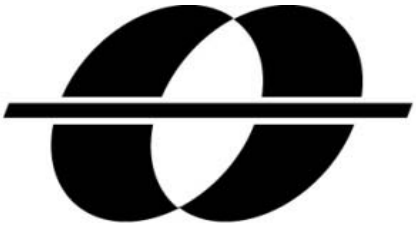
MMA707 – Analytical Finance Ⅰ –Jan R. M. Röman

Monte Carlo Simulation



MÄLARDALEN UNIVERSITY

GROUP

Wej Wang
Maierdan Halifu
Yankai Shao
Arvid Kjellberg

2008-10-09

Department of Mathematics and Physics
Mälardalen University
SE-721 23 Västerås, Sweden

Abstract

This report is designed to describe methods of Monte Carlo simulation to students and teachers of the course: Analytical Finance and other scholars who are interested in Monte Carlo simulation. The report introduces some background and general ideas about Monte Carlo simulation and how it can solve unpredictable stock price movements and endeavors.

Content

1. Aim and Mathematical Approach

1.1 Aim

We aim to create a simulation program to price European call option and put option. The options are pricing with the method of Monte Carlo Simulation algorithm with MATLAB.

1.2 Mathematical approach for European options

First we use Black-Scholes formula to find a series of prices for both European call options and European put options. Then we use Time and Volatility from Black-Scholes formula into our Monte Carlo Simulation to simulate the underlying asset price changes.
The Black-Scholes Formula we use:

$$C = S\Phi(d_1) - Ke^{-rT}\Phi(d_2)$$

Where:

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

$$d_2 = \frac{\ln(S/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}.$$

And here Φ is the standard normal cumulative distribution function:

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{d_n} e^{-\frac{x^2}{2}}\, dx$$

And we can use Put-call Parity to get the price of put options:

$$C(t) + K \cdot B(t,T) = P(t) + S(t)$$

Where:

$$B(t,T) = e^{-r(T-t)}$$

After we get the values, we turn to use Monte Carlo simulation immediately. As we know, the stock price of a share at time T is:

$$S(T) = S(0)\exp\left\{\left(r - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}Z\right\}$$

Where S(0) and S(T) are the prices of the share at moment 0 and T respectively, r is a

risk-free interest rate, σ is volatility, Z~N(0,1).

Then we define prices both for call option and put option:

$$C = e^{-rT}\max\{S(T) - k, 0\}$$

$$P = e^{-rT}\max\{k - S(T), 0\}$$

To find the theoretical option value we calculate the mean value of the discounted pay-off:

$$C_0 = e^{-rT}\frac{1}{N}\sum_{i=1}^{N}\max\{S_{T,i} - X, 0\}$$

2. Numerical Examples:

To get the price of European options use the Monte Carlo simulation program, we take one European call option with the following parameters:

Strike Price K = 95

Spot Price S = 100

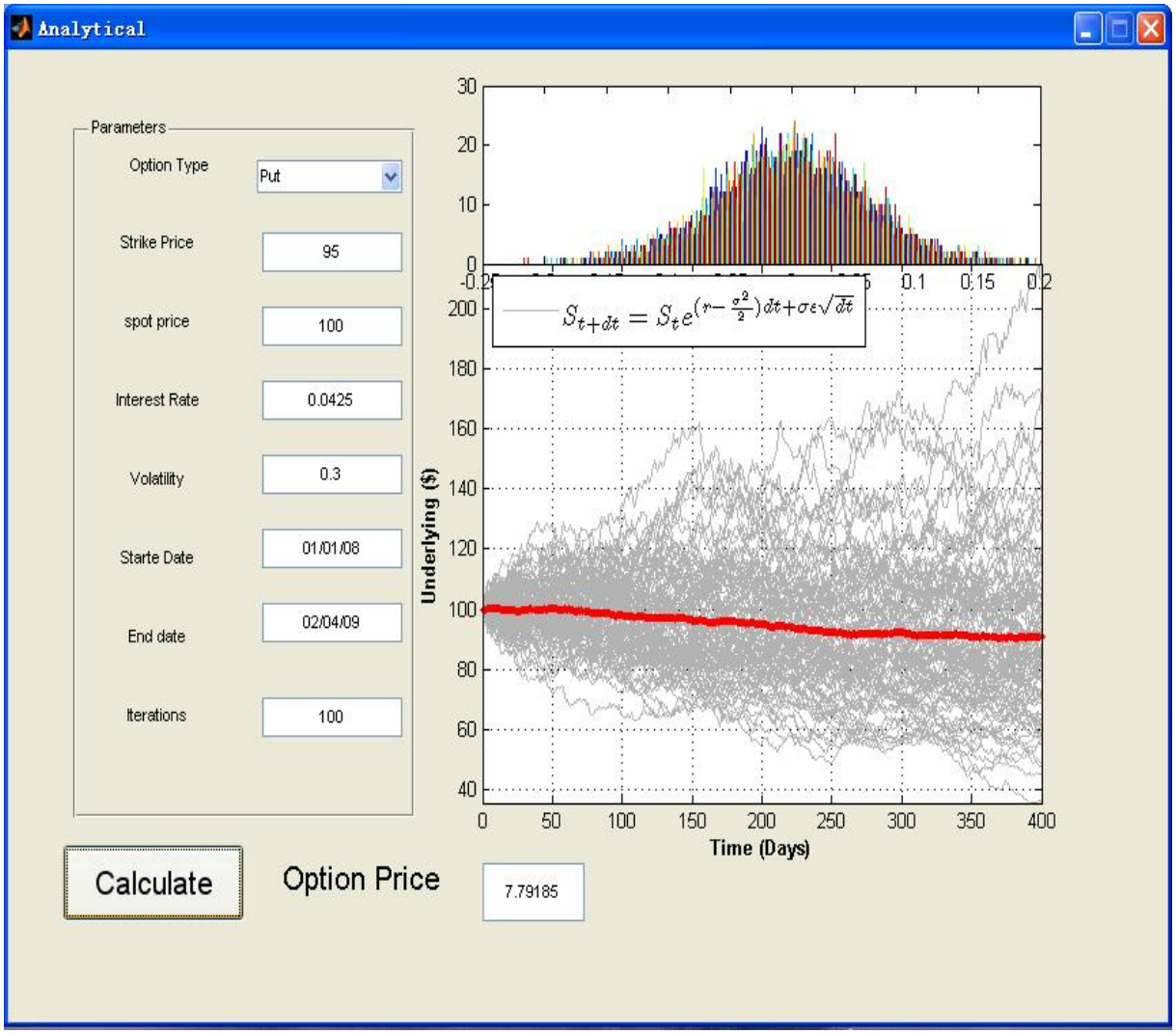
Interest Rate r = 0.0425 (the latest interest rate in Sweden)

