CISC 5352 FINANCIAL PROGRAMMING AND DATA ANALYTICS LECTURE NOTE (2)

Henry Han Ph.D.
Department of Computer and Information Science
Fordham University, New York NY 10023

Last class review

① Algo-trading and HFT concepts
② Model driven analytics to data-driven analytics
   ① BS-model
   ② Volatility
③ Python Basics Review
      ① Linux basics/ Emacs basics
      ② Python's input and output
      ③ sys.maxint, sys.float_info.max (this is related to your quiz 1)
      ④ Python's Math module
      ⑤ Python 2 and 3 differences
      ⑥ Control structures
      ⑦ Loop
      ⑧ Recursion/iteration (mentioned lightly)

Q1: sys.maxint in python is the INT_MAX in C++

Q1: sys.maxint in python is the INT_MAX in C++

No.

INT_MAX: 2^32-1(it also relies on machine)

sys.maxint is only meaningful python2

In python 3: sys.maxsize is just this number: 2^63-1 (LLONG_MAX in C++)

Python 3 has no limits for integers but it does not mean we should forget this!

It is important in programming!

It usually acts as ∞ in programming!

Note: it will cause a slow or even long computing because it may even cause a memory problem: it needs a large memory block.

Why does C++ have LL_ONG_MAX as its maximum integer but python and Java not? They can have arbitrary large numbers (decimals)?

LAB 0: Try this in your computer (Ipython)

① import sys
② sys.maxsize

③ sys.maxsize.bit_length()
④ 2**63 - sys.maxsize
⑤ t=sys.maxsize**10
⑥ t.bit_length()
⑦ M49=2**74207281-1 # The 49th Mersenne number found in 2016
⑧ M49.bit_length()

Q2: what are assumptions of BS model?

Q2: what are assumptions of BS model?

① The stock pays no dividends
② stock returns are normally distributed
③ Interest rates are constant and always known
④ There are no transaction costs (no commission)..
⑤ Options are only European options

Almost all these assumptions about assets do not match the real market situations. It leads to some inconsistency between BS model results and real world results.

Q3: We said BS model assume **Options are only European options: what are European options?**

$$\boxed{\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 - rf = 0}$$

① A partial differential equation describes option price over time t

② f(S; t) is a the price of a derivative (e.g. European option) at time t and its underlying asset price S

③ S: the current value of the underlying asset S .

④ Sigma: the constant volatility (the standard deviation of stock returns)

⑤ r: riskless interest rate

Note: it is called BS model or BSM model sometimes for three people's work: F. Black M. Scholes and R. Merton

How to 'understand' BS model?

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 - rf = 0$$

Option price's change w.r.t. time

Option price 's change w.r.t. underlying asset

The second level option price 's change w.r.t. underlying asset price

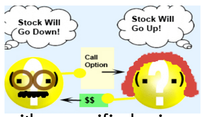
All weighted changes w.r.t. time and underlying asset price is the product of interest rate and option price

Option: a right to buy/sell an asset (e.g. stock)

A financial derivative to allow holders to have the right to buy or sell an asset at a specified price and pre-determined time.

The asset is also called underlying asset. It is usually a stock though not always

Call/put: right to buy/sell



A call option is the right to buy an asset with a specified price (**strike price: K**) at a pre-determined time T (expiration time or maturity time)

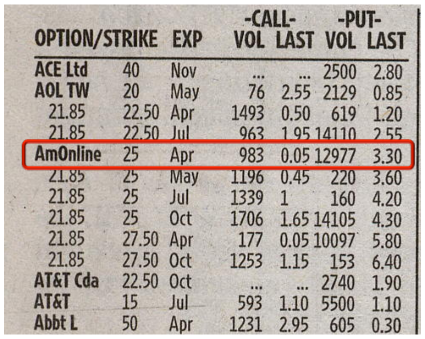
Payoff: max{S_T-K, 0}: S_T Price at time T

A put option is the right to sell an asset with a specified price (**strike price**) at a pre-determined time (expiration time or maturity time)

