TIP ejercicio 1
curso 2012/2013

Carlos Gil Soriano
UAM-HPCN
gilsoriano@gmail.com

21 de febrero de 2013

Resumen

En esta primera entrega de ejercicios se tratan tres problemas que se centran en:

- Comparación del cálculo de momentos de orden n mediante *cuadraturas* con su correspondiente mediante simulación por *Montecarlo*.
- Propiedades del *Browniano aritmético*.
- Aplicaciones del *Browniano geométrico*.

Historial		
Versión	**Estado**	**Fecha**
0.1	Problema 1 resuelto	18 de febrero de 2013
0.1	Problema 2 resuelto	19 de febrero de 2013
0.1	Problema 3 resuelto	21 de febrero de 2013

Índice

Índice de cuadros

Índice de figuras

Listings

1. Cálculo de momentos centrales

Se define un momento central de orden n como:

$$\mu_n = E[(X - E[X])^n] = \int_{-\infty}^{\infty} (x-\mu)^n pdf(x)\mathrm{d}x$$

Para realizar la cuadratura debemos identificar la función $g(x)$ a integrar:

$$I = \int_{-\infty}^{\infty} g(x)\mathrm{d}x$$

En nuestro caso se trata de:

$$g(x) = (x-\mu)^n pdf(x)$$

g(x) se ha definido en Matlab en la función *quadfunction.m*. La integración por cuadratura de Lobatto (cuadratura gaussiana con peso constante de 1) se realiza en Matlab por medio del comando *quadl*. El código del cálculo de $N(\mu, \sigma)$ se encuentra en *centralMomentGaussian.m*.

1.1. Salida del programa y evaluación

```
>> mu = 0; sigma = 1;
   N = 8;
   for n = 1:N;
   moment(n) = centralMomentGaussian(mu,sigma,n);
   end
   moment

moment =
4.0305e-22

moment =
1.0000

moment =
4.0305e-20

moment =
3.0000

moment =
4.0305e-18

moment =
15.0000

moment =
-4.1039e-17

moment =
105.0000

moment =
0.000   1.000   0.000   3.000   0.000   15.000   -0.000   105.000
```

Se observa que el primer y segundo momento ofrecen los valores esperados de cero y varianza, respectivamente. Además, dada la simetría de la función a integrar, g(x), y la simetría del dominio de integración, se comprueba que los momentos de orden impar se anulan.

1.2. Comparación con simulación por Montecarlo

La simulación por el método de Montecarlo ofrece unos resultados similares:

- El histograma de los momentos de orden impar se concentra sobre cero.
- El histograma de momentos de orden 2, se concentra sobre la varianza.

2. Propiedades browniano aritmético

El **browniano aritmético**, también llamado proceso de Wiener con deriva μ y varianza σ^2, es de la forma:

$$\mathrm{d}S_t = \mu \mathrm{d}t + \sigma \mathrm{d}W_t \tag{\textbf{ABM}}$$

Se muestra también el **browniano geométrico**, para ver las diferencias existentes en la formación de la ecuación diferencial estocástica:

$$\frac{\mathrm{d}S_t}{S_t} = \mu \mathrm{d}t + \sigma \mathrm{d}W_t \tag{\textbf{GBM}}$$

2.1. Distribución del browniano aritmético

El browniano aritmético se caracteriza por:

$$\mathrm{E}[S(t)] = S_0 + \mu t \tag{Media}$$

$$\mathrm{E}[(S(t) - \mathrm{E}[S(t)])^2] = \sigma^2 t \tag{Varianza}$$

Por lo tanto, en la gráfica debemos observar una tendencia ascendente junto con una desviación parabólica de las diferentes trayectorias del browniano aritmético:

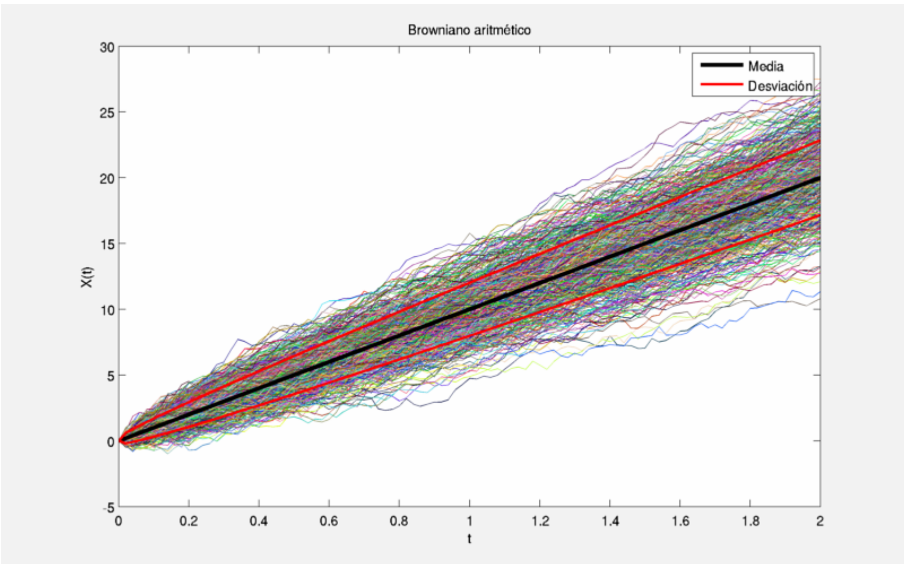

Figura 1: Browniano aritmético

NOTA: *todas las gráficas representadas en este problema han sido obtenidas con los parámetros por defecto que se especifican en los casos de uso de cada uno de los programas suministrados en los anexos.*

2.2. Diagrama de autocovarianzas

Por la definición de autocovarianza:

$$\mathrm{C}_{SS}(t,s) = \mathrm{E}[(S(t) - \mathrm{E}[S(t)])(S(s) - \mathrm{E}[S(s)])]$$

donde si $t = s$ la autocovarianza es la varianza, que ya conocemos a priori por los resultados de la sección anterior y servirá para comprobar si los resultados tienen (algo) de sentido. Para ello comprobaremos que el segundo valor del vector de autocovarianzas es $\sigma^2 \mathrm{d}T$.

Además de la observación anterior, sabemos la autocovarianza del ABM:

$$\mathrm{C}_{SS}(t,s) = \sigma^2 min(t,s) \tag{Autocovarianza}$$

luego, si s es un intervalo en el que t está contenido, la autocovarianza se saturará cuando s sobrepase a t. Por ejemplo, la discretización de la simulación para $n_1 = 30$ se correspondería con t y s estaría relacionado con la discretizaciones en un intervalo de $n_2 \in [1, 50]$.

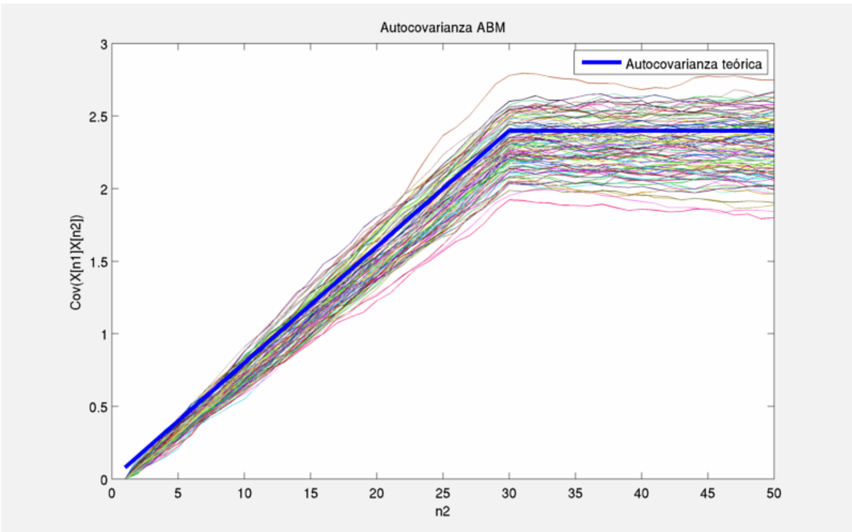

Figura 2: Autocovarianza para un ABM

como así hemos observado en la simulación.

