**LATTICE METHODOLOGIES**

Def. **Lattice:** partially ordered set of points.

**COX-ROSS-RUBISTEIN MODEL**

1. **Price tree generation:** the spot price follows a recombinant binomial tree. At each step, the underlying moves up/down by a factor

by NA: 0 < d < (1+r) < u

At the end of n subperiods, the underlying stock price is: . Assuming the underlying is lognormal: u and d are set in such a way that the moments of the binomial distribution match the moments of the normal distribution.

1. **Compute option value at each FINAL NODE**
2. **Compute option value at each EARLIER NODE until valuation date**

The fair price of the option in the expected value under Q of its future payoff, discounted by r.

Choose delta so that the portfolio is risk-free:

Therefore

Value of the option at each earlier node:

Where the risk neutral probability is:

**AMERICAN PUT OPTION**

Payoff

Price: , where

Going backwards from maturity to valuation date, at each step we must check for early exercise.

The optimal exercise price is the first date in which the *intrinsic value* of the option is > than the *continuation value*.

**MONTE CARLO SIMULATIONS**

When we have a model, there are 3 ways to solve for the price:

1. **Integral:**

Solve with numerical approximation: quadrature methods.

1. **Monte Carlo Simulations**: expectation under *Q* of the payoff of the contract under different simulated scenarios.
2. **Solve the PDE**

**FEYNMAN-KAC THEOREM**

Establishes a link between PDE and Monte Carlo: the solution on the PDE can be seen as an expectation:

where

**MONTE CARLO METHODS**

X random variable with distribution . We wish to estimate**:**

Take a set of *iid* random variables: with distribution Then, we can use as an approximation of .

mean**:** E[ variance: Var[

converges to when n

* **Law of Large Numbers:**
* **Central Limit Theorem :**

The **error of the Monte Carlo** is: *N(0, S*tandard error : . Convergence rate: O()

**Confidence interval**: , with z =

**Random generator:** *Inverse transformation method*

1. Random sampling from uniform U([0,1]) and obtain *u.*
2. Find a transformation ***F*** of U, which must be: monotonically increasing and bijective. ***F*** is the inverse of a CDF. F is the CDF:

Compute**: F(x) = *u***

1. Take **x** as the random number drawn.

Multidimensional extension: if we have to simulate multiple risk factors at the same time:

* INDEPENDENT: apply inverse transformation to each risk factor:
* CORRELATED: solve the following system for

**VARIANCE REDUCTION**

Procedure used to increase the precision of the estimate that can be obtained for a given number of iterations. Since the confidence interval is: . We want to find a new variable that has:

* Same expectation
* Smaller variance, that determines a smaller confidence interval.

1. **Antithetic variables**

Given X: random variable and g(x) monotone function, we want to estimate

The antithetic variable estimator is:

Where and are:

* Sampled from the distribution of X,
* Negatively correlated
* same expectation
* < because cov < 0.

1. **Control variables**

Given X rv, we aim to estimate

***f(x): control variable***

* Quite similar to g(x). i.e. underlying or closed form solution for the price.
* E[f(x)] is known
* E[] = E[g(x)]: same expectation
* Var[]

Since we want Var[], we minimize Var[] with respect to alpha

1. Choose control variable *f*
2. Run a fast MC to estimate the covariance and the expected value of f (if unknown)
3. Run MC to compute the price.

**CONTINUOUS PROCESS SAMPLING**

To sample solutions of an SDE: :

1. Simulate the process at a discrete set of times: create a *time grid* over the interval [0,T], splitting it into -sized intervals. The starting state is known.
2. Interpolate to produce a continuous-time trajectory.
3. **EXACT TRANSFORMATION:** the exact transition density must be known for any pair of consecutive times.
   1. Create time grid.
   2. For i in 1…N, sample
   3. Return as sample of the process X on [0,T]

Example: Vasicek: normal transition density (moments of the distribution can be easily computed from the parameters of the model).

1. **EXACT SOLUTION:** the strong solution of an SDE is explicit if it can be expressed ad analytic functional G of time t and random noise until time t:
   1. Create time grid.
   2. For i in 1…N, sample random noise
   3. Return as sample of the process X on [0,T].

