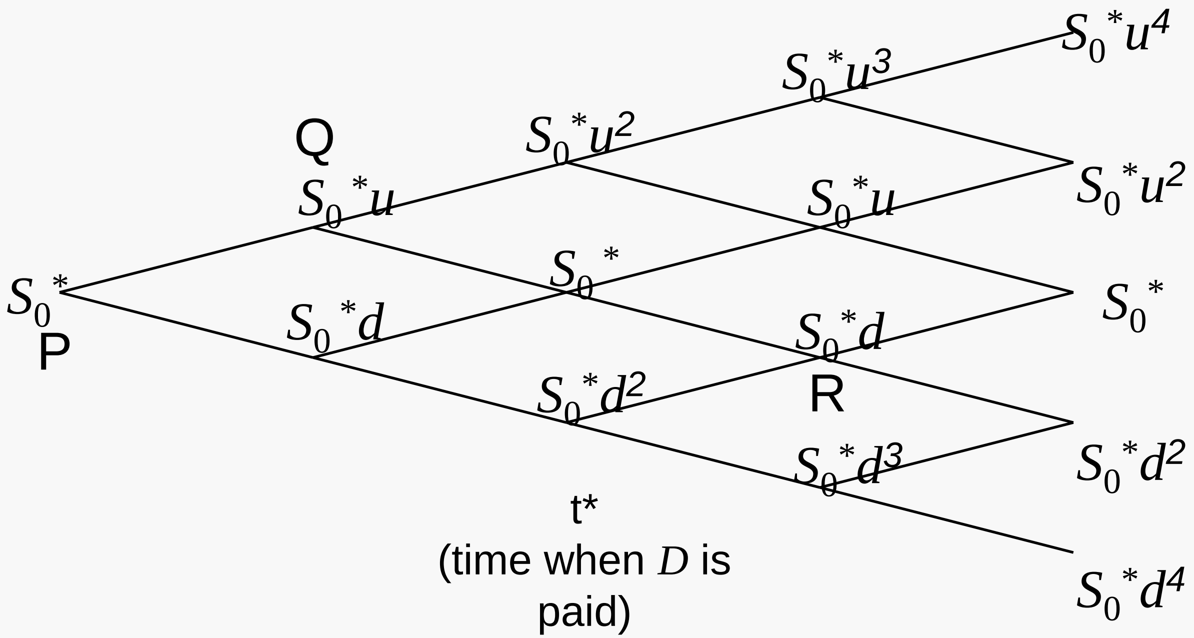
Binomial tree for stock paying known dividends

- Suppose there is a known dividend D and the dividend date is at time t^*
- The asset price S is split into 2 components, a risky part S^* and the known dividend D

$$S_t^* = \begin{cases} S_t - De^{-r(t^* - t)}, t \le t^* \\ S_t, t > t^* \end{cases}$$

- Instead of constructing a tree for S, a tree for S^* is used, where σ^* is the volatility of S^*
  - A recombining tree structure for S^* is thus ensured
  - It is often assumed that the volatilities of S and S^* are identical, although theoretical adjustments can be introduced

Binomial tree for stock paying known dividends



- The asset values at nodes P, Q, R are $S_0{*}{+}De^{-2r\Delta t}$, $S_0{*}u{+}De^{-r\Delta t}$, and $S_0{*}d$ respectively

