CS 179 Lecture 17

Options Pricing

Computational finance

Broad term, can include ideas from machine learning or signals processing.

One part of computational finance:

(1) How should we price things?
(2) How can we efficiently and accurately compute these prices?

What is an option?

Call/put option -

right to buy/sell something at a specific price in a given time period

something = asset or security

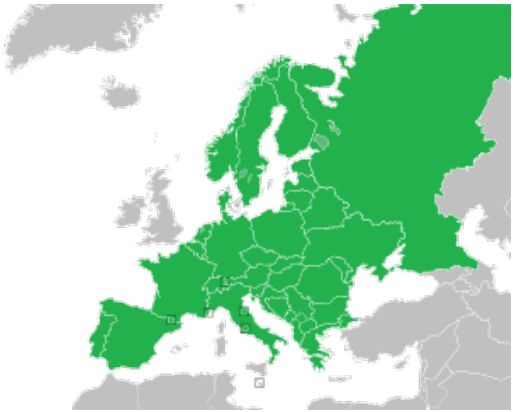
specific price = “striking price” or “exercise price”

time period is often a maturity or expiration date

European options

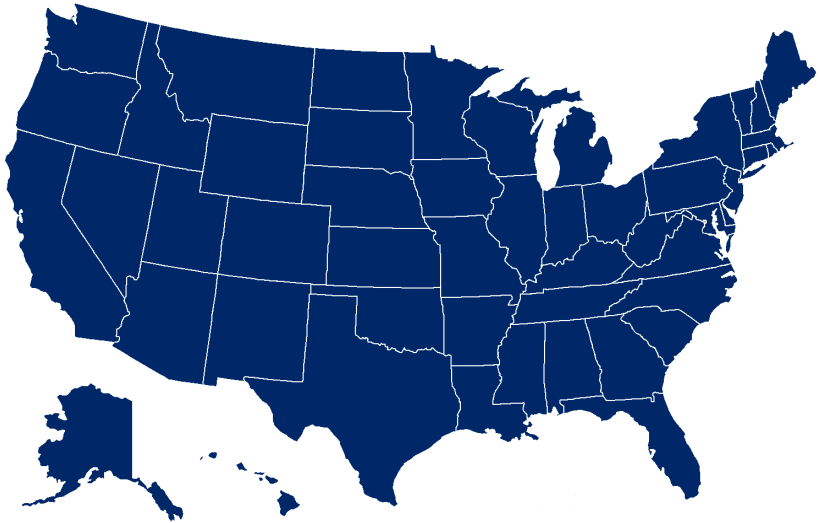
European options are the simplest form of options.

Can only exercise a European option on the expiration date



American options

An American option can be exercised at any time between purchase and the expiration date.



Options pricing

How much should it cost to buy an option?

Example: GOOG currently trades for $540. How much would you pay to be able to sell GOOG for $400 at any point in the next year?

Need to make modelling assumptions about behavior of asset prices.