Example: Geometric Brownian Motion

Exact solution in known:

Discretize:

Where

1. **APPROXIMATE THE DYNAMICS**, by discretizing the SDE:

* **Euler Scheme:**
* **Milstein Scheme:**

**CEV MODEL- constant elasticity of variance**

**HESTON MODEL- stochastic volatility**

Where

Feller condition:

**JUMP PROCESSES**

* **:** determines the jump SIZE. It’s generated through an assigned distribution (i.e. Normal).
* **: COUNTING PROCESS:** number of jumps that have occurred until time t.

Poisson:

**Simulations:**

* 1. **CONDITIONAL SIMULATION:**

1. Simulate number of jumps occurring in [0,T]: N(t)
2. Simulate the exact location of the jumps:
3. Return: jump times and poisson realizations
   1. **COUNTDOWN SIMULATION:**
4. Sample jumps in sequential order:
5. Find location of jumps:
6. Return: number of jumps and exact location

**MIXED JUMP-DIFFUSION MODELS**

Discretization: idea:

* Sample jump part of the process
* Simulate diffusive part with Euler Scheme
* Add the jumps to the diffusive part

**Euler scheme:**

Algorithm

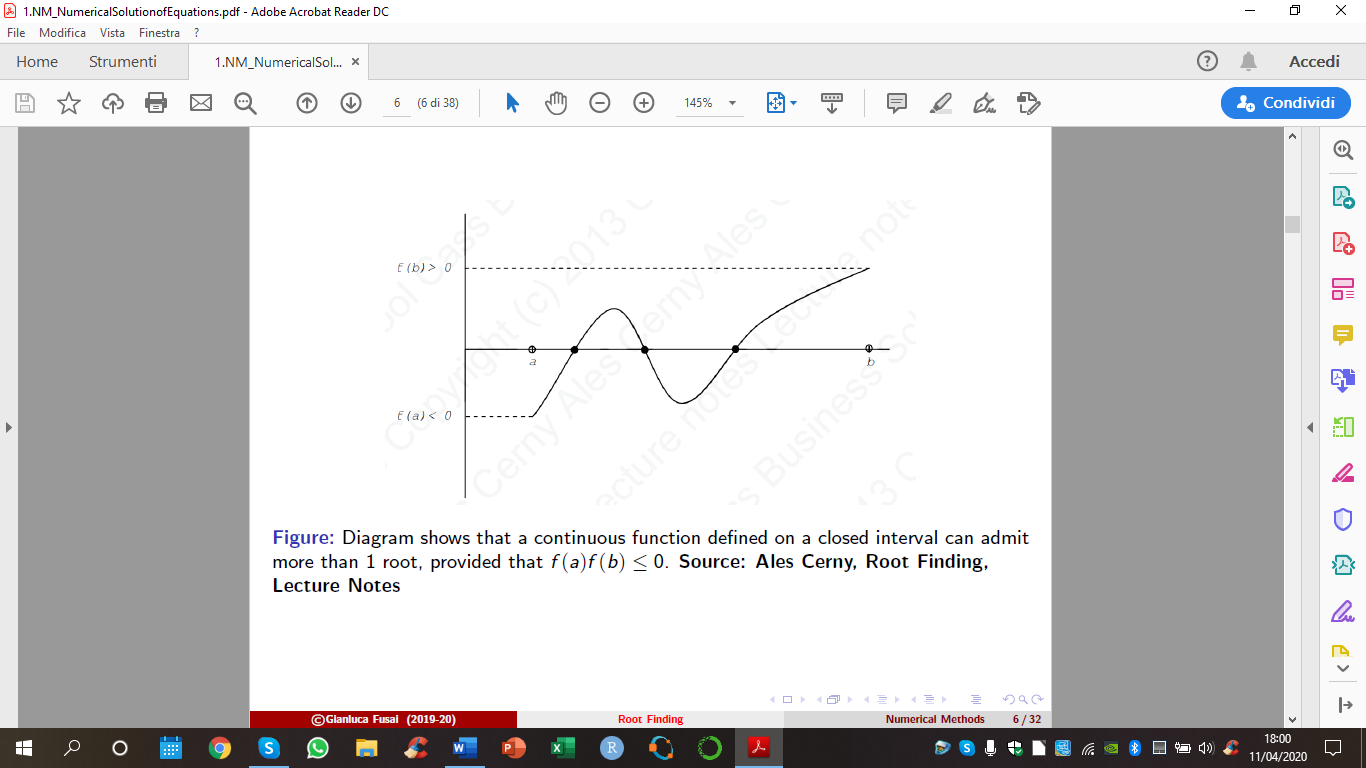
1. Generate random jump times:
2. For j = 0 : (N-1)
   * simulate
   * a jump occurs in *(j+1):*
     + Simulate the jump size , from a distribution

