Máster i2-TIC: Procesamiento de información temporal **2013/02/09**

Entrega: 2013/02/21, 23:55

PROGRAMA EJEMPLO:

Programa para comprobar que la función Matlab que permite calcular la densidad de probabilidad Gaussiana (normpdf) está normalizada

En fichero: demoNormGaussian.m

```
function norm = demoNormGaussian(mu,sigma,tol)
%demoNormGaussian: normalization constant of Gaussian pdf
%
% SYNTAX:   norm = demoNormGaussian(mu,sigma,tol)
%
%          norm  : Normalization of the pdf (should be equal to 1)
%          mu    : Average of the distribution
%          sigma : Standard deviation
%          tol   : Target absolute error of the norm estimate
%
% EXAMPLE 1:
%          m = 0; s = 1;
%          norm = demoNormGaussian(m,s)
%
% EXAMPLE 2:
%          m = -2; s = 3; tol = 1e-10;
%          format long
%          norm = demoNormGaussian(m,s,tol)
%          format short
%
if(nargin ==2)
    tol = 1e-6; % Automatically set precision target
end
%
R = 10; % ~ norminv(1-eps);  % Approximately infty for N(0,1) variable
lower = mu - R*sigma;  % lower limit of effective support interval
upper = mu + R*sigma;  % upper limit of effective support interval
%
% Calculate norm by numerical quadrature
norm = quadl(@(x)integrandNormGaussian(x,mu,sigma),lower,upper,tol);
```

En fichero: integrandNormGaussian.m

```
function  y = integrandNormGaussian(x,mu,sigma)
%integrandNormGaussian: integrando para demoNormGaussian
%
z = (x-mu)/sigma;
y = exp(-0.5*z.*z)/(sqrt(2*pi)*sigma);
```

1. Escribid un programa en Matlab (función principal y funciones auxiliares) que calcule el momento central de orden n para una variable N(μ,σ) mediante cuadratura numérica.

   Funciones a usar: quadl, normpdf

   La función principal debe tener la cabecera

```
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
%
% 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
%
<CÓDIGO DE FUNCIÓN PPAL>
```

   Comparar con la estimación de momentos por Montecarlo

   En fichero: demoCentralMomentGaussianMC.m

```
% demoCentralMomentGaussianMC
normalized = 1; nBins = 40;  % Number of bins for histogram
B = 1e3;     % number of samples generated
M = 1e4;     % size of each sample
mu = 0.0; sigma = 1.0;
X = mu + sigma * randn(B,M); % B samples of size M
for n=1:16
    % B MC estimates of central moment from sample of size M
    moment = mean((X-mu).^n,2);
    avg_moment(n) = mean(moment);
    std_moment(n) = std(moment);
    graphicalComparison(moment,@normpdf,nBins,n,...
        normalized,avg_moment(n),std_moment(n))
    title(sprintf('Moment of order %d',n));
end
avg_moment
```

PROGRAMA EJEMPLO: Simulación de ruido blanco Gaussiano

```
function Z = simGWN(M,N,mu,sigma)
% simGWN : Simulation for Gaussian White Noise
%
% SYNTAX:
%   Z = simGWN(M,N,mu,sigma)
%
%   Z : Matrix (M,N) containing M simulated trajectories,
%        each of length N
%   M : Number of simulated trajectory
%   N : Length of each trajectory
%   mu,sigma : Parámeters of the GWN process
%
% EXAMPLE 1:
%     M = 3; N = 100;
%     mu = 0; sigma = 1;
%     Z = simGWN(M,N,mu,sigma);
%     figure(1); plot(X'); % Plot all trajectories
%     axis([1 Inf -Inf Inf])
%
Z = zeros(M,N);  % Reserve memory
for n = 1:N
    Z(:,n) = randn(M,1); % Update trajectory at each time step.
end
```

2. Simulación de un movimiento Browniano aritmético.

```
function X = 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
```

2.1 Demuestra con los resultados de tu simulación (gráficas) que el browniano aritmético sigue una distribución

$$N(X_0+\mu t,\sigma^2 t)$$

es decir, una normal cuya media es $(X_0+\mu t)$ y cuya varianza es $\sigma^2 t$

2.2 Calcula el diagrama de autocovarianzas.

Es decir, el gráfico:

$$E[(X_{n1}- E[X_{n1}]) (X_{n2}- E[X_{n2}])];$$

Para un n fijo (por ejemplo n1=30, variando n2 (por ejemplo entre 1 y 50)