Formalization

S_t - price of asset at time t, spot price

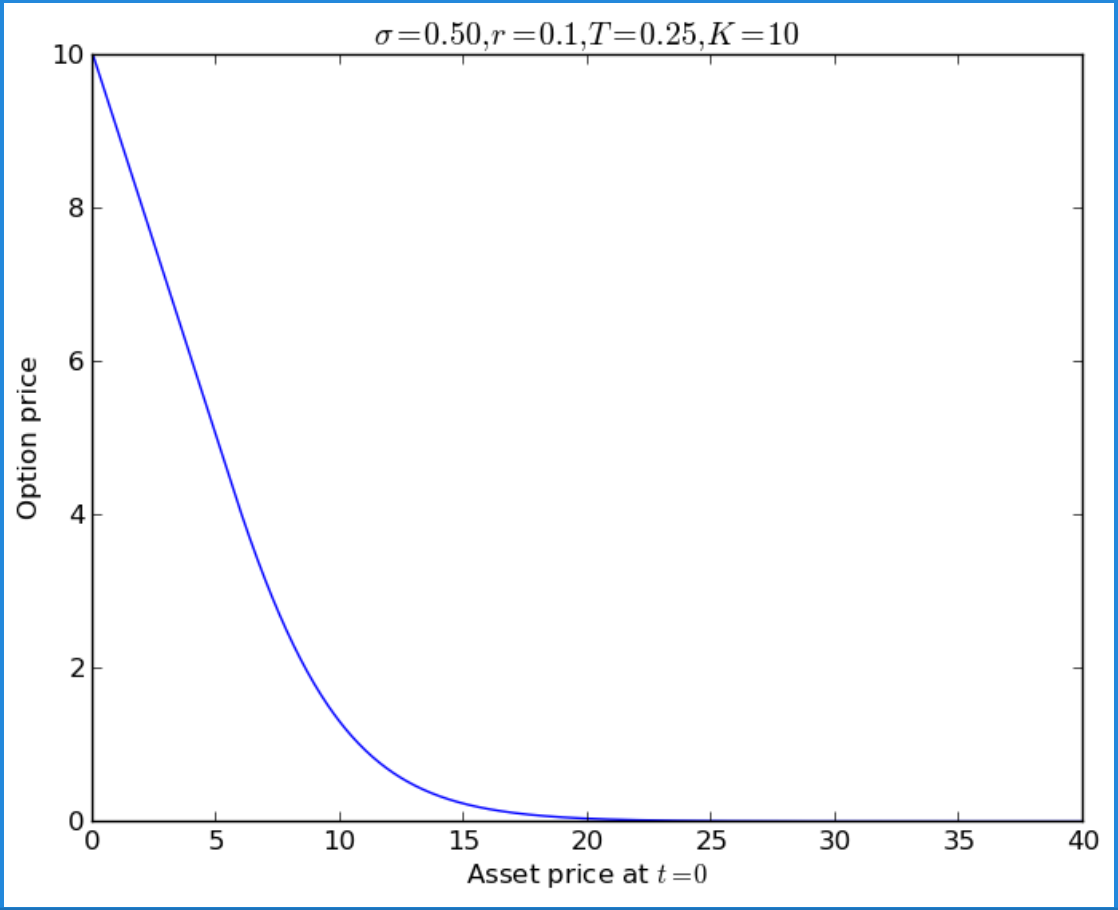
K - striking price

T - time of expiration

Value of option at maturity?

Call option: $\max(S_T - K, 0)$

Put option: $\max(K - S_T, 0)$



American put option pricing by Black-Scholes

Options pricing & GPUs

Sometimes options pricing has an analytic (easy to compute) solution.

Oftentimes not the case…

No closed form for American options under Black-Scholes assumptions. The rest of the lecture is on American options.

Options pricing & GPUs

3 widely used schemes to price options (that can't be analytically priced):

(1) Monte Carlo methods
(2) Finite differencing (differential equations, Black-Scholes)
(3) Binomial option pricing model

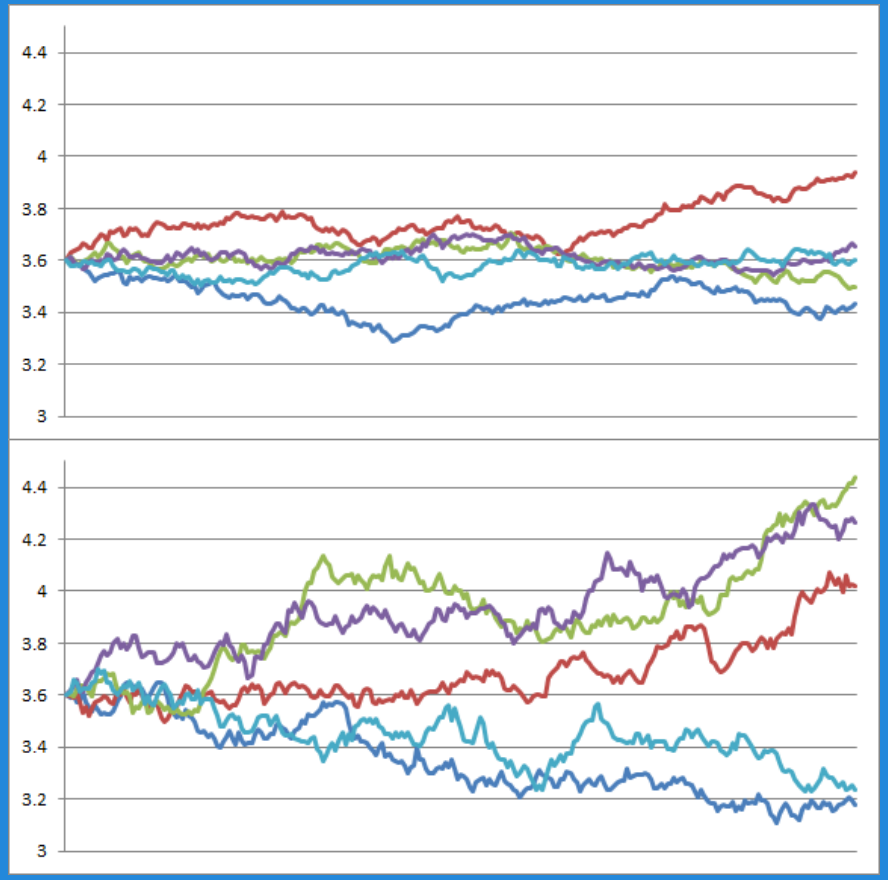
All 3 ways are very different algorithmically!

