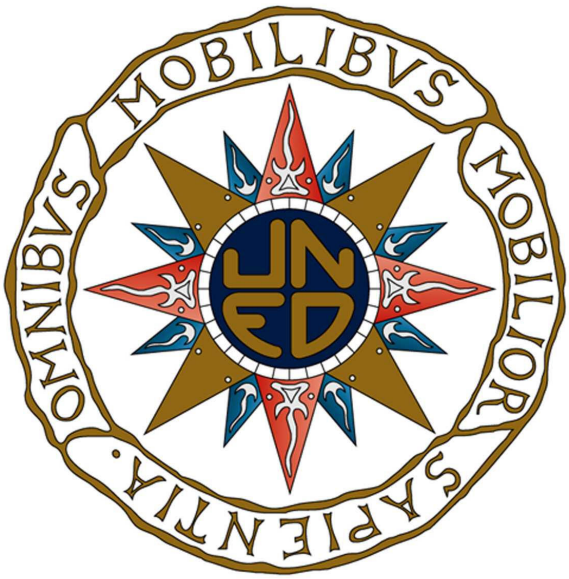

UNED
FACULTAD DE CIENCIAS

DEPARTAMENTO DE MATEMÁTICAS Y ESTADÍSTICA

TRABAJO DE MÁSTER

Aplicaciones del método de Monte Carlo en mercados financieros

Un trabajo realizado por **Rubén Colomina Citoler** para el
Máster en Matemáticas Avanzadas de la
Universidad Nacional de Educación a Distancia.

Supervisado por el Doctor:
Carlos Moreno González

---

22 de octubre de 2013

Agradecimientos

Después de un trabajo tan arduo como éste, no puedo olvidarme de las personas que me han apoyado y animado durante todos estos meses, en los que me he encerrado literalmente, para poder llegar a tiempo a la fecha de entrega. No hubiera sido posible sin vosotros. Doy las gracias:
A mis padres, Pedro y María Josefa, por su cariño, amor y apoyo moral.
A mi hermano Jose Manuel, por su amor y entusiasmo, además de sus correcciones tipográficas y de estilo.
A mi novia Andreza, por su amor y preocupación continua.
A mis amigos matemáticos de la Universidad de Zaragoza, por el interés mostrado.
A mis amigos y comprades de Zaragoza, por sus ánimos.
A mis amigos de Madrid, por su admiración y apoyo.
A mi primo Martín matemático, por su ayuda sin saberlo, en una solución teórica.
Al resto de mi familia por escucharme.
A mis compañeros de trabajo de Indra, quienes ayudaron a mejorar mi programación.
A mis socios, por las prórrogas.
Al resto del mundo por existir ...

Índice general

Índice de figuras

Capítulo 1

Introducción

El actual documento es la memoria del trabajo final, del máster en matemáticas avanzadas impartido en la universidad nacional de ecuación a distancia UNED. El enfoque esta dirigido al desarrollo de aplicaciones de cálculo numérico para ejecutar métodos de tipo **Monte Carlo (MC)** y **Quasi-Monte Carlo (QMC)** sobre ciertos problemas de los mercados financieros, en particular, la valoración de contratos de opciones financieras.

El objetivo principal perseguido, ha sido el desarrollo y aplicación de software orientado al muestreo pseudo-aleatorio y quasi-aleatorio. Se han realizado simulaciones estocásticas de precios económicos para valorar opciones financieras, sin pretender ser exhaustivo, pasando por todos los tipos de opciones existentes, debido a su amplio espectro. Las simulaciones se han realizado para los tipos de opciones más sencillas, *Vanilla*, las conocidas **opciones Europeas**, y para algunas *Exóticas*, como las **Asian options**, **Spread options** y **Lookback options**.

1.1. Contexto y problemática

Los problemas de integración en varias variables en ingeniería o ciencia, suelen tener una alta complejidad analítica a medida que aumenta el número de dimensiones. Los métodos numéricos pueden ser ineficaces, debido al efecto, *curse of dimensionality*, sobre el orden de convergencia, volviéndose extremadamente lento con la dimensión. Por contra, los métodos de MC consiguen evitar este problema, independizándose de la dimensión.

En este trabajo, se hablará de la eficiencia de QMC respecto a MC en bajas dimensiones utilizando conjuntos especiales de puntos, con propiedades de baja discrepancia [1]. Se estudiará su orden de convergencia, y de cómo se ven afectados por ciertas transformaciones no lineales de los dominios de integración de las funciones consideradas, mediante funciones de periodización, sobre el hipercubo $[0,1)^s$, siendo s la dimensión.

[1] En el capítulo 3 se define este concepto con detalle.

1.2. Punto de partida

Las referencias fundamentales y el punto de partida para el trabajo actual, han sido los artículos [1] y [2] de la bibliografía. En ellos, se detallan en profundidad la utilización de métodos QMC mediante reglas **Lattice Rules**, para el primero, y la utilización de estimadores MC deducidos mediante ciertas metodologías, llamadas **Pathwise** y **Likelihood Ratio**, para el segundo.

En las referencias, se presentan resultados numéricos para los distintos casos, junto a sus conclusiones, aunque apenas se detallan las implementaciones de los algoritmos, y la preparación de los problemas en ciertos casos, para ser sometidos a los métodos discutidos.

En las aplicaciones implementadas, en primera instancia, se ha buscado replicar los cálculos realizados en [1] y en [2], tratando de investigar las dificultades finales de implementación de los algoritmos deducidos. Se han comprobado y contrastado los resultados de los autores, obtenidos en problemas de valoración de precios de opciones, y en problemas de cálculo de las griegas de opciones, con otras fuentes de cálculo, como software existente y algoritmos nuevos creados para este trabajo.

1.3. Énfasis del trabajo

A partir de los desarrollos teóricos de las referencias [1] y en [2], se han diseñado algoritmos utilizando el lenguaje *Octave*. En algunos casos, se ha tratado de replicar exactamente los resultados numéricos, obtenidos por sus autores, con el objetivo de verificar la corrección de los algoritmos nuevos implementados, determinando su utilidad en una aplicación práctica.

Una vez contrastadas las implementaciones de estos algoritmos con los resultados de los artículos, se ha tratado de extender las aplicaciones de los programas creados, a otros conjuntos de condiciones iniciales o parámetros de partida de los problemas tratados. Además, se ha buscado hacer comparativas de los métodos en tiempos, y en precisión, en su aplicación a problemas de opciones financieras, más otros problemas de ejemplo, susceptibles de ser solucionados con MC y QMC.

1.4. Las simulaciones

El origen del término **simulación** está asociado a Von Neuman y Ulam, alrededor de 1949. Según la definición de Shannon en 1975, la simulación es el proceso de diseñar un modelo de un sistema real y llevar a cabo experiencias con él, con la finalidad de aprender el comportamiento del sistema original o de evaluar diversas estrategias para el funcionamiento del sistema.

En la práctica, la simulación, es el método de cálculo cuando no existen expresiones analíticas exactas para solucionar un problema. Desde este punto, es necesario utilizar métodos numéricos o de muestreo aleatorio. En el actual trabajo, se han aplicado a modo de pruebas iniciales, y de contraste, métodos MC y QMC para problemas, que sí poseían modelos teóricos exactos. Por ejemplo, para la *Spread Option*, existe la fórmula de Margrabe, la cual, permite evaluar la opción de forma exacta en un caso muy concreto de sus parámetros. Por otro lado, es obligatorio nombrar el famoso modelo de *Black-Scholes*, para la valoración de opciones Europeas de forma exacta.

Una vez ajustadas las implementaciones de los algoritmos, en su aplicación a problemas con modelos exactos, éstas han sido aplicadas con más seguridad a otros problemas. En el caso de *Spread Options* sin expresiones analíticas, se ha tomando como referencia teórica, simulaciones MC con un número grande de muestras junto a la estimación del error estándar[2].

1.5. Entorno de programación: Octave

El entorno de programación elegido en este trabajo ha sido *Octave*, siendo la versión open-source del paquete comercial de software *MatLab*. Las características de este lenguaje están orientadas al cálculo numérico y cuya sintaxis está centrada en el manejo de vectores, matrices y funciones matemáticas predefinidas, por el programa o por el usuario. Todos los números tratados por el sistema de cálculo, están almacenados en doble precisión.

Se ha elegido este lenguaje por comodidad respecto a la exportación de los resultados al procesador de texto LaTeX, además de su facilidad en el uso de funciones matemáticas. También se ha barajado el uso de C++, pero se descartó por ser demasiado genérico, necesitando librerías matemáticas adicionales, reservándose para futuras aplicaciones multiprocesador, ya que MC y QMC son susceptibles de ser abordados por esta tecnología de forma más eficiente. También se consideró Excel(VBA), pero no presenta una buena integración con el procesador de texto, siendo software propietario, y por tanto, necesitando de licencia para su utilización.

Octave es un paquete informático que permite trabajar en modo interactivo, guardando las variables declaradas, en un espacio de trabajo. Además, también es posible trabajar en modo *Batch* o procesamiento por lotes, utilizando para ello, *scripts* y/o *funciones*. Una condición para las funciones, es que deben tener el mismo nombre, que el fichero con las contiene. Los ficheros tendrán extensión **.m**, por ejemplo, **nombre_script.m** y **nombre_funcion.m**.

[2]El error estándar es la desviación estándar de un estadístico. En particular, se define error estándar de la media, como $SE_{\overline{x}} = \frac{s}{\sqrt{n}}$, donde s es la desviación estándar de X, y n el número de muestras.

Octave permite anidar, *scripts* y funciones, produciendo ejecuciones en cascada de forma ordenada. También es posible diseñar una programación estructurada y orientada a objetos, definiendo estructuras y clases de objetos e interfaces, para su reutilización en otros *scripts* o funciones.

En este trabajo, se han codificado una serie de *scripts* de prueba, que permiten la ejecución de otros *scripts* y/o funciones en cascada, utilizando el código fuente *Octave* del anexo A. Es posible rehacer todos los cálculos presentados en este documento, gracias a que en cada sección, se explica detalladamente cuales son los *scripts* y/o *funciones*, y con qué parámetros, es necesario su ejecución.