Volatility σ = 0.3

Start Date: 01/01/08

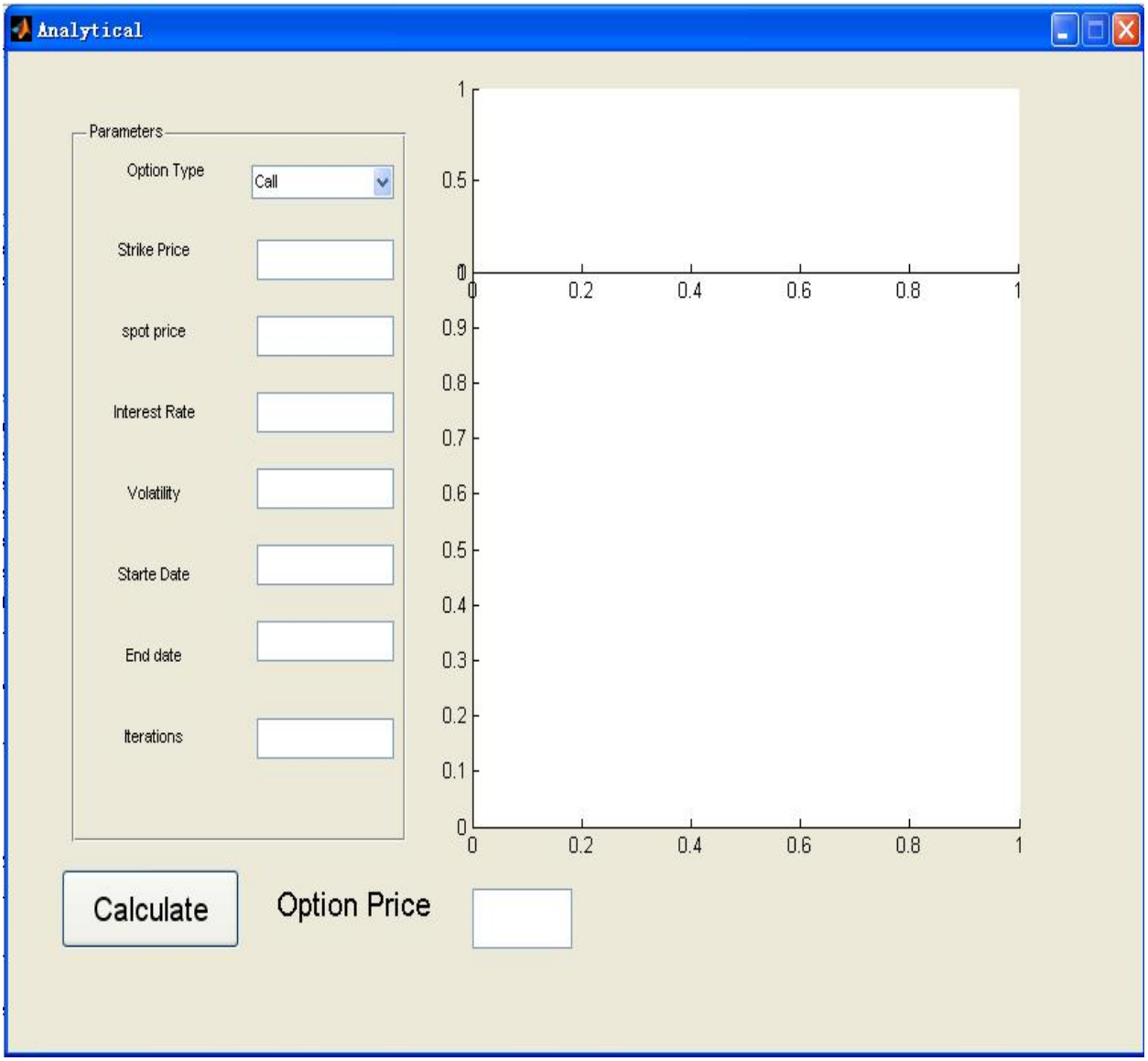
End Date: 02/04/09

Iterations = 100



3. The MATLAB GUI User Interface:

We made a program with MATLAB GUI, where the data are needed to be entered in the corresponding edit boxes. After pressing the Calculate pushbutton, the option price as well as its mean return from simulation and distribution histogram will be displayed.



4. Codes in the MATLAB program:

We only include part of MATLAB code where have been programmed.