American put option example

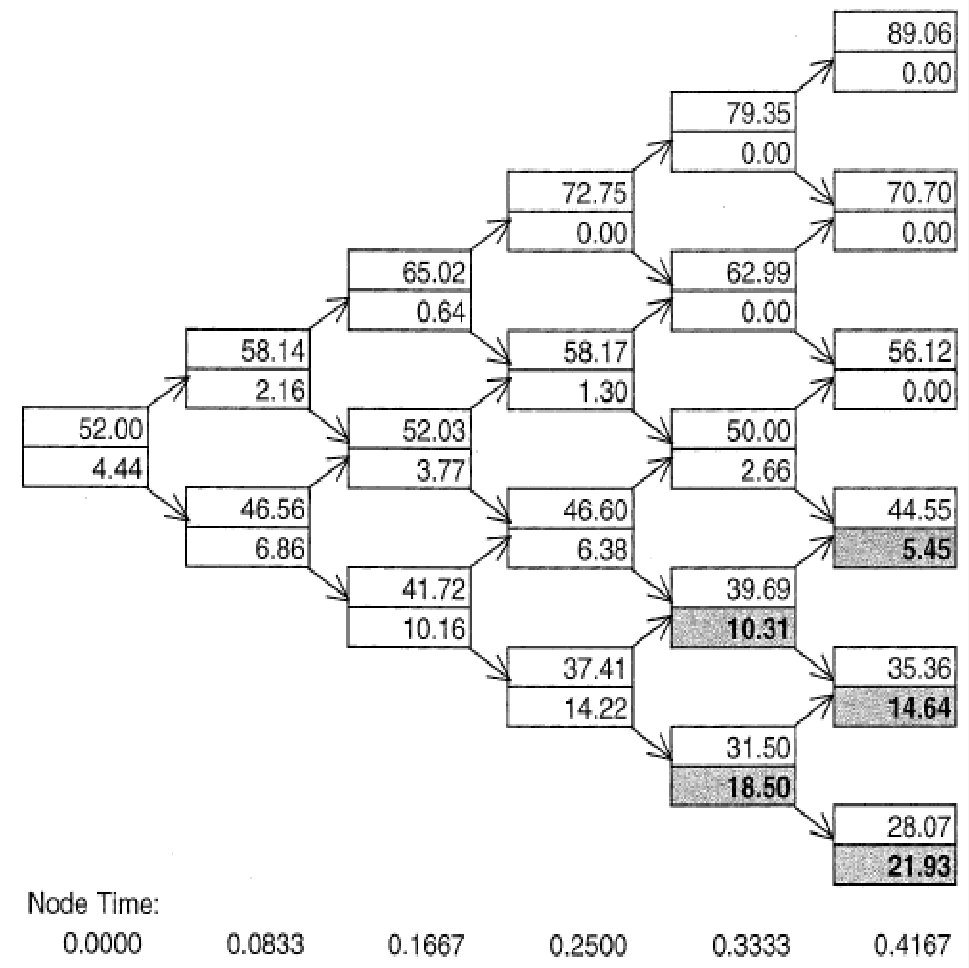
- S_0 = 52; K = 50; r =10%; σ = 40%; D = 2.06, at 3.5 months
  T = 5 months = 0.4167;
  Δt = 1 month = 0.0833
- The parameters imply

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

$$p = \frac{R-d}{u-d} = 0.5073, \quad 1-p = 0.4927.$$

- We construct a tree on S^*; assume that the volatility of S^* is the same as the volatility of S, which is 40%
- The present value of the dividend at time 0 is
  2.06 $e^{-0.10 \times 0.0833 \times 3.5}$ = 2.00, which means that the initial value of S^* is 52–2 = 50

American put option example



- The tree is based on S^*
- E.g. in the upper box at time 0.0833, the stock price is not calculated from 52 x 1.1224 = 58.36, but comes from 50 x 1.1224 + $2.06e^{-0.10\text{x}0.0833\text{x}2.5}$ = 58.14
- Option price is found to be 4.44

Alternative binomial tree geometry

- The parameter choices described earlier were suggested by Cox, Ross and Rubinstein (1979), which are popular among practitioners
- Other ways of constructing the tree are possible; e.g. Jarrow and Rudd (1983) suggest that, instead of setting $u = 1/d$, we can set each of the two probabilities to 0.5 and