1.6. Hipótesis de partida

El trabajo actual, requiere de suponer la hipótesis del mercado eficiente sobre los modelos de precios supuestos de los mercados financieros. Esta hipótesis se basa en algunos conceptos previos que habría que definir, como una estrategia de arbitraje, el precio descontado de un activo y el concepto de martingala para un proceso estocástico.

1.6.1. Estrategia de arbitraje

Una estrategia de **arbitraje** en los mercados financieros, es la práctica de tomar ventaja de la diferencia de precios entre dos o más mercados relacionados. En otras palabras, se trata de realizar una combinación de transacciones complementarias, que capitalizan la diferencia de precios entre los mercados .

Mas formalmente, una estrategia de arbitraje, es una estrategia admisible y autofinanciada con un valor inicial cero de su cartera de activos, un valor no negativo en todo momento y una probabilidad estrictamente positiva de que el valor final sea positivo.

1.6.2. Precio descontado

El **precio descontando** $\tilde{S}_n$ de un activo, en un momento n del futuro, es el precio resultante de descontar al precio S_n,la cantidad generada por unidad de activo sin riesgo con tipo de interés anual fijo r, es decir, $\tilde{S}_n = S_n(1+r)^{-n}$.

También es posible expresarlo en términos del interés continuo. Es sabido que la tasa anual equivalente es: $TAE = (1+\frac{r}{m})^m = (1+\frac{1}{m/r})^{r\cdot m/r}$, con m el número de momento de cobro de la tasa de interés r. Ahora, si se hace tender $m \to \infty$, entonces $TAE \to e^r$, obteniendo el interés continuo para un periodo $T = 1$ de un año y un tipo de interés r anual. De esta forma, el precio descontado, para un momento $0 \leq t \leq T$ será $\tilde{S}_t = e^{-r(T-t)}S_t$. La expresión anterior será la más utilizada en el trabajo actual.

1.6.3. Martingalas

A *grosso modo*, una **martingala**, es un proceso estocástico, cumpliendo que con toda la información de mercado existente en el momento n, sólo cabe esperar que los precios de S_n, sean los mismos que los precios en el momento, S_{n+1}.

Formalmente, considerando un espacio de probabilidad filtrado discreto [3] , $(\Omega, \mathscr{F}, (\mathscr{F}_n)_{n=0}^N, \mathbb{P})$, y una sucesión aleatoria $(X_n)_{n=0}^N$, cumpliendo que, $\forall n, X_n$ es una función $\mathscr{F}_n$ medible [4]. Un proceso estocástico $(X)_{n=0}^N$ real evaluada será:

- Una martingala si $E(X_{n+1}|\mathscr{F}_n) = X_n$ para todo $n \leq N-1$;
- Una supermartingala si $E(X_{n+1}|\mathscr{F}_n) \leq X_n$ para todo $n \leq N-1$;
- Una submartingala si $E(X_{n+1}|\mathscr{F}_n) \geq X_n$ para todo $n \leq N-1$.

Además, es posible extender el concepto a una variable multidimensional, por ejemplo, una secuencia $(X_n^1, \ldots, X_n^d)_{n=0}^N \in \mathbb{R}^d$, será una martingala, si cada componente de la secuencia es una martingala real evaluada.

Ejemplo de Martingala: *Camino aleatorio*

Sea una sucesión de variables aleatorias independientes e idénticamente distribuidas, $(X_i)_{i\geq 1}$, $\forall i$. Formar la sucesión de variables aleatorias $Y_n = \frac{1}{n}\sum_{i=1}^n X_i$.
Sea $\mathscr{F}_n = \sigma(X_1, \ldots, X_n)$, la σ-álgebra generada por los (X_i), entonces la sucesión aleatoria, $(Y_n - nE[X_1])_{n\geq 1}$ será una $\mathscr{F}_n$-martingala. En efecto, por definición de Y_n y linealidad del operador esperanza respecto una σ-álgebra,

$$\begin{aligned} E[Y_{n+1} - (n+1)E[X_1]|\mathscr{F}_n] &= E[X_{n+1} + Y_n - (n+1)E[X_1]|\mathscr{F}_n] \\ &= E[X_{n+1}|\mathscr{F}_n] + E[Y_n|\mathscr{F}_n] - (n+1)E[X_1] \end{aligned}$$

Ahora como Y_n es $\mathscr{F}_n$-medible,

$$E[Y_n|\mathscr{F}_n] = Y_n, a.s., (\text{a.s=almost sure})$$

De esta manera,

$$E[X_{n+1}|\mathscr{F}_n] + E[Y_n|\mathscr{F}_n] - (n+1)E[X_1] = E[X_{n+1}|\mathscr{F}_n] + Y_n - (n+1)E[X_1]$$

[3] Un espacio de probabilidad, $(\Omega, \mathscr{F}, \mathbb{P})$, se dirá filtrado si está equipado con una sucesión incremental de σ-álgebras contenidas en $\mathscr{F}$: $\mathscr{F}_0, \ldots, \mathscr{F}_N$. Una sigma álgebra $\mathscr{F}_n$, puede ser interpretada como la información disponible en el momento n. El momento N, corresponde con la maduración de posibles opciones financieras. Se asume que $\mathscr{F}_0 = \{\emptyset, \Omega\}$, $\mathscr{F}_N = \mathscr{P}(\Omega)$, siendo $\mathscr{P}(\Omega)$, conjunto de todos los subconjuntos de un espacio finito Ω.

[4] Una función entre dos espacios medibles, se dice medible, si la preimagen de cualquier conjunto medible es medible.

Aplicando que las X_i son variables aleatorias independientes, X_{n+1} es independiente de $\mathscr{F}_n = \sigma(X_1, \ldots, X_n)$, luego

$$E[X_{n+1}|\mathscr{F}_n] = E[X_{n+1}], a.s.$$

Además como las X_i están idénticamente distribuidas, $E[X_{n+1}] = E[X_1]$. Se concluye,

$$\begin{aligned} E[X_{n+1}|\mathscr{F}_n] + Y_n - (n+1)E[X_1] &= E[X_{n+1}] + Y_n - nE[X_1] - E[X_1] \\ &= E[X_1] + Y_n - nE[X_1] - E[X_1] \\ &= Y_n - nE[X_1] \end{aligned}$$

Como se quería demostrar.

1.6.4. Marco teórico

La teoría que sustenta los cálculos y algoritmos estudiados, se basa en la **hipótesis del mercado eficiente**. Su explicación detallada, es un tema que se sale del ámbito de este trabajo, pero en teoría, hay que suponer que se está trabajando con un mercado **viable** y **completo**. Son conceptos que tratan sobre un mercado libre de estrategias de arbitraje (mercado viable), en el que existe una única medida de probabilidad equivalente a la medida histórica, bajo la cual, los precios descontados de los activos con riesgo de un mercado, son martingalas (mercado completo).

Adicionalmente, se supone que es posible realizar transacciones financieras en todo momento, pudiendo comprar fracciones de activos y asumiendo que es posible hacer ventas en corto.

1.6.5. Movimiento Browniano

Un fenómeno aleatorio indispensable a conocer para este estudio, y con aplicaciones en problemas a los mercados financieros, es el **movimiento Browniano**. Es aplicado a la modelización de la parte estocásticos de los precios económicos, siendo el punto de partida en la solución de los problemas de valoración de opciones financieras.

El movimiento browniano es el movimiento aleatorio que se observa en algunas partículas microscópicas que se hallan en un medio fluido (por ejemplo, polen en una gota de agua). Recibe su nombre en honor al escocés Robert Brown, biólogo y botánico que descubrió este fenómeno en 1827 y observó que pequeñas partículas de polen se desplazaban en movimientos aleatorios sin razón aparente. El movimiento estocástico de estas partículas se debe a que su superficie es bombardeada incesantemente por las moléculas (átomos) del fluido sometidas a una agitación térmica. Este bombardeo a escala atómica no es siempre completamente uniforme y sufre variaciones estadísticas importantes. Así, la presión ejercida sobre los lados puede variar ligeramente con el tiempo, y así se genera

el movimiento observado.

La descripción matemática del fenómeno fue elaborada por Albert Einstein y constituye el primero de sus artículos del que, en la obra de Einstein, 1905. La teoría de Einstein demostraba la teoría atómica, todavía en disputa a principios del siglo XX, e iniciaba el campo de la física estadística.

El primero en describir matemáticamente el movimiento browniano fue Thorvald N. Thiele en 1880, en un documento sobre el método de los mínimos cuadrados. Fue seguido independientemente por Louis Bachelier en 1900, en su tesis doctoral *Théorie de la spéculation*, en la que se presenta un análisis estocástico de acción y opción de mercados. Sin embargo, fue el estudio independiente de Albert Einstein en su artículo de 1905 el que mostró la solución a los físicos, como una forma indirecta de confirmar la existencia de átomos y moléculas. Desde el punto de vista matemático, fueron Norbert Wiener, Andrei Kolmogorov y Paul Lévy, quienes desarrollaron la teoría de procesos estocásticos.

En esa época la naturaleza atómica de la materia aún era una idea controvertida. Einstein y Marian Smoluchowski dedujeron que, si la teoría cinética de los fluidos era correcta, entonces las moléculas de agua tendrían movimientos aleatorios. Por lo tanto, las partículas pequeñas podrían recibir un número aleatorio de impactos, de fuerza aleatoria y de direcciones aleatorias, en cortos períodos de tiempo. Este bombardeo aleatorio por las moléculas del fluido podría ser suficiente para que las partículas pequeñas se moviesen de la manera exacta que Brown había descrito.