Memoria:

- **Código.**
- **Gráficas:** Comparaciones entre la media, varianza, autocorrelaciones empíricas.
- **Comentarios sobre implementación.**

3. Valoración de productos derivados dentro del modelo Black-Scholes.

En finanzas, un producto derivado consiste en una transacción que involucra el precio de un activo (el subyacente. Por ejemplo, acciones) en el futuro.
Muchos de estos productos son opciones. En una opción el comprador adquiere el derecho a realizar una transacción que involucra al subyacente en el futuro, pero cuyas condiciones se especifican en el presente. Se denomina vencimiento o tiempo de vida de la opción (T) al plazo máximo en el que el comprador debe elegir ejercer o no dicha opción.

El producto es de tipo **americano** si el derecho puede ser ejercido en cualquier momento durante la vida de la opción. Es de tipo **europeo** en caso de que el derecho sólo puede ser ejercido a vencimiento.

Las opciones de tipo **vainilla** son productos con condiciones estandarizadas, que únicamente involucran comprar (**call**) o vender (**put**) una cantidad especificada de subyacente, a un precio también prefijado.

Las especificaciones de las opciones **exóticas** son más complejas. Ejemplos de exóticas son las opciones con **barrera,** en las que el derecho a realizar la transacción depende de si el subyacente ha alcanzado una determinada cota (por arriba o por abajo) durante el tiempo de vida de la opción. Otro tipo de exóticas son las **asiáticas,** en las que el pago que se realiza a vencimiento depende de toda la trayectoria del subyacente, no solo de su precio final. Ejemplos de asiáticas son aquellas en las que los pagos se realizan en función de la **media aritmética** de precios del producto durante el tiempo de vida de la opción. De manera análoga, se pueden definir opciones asiáticas cuyo pago dependa de la **media geométrica**.

Call europea

La adquisición de una call europea otorga el derecho al comprador de comprar una cantidad dada de subyacente (precio inicial: S_0 ; volatilidad: σ) a vencimiento (T) por un precio de ejercicio (K) prefijado.

La evolución del subyacente es un proceso Browniano geométrico, de acuerdo con el modelo de Black-Scholes:

$$S(T) = S_0 \exp\left[\left(r - \frac{1}{2}\sigma^2\right)T + \sigma\sqrt{T}X\right]; \qquad X \sim N(0,1),$$

donde *r* es el tipo de interés libre de riesgo.

El pago descontado (es decir, con un precio expresado en moneda en el presente, no en el futuro) en una call europea es

$$\text{pago}\big(S(T;X)\big) = e^{-rT}\big(S(T;X) - K\big)_+ = e^{-rT}\max\big(S(T;X) - K, 0\big)$$

Valoración de una call europea por cuadratura

$$\text{precio} = \mathbf{E}_X\left[pago(S(T;X))\right] = e^{-rT}\int_{-\infty}^{\infty} dx\,(S(T;x)-K)_+\, normpdf(x)$$

Diseñar una función de Matlab para realizar valoración de una call europea por cuadratura

```
function precio = precioCallEU(S0,K,r,T,sigma)
%precioCallEU: Precio de una call europea
%
% SINTAXIS:
%    precio = precioCallEU(S0,K,r,T,sigma)
%
%  precio: Precio de una call europea
%     S0 : Precio inicial del subyacente
%      K : Precio de ejercicio (Strike)
%      r : tipo de interés libre de riesgo
%      T : tiempo de vencimiento
%  sigma : volatilidad
%
% EJEMPLO:
%  S0 = 100; K = 90; r = 0.03; T = 2; sigma = 0.4;
%  precioCallEU(S0,K,r,T,sigma)
%

[CÓDIGO: Utilizad la función quadl]
```