$$u = e^{(r-q-\sigma^2/2)\Delta t+\sigma\sqrt{\Delta t}}$$

$$d = e^{(r-q-\sigma^2/2)\Delta t-\sigma\sqrt{\Delta t}}$$

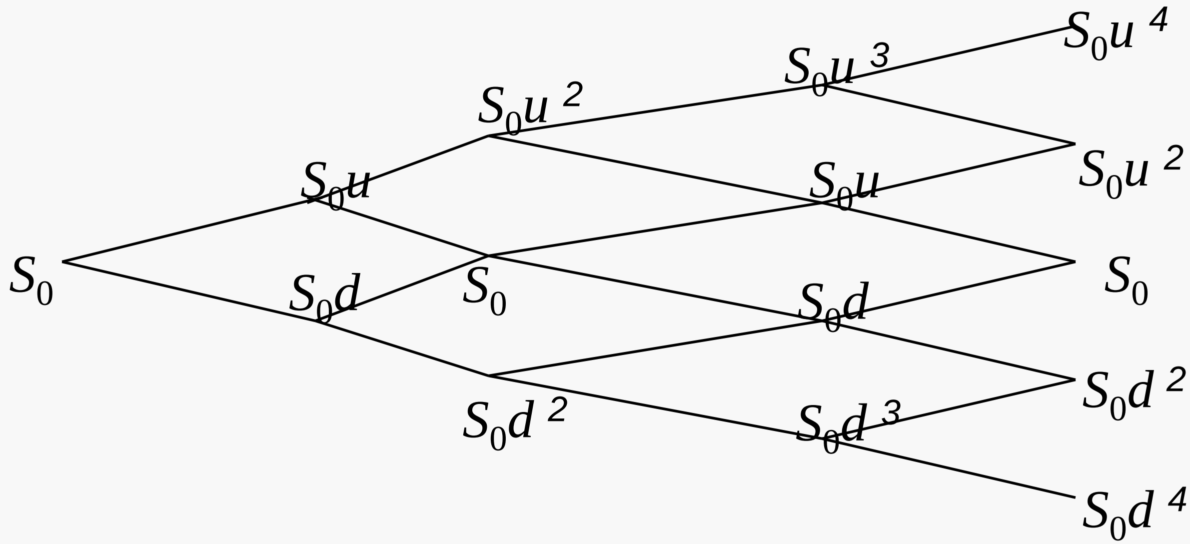
Extensions of Tree Approach

- Time dependent interest rates and volatilities
  - i.e. use $r(t)$ and $\sigma(t)$ in constructing the tree

- Pricing barrier options with trees
  - Tricks adopted to make the algorithm converge faster, e.g. by placing the nodes right next to the barrier level

- Pricing path-dependent options with trees
  - E.g. lookback options, by storing one or more additional values at each tree node representing the path-dependent state variables
    - First proposed by Hull and White (1993); sometimes known as the forward shooting grid algorithm

Time Dependent Parameters in a Binomial Tree



- Making r or q a function of time does not affect the geometry of the tree; the probabilities on the tree become functions of time
- We can make σ a function of time by making the lengths of the time steps inversely proportional to the variance rate
  - The tree will recombine as long as $\sigma_i^2 \Delta t_i$ is constant

Theoretical basis of Monte Carlo method

- Strong law of large numbers: The arithmetic mean of the realizations of X_{i} tends to its expectation μ

$$\mu = E(X_i)$$

$$\frac{1}{n}\sum_{i=1}^{n} X_i \to \mu \ \text{ as } \ n \to \infty$$

- Central Limit Theorem: the normalized error tends to the standard normal distribution

$$\frac{\sum_{i=1}^{N} X_i - N\mu}{\sqrt{N}\sigma} \to \Phi(0,1) \ \text{ as } \ N \to \infty$$

How do we get random numbers?

- The Monte Carlo method is a numerical procedure for estimating the expected value of a random variable, thus the quality of the random number used is very important

- In the past, random number generators with short periods can often give rise to erroneous conclusions as the period of the numbers generated are not well understood
  - Although the test is run many times, the period can be too short

Some quality criteria for random number generators

- Generate numbers that are evenly distributed
- Speed and memory requirements
- Generate independent, identically distributed (i.i.d.) samples
- Reproducible: useful for debugging purposes
- Portability – one that is not machine dependent
- Structure of the random points – not all lie in the same hyperplane

- Source: Ralf Korn, Elke Korn, and Gerald Kroisandt, Monte Carlo Methods and Models in Finance and Insurance, Chapman, 2009, pp.6-7