Una metáfora intuitiva puede aclarar el concepto: considerando un gran balón de 10 metros de diámetro. Se puede imagira este balón en un estadio de fútbol o cualquier otra área llena de gente. El balón es tan grande y ligero que permanece por encima de la muchedumbre. Las personas aciertan a golpear el balón en diferentes momentos y direcciones de manera completamente aleatoria. Por ello, el balón no sigue una trayectoria definida. Ahora, considerando una fuerza ejercida durante un cierto tiempo; es posible imaginar 20 personas empujando para la derecha y 21 para la izquierda y que cada persona está ejerciendo cantidades de fuerza equivalentes. En este caso las fuerzas ejercidas por el lado izquierdo y por el lado derecho no están equilibradas, favoreciendo al lado izquierdo, por lo que el balón se moverá ligeramente hacia la izquierda. Esta desproporción siempre existe, y es lo que causa el movimiento aleatorio. Si se observa la situación desde arriba, de modo que no se puedan ver a las personas, se vería el gran balón moviéndose en un plano como un objeto animado por movimientos erráticos.

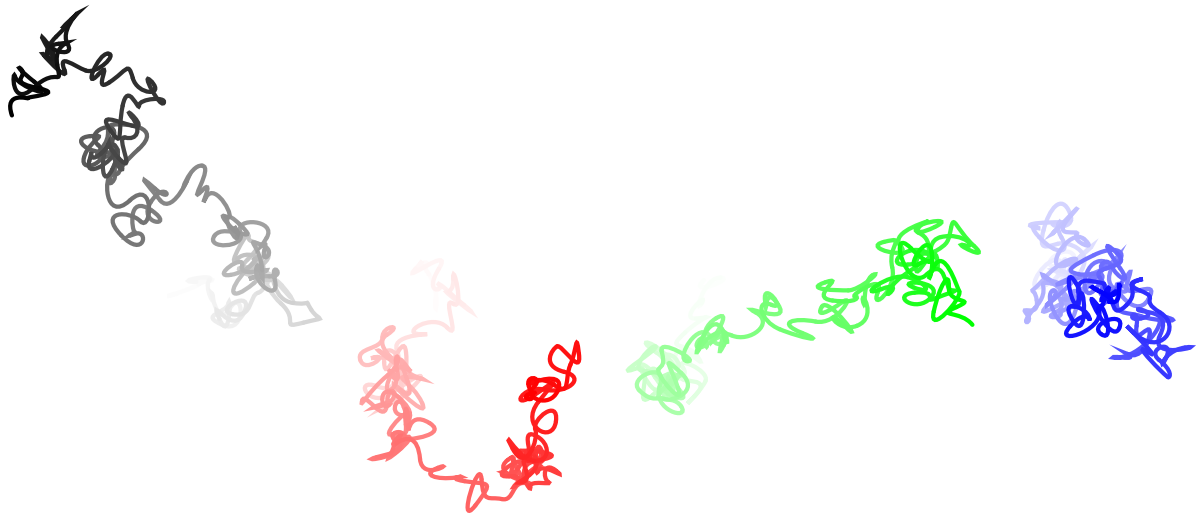

Figura 1.1: Ejemplos de movimientos Brownianos bidimensionales.

Aproximación heurística

Aunque el movimiento Browniano tiene lugar en tres dimensiones, el núcleo del problema se encuentra en una dimensión. Así, suponer una partícula que se mueve a lo largo de una línea recta, partiendo de un origen, y que en cada intervalo de tiempo h, la partícula puede realizar un salto de amplitud Δ, a derecha o a izquierda con probabilidades p y q respectivamente $(p+q=1)$.
Los saltos sucesivos Z_n, son variables aleatorias independientes,

$$Z_n = \begin{cases} +\Delta & \textit{con probabilidad } p \\ -\Delta & \textit{con probabilidad } q \end{cases}$$

Calculando su esperanza y varianza,

$$E[Z_n] = (p-q)\Delta$$
$$V[Z_n] = E[Z_n^2] - E[Z_n]^2 = \Delta^2 - (p-q)^2\Delta^2 = 4pq\Delta^2$$

La partícula saltando, al cabo de un tiempo t, habrá realizado $n = t/h$ saltos, encontrándose en la posición

$$X_t = \sum_{k=1}^{n} Z_k$$

cuya media y varianza son,

$$E[X_t] = (p-q)\frac{\Delta}{h}t$$
$$V[X_t] = \sum_{k=1}^{n} V[Z_t] = 4pq\frac{\Delta^2}{h}t$$

Lo anterior, describe para la partícula un recorrido aleatorio discreto, ocupando en instantes múltiplos de h, posiciones múltiplos de Δ. Esto genera un movimiento discreto zigzagueante en una rejilla bidimensional de tiempo y desplazamiento del origen de la recta. Ahora, para transformar el movimiento en continuo, puede hacerse tender Δ y h a cero, con intención de que:

$$E[X_t] \to \mu t$$
$$V[X_T] \to \sigma^2 t$$

siendo ambos parámetros proporcionales a t. Para conseguir lo anterior, se puede tomar $q = 1-p$, $\Delta = \sigma\sqrt{h}$, de esta forma,

$$(p-q)\frac{\Delta}{h} = (2p-1)\frac{\sigma}{\sqrt{h}} = \mu$$

de esta forma

$$p = \frac{1}{2}\left(1 + \frac{\mu\sqrt{h}}{\sigma}\right)$$
$$q = \frac{1}{2}\left(1 - \frac{\mu\sqrt{h}}{\sigma}\right)$$

En estas condiciones, volviendo a los parámetros de Z_n, se tiene que,

$$E[Z_n] = (2p-1)\Delta = \frac{\mu\sqrt{h}}{\sigma}\sigma\sqrt{h} = \mu h$$
$$V(Z_n) = \left(1 - \frac{\mu^2 h}{\sigma^2}\right)\sigma^2 h = \sigma^2 h - \mu^2 h^2$$

Tipificando la variables X_t,

$$\frac{X_t - \mu t}{\sigma\sqrt{t}} = \frac{\sum_{k=1}^{t/h} Z_k - \frac{t}{h}\mu h}{\sqrt{t/h}\sqrt{\sigma^2 h - \mu^2 h^2}} \frac{\sqrt{t/h}\sqrt{\sigma^2 h - \mu^2 h^2}}{\sigma\sqrt{t}}$$

El primer término suma tipificada de variables aleatorias independientes e igualmente distribuidas, así que por el teorema central del límite [5] tiende a una distribución $N(0,1)$. El segundo factor $\frac{\sqrt{t/h}\sqrt{\sigma^2 h-\mu^2 h^2}}{\sigma\sqrt{t}} = \sqrt{1-\frac{\mu^2 h}{\sigma^2}} \to 1$, cuando $h \to 0$. Por tanto, en resumen, X_t tiene una distribución $N(\mu t, \sigma\sqrt{t})$.
El resultado obtenido es la distribución de la posición de la partícula en función del tiempo, pero además, permite describir cómo se desplaza a lo largo de su trayectoria. Si en un instante cualquiera s, se observa que la partícula tiene una posición X_s, la evolución posterior al instante s se producirá exactamente igual que antes de s. De esta forma, el modelo de movimiento Browniano en una dimensión se puede plantear como un proceso estocástico $(X_t)_{t\geq 0}$ con las siguientes propiedades:

1. $X_0 = 0$.
2. $\forall s < t$, la variable aleatoria $X_t - X_s$ se distribuye como $N(\mu\sqrt{t-s}, \sigma\sqrt{t-s})$.
3. Dados los intervalos disjuntos, $(t_1, s_1), \ldots, (t_n, s_n)$, las variables aleatorias, $X_{s_1} - X_{t_1}, X_{s_2} - X_{t_2}, \ldots, X_{s_n} - X_{t_n}$, son independientes (incrementos estacionarios e independientes).

Definición formal

Se ha visto que en general un *Movimiento Browniano*, se puede definir como un proceso estocástico continuo con incrementos estacionarios e independientes, siendo el núcleo fundamental de la mayor parte de los modelos financieros, en particular, el modelo de Black-Scholes. Más formalmente, se pueden definir introduciendo los siguientes conceptos:

Definición 1. *Una filtración sobre un espacio de probabilidad, $(\Omega, \mathscr{A}, \mathbb{P})$ es una familia creciente $(\mathscr{F}_t)_{t\geq 0}$ de σ-álgebras incluidas en $\mathscr{A}$.*

La σ-álgebra $\mathscr{F}_t$ representa la información disponible en el instante t.

Definición 2. *Un proceso estocástico $(X_t)_{t\geq 0}$ se dirá **adaptado** a $(\mathscr{F})_{t\geq 0}$, si para cualquier t, X_t es $\mathscr{F}_t$-medible.*

Ahora, utilizando las definiciones 1 y 2,

Definición 3. *Un Movimiento Browniano es un proceso estocástico real evaluado, $(X_t)_{t\geq 0}$ continuo con incrementos independientes y estacionarios:*

[5] **Teorema Central del Límite:** Sean $X_1, ldots, X_n$ un conjunto de variables aleatorias, independientes e idénticamente distribuidas con media μ y varianza σ^2 distinta de cero. Sea $S_n = X_1 + \cdots + X_n$, entonces

$$\lim_{n\to\infty} P\left(\frac{S_n - n\mu}{\sigma\sqrt{n}}\right) = \Phi(z).$$

- *Continuidad: Si $\omega \in \Omega$, las trayectorias $X_t(\omega)$ son funciones continuas para todo $t \geq 0$, c.s.* [6] *respecto de la medida de probabilidad $\mathbb{P}$.*

- *Incrementos independientes: Si $s \leq t$, $X_t - X_s$, es independiente de*

$$\mathscr{F}_s = \sigma(X_u, u \leq s)$$

- *Incrementos estacionarios: Si $s \leq t$, $X_t - X_s$ y $X_{t-s} - X_0$ se distribuyen con la misma probabilidad.*

Recíprocamente, desde (1),(2) y (3), se puede probar que la definición anterior induce la distribución del proceso estocástico X_t.

Teorema 1. *Si $(X_t)_{t\geq 0}$ es un movimiento Browniano, y si $0 \leq t_1 < \ldots < t_n$, entonces $(X_{t_1}, \ldots, X_{t_n})$ es un vector Gausiano.*

También existe una definición de un Movimiento Browniano en términos de una filtración $\mathscr{F}_t$.