4.1 Strike price:

```
function edit1strikeprice_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'String'))
New1=get(hObject,'String');
New2=str2double(New1);
handles.edit1strikeprice=New2;
guidata(hObject,handles);

function edit1strikeprice_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

4.2 Spot price

```
function edit2spotprice_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
New1=get(hObject,'String');
New2=str2double(New1);
handles.edit2spotprice=New2;
guidata(hObject,handles);
```

4.3 Interest Rate:

```
function edit3interestrate_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));  %#ok<NASGU>
New1=get(hObject,'String');
New2=str2double(New1);
handles.edit3interestrate=New2;
guidata(hObject,handles);
```

4.4 Volatility

```
function edit4volatility_Callback(hObject, eventdata, handles)
```

```
user_entry = str2double(get(hObject,'String'));
New1=get(hObject,'String');
New2=str2double(New1);
handles.edit4volatility=New2;
guidata(hObject,handles);
```

4.5 Start date & End date:

The length between start and end date is calculated by MATLAB function yearfrac, which is programmed within Pushbutton1' Callback instead.

4.6 Iterations:

```
function edit7iterations_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'String'));
New1=get(hObject,'String');
New2=str2double(New1);
handles.edit7iterations=New2;
guidata(hObject,handles);
```

4.7 This is the most important part in our program which generates the Monte Carlo simulation:

```
function pushbutton1_Callback(hObject, eventdata, handles)
K = handles.edit1strikeprice;
S = handles.edit2spotprice;
r = handles.edit3interestrate;
SIG = handles.edit4volatility;
iter = handles.edit7iterations
startDate = get(handles.edit5startdate, 'String');
endDate = get(handles.edit6enddate, 'String');
try
       T = yearfrac(startDate, endDate);
   catch
       error('optionPriceGui:InvalidDates', ...
          'Dates must be in a valid format')
end
[call,put] = blsprice(S,K,r,T,SIG);
```

```
if handles.bCall == true
    set(handles.optionprice,'string',call);
else
    set(handles.optionprice,'string',put);
end
guidata(hObject,handles);
%now run the Monte Carlo simulation
numDays = daysact(startDate, endDate);
TT = sqrt(T/numDays) * ones(numDays, iter);
rNum = randn(numDays, iter);
BT = TT.*rNum;
TT = [zeros(1, iter); TT];
BT = [zeros(1, iter); BT];
ST = S* exp(cumsum((r - 0.5 * SIG^2) * TT + SIG * BT));
% Plot the Monte Carlo simulation
set(handles.axesHist,'yaxislocation','right','xaxislocation','top',..
.
    'xtick',[],'xlimmode','auto')
if size(ST,2)>1000
    ind = randperm(size(ST,2))';
    STplot = ST(:,ind(1:1000));
else
    STplot = ST;
end
% Plot the monte-carlo underlying price paths
handles.pricePaths = plot(handles.axesMonte, (0:numDays), STplot,...
    'color', [.7,.7,.7]);
%plot the mean underlying Monte carlo price path
STmean = mean(ST,2);
handles.meanLine = line('XData', (0:numDays), 'YData', STmean, ...
    'Parent', handles.axesMonte, 'Marker', '.' ,'MarkerSize', 10, 'Color',
'r');
%Monte Carlo Plots
xlabel(handles.axesMonte, 'Time (Days)','fontweight','bold')
ylabel(handles.axesMonte, 'Underlying ($)','fontweight','bold')
axis(handles.axesMonte, 'tight')
grid(handles.axesMonte, 'on')
legend(handles.axesMonte, ...
    {'$$S_{t+dt} =
S_te^{(r-\frac{{\sigma^2}}{2})dt+\sigma\epsilon\sqrt{dt}}$$'}, ...
    'fontsize',14,'interpreter','latex', 'Location', 'NorthWest');
x = handles.meanLine;
y = [zeros(1, iter); BT];
hist(y,x)
```

5. References:

1. Analytical Finance 1 Lecture Note, Jan R. M. Röman, 2007
2. http://en.wikipedia.org/wiki/Black-scholes, latest visit Oct 08, 2008
3. http://en.wikipedia.org/wiki/Monte_Carlo_simulation, latest visit Oct 08, 2008
4. www.mathworks.com, latest visit Oct 08, 2008

6. Appendix

We use the same interface to find the price one European put option using the data from the previous section.