Payoff: max{K- S_T, 0}: S_T Price at time T

Almost all big business use options to hedge their cost

Options prices from WSJ April 10, 2002



OPTION/STRIKE		EXP	-CALL- VOL	-CALL- LAST	-PUT- VOL	-PUT- LAST
ACE Ltd	40	Nov	...	...	2500	2.80
AOL TW	20	May	76	2.55	2129	0.85
21.85	22.50	Apr	1493	0.50	619	1.20
21.85	22.50	Jul	963	1.95	14110	2.55
AmOnline	25	Apr	983	0.05	12977	3.30
21.85	25	May	1196	0.45	220	3.60
21.85	25	Jul	1339	1	160	4.20
21.85	25	Oct	1706	1.65	14105	4.30
21.85	27.50	Apr	177	0.05	10097	5.80
21.85	27.50	Oct	1253	1.15	153	6.40
AT&T Cda	22.50	Oct	...	...	2740	1.90
AT&T	15	Jul	593	1.10	5500	1.10
Abbt L	50	Apr	1231	2.95	605	0.30

A Call example

- **Suppose Google stock price is now $702.45 on 02/25/2016. John expects the price will go up. Then he buys a call option with strike price $750 and the call's expiration time is 08/25/2016**

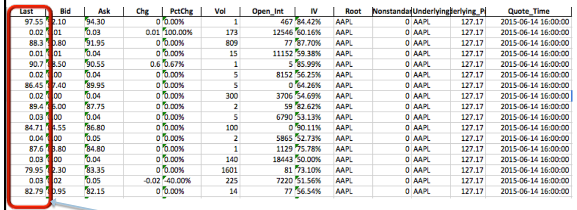
A Call example

- **Suppose Google stock price is now $702.45 on 02/25/2016. John expects the price will go up. Then he buys a call option with strike price $750 and the call's expiration time is 08/25/2016**
- **It means that he has the right to buy Google stock with $750 in a half year later.**

A Call example

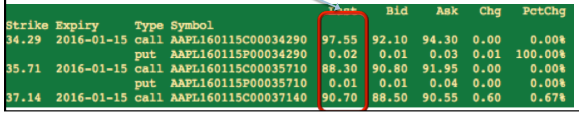
- **Suppose Google stock price is now $702.45 on 02/25/2016. John expects the price will go up. Then he buys a call option with strike price $750 and the call's expiration time is 08/25/2016**
- **It means that he has right to buy Google stock with $750 in a half year later.**
- **If Google stock price was $800 at that time, then he would exercise his call: he would earn $50 (payoff)**
- **If half year later Google stock price was $700, then he would not do anything. His call payoff is 0.**

All AAPL (Apple stock) Options with expiration date 2016/01/15 Retrieved on June 14, 2015 from Yahoo! Finance by using Python's analytics package: Pandas



Last	Bid	Ask	Chg	PctChg	Vol	Open_Int	IV	Root	Nonstandar	Underlying	erlying_P	Quote_Time
97.55	92.10	94.30	0	0.00%	1	467	84.42%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
0.02	0.01	0.03	0.01	100.00%	173	12546	60.16%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
88.3	90.80	91.95	0	0.00%	809	77	87.70%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
0.01	0.01	0.04	0	0.00%	15	11152	59.38%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
90.7	88.50	90.55	0.6	0.67%	1	5	85.99%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
0.02	0.00	0.04	0	0.00%	5	8152	56.25%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
86.45	87.40	89.95	0	0.00%	5	0	64.26%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
0.02	0.00	0.04	0	0.00%	300	3706	54.69%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
89.4	86.00	87.75	0	0.00%	2	59	82.62%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
0.03	0.00	0.04	0	0.00%	5	6790	53.13%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
84.71	84.55	86.80	0	0.00%	100	0	90.11%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
0.04	0.00	0.05	0	0.00%	2	5865	52.73%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
87.6	83.80	84.80	0	0.00%	1	1129	75.78%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
0.03	0.00	0.04	0	0.00%	140	18443	50.00%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
79.95	82.30	83.35	0	0.00%	1601	81	73.10%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
0.03	0.02	0.05	-0.02	-40.00%	225	7220	51.56%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00
82.79	80.95	82.15	0	0.00%	14	77	56.54%	AAPL	0	AAPL	127.17	2015-06-14 16:00:00