2.3. Código empleado

Se han codificado tres archivos en Matlab:

- **simBM.m**
  Función que calcula el browniano arimético.
- **autocovBM.m**
  Función que calcula una matriz de autocovarianzas para un intervalo de tiempo dado sobre una referencia temporal.
- **plotautocovBM.m**
  Pequeño programa para dibujar de una manera más clara la autocovarancia calculada por medio de las simulaciones y la comparación con la teórica.

3. Valoración de productos derivados

3.1. Valoración de una call europea por cuadratura

Para este problema procedemos idénticamente a cómo lo realizamos para el problema 1: identificamos la función a integrar y realizamos una integración por Lobatto.

Antes de obtener el precio, que es algo inmediato, analizamos de un modo un tanto grosero la *call* europea con los parámetros dados para su simulación.

Primeramente observamos el precio del ejercicio a día de hoy que es:

$$K_{hoy} = \mathrm{e}^{-rT} K = 84{,}7588$$

Luego comparamos con el precio a día de hoy del máximo, valor medio y mínimo de la evolución del subyacente por Black-Scholes:

Valor del subyacente a día de hoy	
Máximo	150.0329
Medio	114.5944
Mínimo	85.2144

Es decir, vamos a pagar menos que el valor mínimo estimado, todo ello referido a día de hoy. Podemos pensar que va a ser una operación beneficiosa. Por último, ojeamos un histograma del subyacente referido al presente. Para su elaboración se tomaron 10^6 muestras de una distribución $\mathrm{N}(0,1)$:

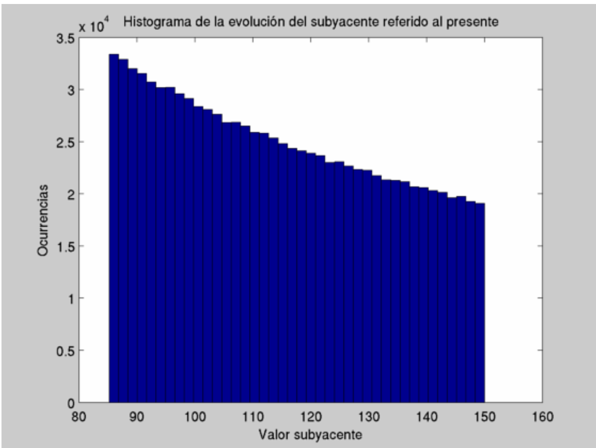

Figura 3: Evolución del subyacente referido al presente

Finalmente computamos el precio resultando 15,2187. La operación será beneficiosa.

3.2. Valoración de una call europea por simulación

A. Problema 1

A.1. centralMomentGaussian.m

```
function moment = centralMomentGaussian(mu,sigma,n)
%centralMomentGaussian: central moment for Gaussian distribution
%
%SYNTAX: moment = centralMomentGaussian(mu,sigma,n)
%
%INPUT:    mu     : Average of the distribution
%          sigma  : Standard deviation
%          n      : Order of the moment
%
%STEPS
%   1.- We create the quadrature function to be integrated
%   2.- We define bounds for integration
%   3.- Lastly, we quadrate the function via quadl
%CHECKS
%   A.- The first moment is always zero
%   B.- The second moment equals the variance
%
%EXAMPLE:
%   mu = 0; sigma = 1;
%   N = 8;
%   for n = 1:N;
%       moment(n) = centralMomentGaussian(mu,sigma,n);
%   end
%   moment

    TOL        = 1e-6; % Automatically set precision target

    R          = 10;
    lowerBound = mu - R*sigma;
    upperBound = mu + R*sigma;
    x          = -100:0.1:100;

    moment     = quadl(@(x)quadfunction(x, mu, sigma, n), lowerBound,
         upperBound, TOL)
    plot(x, moment);
```