Random number generator example

- The earliest and easiest type of random number generator is called the Linear Congruential Generator (LCG), which has the form

$$s_{n+1} = (as_n + c) \bmod m,\ n \in \mathrm{N}$$

$$u_n = \frac{s_n}{m} \text{ where } u_n \text{ lies between } [0,1)$$

- In a version proposed by L'Ecuyer, $m = 2^{31}-249$, $a = 40692$, $c = 0$; this generator can generate numbers with a period of $2^{31}-250$
- A state-of-the-art generator was proposed by Matsumoto and Nishimura in 1998, known as Mersenne Twister MT19937; the period of this generator is tremendously large, equal to $2^{19937} - 1$, and there is minimal serial correlation up to a dimension of 625
  - It is a standard generator in packages such as Matlab and S-plus

Euler-Maruyama scheme for stochastic differential equations

- Assume that a state variable $X(t)$ follows the stochastic differential equation (SDE):

$$dX(t) = a(t, X(t))dt + \sigma(t, X(t))dW(t)$$

- The SDE can be approximated by the following algorithm:

1) Let $\Delta t = T / N$, set $Y_N(0) = X(0) = x_0$

2) For $j = 0$ to $N-1$ do

(a) Simulate a standard normally distributed variable Z_j

(b) Set $\Delta W(j\Delta t) = \sqrt{\Delta t} Z_j$ and

$$Y_N((j+1)\Delta t) = Y_N(j\Delta t) + a(j\Delta t, Y_N(j\Delta t))\Delta t + \sigma(j\Delta t, Y_N(j\Delta t))\Delta W(j\Delta t)$$

Applications of Monte Carlo simulation

- Compared with the other numerical methods, Monte Carlo simulation can deal with
  - path dependent options
  - options dependent on several underlying state variables (no curse of dimensionality)
  - options with complex payoffs
  - Complex underlying stochastic processes

- If there is only one state variable, Monte Carlo methods can be time consuming, and the standard method cannot easily deal with American-style options

Remarks on computation

- Number of time steps required depend on whether there are intermediate cash flows or event dates

- Aim to perform the minimum number of computations within the main calculation routine
  - E.g. if we need to price European options, the discounting factor can be applied to the average payoff instead of being applied per path

Crude Monte Carlo method

- Example: European stock options
  - The required calculation is $e^{-rT} E\left[\max(S_T - K, 0)\right]$

- Monte Carlo simulation involves the following steps:
  - Simulate 1 path for the stock price in a risk neutral world
  - Calculate the payoff from the stock option
  - Repeat these steps many times to get many sample payoffs
  - Calculate mean payoff
  - Discount mean payoff at risk free rate

Stock price movements

- In a risk neutral world the process for a stock price is (assume dividend = 0)

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

- We can simulate a path by choosing time steps of length Δt and using the discrete version of this

$$\Delta S = rS\,\Delta t + \sigma S\,\varepsilon\sqrt{\Delta t}$$

where ε is a random sample from $\phi(0,1)$

Example 1: stock price path

Stock price at start of period	Random sample for ϵ	Change in stock price during period
100.00	0.52	2.45
102.45	1.44	6.43
108.88	−0.86	−3.58
105.30	1.46	6.70
112.00	−0.69	−2.89
109.11	−0.74	−3.04
106.06	0.21	1.23
107.30	−1.10	−4.60
102.69	0.73	3.41
106.11	1.16	5.43
111.54	2.56	12.20