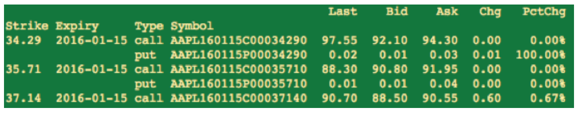
Final price



```
                                            Last    Bid    Ask   Chg   PctChg
Strike Expiry     Type Symbol
34.29  2016-01-15 call AAPL160115C00034290  97.55  92.10  94.30  0.00    0.00%
                  put  AAPL160115P00034290   0.02   0.01   0.03  0.01  100.00%
35.71  2016-01-15 call AAPL160115C00035710  88.30  90.80  91.95  0.00    0.00%
                  put  AAPL160115P00035710   0.01   0.01   0.04  0.00    0.00%
37.14  2016-01-15 call AAPL160115C00037140  90.70  88.50  90.55  0.60    0.67%
```

Two Basic types of options

European options – can be exercised only on exercise date (expiration time)



```
                                            Last    Bid    Ask   Chg   PctChg
Strike Expiry     Type Symbol
34.29  2016-01-15 call AAPL160115C00034290  97.55  92.10  94.30  0.00    0.00%
                  put  AAPL160115P00034290   0.02   0.01   0.03  0.01  100.00%
35.71  2016-01-15 call AAPL160115C00035710  88.30  90.80  91.95  0.00    0.00%
                  put  AAPL160115P00035710   0.01   0.01   0.04  0.00    0.00%
37.14  2016-01-15 call AAPL160115C00037140  90.70  88.50  90.55  0.60    0.67%
```

American options – can be exercised any time until the exercise date

How to do option pricing?

- **BS model firstly presented a formula to provide answer to European options.**

How to do option pricing?

- **BS model firstly presented a formula to provide answer to European options.**
- It was then extended to American options by other people's work (e.g., Barone-Adesi and Whaley's work in 1987) .
- A milestone model in Finance and business!

How to do option pricing?

- **BS model firstly presented a formula to provide answer to European options.**
- It was then extended to American options by other people's work (e.g., Barone-Adesi and Whaley's work in 1987) .
- A milestone model in Finance and business!
- Still widely used in practice though its assumptions are not matched with reality in market.

How to do option pricing?

- **BS model firstly presented a formula to provide answer to European options.**
- It was then extended to American options by other people's work (e.g., Barone-Adesi and Whaley's work in 1987) .
- A milestone model in Finance and business!
- Still widely used in practice though its assumptions are not matched with reality in market.
  - Almost all stock implied volatilities (IV) are computed by using BS model in market!
  - We will use this model and machine learning ways to handle I

How to price options?

Each option has the basic parameters:

1. Type: buy or sell
2. Stock initial value
3. Strike price
4. Life: time-to-maturity:
5. Market interest rate
6. Stock return in the future
7. Dividend paid by stock
8. Transaction fee
9. Other future contracts' price
10. …

How to value an European call option at time T?

1. Initial stock price $S_0 = 89.0$
2. Strike price of the European call option $K = 100.0$
3. Time-to-maturity $T = 0.5$ year
4. Constant, riskless short rate $r = 5\%$
5. Constant volatility $\sigma = 20\%$ ((**i.e., standard deviation of returns of the underlying property (e.g. stock)**)

BS model's answer: BS formula

Although most PDE equations don't have analytic solutions, BS model has its analytic solutions for European options

$$C = SN(d_1) - Ke^{-rT}N(d_2)$$

$$P = Ke^{-rT}N(-d_2) - SN(-d_1)$$

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

$$d_2 = d_1 - \sigma\sqrt{T}$$

C and P are just call and put option price given by the model.

Technically C→ C(S, t) ; P→P(S,t)

S: is the current price of the underlying asset (stock); K is the strike price

What's N(x)?

What's N(x)?

A cumulative distribution function for standard normal distribution:

$$N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt$$

Such an integration has no an analytic solution and must be done through numerical approaches.

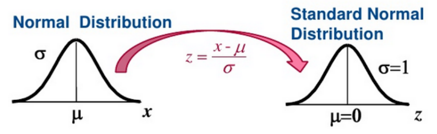
It is widely used in option pricing.