Listing 1: centralMomentGaussian.m

A.2. quadfunction.m

```
function y = quadfunction(x, mu, sigma, n)
    y = (x - mu).^n.*normpdf(x, mu, sigma);
```

Listing 2: quadfunction.m

B. Problema 2

B.1. simBM.m

```
function Z = simBM(M,X0,N,dT,mu,sigma)
    %simBM : Simulation for Brownian motion
    %
    %SYNTAX:
    %   X = simBM(M,X0,N,dT,mu,sigma)
    %
    %   X                   : Matrix (M,(N+1)) containing M simulated
    %                         trajectories, each of length N+1
    %   X0                  : Initial value of the simulation
    %   M                   : Number of simulated trajectory
    %   N                   : Number of steps in the simulation
    %   dT                  : Size of time step in simulation
    %   mu,sigma            : Parameters of the GWN process
    %
    %   Simulation parameters
    %     M       = 500;
    %     X0      = 0;
    %     N       = 1e2;
    %     dT      = 2e-2;
    %     mu      = 10;
    %     sigma   = 2;
    %     BM = simBM(M,X0,N,dT,mu,sigma);
Z        = zeros (M, N+1);
X        = randn (M, N+1);
Z(:, 1) = X0;
for n = 1:N
    %And here we place the arithmetic brownian motion
    Z(:, n+1) = Z(:, n)+mu*dT+(sigma*sqrt(dT)).*X(:,n);
    %Here we place the geometric brownian motion, as well
    %Z(:, n+1) = Z(:, n)*(1+mu*dT)+Z(:, n)*(sigma*sqrt(dT)).*X(:,n);
end
x_axis = 0:dT:N*dT;
hFig = figure(1);
set(hFig, 'Position', [100 100 800 500], 'Color', [0.955 0.955 0.955])
for m = 1:M
    plot(x_axis,Z(m,:), 'Color', rand([3,1]));
    hold on;
end
%Now we overlap the average value of the ABM
mean_ABM = (0:mu*dT:N*mu*dT);
dev_ABM_upper = zeros(N,1);
dev_ABM_lower = zeros(N,1);
for n = 2:(N+1)
    dev_ABM_upper(n)  =  mean_ABM(n) + sqrt(sigma^2*dT*n);
    dev_ABM_lower(n)  =  mean_ABM(n) - sqrt(sigma^2*dT*n);
end
mean_plot = plot(x_axis, mean_ABM, 'k', 'linewidth', 3);
    hold on;
dev_plot = plot(x_axis, dev_ABM_upper, 'r', 'linewidth', 2);
    hold on;
plot(x_axis, dev_ABM_lower, 'r', 'linewidth', 2);
    hold off;
    legend([mean_plot, dev_plot], 'Media', 'Desviaci n');
title('Browniano aritm tico');
xlabel('t'); ylabel('X(t)');
```