Valoración de una call europea por simulación

(i) Simular M trayectorias del browniano geométrico entre [0,T] en N pasos

$$\Delta T = \frac{T}{N}, \quad t_n \equiv n\,\Delta T, \quad S_n \equiv S(t_n)$$

$$S_0^{(m)} = S_0;$$

$$S_n^{(m)} = S_{n-1}^{(m)} \exp\left[\left(r-\frac{1}{2}\sigma^2\right)\Delta T + \sigma\sqrt{\Delta T}\,X_n^{(m)}\right]; \quad X_n^{(m)} \sim N(0,1) \qquad n = 1,2,\ldots,N;$$

$$m = 1,2,\ldots,M$$

(ii) Calcular los pagos descontados a vencimiento ($t_N = T$; S_N=S(T)) para cada trayectoria

$$pago\left(S_N^{(m)}\right) = e^{-rT}\max\left(S_N^{(m)}-K,0\right); \quad m = 1,2,\ldots,M$$

(iii) El precio se calcula como el promedio sobre las trayectorias simuladas de los los pagos descontados

$$\text{precio}_M = \frac{1}{M}\sum_{m=1}^{M} \text{pago}\left(S_N^{(m)}\right), \quad m = 1,2,\ldots,M$$

precio_M es una variable aleatoria cuya distribución, por el teorema central del límite se aproxima a una Gaussiana cuando $M \to \infty$. Por esta razón se puede

tomar como una estimación del error la desviación estándar de esta variable aleatoria

$$\text{error}_M \approx \frac{1}{\sqrt{M}} \text{stdve}\left[\left\{\text{pago}\left(S_N^{(m)}\right)\right\}_{m=1}^{M}\right]$$

Diseñar una función de Matlab para realizar valoración de una call europea por simulación

```
function [precio,err] = precioCallEU_MC(M,S0,K,r,T,sigma)
%precioCallEU_MC: Precio de una call europea mediante MC
%
% EJEMPLO 1:
%  S0 = 100; K = 90; r = 0.03; T = 2; sigma = 0.4;
%  M = 1e4;
%  [precio,err] = precioCallEU_MC(M,S0,K,r,T,sigma)
%  blsprice(S0,K,r,T,sigma)
%
% EJEMPLO 2:
%  S0 = 100; K = 90; r = 0.03; T = 2; sigma = 0.4;
%  M = 1e4; B = 10000;
%  for b =1:B
%     [precio(b), err(b)] =
precioCallEU_MC(M,S0,K,r,T,sigma);
%  end
%
%  % La distribución de precios estimados por MC es
Gaussiana
%  % con los parámetros
%  media = mean(precio); desvEst = mean(err);
%  figure(1); nBins = 40; hist(precio,nBins);
%  nPlot = 1e3; a = 4;
%  xPlot = linspace(media-a*desvEst,media+a*desvEst,nPlot);
%  factor = (max(precio)-min(precio))*B/nBins;
%  yPlot = factor*normpdf(xPlot,media,desvEst);
%  hold on; plot(xPlot,yPlot,'r','linewidth',2); hold off;
%
```

Valoración de una call asiática de media aritmética por simulación

El pago descontado de este producto derivado es

$$pago\left(\left\{S_n^{(m)}\right\}_{n=0}^{N}\right) = e^{-rT}\max\left(\frac{1}{N}\sum_{n=1}^{N} S_n^{(m)} - K, 0\right); \quad m = 1,2,\ldots,M$$

El procedimiento de valoración es similar al de una call europea, pero utilizando estos pagos.

Diseñar una función de Matlab para realizar valoración de una call europea por simulación

```
function[price,err]=Asiatica_Call_Aritmetica_BS_MC(S0,K,r,T,sigma,N,M)
% Asiatica_Call_Aritmetica_BS_MC: Call Asiática por MC
%
% Sintaxis:
%
% [precio,err]  = Asiatica_Call_Aritmetica_BS_MC(S0,K,r,T,sigma,N,M)
%
%        S0    : Valor inicial del subyacente
%        K     : Strike
%        r     : Tipo de interés anual
%        T     : Vencimiento de la opción
%        sigma : Volatilidad
%        N     : Número de pasos en cada trayectoria
%        M     : Número de trayectorias
%
%        precio: Estimación MC del precio
%        error:  Estimación MC del error en el precio
%
%  Ejemplo:
%  S0 = 100; K = 110; r = 0.09; T = 1; sigma = 0.5;
%  N = 52; M = 1e4;
%  [precio,error] = Asiatica_Call_Aritmetica_BS_MC(S0,K,r,T,sigma,N,M)
%
%
```