Monte Carlo methods

Monte Carlo methods approximate difficult to compute statistics by sampling from the generating distribution.

Let's randomly generate asset prices over time.

Parallel Forall blog post on Monte Carlo options pricing



Simulated asset prices (from NVIDIA blog post)

Predicting exercise time

Besides generating the asset price paths, we also need to predict at which point the option would be exercised for each path.

The Longstaff-Schwartz algorithm uses a linear regression at each time-step to decide whether to exercise.

Monte Carlo parallelism

Run different path simulations in parallel.

Can use cuRAND for path generation and cuSOLVER for linear regressions.

Pricing options with PDEs

Can model options price as a PDE in 2 variables: time and asset price.

Known boundary conditions at:

- time=expiration
- spot price = 0
- spot price = ∞

Price is solution to equation at time = 0.

Black-Scholes & American options

The Black-Scholes equation dictates how option price changes in time and spot price. Same equation for European and American options.

B-S requires some stochastic calculus to derive, not going to go into it.

B-S used to compute analytic solution for European options, but boundary conditions are different for American options.

Finite Differencing

Finite differencing is a simple way to numerically solve differential equations

Discretize the equation and solve on a grid.

Replace derivatives with approximations (use Taylor series)

Finite differencing example

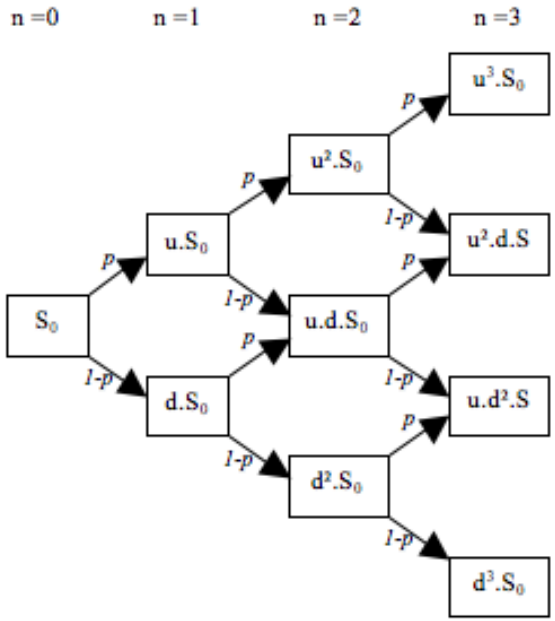
$$u_j^{n+1} = (1-2r)u_j^n + ru_{j-1}^n + ru_{j+1}^n$$

Compute value at time (n+1) from 3 different locations at time n. This is a sparse matrix multiplication! r depends on grid size in both dimensions.

Above is an “explicit” method as time (n+1) depends only on time n. There also implicit schemes that require solving a sparse linear system at both time steps.

Finite differencing parallelization

Perform sparse matrix multiplication or sparse system solving on GPU.



$$p = \frac{e^{rt/n} - d}{u - d}$$

$$u = e^{\sigma\sqrt{t/n}}$$

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

Binomial options pricing model diagram (from Wikipedia)

Binomial options pricing

A dynamic programming problem with convenient assumptions.

At each time step, asset price goes up by factor “u” or down by factor “d”.

Let u * d = 1.

Finishing binomial pricing

(1) Assign option values to each node at expiration time
(2) Work backwards in time to assign each node an option value based on the values of its two children based on interest-discounted expected value. For American option, take max at each node of exercising or keeping option.

Parallelizing binomial pricing

Less easy than parallelizing other models because of serial algorithm across times.

Can use data parallelism to compute time i prices from time (i+1).

Can also use task parallelism to directly compute time (i-1), i directly from (i+1). Each (i-1) node just depends on 4 nodes rather than 2. Repeating work isn't illegal!

Conclusion

GPUs are useful for option pricing and computational finance.

Some of these algorithms exhibit “MATLAB-parallelism”

Some parts of finance can greatly benefit from the parallel computing ideas we’ve discussed in this course.