Definición 4. *Un proceso estocástico continuo real evaluado será un movimiento $\mathscr{F}_t$-Browniano si satisface:*

- *$\forall, t \geq 0$, X_t es $\mathscr{F}_t$-medible.*

- *Si $s \leq t$, $X_t - X_s$ es independiente de la σ-álgebra $\mathscr{F}_t$.*

- *Si $s \leq t$, $X_t - X_s$, y $X_{t-s} - X_0$ tienen la misma ley de distribución.*

Se puede probar que con definiciones 4 y 3, que un movimiento Browniano respecto de una filtración y respecto de su filtración natural, $\sigma(X_u, u \leq t) \subset \mathscr{F}_t$ son equivalentes.

Definición 5. *Un movimiento Browniano $(X_t)_{t\geq 0}$ se dirá **estándar** si $X_0 = 0$, $E(X_t) = 0$, y $E(X_t^2) = t$. Se denotará como $(B_t)_{t\geq 0}$.*

Ahora, la definición de martingala en tiempo continuo se puede definir como,

Definición 6. *Sea un espacio de probabilidad $(\Omega, \mathscr{A}, \mathbb{P})$ y una filtración $(\mathscr{F}_t)_{t\geq 0}$ sobre este espacio. Una familia $(M_t)_{t\geq 0}$ adaptada de variables aleatorias integrables $E(|M_t|) < +\infty$, para cada t será:*

- *martingala si, $\forall, s \leq t, E(M_t|\mathscr{F}_s) = M_s$*

- *supermartingala si, $\forall, s \leq t, E(M_t|\mathscr{F}_s) \leq M_s$*

- *submartingala si, $\forall, s \leq t, E(M_t|\mathscr{F}_s) \geq M_s$*

[6] c.s.: casi seguramente.

1.6.6. Modelo matemático de los precios económicos

Supuestas las hipótesis del marco teórico de la sección 1.6.4, es necesario fijar un modelo matemático para los precios de los activos. El modelo sugerido por Black and Scholes (1973), trata el problema de la valoración, y la cobertura de una opción Europea sin dividendos pagados por un activo. Su modelo para describir el comportamiento de los precios, es un modelo continuo en el tiempo, formado por un activo sin riesgo, y otro activo con riesgo, representándose con el vector de precios, (S_t^0, S_t), en el momento t.

El modelo tiene dos partes, una determinista y otra estocástica. La parte determinista será la solución de la ecuación diferencial ordinaria ,

$$dS_t^0 = rS_t^0 dt \tag{1.1}$$

,donde r es una constante no negativa representando el tipo de interés instantáneo. Fijando $S_0^0 = 1$, se tiene que la solución de 1.1 será $S_t^0 = e^{rt}$ para $t \geq 0$.
Por otro lado, el comportamiento del activo con riesgo, es decir, la parte aleatoria, supone que su comportamiento se basa en la ecuación diferencial estocástica siguiente,

$$dS_t = S_t(\mu dt + \sigma dB_t) \tag{1.2}$$

siendo μ y σ dos constantes, y B_t un movimiento Browniano estándar. La constante σ es llamada *volatilidad* del activo. El modelo se plantea para un intervalo $[0, T]$, donde T es el momento de maduración de una opción financiera. El intento de resolver la ecuación 1.2 mediante los métodos clásicos resultará en fracaso, así que se verá a continuación la forma de resolver la ecuación mediante otras técnicas.

Definición 7. *Sea un espacio de probabilidad filtrado* $(\Omega, \mathscr{A}, (\mathscr{F}_t)_{t\geq 0}, \mathbb{P})$ *y* $(W_t)_{t\geq 0}$ *un movimiento* $\mathscr{F}_t$*-Browniano. Un proceso estocástico* $(X_t)_{0\leq t\leq T}$ *real evaluado se dirá un* ***proceso de Itô*** *si puede escribirse como,*

$$\mathbb{P}, c.s., \forall t \leq T, X_t = X_0 + \int_0^t K_s ds + \int_0^t H_s dW_s,$$

siendo,

- X_0, $\mathscr{F}_t$*-medible,*
- $(K_t)_{0\leq t\leq T}$ *y* $(H_t)_{0\leq t\leq T}$ *procesos* $\mathscr{F}_t$*-adaptados,*
- $\int_0^T |K_s| ds < +\infty$, $\mathbb{P}$ *c.s.*
- $\int_0^T |H_s|^2 ds < +\infty$, $\mathbb{P}$ *c.s.*

Teorema 2. *Sea* $(X_t)_{0\leq t\leq T}$ *un proceso de Itô,*

$$X_t = X_0 + \int_0^t K_s ds + \int_0^t H_s dW_s$$

y f una función doblemente diferenciable. Entonces,

$$f(X_t) = f(X_0) + \int_0^t f'(X_s)ds + \frac{1}{2}\int_0^t f''(X_s)d\langle X, X\rangle_s$$

siendo por definición,

$$\langle X, X\rangle_t = \int_0^t H_s^2 ds$$

y la integral $\int_0^t f'(X_s)dX_s = \int_0^t f'(X_s)K_s ds + \int_0^t f'(X_s)H_s dW_s$.

Proposición 1. ***Fórmula de integración por partes**: Sean los procesos de Itô X_t y Y_t,*

$$X_t = X_0 + \int_0^t K_s ds + \int_0^t H_s dW_s$$
$$Y_t = Y_0 + \int_0^t K_s^{'} ds + \int_0^t H_s^{'} dW_s.$$

Entonces,

$$X_t Y_t = X_0 Y_0 + \int_0^t X_s dY_s + \int_0^t Y_s dX_s + \langle X, Y\rangle_t$$

Con la convención,

$$\langle X, Y\rangle_t = \int_0^t H_s H_s^{'} ds.$$

Ahora, volviendo sobre el problema de la ecuación 1.2, pero en lugar de $(B_t)_{0\leq t\leq T}$, con un $(W_t)_{0\leq t\leq T}$ un movimiento $\mathscr{F}_t$-Browniano, con objeto de aplicar la teoría expuesta,

$$dS_t = S_t(\mu dt + \sigma dW_t) \tag{1.3}$$

pero que también se puede poner en forma integral,

$$S_t = S_0 + \int_0^t S_s(\mu ds + \sigma dW_s). \tag{1.4}$$

De esta forma, se busca una solución que sea un proceso adaptado S_t, tal que existan las integrales siguientes $\int_0^t S_s ds$ y $\int_0^t S_s dW_s$ para todo t.

$$\mathbb{P}, \text{c.s.}, S_t = S_0 + \int_0^t \mu S_s ds + \int_0^t \sigma S_s dW_s.$$

S_t es un proceso de Itô con $K_s = \mu S_s$ y $H_s = \sigma S_s$. Suponiendo que S_t es no negativo, se puede aplicar la fórmula de Itô con la función $f(x) = \log(x) \in C^2$.

$$\log(S_t) = \log(S_0) + \int_0^t \frac{dS_s}{S_s} + \frac{1}{2}\int_0^t (\frac{-1}{S_s^2})\sigma^2 S_s^2 ds. \tag{1.5}$$

Considerando $Y_t = \log(S_t)$, y usando 1.3 se obtiene,

$$Y_t = Y_0 + \int_0^t (\mu - \sigma^2/2)dt + \int_0^t \sigma dW_t.$$

Finalmente,

$$Y_t = log(S_t) = log(S_0) + (\mu - \sigma^2/2)t + \sigma W_t.$$

De esta forma,

$$S_t = S_0 e^{(\mu-\sigma^2/2)t+\sigma W_t}.$$

siendo S_0 el precio observado en el momento 0. El resultado del modelo implica que la ley para S_t es lognormal $\forall t \in [0, T]$ (su logaritmo sigue una distribución normal). Llamando $b(t) = \log(S_0) - (\mu - \sigma^2/2)t$,

$$F_{\log(S_t)}(y) = \frac{1}{\sqrt{2\pi t}\sigma} \int_{-\infty}^{y} \exp\left(- \frac{(u - b(t))^2}{2\sigma^2 t} \right) du,$$

Lo que implica que $\log(S_t)$ sigue una distribución normal $b(t)$ y $t\sigma^2$.

Por lo tanto, se tiene (S_t) es un proceso solución de una ecuación de tipo 1.3, si y sólo sí, el proceso $\log(S_t)$ es un movimiento Browniano (no necesariamente estándar).

Deacuerdo con las propiedades vistas en 3, el proceso (S_t) cumple las siguientes propiedades:

- Continuidad del camino del proceso.
- Incrementos relativos independientes: $\forall u \leq t$, $(S_t - S_u)/S_u$ es independiente de la σ-álgebra generada $\sigma(S_v; u \leq v)$;
- Incrementos relativos estacionarios: $\forall u \leq t$, la ley de $(S_t - S_u)/S_u$ es idéntica a la ley de $(S_{t-u} - S_0)/S_0$

Las tres propiedades anteriores, expresan las hipótesis del comportamiento de los precios de los activos con el modelo de Black and Scholes. Si $\mu = 0$, entonces (S_t) será una martingala respecto de $(\mathscr{F}_t)_{0 \leq t \leq T}$ y respecto de $\mathbb{P}$.

1.6.7. Estrategias Auto-Financiadas

En este párrafo, se definirá formalmente las componentes de una cartera de activos invertidos en un mercado y qué es una estrategia autofinanciada.