$$r = 15\%,\ \sigma = 30\%,\ \Delta t = 1\,\text{week} = 0.0192$$

$$\frac{\Delta S}{S} = 0.15\Delta t + 0.30\varepsilon\sqrt{\Delta t}$$

$$\Rightarrow \Delta S = 0.00288S + 0.0416S\varepsilon$$

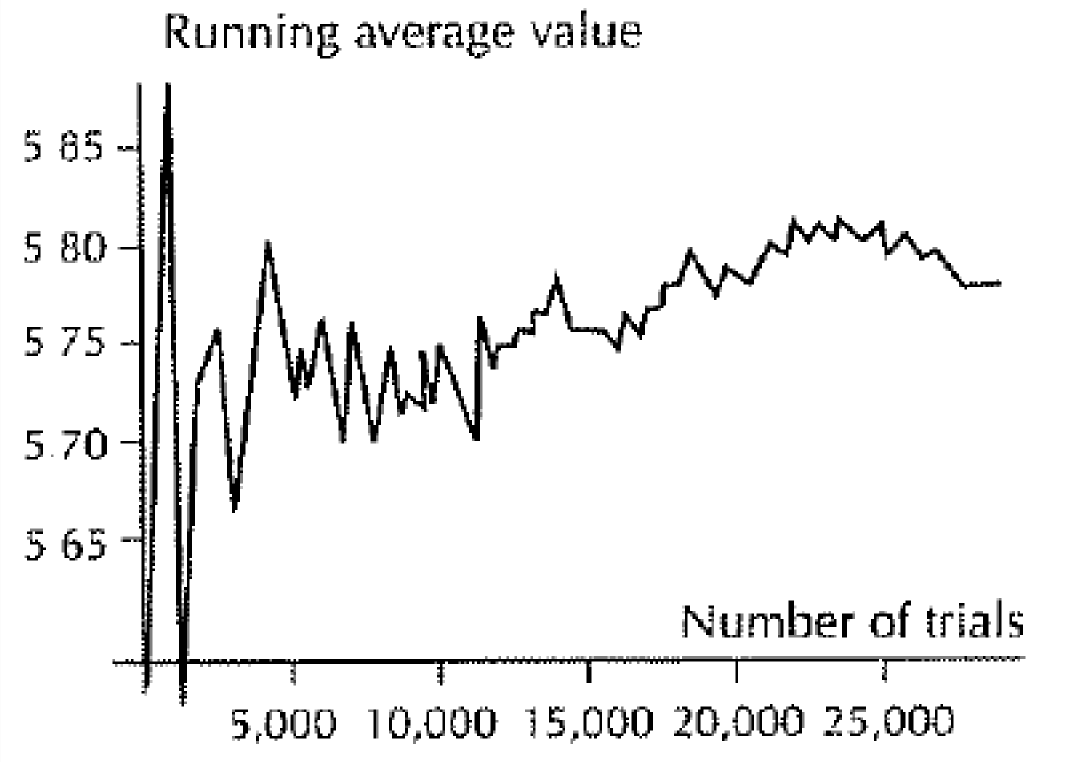
- At the start of the period, S=100
- In the next period,

  ΔS=0.00288x100

  +0.0416x100x0.52

  = 2.45

Example 2: a test of the Black-Scholes formula



- European call option, S_0=\$62, K=\$60, σ=20%, r=12%, T=5 months = 0.417 years divided into 80 time intervals, Black-Scholes price = \$5.80

A more accurate approach

- When possible, it is more accurate to use an exact solution of the SDE rather than relying on discretization
- In the standard dynamics above, we can use the following:

$$d \ln S = \left(r - \sigma^2/2\right)dt + \sigma\, dz$$

The discrete version of this is

$$\ln S(t+\Delta t) - \ln S(t) = \left(r - \sigma^2/2\right)\Delta t + \sigma\varepsilon\sqrt{\Delta t}$$

or

$$S(t+\Delta t) = S(t)\, e^{\left(r-\sigma^2/2\right)\Delta t + \sigma\varepsilon\sqrt{\Delta t}}$$

Example 2: another test of the Black-Scholes formula

	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	BS 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					

- If we have constant parameters, the solution to the SDE can directly be written as:

$$S(T) = S(0)\exp\left[\left(r - \frac{\sigma^2}{2}\right)T + \sigma\varepsilon\sqrt{T}\right]$$

- In the above example, we only simulate 1000 times, where each path has only 1 time step; column B represents the discounted payoff given the simulated stock price at column A

Example 3: the Heston model

- Refer to the notes of lecture 8 for the Heston model (assume q=0)
- The following is based on the Euler-Maruyama scheme

$$1) \text{ Let } \Delta t = T/N, \text{ set } V(0) = V_0, S(0) = S_0$$

$$2) \text{ For } j = 0 \text{ to } N-1 \text{ do}$$

(a) Simulate two standard normally distributed variables Y and Z

(b) Set $W = \rho Z + \sqrt{1-\rho^2}\,Y$

(c) Update the volatility

$$V(j\Delta t) = V((j-1)\Delta t) + a(V_L - V((j-1)\Delta t))\Delta t + \xi\sqrt{V((j-1)\Delta t))}\sqrt{\Delta t}W$$