Listing 3: simBM.m

B.2. autocovBM.m

```
function [Z, meanABM_tstep0, meanABM_tsweep, sampledeviation_tstep0, sampledeviation_tsweep] = autocovBM(ABM, tstep0, tsweep0, tsweep1)
    %autocovBM : Returns the autocovariance matrix of a ABM
    %
    %SYNTAX:
    %   Z = autocovBM(BM, rows, columns);
    %
    %   ABM      : ABM of [rows] trajectories and [columns] time-steps
    %   tstep0   : Reference time-step to perfom autocovariance
    %   tsweep0  : Lower bound time-step to perform autocovariance
    %   tsweep1  : Upper bound time-step to perform autocovariance
    %
    %STEPS
    % 1.- Calculate sample mean two given time-steps (tstep0 and tstep1)
    % 2.- Calculate sample deviation of all the trajectories of the ABM
    %      at two given time-steps (tstep0 and tstep1)
    %
    %Here we calculate the sample mean at time-step tstep0 and tstep1
    [rows, columns] = size(ABM);
    meanABM_tstep0 = (1/rows)*sum(ABM(:, tstep0));
    meanABM_tsweep = zeros(1, tsweep1 - tsweep0 + 1);
    for sweep = 1:(tsweep1-tsweep0+1)
       meanABM_tsweep(sweep) = (1/rows)*sum(ABM(:, sweep));
    end

    %Here we calculate the arrays of sample deviations
    sampledeviation_tstep0 = ABM(:, tstep0) - meanABM_tstep0;
    sampledeviation_tsweep  = zeros(rows, tsweep1 - tsweep0 + 1);
    for sweep = 1:(tsweep1-tsweep0+1)
       sampledeviation_tsweep(:,sweep) = ABM(:, sweep) - meanABM_tsweep(sweep);
    end
    Z = zeros(tsweep1 - tsweep0 + 1, 1);
    %Finally we calculate the autocovariance
    for sweep = 1:(tsweep1-tsweep0+1)
       Z(sweep) = (1/rows)*sum(sampledeviation_tstep0.*sampledeviation_tsweep(:,sweep));
    end
```

Listing 4: simBM.m

B.3. plotautocovBM.m

```
function graph = plotautocovBM(M,X0,N,dT,mu,sigma,tstep0, tsweep0, tsweep1, nplotcov)
    %plotautocovBM : functions that plots together several autocovariances of ABMs
    %
    %M            :  Number of simulated trajectory
    %X0           :  Starting point for the ABM
    %N            :  Number of steps in the simulation
    %dT           :  Size of time step in simulation
    %mu, sigma    :  Parameters of the GWN process
    %tstep0       : Time step for the reference autocovariance ABM sample
```

```
    % tsweep0     : Initial time step for the sweep ABM used in the autocovariance
    % tsweep1     : End time step for the sweep ABM used in the autocovariance
    % nplotcov    : Number of ABM autocovariance plots to be displayed
    %
    % SYNTAX:
    %   Z = autocovBM(BM, rows, columns);
    %
    %  ABM       : ABM of [rows] trajectories and [columns] time-steps
    %   tstep0   : Reference time-step to perfom autocovariance
    %   tsweep0  : Lower bound time-step to perform autocovariance
    %   tsweep1  : Upper bound time-step to perform autocovariance
    %
    % SIMULATION PARAMETERS
    %     M         = 500;
    %     X0        = 0;
    %     N         = 1e2;
    %     dT        = 2e-2;
    %     mu        = 10;
    %     sigma     = 2;
    %     tstep0    = 30;
    %     tsweep0   = 1;
    %     tsweep1   = 50;
    %     nplotcov  = 100;
    %
    %     plotautocovBM(M,X0,N,dT,mu,sigma,tstep0, tsweep0, tsweep1, nplotcov)
hFig = figure(1);
set(hFig, 'Position', [100 100 800 500], 'Color', [0.955 0.955 0.955])
autocov = zeros(tsweep1-tsweep0+1, nplotcov);
for i = 1:nplotcov
    BM       = simBM(M,X0,N,dT,mu,sigma);
    autocov(:,i) = autocovBM(BM, tstep0, tsweep0, tsweep1);
end
exact_autocov = zeros(tsweep1-tsweep0+1, 1);
for i = 1:tsweep1-tsweep0+1
    exact_autocov(i) = sigma^2*min(tstep0, i-1+tsweep0)*dT;
end
x_axis = tsweep0:tsweep1;
for i = 1:nplotcov
    plot(x_axis, autocov(:,i),'Color', rand([3,1]));
    hold on;
end
exact_plot = plot(x_axis, exact_autocov, 'b', 'linewidth', 3);
hold off;
title('Autocovarianza ABM');
xlabel('n2'); ylabel('Cov(X[n1]X[n2])');
legend(exact_plot, 'Autocovarianza te rica');
```

Listing 5: simBM.m

C. Problema 3