Definición 8. *Una estrategia es un proceso $\phi = (\Phi)_{0 \leq t \leq T} = (H_t^0, H_t)_{0 \leq t \leq T}$ con valores en $\mathbb{R}^2$, adaptada a la filtración natural $(\mathscr{F}_t)$ del movimiento Browniano. Las componentes de la estrategia, H_t^0 y H_t son cantidades de los activos libre de riesgo y con riesgo, respectívamente. El valor de la cartera en el momento t vendrá dado por:*

$$V_t(\phi) = H_t^0 S_t^0 + H_t dS_t$$

Se supondrán las condiciones siguientes:

$$\int_0^T |H_t^0| dt < +\infty, \mathbb{P} \ c.s.$$
$$\int_0^T |H_t^2| dt < +\infty, \mathbb{P} \ c.s.$$

Definición 9. *Una estrategia autofinanciada se define como un par* $\phi = (H_t^0, H_t)_{0 \leq t \leq T}$ *de procesos adaptados* $(H_t^0)_{0 \leq t \leq T}$ *y* $(H_t)_{0 \leq t \leq T}$ *cumpliendo:*

1. $\int_0^T |H_t^0| dt + \int_0^T H_t^2 dt < \infty, \mathbb{P} c.s.$
2. $H_t^0 S_t^0 + H_t S_t = H_t^0 S_t^0 + H_t S_t + \int_0^t H_u^0 dS_u^0 + \int_0^t H_u dS_u, \mathbb{P} c.s., \forall, t \in [0, T]$

La siguiente proposición muestra, que si un inversor sigue una estrategia autofinanciada, el valor descontado de su cartera, está completamente definido por el valor inicial y la estrategia $(\Phi)_{0 \leq t \leq T} = (H_t^0, H_t)_{0 \leq t \leq T}$

Proposición 2. *Sea* $(\phi)_{0 \leq t \leq T} = (H_t^0, H_t)_{0 \leq t \leq T}$ *un proceso adaptado con valor en* $\mathbb{R}^2$, *satisfaciendo la condición,* $\int_0^T |H_t^0| dt + \int_0^T H_t^2 dt < \infty, \mathbb{P} c.s.$. *Sea* $V_t(\phi) = H_t^0 S_t^0 + H_t S_t$ *y* $\tilde{V}_t(\phi) = e^{-rT} V_t(\phi)$. *Entonces,* ϕ *define una estrategia autofinanciada sí y solo sí*

$$\tilde{V}_t(\phi) = V_0(\phi) + \int_0^t H_u d\tilde{S}_u, \ c.s., \forall, t \in [0, T]$$

1.6.8. Probabilidades equivalentes

Es necesario conocer cómo se relacionan diferentes medidas de probabilidad, se verá a continuación:

Definición 10. *Sea* $(\Omega, \mathscr{A}, \mathbb{P})$ *un espacio de probabilidad. Una medida de probabilidad* $\mathbb{Q}$ *en* $(\Omega, \mathscr{A})$ *es absolutamente continua con respecto a* $\mathbb{P}$ *si*

$$\forall A \in \mathscr{A}, \mathbb{P}(A) = 0, \Rightarrow \mathbb{Q} = 0.$$

Teorema 3. *Una medida de probabilidad* $\mathbb{Q}$ *es absolutamente continua con respecto* $\mathbb{P}$ *sí y solo sí existe un variable aleatoria no negativa* Z *en* $(\Omega, \mathscr{A})$ *tal que:*

$$\forall, A \in \mathscr{A}, \mathbb{Q} = \int_A Z(\omega) d\mathbb{P}(\omega).$$

Z *se llama densidad de* $\mathbb{Q}$ *con respecto de* $\mathbb{P}$ *y se denota como* $\frac{d\mathbb{Q}}{d\mathbb{P}}$.

En tiempo continuo, el teorema de Girsanov, permite hacer el cambio de probabilidad conveniente, para convertir los precios descontados en martingalas.

1.6.9. Teorema de Girsanov

Sea $(\Omega, \mathscr{A}, (\mathscr{F}_t)_{t\geq 0}, \mathbb{P})$ un espacio de probabilidad filtrado y $(B_t)_{0\leq t\leq T}$ un $\mathscr{F}_t$-estándar movimiento Browniano. Se define el siguiente teorema que relaciona medidas de probabilidad equivalentes:

Teorema 4. *Sea $(\theta_t)_{0\leq t\leq T}$ un proceso adaptado cumpliendo $\int_0^T \theta_s^2 ds < +\infty$, c.s., tal que el proceso $(L_t)_{0\leq t\leq T}$ definido por,*

$$L_t = \exp\left(-\int_0^t \theta_s dB_s - \frac{1}{2}\int_0^t \theta_s^2 ds \right)$$

sea una martingala. Entonces, bajo la medida de probabilidad $\mathbb{P}^{(L)}$ con densidad L_T respecto de $\mathbb{P}$, el proceso $(W_t)_{0\leq t\leq T}$ definido por $W_t = B_t + \int_0^t \theta_s ds$ es un $\mathscr{F}_t$-estándar movimiento Browniano.

Desde el teorema anterior, se tiene que $\mathbb{P}^{(L)}$ y $\mathbb{P}$ son equivalentes, es decir,

$$\forall A \in \mathscr{F}, \mathbb{P}^{(L)}(A) = \int_A L_T(\omega) d\mathbb{P}(\omega)$$

Por el criterio de Novikov, [7], una condición suficiente para que $(L_t)_{0\leq t\leq T}$ sea una martingala es

$$E\left(\exp\left(\frac{1}{2}\int_0^T \theta_t^2 dt \right) \right) < \infty,$$

1.6.10. Probabilidad bajo la cual $\tilde{S}_t$ es una martingala

Volviendo al modelo de la ecuación 1.2, el activo con riesgo S_t, se va a probar que existe una probabilidad equivalente a $\mathbb{P}$, bajo la cual, el precio descontado de un activo $\tilde{S}_t = e^{-rt}S_t$ es una martingala. Aplicando la fórmula de integración por partes a los procesos de Itô, e^{-rt} y S_t respecto de (B_t),

$$\begin{aligned} d\tilde{S}_t &= d(e^{-rt}S_t) \\ &= -re^{-rt}S_t dt + e^{-rt}dS_t \\ &= -re^{-rt}S_t dt + e^{-rt}d(S_t\mu dt + S_t\sigma dB_t) \\ &= \tilde{S}_t((\mu - r)dt + \sigma dB_t). \end{aligned}$$

Tomando $W_t = B_t + \frac{(\mu-r)}{\sigma}t$, la anterior se simplifica a la siguiente ecuación,

$$d\tilde{S}_t = \sigma\tilde{S}_t dW_t \tag{1.6}$$

[7] Ver Karatzas and Shreve (1988)

Aplicando el teorema de Girsanov 4, con

$$\theta_t = (\mu - r)/\sigma, (:= \rho)$$
$$L_t = \exp(-\rho B_t - \frac{1}{2} t\rho^2),$$

,entonces existe una probabilidad $\mathbb{P}^*$ equivalente a $\mathbb{P}$, bajo la cual, $W_t = B_t + \rho t$ es un $\mathscr{F}_t$-estándar movimiento Browniano. Por lo tanto, bajo la probabilidad $\mathbb{P}^*$, se deduce desde 1.6 que $(\tilde{S}_t)$ es una martingala cumpliendo la ecuación,

$$\tilde{S}_t = \tilde{S}_0 \exp(\sigma W_t - \sigma^2 t/2)$$

o bien,

$$S_t = S_0 \exp(\sigma W_t + (r - \sigma^2/2)t) \tag{1.7}$$

Se asumirá el modelo anterior para los precios de éste trabajo, llamado **Movimiento Geométrico Browniano**, (GBM).

1.7. Las opciones financieras

En esta sección se explicarán las opciones financieras y sus diferentes tipos, sus variables o parámetros asociados, además de cómo se negocian según las expectativas del comprador y el vendedor, explicando su valoración según las condiciones de mercado y los diferentes tipos existentes sin pretender ser exhaustivo.

Las **opciones financieras** son contratos emitidos por organismos reguladores con posibilidad de ser ejercidos por el tenedor o negociadas en mercados secundarios. La parte compradora, tiene el derecho, pero no la obligación, de comprar o vender a un precio determinado una cantidad de unidades de un subyacente [8] asociado. Por contra, la parte vendedora o suscriptora, en caso de que la opción sea ejecutada por el comprador, tiene la obligación de vender o comprar el subyacente al precio de ejercicio. El valor de una opción se llama *Prima* o *Premium*. Cuando una opción es comerciada, la prima es recalculada para las nuevas circunstancias de mercado.

El precio del subyacente asociado a una opción financiera, se suele llamar *Spot Price*, S_t, y el nivel de precios al que se ejecuta la opción precio de ejercicio, o *Strike Price*, K, siendo fijo durante toda la vida de la opción. Existen dos tipos generales de opciones, las que dan derecho a comprar, *Call* y las que dan derecho a vender, *Put*.

Las opciones poseen un periodo de vida determinado con una fecha de vencimiento o de ejercicio, T, a partir de la cual, ya sólo quedan dos posibilidades, o bien el contrato se ejecuta, o bien expira. El momento de ejecución dependerá del tipo de opción, que

[8] Subyacente es el nombre que recibe el tipo de activo con riesgo sobre el que se aplica la opción financiera.

podrá ser ejecutada únicamente en la fecha de ejercicio o el caso del tipo Americanas, en cualquier momento antes de tal fecha T. En particular, en el caso de una *Call* sobre una acción, si en el momento de expiración, $S_T < K$, entonces el comprador no le interesará ejercer su opción, ya que puede comprar a mercado. En cambio, si $S_T > K$, entonces el comprador ejercerá la opción comprando al precio K una acción , y vendiendo al precio de mercado S_T, obteniendo así, un beneficio de $S_T - K$ por acción, por lo tanto, el valor de una opción *Call* Europea [9] en el momento de su maduración es:

$$\text{máx}\{S_T - K, 0\}$$

Si una opción es vendida, la parte vendedora, también llamada suscriptora de la opción, por ejemplo, en el caso de una *Call*, debe ser capaz de entregar al comprador la cantidad de dinero máx$\{S_T - K, 0\}$ en el momento de maduración. Existe la incertidumbre sobre el valor de S_T en el momento de la suscripción de una opción, de esta forma surgen las siguientes preguntas:

1. ¿Cuanto debería pagar el comprador de una opción?, es decir, ¿cómo se debería valorar el precio de una opción en el momento $t = 0$ sobre un activo valiendo máx$\{S_T - K, 0\}$ en el momento de expiración T?. Este es el problema de *Valoración* de opciones.

2. ¿Cómo debería el suscriptor, que recibe el prima inicial, generar una cantidad de máx$\{S_T - K, 0\}$ en el momento T?. Este es el problema de *Cobertura* de una opción.