(d) Update the log-stock price $X(t) = \ln(S(t))$

$$X(j\Delta t) = X((j-1)\Delta t) + \left(r - \frac{1}{2}V((j-1)\Delta t)\right)\Delta t + \sqrt{V((j-1)\Delta t))}\sqrt{\Delta t}Z$$

- Note that there is one potential problem – step (c) can produce a negative number; special tricks will be required to resolve this issue

Variance reduction techniques

- The naïve Monte Carlo method may only converge slowly to a correct answer

- Some techniques have been applied to speed up the convergence, examples include:
  - Antithetic variate
  - Control variate
  - Stratified sampling
  - Importance sampling
  - Quasi-random sequences

Antithetic variate

- Variance reduction is achieved by introducing symmetry
- If X is a random variable uniformly distributed on [0,1], the crude Monte Carlo estimate is:

$$\overline{f}(X) = \frac{1}{N}\sum_{i=1}^{N} f(X_i)$$

- The antithetic estimator is given by

$$\overline{f}_{anti}(X) = \frac{1}{2}\left(\frac{1}{N}\sum_{i=1}^{N} f(X_i) + \frac{1}{N}\sum_{i=1}^{N} f(1-X_i)\right)$$

- Note that as both $f(X_i)$ and $f(1-X_i)$ are unbiased estimators of $f(X)$ the antithetic estimate is also unbiased

Antithetic variate

- If the variance of $f(X)$ with N samples = σ^2, the variance of the antithetic estimator is:

$$\mathrm{var}(\overline{f}(X)+\overline{f}(1-X)) = 2\sigma^2 + 2\,\mathrm{cov}(\overline{f}(X), \overline{f}(1-X))$$

- i.e. there is a reduction in variance if $f(X)$ and $f(1-X_\mathrm{i})$ are negatively correlated
  - Heuristically this is reasonable as errors from the first simulation can be negated by the errors of the opposite simulation
- Furthermore, the random inputs obtained from the collection of the antithetic pair are more evenly distributed than a collection of $2N$ independent samples
- However, the improvement in convergence is not too significant

Control variate

- We want to price an option X with Monte Carlo, whereas the price of a similar option Y can be obtained accurately (maybe through an analytical formula)
- Use the following notation

$$V_X, V_Y : \text{true values of options X and Y}$$

$$\overline{V_X}, \overline{V_Y} : \text{estimated values of the options using Monte Carlo}$$

- The control variate method assumes that

$$V_X - \overline{V_X} \approx V_Y - \overline{V_Y} \Rightarrow \overline{V_X^{CV}} = \overline{V_X} + (V_Y - \overline{V_Y})$$

so we can use the error in the control variate to provide a better estimate of V_X

Control variate

- The variance of the estimator can easily be calculated:

$$\overline{V_X^{CV}} = \overline{V_X} - \overline{V_Y} + V_Y$$

$$\mathrm{var}(\overline{V_X^{CV}}) = \mathrm{var}(\overline{V_X}) + \mathrm{var}(\overline{V_Y}) - 2\,\mathrm{cov}(\overline{V_X}, \overline{V_Y})$$

  - V_Y is known (usually from analytical formula)

- Variance reduction is achieved if $\mathrm{var}(\overline{V_Y}) < 2\,\mathrm{cov}(\overline{V_X}, \overline{V_Y})$
  - This is true if Y is such chosen that it is closely related to X

Control variate

- Some examples of control variates include:
  - Use geometric averaging Asian option as the control variate for arithmetic Asian option
  - Similarly, use geometric averaging basket option as the control variate for arithmetic basket option
  - Use a constant volatility model as the control variate for the stochastic volatility model