The classic method to conduct numerical integration is called Hart algorithm invented in1968

For CS undergraduate students only: what is a standard normal distribution?

A bunch of numbers that are subject to normal distribution but with zero mean and standard deviation 1

It is prototype of a normal distribution



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

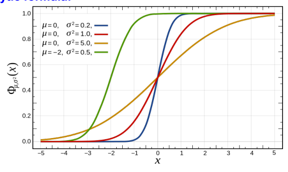
For CS undergraduate students only: what's cumulative distribution function CDF?

The function to compute probability. The probability is an area under the CDF (curve)

$$\boxed{N(x)} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt$$

CDF(an integration of pdf)
Not all integrations can have an analytic formula!

Probability density function (PDF) usually known



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

Cumulative Distribution Function for the normal distribution

Hart algorithm: three cases

- Hart gave the following approximation for this cumulative distribution function (CDF) $N(x)$

$$N(x) = \left\{ \begin{array}{c} e^{-t^2/2}\frac{A}{B}, \text{when } t < t_1 \\ e^{-t^2/2}\frac{1}{t_3 C}\text{when } t_1 \le t \le t_2 \\ 0, when t > t_2 \end{array} \right)$$

where $t = |x|$ and $t_1 = 7.07106781186547$, $t_2 = 37$, $t_3 = 2.506628274631$

How to compute A, B, C?

We only need to consider when $x < 0$ for convenience of computing. When $x > 0$, we use basic property of a normal distribution and get $N(x) = 1 - N(x)$

$$\boxed{\begin{array}{l} \texttt{A = } ((((((a_1 t + a_2)t + a_3)t + a_4)t + a_5)t + a_6)t + a_7) \\ \texttt{B = } (((((((b_1 t + b_2)t + b_3)t + b_4)t + b_5)t + b_6)t + b_7)t + b_8) \\ \texttt{C = } t + 1/(t + 2/(t + 3/(t + 4/(t + 0.65)))) \end{array}}$$

a and b are the following arrays

a_1 = 0.0352624965998911
a_2 = 0.700383064443688
a_3 = 6.37396220353165
a_4 = 33.912866078383
a_5 = 112.079291497871
a_6 = 221.213596169931
a_7 = 220.206867912376

b_1 = 0.0883883476483184
b_2 = 1.75566716318264
b_3 = 16.064177579207
b_4 = 86.7807322029461
b_5 = 296.564248779674
b_6 = 637.333633378831
b_7 = 793.826512519948
b_8 = 440.413735824752

You will implement Hart's algorithm by python in your quiz (2)

Two recent improvements on such CDF numerical approximation

2005: West's work: Better approximations to cumulative normal functions,

Wilmott Magazine, May, pp. 70-76.

Code: http://www.finmod.co.za/research.htm

2010: Meyer's work: Recursive Numerical Evaluation of the Cumulative Bivariate Normal Distribution

Does python have a module for this CDF?

Does python have a module for this CDF?

Yes, there is one in the **stat class of scipy**

SciPy (a numerical computing library)

The standard normal distribution 's pdf:

```
norm.pdf(x) = exp(-x**2/2)/sqrt(2*pi)
```

More general normal distribution

```
norm.pdf(x, mu, sigma)
```

The corresponding CDF

```
norm.cdf(x, mu, sigma)
```

The CDF for standard normal distribution

```
norm.cdf(x, 0.0, 1.0)
```

What we need to code BS formula?

log, exp, sqrt functions from math module
Stat module from Scipy

from math import log, sqrt, exp
from scipy import stats

$$C = SN(d_1) - Ke^{-rT}N(d_2)$$

$$P = Ke^{-rT}N(-d_2) - SN(-d_1)$$

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

$$d_2 = d_1 - \sigma\sqrt{T}$$

A formula should be always coded as a function

```
from  math  import log, sqrt, exp
from  scipy import stats

def bs_call_value(S, K, T, r, sigma):

    d1 = (log(S/K) + (r + 0.5 * sigma ** 2) * T)/ ( sigma * sqrt(T))
    d2 = d1 - sigma * sqrt(T)

    N_d1 = stats.norm.cdf(d1, 0.0, 1.0)
    N_d2 = stats.norm.cdf(d2, 0.0, 1.0)

    call_price = (S * N_d1 - K* exp(-r * T)* N_d2)

    return call_price
```

```
#############################
# test for an European call
#############################

S0    = 89.0   # keep it as a float number even if it is an integer
K     = 100.0
T     = 0.5
r     = 0.02
sigma = 0.20

#############################
# format output
#############################

call_price = bs_call_value(S0, K, T,r, sigma)
print("This Europen Call price is: ${:20.16f}".format(call_price))
```