Una aplicación común de las opciones en los mercados financieros, es la de servir como coberturas para un número de acciones durante un tiempo T, con la ventaja de no tener que venderlas. Dicho de forma más específica, si se posee una cantidad n de acciones que cotizan a un precio S_t actualmente en el mercado, y el objetivo es cubrir los beneficios de las acciones a una caída de la cotización, el comprador, deberían comprar un número m de opciones *Put* con precio de ejercicio $K = S_t$, de tal forma que le otorguen el derecho de vender a precio K, el número n de acciones a cubrir. En caso de que la cotización no caiga por debajo de K, el comprador puede transferir esas opciones, antes de su expiración, a otros participantes del mercado que tengan diferentes expectativas sobre la cotización de la acción.

Los incentivos para que se produzca una transacción de opciones, son las diferentes perspectivas futuras de los precios por parte del comprador y del vendedor. Dependiendo del tipo de opción, *Call* o *Put*,y si se compra o se vende, se pueden listar las siguientes perspectivas:

- *Alcista* comprando una *Call*,
- *Bajista* comprando una *Put*,
- *Bajista* comprando una *Call*,

[9] Este tipo de opción, será definida en detalle en la subsección 1.7.4

- *Alcista* comprando una *Put*.

Adicionalmente, es posible realizar estrategias complejas de mercado combinando compras y ventas de diferentes opciones de los tipos *Call* o *Put*, con diferentes precios de ejercicio. Por ejemplo, es posible nombrar dos estrategias: (1) apostar a que un activo se va a mantener en una franja de precios, o por contra, (2) apostar a que el precio de un activo va a tener un gran movimiento al alza o a la baja.

1.7.1. Ausencia de arbitraje por paridad Put/Call

Generalmente se acepta la ausencia de oportunidades de arbitraje en los mercados financieros líquidos, y por tanto, no es posible realizar beneficios sin riesgo a los participantes del mercado. Existen modelos matemáticos sobre este concepto en general. Más concretamente, es posible deducir una fórmula que relaciona los precios de una Call y una Put Europeas, asumiendo la no existencia de arbitraje. Considerar una Call y una Put, sobre el mismo subyacente con precio S_t, con el mismo precio de ejercicio $K_c = K_p = K$, y con la misma fecha de ejercicio, $T_c = T_p = T$. Asumir que es posible prestar dinero a un tipo de interés r. Si C_t y P_t son los precios de la Call y la Put consideradas en el momento t, a razón de que existe ausencia de arbitraje, se cumple la siguiente ecuación con $t < T$, llamada **paridad de put/call**:

$$C_t - P_t = S_t - Ke^{-r(T-t)}$$

Ahora, suponer que no se cumpliera la ecuación anterior, por ejemplo, $C_t - P_t > S_t - Ke^{-r(T-t)}$, entonces es posible hacer un beneficio sin riesgo de la siguiente manera:
En el momento t, se compran 100 acciones (paquete habitual que se permite comprar o vender con una opción), se compra una Put, y se vende una Call. Entonces el valor líquido de la operación es,

$$C_t - P_t - S_t$$

Ahora, en la expresión anterior, si la cantidad es positiva, se invierte a un tipo de interés r hasta el momento T, pero si es negativa, se presta al mismo tipo de interés. En el momento T existen dos posibilidades:

- Si $S_T > K$, la Put no se ejerce, en cambio la Call sí se ejerce, entregándose las acciones al poseedor, y recibiendo la cantidad K a cambio por la venta y retirando los fondos de la cuenta bancaria. Finalmente obteniendo el beneficio de $K + e^{r(T-t)}(C_t - P_t - S_t) > 0$.

- Si $S_T < K$, la Call no se ejerce, en cambio la Put sí se ejerce, obteniendo K por la venta y retirando los fondo de la cuenta bancaria. Finalmente obteniendo el beneficio de $K + e^{r(T-t)}(C_t - P_t - S_t) > 0$.

En ambos casos, se obtiene un beneficio positivo, sin correr riesgos, lo cual es un ejemplo de estrategia de arbitraje [10].

[10] Se pueden ver ejemplos similares en el libro de Cox y Rubinstein (1985)

1.7.2. Replicando el PayOff de una opción Europea

Dado un espacio de probabilidad filtrado, $(\Omega, \mathscr{A}, (\mathscr{F}_t)_{t\geq 0}, \mathbb{P})$ Una opción Europea se define por una variable aleatoria h (**PayOff**) no negativa, $\mathscr{F}_T$-medible. Se escribe a menudo $h = f(S_T)$, siendo $f(x) = \text{máx}\{x-K, 0\}$, en el caso de una Call y $f(x) = \text{máx}\{K-x, 0\}$ en el de una Put. A continuación, se definen las estrategias admisibles, y cómo hacer la valoración de una opción Europea para estas estrategias.

Definición 11. *Una estrategia $\phi = (H_t^0, H_t)_{0\leq t\leq T}$ es admisible, si es auto-financiable y su valor descontado $\tilde{V}_t = H_t^0 + H_t\tilde{S}_t$ de la cartera correspondiente es, $\forall, t \geq 0$, tal que, $\sup_{t\in[0,T]} \tilde{V}_t$ sea de cuadrado integrable bajo $\mathbb{P}^*$.*

Una opción se dice reproducible si su payoff en el momento de maduración, es igual al valor final de una estrategia admisible. Es claro, que para que una opción definida por h sea reproducible, es necesario que h ($h = \text{máx}\{S_T - K, 0\}$ caso de una Call) sea de cuadrado integrable bajo la medida de probabilidad $\mathbb{P}^*$, propiedad que se cumple, desde que $E^*(h^2) < E^*(S_T^2) < \infty$ [11]. Para el caso de una Put, h está incluso acotada por 0.

Teorema 5. *En el modelo de Black-Scholes, cualquier opción definida por una variable aleatoria h no negativa $\mathscr{F}_T$-medible, la cual, es de cuadrado integrable bajo $\mathbb{P}^*$, es reproducible y su valor en un momento t de cualquier cartera reproducible es dado por,*

$$V_t = E^*\left(e^{-r(T-t)}h|\mathscr{F}_t\right) \tag{1.8}$$

Así, el valor de una opción en el momento t, puede ser definido naturalmente por la expresión $E^*\left(e^{-r(T-t)}h|\mathscr{F}_t\right)$.

Como corolario, cuando la variable aleatoria h pueda ser escrita como $h = f(S_T)$, se podrá expresar el valor de la opción V_t en el momento t como una función de t y S_t.

$$V_t = E^*\left(e^{-r(T-t)}f(S_T)|\mathscr{F}_t\right) \tag{1.9}$$

$$= E^*\left(e^{-r(T-t)}f(S_t e^{r(T-t)}e^{\sigma(W_T-W_t)-(\sigma^2/2)(T-t)})|\mathscr{F}_t\right) \tag{1.10}$$

$$\tag{1.11}$$

1.7.3. Valoración de Opciones

El problema de la valoración de una opción financiera, es el problema de asignar un precio de negociación, o también llamado **Fair Price**, a una opción en el momento de su adquisición o transferencia durante su periodo de vida. En general, es un problema difícil, teniendo una formulación exacta en muy pocos casos del amplio espectro de tipos

[11] Ver teorema de Itô [7, 3.5.1]

opciones existentes. Como complejidad adicional, y con la finalidad de adaptar el modelo de valoración a la realidad, el problema también puede tratarse, considerando algunos de los parámetros anteriores, como variables adicionales, como la volatilidad o el tipo de interés, durante el periodo vida de la opción. En todos los problemas tratados para este trabajo, esos parámetros se considerarán constantes.

El valor de una opción en un momento t dependerá de los siguientes parámetros:

- El precio actual del subyacente o *Spot Price*, S_t.
- El precio de ejercicio o *Strike Price*, K.
- El tipo de interés, r.
- La volatilidad del subyacente, σ.
- El dividendo obtenido por el subyacente, δ.
- La fecha de maduración T.

La obtención del valor objetivo de una opción para replicar una cartera de activos, pasa por calcular la esperanza matemática de su variable aleatoria asociada, que es función de uno o varios precios subyacentes, teniendo estos últimos, en general, una distribución logarítmica normal multivariante (visto caso unidimensional en 1.6.6, extensible a varias variables), y según la hipótesis de partida de la sección 1.6.

Existe una amplia teoría analítica para el tratamiento de las opciones financieras, para valorar su precio y sus parámetros derivados llamados griegas [12] ,pero en este trabajo, se focalizará su estudio, en los métodos numéricos basados en MC y QMC. Para el caso de las opciones con momento de ejecución únicamente en la fecha de ejercicio, en general, para calcular su precio descontado $\tilde{V}$ en un momento de tiempo $0 \leq t \leq T$, se planteará el siguiente problema de probabilidad:

Sea un instante de tiempo $t \in [0, T]$, y un vector de subyacentes $\overline{S}_t = (S_t^1, S_t^2, \ldots, S_t^s)$, sea una función $h = p(\overline{S}_t, K)$ [13], representando la cantidad pagada al comprador de la opción en la fecha de maduración. Entonces, el valor descontado de una opción, se calculará con la esperanza matemática de $h = p(\overline{S}_t, K)$ bajo $\mathbb{P}^*$:

$$\tilde{V} = E_{\mathbb{P}^*}[e^{-rT} p(\overline{S})] \tag{1.12}$$

$$= e^{-rT} E_{\mathbb{P}^*}[p(\overline{S})] \tag{1.13}$$

$$= e^{-rT} \int_{\mathbb{R}^s} p(\overline{S}) f(\overline{S}) d\overline{S}, \tag{1.14}$$

$$\tag{1.15}$$

donde,

[12] Derivadas parciales del valor de una opción respecto de uno de sus parámetros, $\frac{\partial V}{\partial \alpha}$.

[13] Caso multidimensional del payoff visto en subsección 1.7.2

- $E[\cdot]$ operador esperanza matemática de una variable aleatoria.
- $p(\cdot)$ función de la variable aleatoria $\overline{S}_t$.
- f es la función de densidad s-variada de $\overline{S}_t$, siendo log-normal en la mayoría de los casos tratados.
- $\mathbb{P}^*$ la medida de probabilidad de riesgo neutro[14].