Control variate example

- To calculate the price of an Asian option based on arithmetic average = V_X
- Given:
  - Price of an Asian option based on the geometric average, calculated with an exact pricing formula = V_Y = 0.1368
  - Price of an Asian option based on the geometric average, calculated with Monte Carlo simulation = $\bar{V}_Y$ = 0.1507
  - Price of an Asian option based on the arithmetic average, calculated with Monte Carlo simulation = $\bar{V}_X$ = 0.1335
  - Variance of the Asian option based on geometric average, calculated with Monte Carlo simulation = $\mathrm{var}(\bar{V}_Y)$ = 0.000144
- Results:
  - Estimate of V_X = 0.1335 – 0.1507 + 0.1368 = 0.1196
  - Variance reduction is achieved if covariance between covariance between $\bar{V}_X$ and $\bar{V}_Y$: $\mathrm{cov}(V_X, V_Y) > 0.000144/2 = 0.000072$

Control variate

- Better convergence is possible if we optimize the control variate via:

$$\overline{V_X^{\beta}} = \overline{V_X} + \beta(V_Y - \overline{V_Y})$$

$$\mathrm{var}(\overline{V_X^{\beta}}) = \mathrm{var}(\overline{V_X}) + \beta^2\,\mathrm{var}(\overline{V_Y}) - 2\beta\,\mathrm{cov}(\overline{V_X}, \overline{V_Y})$$

$$\mathrm{var}(\overline{V_X^{\beta}})\text{ is minimzed when }\beta^* = \frac{\mathrm{cov}(\overline{V_X}, \overline{V_Y})}{\mathrm{var}(\overline{V_Y})}\text{, so that}$$

$$\mathrm{var}(\overline{V_X^{\beta}})\Big|_{\beta=\beta^*} = \mathrm{var}(\overline{V_X}) - \frac{\mathrm{cov}(\overline{V_X}, \overline{V_Y})^2}{\mathrm{var}(\overline{V_Y})}$$

- The difficulty is that in general, the covariance term is not available, and it has to be estimated through regression

Stratified sampling

- Aim to sample in a small sub-population that mirrors the properties of the total population

- Divide the distribution of the random variable X into different parts, and apply Monte Carlo simulations within each part

- However, this method may not work well if the simulation is based on more than one underlying variable (the curse of dimensionality)

Importance sampling

- Main idea: find a distribution of the random variable that assigns a high probability to those values that are important for computing the expectation

- Popular methods to obtain importance sampling density include:
  - Shifting and/or scaling the density
  - Conditional sampling restricted to the important area

- However, this method relies on the fact that the characteristics of the distribution are well known before the simulation is performed