What does it mean?

```
print("This Europen Call price is: ${:20.16f}".format(call_price))
```

What does it mean?

```
print("This Europen Call price is: ${:20.16f}".format(call_price))
```

Output the call price by following the format: 20.16f → a float number with total 20 digits and 16 digits after the point.

Note: such a high precision is just for programming practice.

Q4: Can we use C style output format instead of '{}.format()' output format?

```
print("This European Call price is: $%20.16f" %call_price)
```

Q4: Can we use C style output format instead of '{}.format()' output format?

```
print("This European Call price is: $%20.16f" %call_price)
```

You can, but such a C-style like (printf(…)) format output is an old style one.

Not recommended!

What are their differences?

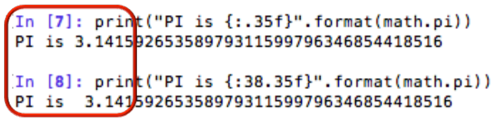
Type it in your computer to find answers!

```
print("PI is {:.35f}".format(math.pi))

print("PI is {:38.35f}".format(math.pi))
```

What are their differences?

Type it in your computer to find answers!

```
print("PI is {:.35f}".format(math.pi))

print("PI is {:38.35f}".format(math.pi))
```



```
In [7]: print("PI is {:.35f}".format(math.pi))
PI is 3.14159265358979311599796346854418516

In [8]: print("PI is {:38.35f}".format(math.pi))
PI is  3.14159265358979311599796346854418516
```

Lab 1: Write a bs_put_formula function(...), and test the following European put

S0 = 102.5 (initial stock price)
K = 88.5
r = 0 .03
T = 0.25
sigma= 0 .3

```
####################################
# BS-European put formula
####################################
def bs_put_value(S, K, T, r, sigma):

    d1 = (log(S/K) + (r + 0.5 * sigma ** 2) * T)/ ( sigma * sqrt(T))
    d2 = d1 - sigma * sqrt(T)

    N_n_d2 = stats.norm.cdf(-d2, 0.0, 1.0)
    N_n_d1 = stats.norm.cdf(-d1, 0.0, 1.0)

    put_price = K* exp(-r * T)* N_n_d2 - S * N_n_d1

    return put_price
```

This Europen Put price is: $ 1.1196098062956548

Extensions of the BS model

Merton did a lot work to extend the original BS model such that it is widely called BSM model. That 's why he also shared the Nobel prize with Scholes in 1997.

These extensions make such a model more practical in the market!

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 - rf = 0$$

$$C = SN(d_1) - Ke^{-rT}N(d_2)$$

$$P = Ke^{-rT}N(-d_2) - SN(-d_1)$$

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

$$d_2 = d_1 - \sigma\sqrt{T}$$

Extension 1: considering dividend (BSM model)

Merton expanded the original BS model by adding dividend yield q in the original PDE in 1970s and got equation to price European options on a stock or stock index paying dividend yield q:

$$\frac{\partial f}{\partial t} + (r-q)S\frac{\partial f}{\partial S} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 - rf = 0$$

The original BS model is a special case of this model when q = 0:

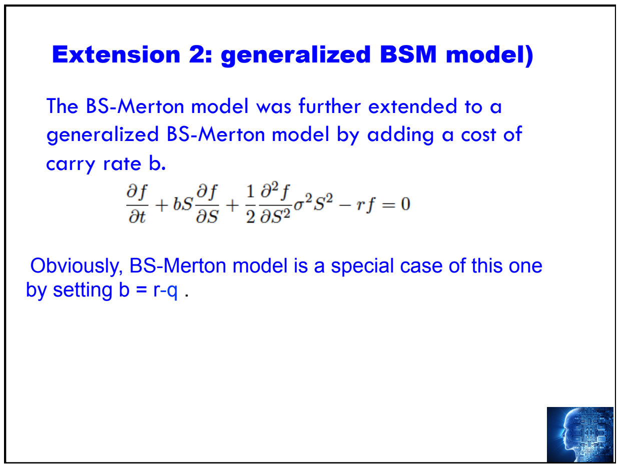
This model (BSM model) has the following analytic solutions for an European options on a stock ot stock index paying dividend

$$C = Se^{-qT}N(d_1) - Ke^{-rT}N(d_2)$$

$$P = Ke^{-rT}N(-d_2) - Se^{-qT}N(-d_1)$$

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

$$d_2 = d_1 - \sigma\sqrt{T}$$

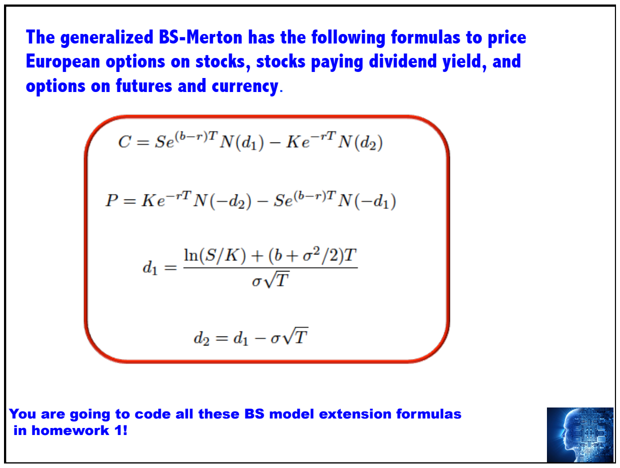


Extension 2: generalized BSM model)

The BS-Merton model was further extended to a generalized BS-Merton model by adding a cost of carry rate b.

$$\frac{\partial f}{\partial t} + bS\frac{\partial f}{\partial S} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 - rf = 0$$

Obviously, BS-Merton model is a special case of this one by setting b = r-q .



The generalized BS-Merton has the following formulas to price European options on stocks, stocks paying dividend yield, and options on futures and currency.

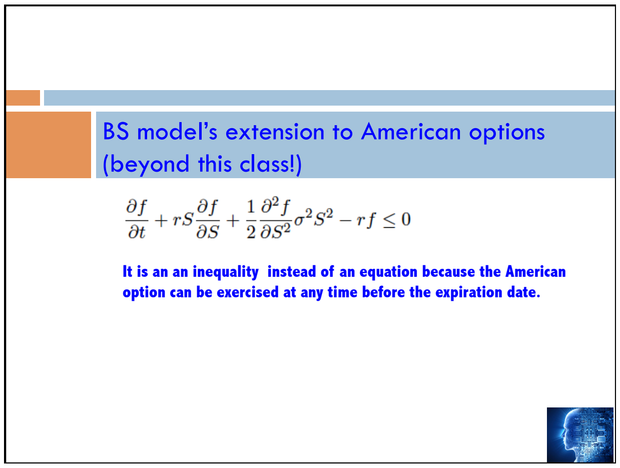
$$C = Se^{(b-r)T}N(d_1) - Ke^{-rT}N(d_2)$$

$$P = Ke^{-rT}N(-d_2) - Se^{(b-r)T}N(-d_1)$$

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

$$d_2 = d_1 - \sigma\sqrt{T}$$

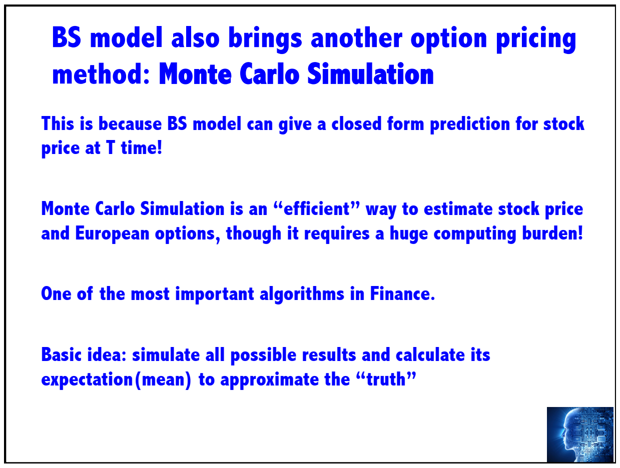
You are going to code all these BS model extension formulas in homework 1!



