An Introductory Course in MATLAB with Financial Case Studies

Panayiotis C. Andreou, PhD

Monte Carlo Simulation

- We shall assume that our share price S(t) is a random variable given by the following formula:

$$S(t) = S(0)e^{(r-\sigma^2/2)t+\sigma\sqrt{t}Z}, \qquad \text{for } t > 0$$

 where *Z* is a standard Gaussian random variable.

- The above process is also assumed to govern the Black and Scholes model.
- We cannot predict the future price S(T) of our stock at time T, but we can approximate the distribution of its possible values. In other words, we can predict the likely behaviour of our asset in many possible futures.

Monte Carlo Simulation

Question: Illustrate the distribution of the future values for $S(T)$ with:

$S = 42$, $r = 0.1$, $T = 0.5$ and $\sigma = 0.2$.

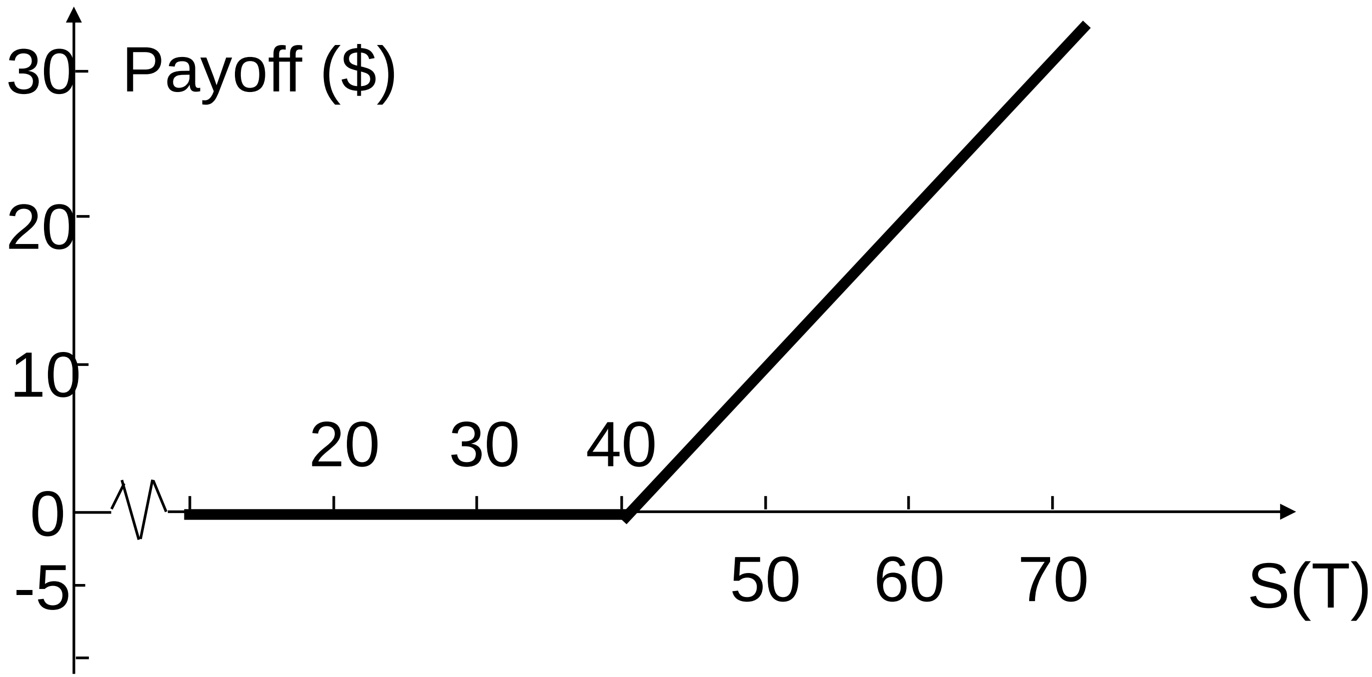
Monte Carlo Simulation

<u>Answer</u>:

```
S0 = 42; r = 0.1; T = 0.5; sigma = 0.2;
N = 100000;
% generate asset prices at expiry
Z = randn(N,1);
ST = S0*exp( (r-(sigma^2)/2)*T + sigma*sqrt(T)*Z );
% display histogram of possible prices at expiry
nbins=40;
hist(ST,nbins);
```

Monte Carlo Simulation

<u>Question</u>: Value a European call option with a strike price K = 40.



Monte Carlo Simulation

<u>Answer</u>: The payoff from a call option is: max(S(T) – X, 0).

```
% calculate call contract values at expiry
fcall = max( ST - 40,0.0);
% average call values at expiry and discount to present
call = exp(-r*T)*sum(fcall)/N
```

Monte Carlo Simulation

Question: How can you check whether the call price you got makes sense?