La justificación de la ecuación 1.12 proviene del modelo de Black-Scholes, visto en el corolario 1.9, desde que una opción es reproducible bajo $\mathbb{P}^*$, extendida al caso multidimensional con dividendos.

El problema de la ecuación 1.12, puede ser tratado por la vía analítica, o bien, como se pretende en este trabajo, formular el problema para ser sometido a un MC o QMC. Para esto último, se requiere una discretización del dominio de integración, evaluando el integrando en una red de puntos uniformemente distribuida, convirtiendo un problema analítico en uno numérico. Entre los métodos MC y QMC, la diferencia fundamental estriba en la forma de obtener la red de puntos de integración, siendo generada aleatoriamente en MC, y de forma determinista mediante reglas de generación en QMC. La calidad de la aproximación obtenida por los métodos numéricos, dependerá de la calidad de los puntos generados del dominio de integración. El orden de convergencia y la precisión obtenida, variará enormemente según casos tratados, que dependerá de la variabilidad o suavidad del integrando, la discrepancia de la red de puntos utilizada, la cantidad de puntos de la red y la dimensión del problema. Se verán más detalles sobre MC y QMC, en las secciones 2.1 y 3, exponiendo sus definiciones formales.

Para el caso de las opciones Americanas, que pueden ser ejecutadas en cualquier momento $0 \leq n \leq T$, la valoración será diferente a la anteriormente planteada.

1.7.4. Opciones Vanilla

Las opciones *Vanilla* son las opciones más básicas y elementales, habiendo de dos tipos *Europeas* y *Americanas*. Las primeras permiten ser ejecutadas solamente en la fecha del ejercicio, en cambio las segundas, pueden ser ejecutadas en cualquier momento $0 \leq t \leq T$. En el segundo caso, la valoración es totalmente distinta. Las opciones Europeas son evaluadas según el modelo de Black-Scholes, en cambio, la valoración para las Americanas, se basará en la envoltura de Snell para el caso discreto, o el problema de parada óptima para el continuo.

Opciones Europeas

Las opciones Europeas están caracterizadas por su función de pago o Pay-Off, p no negativa, siendo de los tipos *Call* o *Put* sobre un subyacente asociado. La función de pago

[14]Probabilidad bajo la cual los precios descontados de los subyacentes son martingalas.

depende del precio en el momento de maduración, S_T y del precio de strike K:

$$p_{call}(S_T) = \text{máx}\{S_T - K, 0\},$$
$$p_{put}(S_T) = \text{máx}\{K - S_T, 0\},$$

El vendedor, obtendrá el beneficio siguiente según tipo:

$$(Call) : -\text{máx}\{S_T - K, 0\} + C(K, T),$$
$$(Put) : -\text{máx}\{K - S_T, 0\} + P(K, T),$$

siendo $C(K, T)$ y $P(K, T)$ el valor descontado de una *Call* y una *Put* respectívamente con precio de ejercicio K y fecha de ejercicio T. Estos últimos valores, $C(K, T)$ y $P(K, T)$ son las incógnitas a calcular, siendo el objetivo de las aplicaciones de MC y QMC en este trabajo.

Ahora, representando gráficamente el precio final del subyacente en las ordenadas S_T, y en las ordenadas la función de pago $p_{Call}(S_T)$, pueden pintarse las funciones de pago anteriores. El caso de un comprador que adquiere una opción *Call* Europea se presenta en la figura 1.2. Es un diagrama *PayOff-Beneficio*, mostrando en rojo el pago realizado al comprador por ejercer la opción en un momento S_T de maduración, es decir, $\text{máx}\{S_T - K, 0\}$, y mostrando en azul, el beneficio obtenido por el comprador, $\text{máx}\{S_T - K, 0\} - C(K, T)$, que resulta de restar al Pay-Off, el valor descontado de la cantidad pagada como prima de una opción *Call* entregada al vendedor, y también llamada *Prima*. Se presenta en líneas punteadas el precio de ejercicio o strike, K. El vendedor de la *Call*, tendría como Pay-Off, $\text{máx}\{K - S_T, 0\}$, y como beneficio, $-\text{máx}\{S_T - K, 0\} + C(K, T)$, presentado en la figura 1.3. Es importante observar que el comprador de una opción posee un riesgo limitado, pero en cambio posee un beneficio ilimitado, por contra el vendedor, tiene un riesgo ilimitado, pero un beneficio limitado.

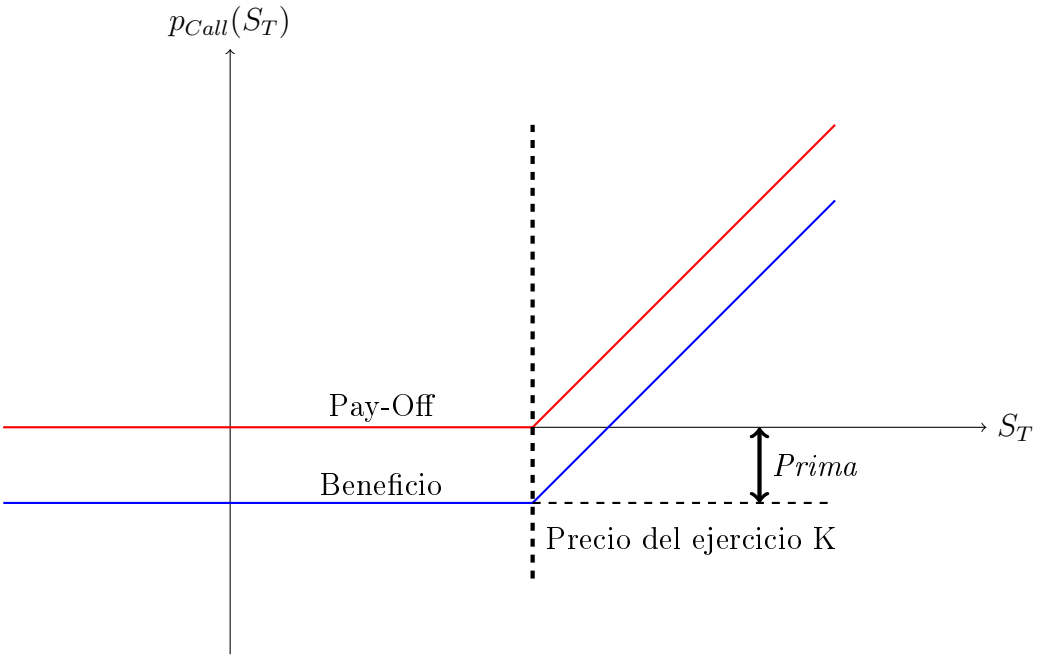

Figura 1.2: Diagrama Pay-off-Beneficio del comprador de una opción Call Europea.

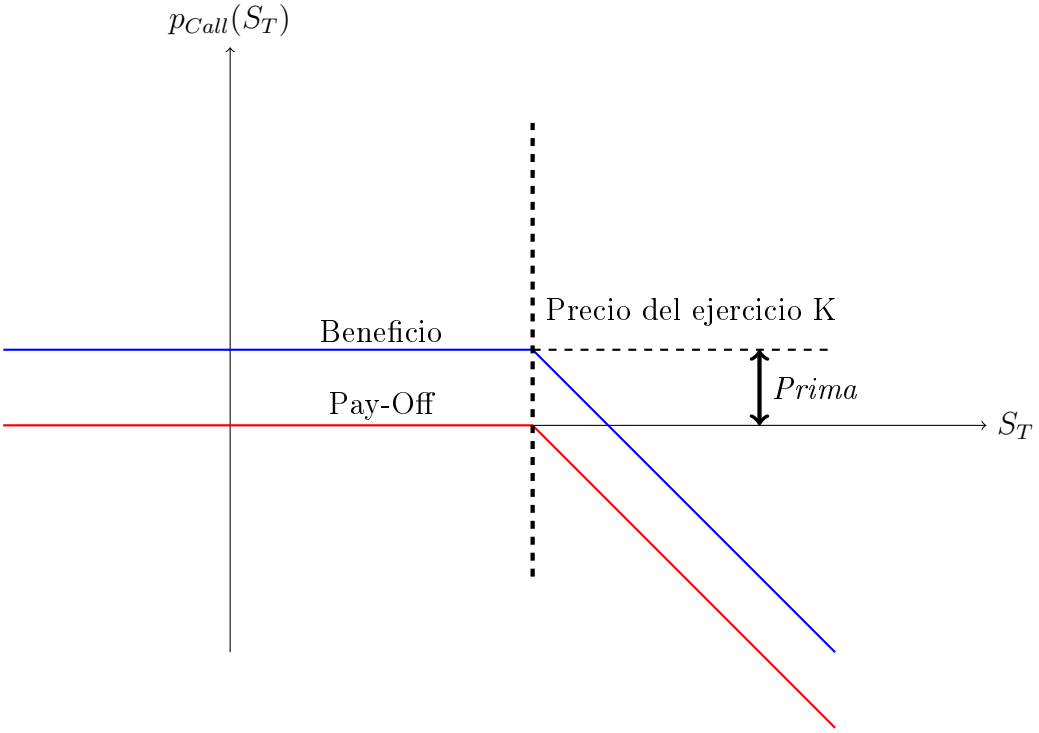

Figura 1.3: Diagrama Pay-off-Beneficio del vendedor de una opción *Call* Europea.

El mismo tipo de gráficos se presenta para una *Put*, tanto desde el punto de vista del

comprador como del vendedor, en las figuras 1.4 y 1.5.