**NUMERICAL SOLUTION OF EQUATION**

Solve: *f(x) = 0*

1. **BISECTION METHOD**

If *f* is:

* Continuous on [a,b]
* opposite signs at the endpoints

then,

If there are *n roots:*

* Even roots: you find them with *probability = 0*
* Odd roots: you fine them with *equal probabilities*

ALGORITHM:

1. **Halve** the original interval →
2. If → the new interval is:

If → the new interval is:

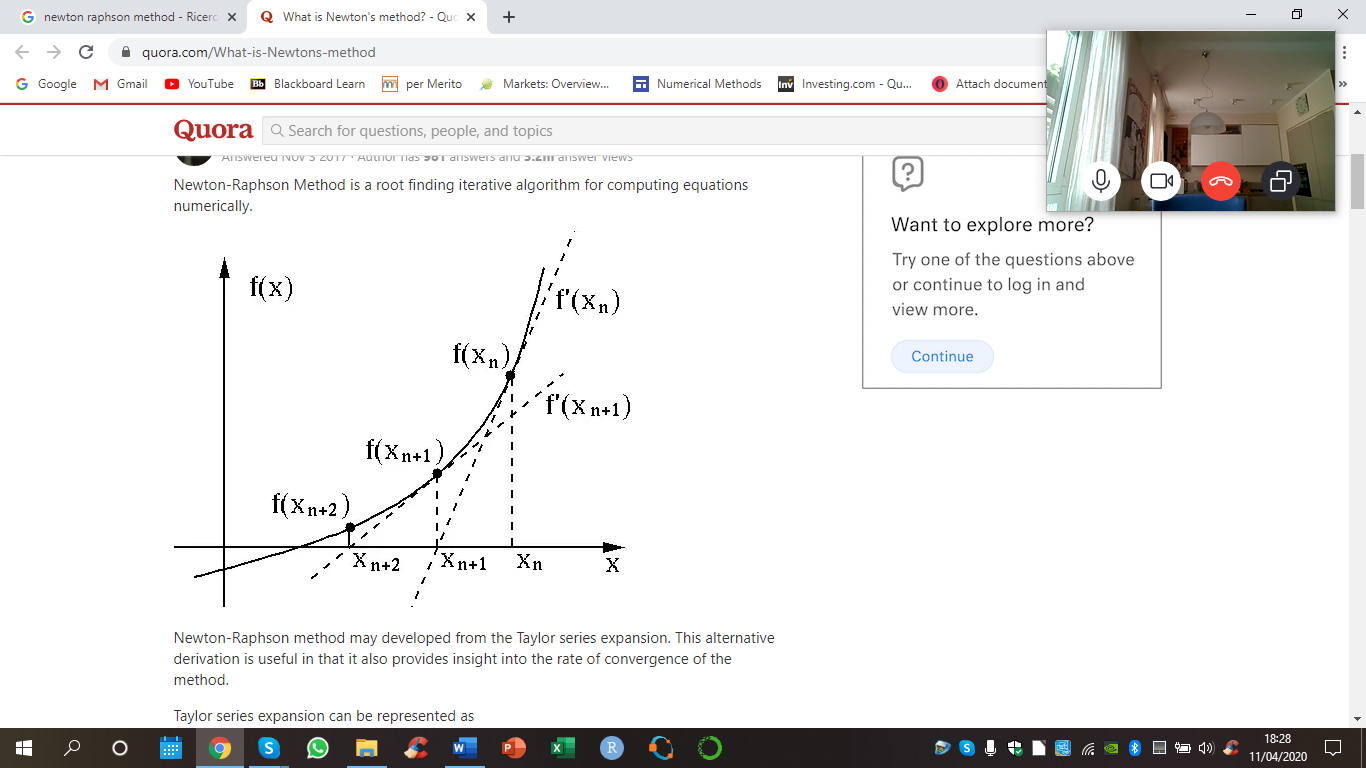
1. The iterations stop when a preassigned level of tolerance is obtained:|b – a |

**Convergence:**

If we want a 1-digit precision:

→ slow convergence

1. **NEWTON-RAPHSON METHOD**

*f(x)*: continuous and differentiable.