BS model's extension to American options (beyond this class!)

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 - rf \leq 0$$

It is an an inequality instead of an equation because the American option can be exercised at any time before the expiration date.



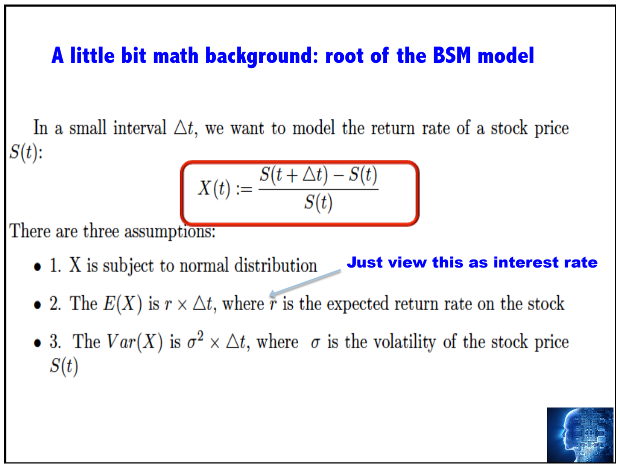
BS model also brings another option pricing method: Monte Carlo Simulation

This is because BS model can give a closed form prediction for stock price at T time!

Monte Carlo Simulation is an "efficient" way to estimate stock price and European options, though it requires a huge computing burden!

One of the most important algorithms in Finance.

Basic idea: simulate all possible results and calculate its expectation(mean) to approximate the "truth"



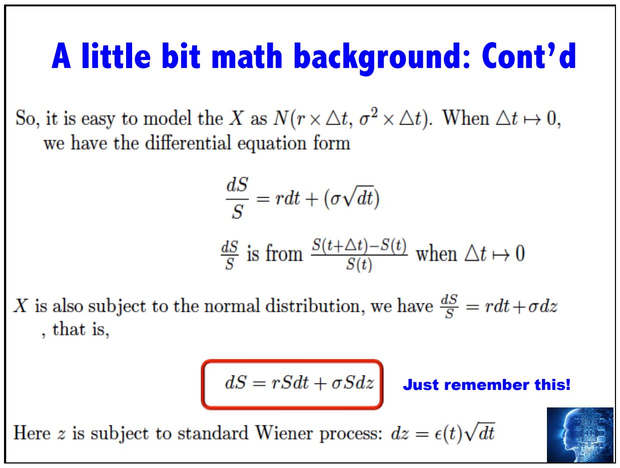
A little bit math background: root of the BSM model

In a small interval Δt, we want to model the return rate of a stock price $S(t)$:

$$\boxed{X(t) := \frac{S(t+\Delta t) - S(t)}{S(t)}}$$

There are three assumptions:

- 1. X is subject to normal distribution **Just view this as interest rate**
- 2. The $E(X)$ is $r \times \Delta t$, where r is the expected return rate on the stock
- 3. The $Var(X)$ is $\sigma^2 \times \Delta t$, where σ is the volatility of the stock price $S(t)$



A little bit math background: Cont'd

So, it is easy to model the X as $N(r \times \Delta t, \sigma^2 \times \Delta t)$. When $\Delta t \mapsto 0$, we have the differential equation form

$$\frac{dS}{S} = rdt + (\sigma\sqrt{dt})$$

$$\tfrac{dS}{S} \text{ is from } \tfrac{S(t+\Delta t)-S(t)}{S(t)} \text{ when } \Delta t \mapsto 0$$

X is also subject to the normal distribution, we have $\frac{dS}{S} = rdt + \sigma dz$, that is,

$$\boxed{dS = rSdt + \sigma Sdz}$$ **Just remember this!**

Here z is subject to standard Wiener process: $dz = \epsilon(t)\sqrt{dt}$

Wiener process: also called standard standard Brownian motion

Brownian motion: random walk at random step

It is used to describe stochastic movements such as like stock price movement in market→ random walk!