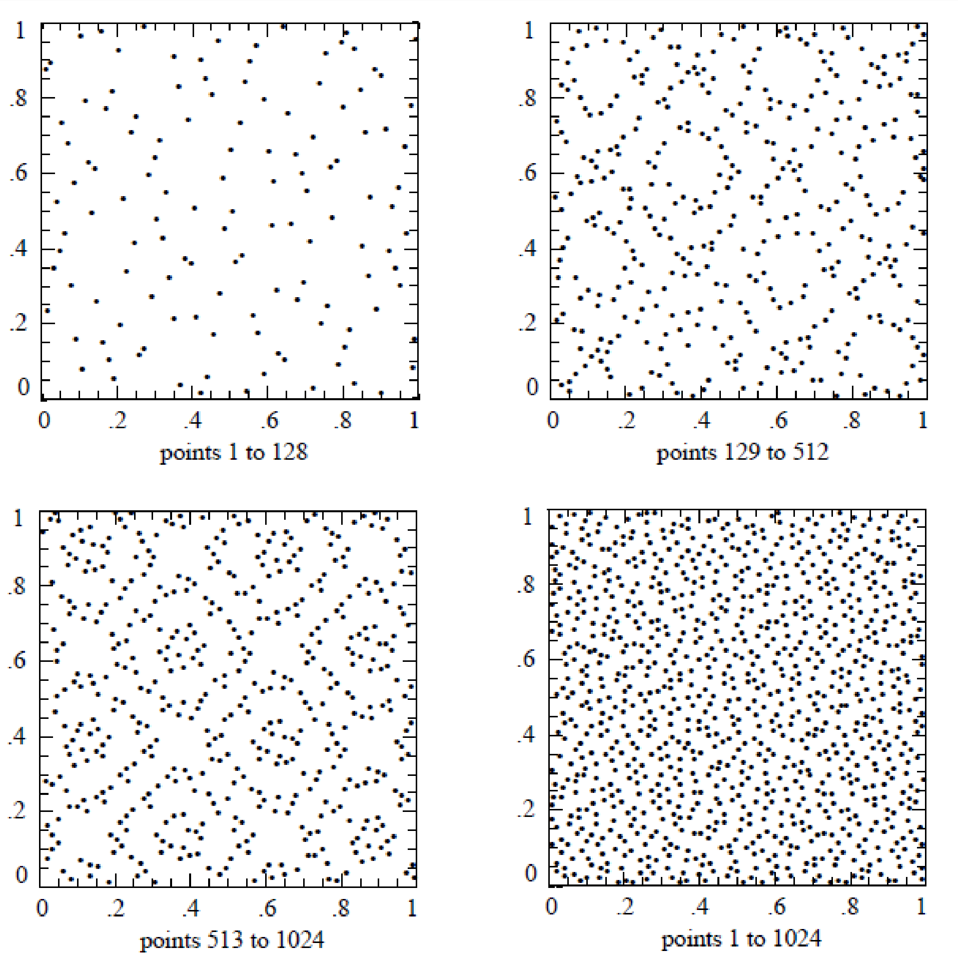
Quasi-random sequences

- Quasi-random sequences are completely deterministic (i.e. not random), but they are *incrementally* evenly distributed
  - These are clever ways to insert additional points in the sampling space without clustering
- It is not necessary to decide in advance how many points are needed – the sampling could continue "until" some convergence or termination criterion is met
- Examples of quasi-random sequences include Halton, Faure, Sobol, and Niederreiter
  - These are also known as low-discrepancy sequences

Quasi-random sequences

- Example: 1-dimensional Halton sequence (van der Corput sequence)
- The j^{th} number H_j in the sequence is obtained from:
  1) write j as a number in base b, where b is prime
  2) reverse the digits and put a radix (decimal) point in front of the number, convert back to base 10 and this is H_j
- E.g. j = 17 in base b =5 is 32; reverse the digits and put a decimal to become 0.23 base 5; convert back to base 10 via 2x1/5 + 3x1/25 = 0.52
- It is noted that as j increases by 1, the most significant digit in H_j is increased (by 1/5 in this example); this results in a kind of "maximally spread-out" order on each grid
- The first 29 numbers in this example are:
  - 0.2, 0.4, 0.6, 0.8, 0.04, 0.24, 0.44, 0.64, 0.84, 0.08, 0.28, 0.48, 0.68, 0.88, 0.12, 0.32, 0.52, 0.72, 0.92, 0.16, 0.36, 0.56, 0.76, 0.96, 0.008, 0.208, 0.408, 0.608, 0.808…

Quasi-random sequences



- The first 1024 points of a 2-dimensional Sobol sequence

- Reference: Press et al., Numerical Recipes in C, 2nd edition, p.310

Valuation of American options

- There used to be a general belief that simulation methods can only be used to price European style options, apparently due to the backward nature of the early exercise feature
  - There is no way to know whether exercise is optimal at a particular level of the state variable
- Algorithms dealing with this problem started to appear in the mid-1990s
- The problem is now considered to be mostly solved, and the most popular approach in recent years was proposed by Longstaff and Schwartz (2001), where basis functions and regression techniques are used to approximate the early exercise boundary
- Other parameterizations of this boundary are also available
- Note that these methods produce a lower bound on the price of the option, as the exercise policy can only be sub-optimal

Tree methods and Monte Carlo

- Tree methods are suited for American style payoffs due to backward induction
- However, they cannot easily accommodate multiple underlyings or path dependency
  - This is an intrinsic problem with the tree approach where no solution will be possible
- If an efficient Monte Carlo method can be found which can handle American options, this is considered as the favored outcome as it means that products which are path dependent, American, and depend on multiple underlyings can be managed