1. Starting point:
2. Exploit *first order Taylor expansion* of *f* around *xk*.
3. Iterations stop when is small

**Convergence Conditions:**

Then, all the iterations also satisfy:

There exist a constant c, such that: → **quadratic converge**

1. **SECANT METHOD**

Numerical approximation of the derivative: backward method

**Superlinear convergence:**

**Derivatives approximation: FINITE DIFFERENCES**

* **FORWARD METHOD:**
* **CENTRED METHOD:**
* **BACKWARD METHOD:**

**MODEL CALIBRATION**

Choosing the parameter set Θ that minimizes the distance between: cross-section of model quantities and market observables.

1. **Non linear least squared regression**

NLLS estimator is asymptotically normally distributed: -

Where Q is consistent estimate of the asymptotic Variance-Covariance matrix:

: Jacobian matrix (matrix of partial derivatives with respect to the parameter set)

1. **Quadratic loss function** to be minimized in order to estimate the parameters:

Mean-squared error =

Percentage mean-squared error =

**NELSON-SIEGEL MODEL**



The set of parameters is chosen in such a way that:

**NEWTON’S METHOD: minimization of a function**

Locally approximate the function *f* with a quadratic function. Exploit Second order Taylor expansion:

* GRADIENT VECTOR: vector of first order partial derivatives
* HESSIAN MATRIX: matrix of second order partial derivatives

1. ***First order condition:***
2. ***Second order condition***

* H positive definite

**Convergence:** if we start sufficiently close to the minimum, the convergence is **quadratic.**

We can define a general class of algorithms:

* **:** search parameter
* **M:**
  + If M = , Newton Method
  + If M = 1, Steepest Descent Method

**NUMERICAL INTEGRATION**

**NEWTON-COTES INTEGRATION**

1. Fix the abscissas: spit interval into evenly spaced subintervals →
2. Approximate the function with an interpolating polynomial:
3. The integral is the sum of the integrals computed in each subinterval:
4. **RECTANGLE RULE:** sample the function at the beginning of each subinterval

convergence: order 1

1. **MIDPOINT RULE:** sample the function in the middle of each subinterval

convergence: order 2

1. **TRAPEZOID RULE:** convergence: order 2

**GAUSSIAN QUADRATURE**: automatically selects n abscissas and n weights. Convergence of order **2n -1**

**ADAPTIVE QUADRATURE:** integrate f(x) using 2 integration methods and . Split the interval until the difference between the two approximation is small enough:.

**IMPLIED VOLATILITY**

**SKEWNESS PREMIUM:** percentage deviation of OTM call prices from OTM put prices:

|  |  |  |
| --- | --- | --- |
| **SK < 0** | **SK = 0** | **SK > 0** |
| **LEVERAGE effect:**   * Positive returns, volatility * Negative returns, volatility |  | **INVERSE LEVERAGE effect:**   * Positive returns, volatility * Negative returns, volatility |
| **LEFT skew** | **SYMMETRIC - leptokurtic** | **RIGHT skew** |
| **Skew** | **Smile** | **Skew** |
| **CEV** | **CEV** | **CEV** |
| **HESTON** | **HESTON** | **HESTON** |
| **JUMP PROCESS**  Negative jumps | **JUMP PROCESS**  No jumps (or 0 expected value) | **JUMP PROCESS**  Positive jumps |

**SHIMKO APPROACH**

1. Solve Black-Scholes for implied volatility.
2. Approximate the implied volatility curve with a quadratic interpolation: OLS regression
3. Generate the corresponding option prices using with Black and Scholes.
4. Obtain the implied density: *Breeden and Litzenberger Approach:*

* Implied CDF:
* Implied DENSITY

**OPTION PRICING with INTEGRATION**

= =

**Payoff**

**Density:** under BS follows Geometric Brownian Motion: is lognormal

**Price**