BSM starts from this stochastic differential equation:

$$dS = rSdt + \sigma Sdz$$

Small change in stock price has two parts:

1. rSdt (dt: very small time interval almost 0)
2. a random walk times volatility and current price.

BSM model gives an analytic solution for this equation:

The stock price at time T can be calculated as

$$S_T = S_0 \exp\left(\left(r - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}z\right)$$

where z is a standard normal distribution

Basic idea

We know time T stock price S_T and strike price K, we can calculate payoff for an option (e.g. European call)!

payOff = maximum(S_T - K, 0)

S_T has a standard normal distribution term z.

Idea: try all possible cases for S_T through simulations to calculate option price!

Monte Carlo simulation Idea

① Generate N (e.g. 100000) numbers of random numbers from N(0,1)→ This N partitions the time interval [0, T] as N subintervals

Monte Carlo simulation Idea

① Generate N (e.g. 100000) numbers of random numbers from N(0,1)→ This N partitions the time interval [0, T] as N subintervals

② Compute all possible stock prices at T for each random number by using the result from BSM model.

Monte Carlo simulation Idea

1. Generate N (e.g. 100000) numbers of random numbers from N(0,1)→ This N partitions the time interval [0, T] as N subintervals
2. Compute all possible stock prices at T for each random number by using the result from BSM model.
3. Calculate all possible payoffs for the prices you get from the BSM model

Monte Carlo simulation Idea

1. Generate N (e.g. 100000) numbers of random numbers from N(0,1)→ This N partitions the time interval [0, T] as N subintervals
2. Compute all possible stock prices at T for each random number by using the result from BSM model.
3. Calculate all possible payoffs for the prices you get from the BSM model
4. Simulations: calculate the average of all payoffs and divided it by e^rT (r: interest rate)
   1. e^rT: total asset increase in the time interval[0 T]

Q5: How to generate random numbers?

Use numpy function:

random.seed()

To generate different sequence of random numbers→ we need different seeds

random.standard_normal(N)

Generate N number of random numbers under standard_normal distribution

Run the following codes to know more about random.standard_normal(N)

```
① from numpy import *
② random.seed(1000)
③ N=100000
④ z=random.standard_normal(N)

⑤ for i in range(0, 20):
       print("{:15.8f}".format(z[i]))
```

```
-0.80445830
 0.32093155
-0.02548288
 0.64432383
-0.30079667
 0.38947455
-0.10743730
-0.47998308
 0.59503550
-0.46466753
 0.66728131
-0.80611561
-1.19606983
-0.40596016
-0.18237734
 0.10319289
-0.13842199
 0.70569237
 1.27179528
-0.98674733
```

```
from numpy import *

def mc_simluation_eu_call(s0, K, T, r, sigma, no_trial):

### INPUT:
##      s0      : initial stock price
##      K       : strike price
##      T       : time to maurity time
##      r       : riskless interest rate
##      no_trial: # simulation steps

### OUTPUT:
##       european call price

## set seed will make random number sequence repeatable each time

    random.seed(1000)  # 1000 seed states

    z = random.standard_normal(no_trial)

    ## Stock price at Time T

    ST = S0 * exp((r - 0.5 * sigma ** 2) *T + sigma * sqrt(T) * z)

    ## Call payoff
    payOff = maximum(ST - K, 0)

    ## calculate expectation
    eu_call_price = exp(-r * T) * sum(payOff) / no_trial

    return eu_call_price
```

```
############################
# An European call sample
############################

S0     = 89.0
K      = 102.0
T      = 0.5
r      = 0.03
sigma   = 0.3
no_trial = 1000000  # simulation steps

eu_call_price = mc_simluation_eu_call(S0, K, T, r, sigma, no_trial)  # MC simulation

print("\nThe European Call Price is  ${:12.8f}\n".format(eu_call_price))
```

Lab 2: Modify the program such that it works for European put

BS and Monte Carlo simulation, which one will be better?

BS and Monte Carlo simulation, which one will be better?

Monte Carlo simulation will be better than BS model for its more comprehensive information collection in simulation.

Note: there are different Monte Carlo simulation algorithms!