Monte Carlo Simulation

Answer: Use the function: "pres_BSMprice.m"

Input:

```
[Call, Put]=pres_BSMprice(42,40,0.5,0.2,0.1,0)
```

Regression Analysis

- Matlab offers a variety of functions to perform different type of regressions like Linear Regression, Generalized Linear Models and Nonlinear Regression. Those functions are mostly located in the Statistical Toolbox, however the Econometric Toolbox has also a number of function to perform a particular regression analysis. In this section only linear and logistic models are presented.
- Matlab has two basic functions for performing this linear regression: ***regress*** and ***regstats***.
- **Review : “RegressionAnalysisExamples.m”**

System of Linear Equations

- System of linear equations:

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_m = b_n \end{gathered}$$

- Can you figure out how to solve in Matlab the following problem?

$$\begin{gathered} 2X_1 + 4X_2 + 3X_3 = 2 \\ -2X_1 - 4X_2 + 2X_3 = 4 \\ 6X_1 + 4X_2 + 2X_3 = 1 \end{gathered}$$

Function Minimization

- There is a variety of optimization algorithms that can be used to minimize (or to maximize) a univariate or a multivariate function (note that the maximization of a function, *f(x),* is the same as the minimization of *-f(x)).*
- Two well known and widely used algorithms are the ***gradient descent**,* that minimizes a function by relying solely on a function's first partial derivatives (gradient vector), and the ***Newton's descent*** algorithm that utilizes a function's second order partial derivatives (Hessian matrix).

Function Minimization

The <u>gradient descent algorithm</u> can be summarized as follows:

- ***Step #1***: *give an initial guess for the coordinates of the minimum point x^0 (e.g. if the function under consideration has two unknowns x1 and x2, then x should be a two element row vector:* $\boldsymbol{x^0} = [\boldsymbol{x_1^0}, \boldsymbol{x_2^0}]$

- ***Step #2***: *perform an algorithm iteration, k, based on the following formula:*

$$x^{k+1} = x^k - af'(x^k)$$

for k=1,2,3,....

Specifically, f'(x) for a function with n variables is:

$$f'(x) = \left[\frac{\partial f(x)}{\partial x_1} \quad \frac{\partial f(x)}{\partial x_2} \quad \cdots \quad \frac{\partial f(x)}{\partial x_n} \right]$$

Function Minimization

The <u>gradient descent algorithm</u> can be summarized as follows:

- ***Step #3***: Increase: k ← k+1 and if the current iteration index k is larger than the maximum number of iterations or if $\|\boldsymbol{a}\boldsymbol{f}'(\boldsymbol{x}^{k})\| < \boldsymbol{e}$ then stop and return $\boldsymbol{x}^{k+1}$, ***otherwise*** go to Step #2 and perform one more iteration of the algorithm (the e is the desire accuracy, usually set to a small quantity such as *1*e-*6*, whereas the maximum number of iterations depends solely by the experience of the researcher).

Function Minimization

The <u>Newton descent algorithm</u> can be summarized as follows:

- ***Step #1:*** *give an initial guess for the coordinates of the minimum point* x^0 *(e.g. if the function under consideration has two unknowns x1 and x2, then x should be a two element row vector:* $\mathbf{x}^0 = [\mathbf{x}_1^0, \mathbf{x}_2^0]$*)*

- ***Step #2****: perform an algorithm iteration, k, based on the following formula:*

$$x^{k+1} = x^k - f'(x^k)[f''(x^k)]^{-1}$$

for k=1,2,3,….

Function Minimization

The Newton descent algorithm can be summarized as follows:

- ***Step #2****: perform an algorithm iteration, k, based on the following formula:*

$$x^{k+1} = x^k - f'(x^k)[f''(x^k)]^{-1}$$

for k=1,2,3,....

Specifically, $f''(x)$ for a Function with n variables is:

$$f''(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$

Function Minimization

The <u>Newton descent algorithm</u> can be summarized as follows:

- ***Step #3***: Increase k ← k+1 and if the current iteration index k is larger than the maximum number of iterations or if $\|f'(x^k)[f''(x^k)]^{-1}\| < e$. Then stop and return x^{k+1}, ***otherwise*** go to Step#2 and perform one more iteration of the algorithm (the e is the desire accuracy, usually set to a small quantity such as 1e-6, whereas the maximum number of iterations depends solely by the experience of the researcher).

Function Minimization

- Find the minimum for the following function:

$$\boldsymbol{f(x, y) = e^{x}(4x^2 + 2y^2 + 4xy + 2y + 1)}$$