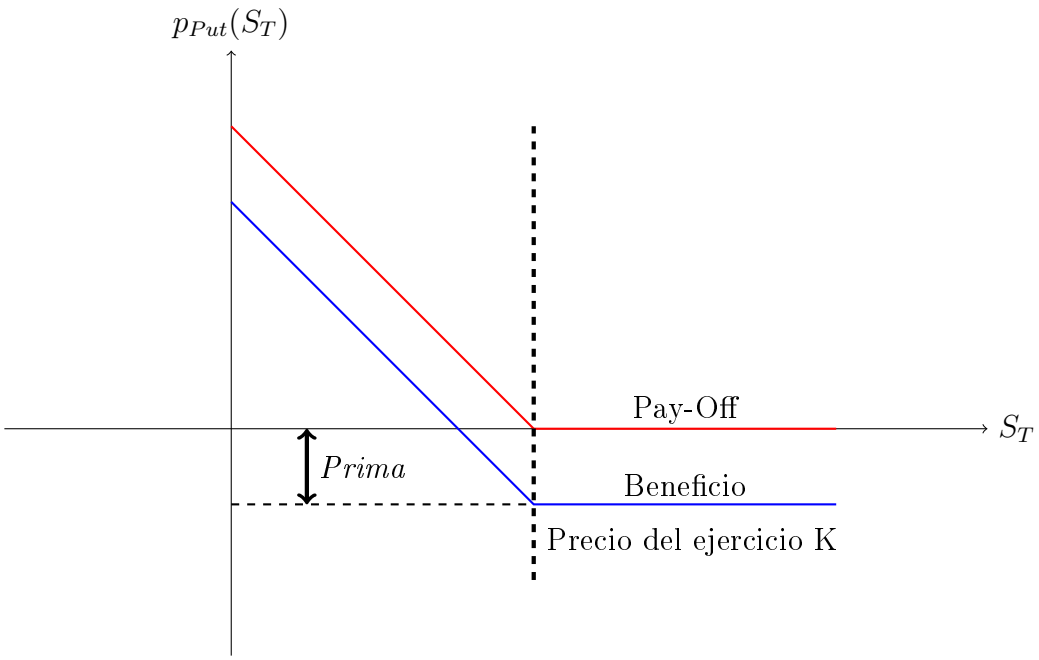

Figura 1.4: Diagrama Pay-off-Beneficio del comprador de una opción Put Europea.

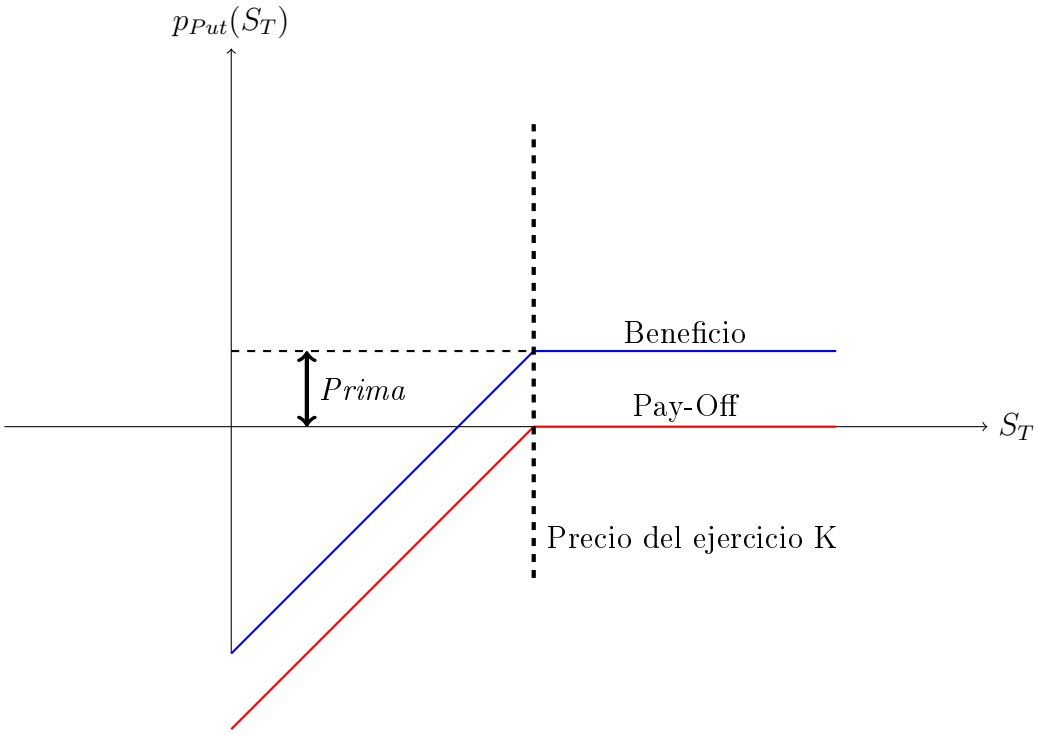

Figura 1.5: Diagrama Pay-off-Beneficio del vendedor de una opción *Put* Europea.

Las aplicación de MC y QMC para opciones Europeas se hará simulando, una cantidad N de variables aleatorias, de tipo log-normal S_T. Seguidamente, los valores generados se insertan en la función de pago $p(S_T)$, y promediando, se obtendrá una aproximación a su valor de tipo MC o QMC.

Fórmula de Black-Scholes para una opción Europea

Black and Sholes fueron los primeros en sugerir un modelo a partir del cual, sea posible derivar una fórmula explícita del precio para una opción Europea sobre una acción que no pague dividendo. Deacuerdo con su fórmula, el suscriptor de una opción Call, puede replicarse perfectamente, para que la *Prima*, sea la cantidad de dinero necesitada en el momento cero, para replicar exactamente el PayOff máx$\{S_T - K, 0\}$, siguiendo una estrategia dinámica de coberturas hasta el momento de maduración. El punto débil de la fórmula, es que depende de un parámetro no observable (estimado estadísticamente), la conocida *Volatilidad*.

De esta forma, gracias a Merton-Black-Scholes(1973), existe una fórmula exacta para calcular el precio actual de una opción *Call* o *Put* Europea. Sean C y P las valoraciones de opciones Call y Put respectivamente, siendo funciones del tiempo $t < T$ y del precio del subyacente S_t, entonces la fórmula de Black-Scholes para ambos tipos puede expresarse como:

$$C(S,t) = S_t e^{-\delta(t-T)} N(d_1) - K e^{r(t-T)} N(d_2)$$
$$P(S,t) = C(S,t) + K e^{r(t-T)} - S_t$$

donde,

$$d_1(x) = \frac{\frac{ln(S_0)}{K} + (r - \delta + \frac{1}{2}\sigma^2)T}{\sigma\sqrt{t-T}}$$
$$d_2(x) = d_1 - \sigma\sqrt{T-t}$$

y los parámetros,

- $N(\cdot)$ es la función de distribución de la variable aleatoria normal estándar.
- S_t es el precio del subyacente en el momento de tiempo t.
- K precio del ejercicio o strike price.
- r tipo de interés.
- σ volatilidad del subyacente.
- δ dividendo pagado por el subyacente.
- T momento de maduración.

Opciones Americanas

A diferencia de las Europeas, la valoración de una opción una Americana resultará mucho más compleja de describir. En principio, se puede definir una secuencia no negativa estocástica Z_n, representando el valor intrínseco obtenido por la ejecución de la opción en el momento n, pero éste no será el valor de la opción. En el caso de una *Call* Americana sobre un subyacente S_n, $Z_n = \text{máx}\{S_n - K, 0\}$, con K el precio del ejercicio, y para el caso *Put*, $Z_n = \text{máx}\{K - S_n, 0\}$. La realidad es que el valor de una opción Americana en cada instante de tiempo será superior a este valor intrínseco, problema que será resuelto con la envoltura de Snell, mediante inducción regresiva, o como problema de parada óptimo en el caso continuo. Seguidamente se presenta la definición formal de una envoltura de Snell:

Dado un espacio de probabilidad filtrado, $(\Omega, \mathscr{F}, (\mathscr{F}_n)_{n=0}^N, \mathbb{P})$, y $\mathbb{Q}$, una medida de probabilidad absolutamente continua respecto de $\mathbb{P}$, de definirá una **envoltura de Snell** como un proceso estocástico U_n, respecto una medida de probabilidad de riesgo neutro, de tal forma que se cumple el siguiente esquema recursivo:

$$\begin{aligned} U_N &= Z_N \\ U_n &= max\{Z_n, E_Q[U_{n+1}|\mathscr{F}_n]\} \end{aligned}$$

con $n = N-1, \ldots, 0$, y Q medida de probabilidad de riesgo neutro, bajo la cual los precios descontandos son martingalas.

Un ejemplo discreto de la valoración para una opción Americana se puede resolver con una envoltura de Snell. Suponer que se dispone de un subyacente S_n, con $1 \leq n \leq 3$ momentos discretos del tiempo cuyo precio inicial es $S_1 = 10$. Si se tiene un modelo de precios discreto tal que, éste puede incrementar 2 unidades o disminuir 1 unidad su valor, con una probabilidades de 1/3 y 2/3 respectivamente. Dada una opción americana sobre el subyacente S_n, con precio de ejercicio $K = 8.5$ y fecha de maduración en el periodo $n = 3$, ¿qué valor posee en los momentos $n \leq 3$?, ¿y cual es la estrategia a seguir en cada momento?.

La solución al problema se puede realizar aplicando la inducción regresiva sobre el precio. Se comienza desde el último periodo, trabajando hacia atrás. Desde el enunciado, en el periodo $n = 3$, caben 3 posiblidades, que $S_3 = 14$, $S_3 = 11$ o $S_3 = 8$, siendo el valor de la opción americana en ese momento de $a_3 = 5.5$,$b_3 = 2.5$ y $c_3 = 0$ unidades respectivamente. En el momento final, $U_3 = Z_3$.

En el momento $n = 2$ y $S_2 = 11$, ejercer la opción Americana, permite obtener $S_2 - K = 2.5$ unidades, así que el valor de la opción americana en el momento $n = 2$ con $S_2 = 12$ será: $U_2(S_2 = 11) = \text{máx}\{Z_2, E[U_3|S_2 = 11]\} = \text{máx}\{S_2 - K, 1/3 \cdot a_3 + 2/3 \cdot b_3\} = \text{máx}\{7/2, 7/2\} = 7/2$, lo que implica que se debería parar, ya que es igual seguir que ejercer la opción. En el caso de que $S_2 = 9$, se procede análogamente, $U_2(S_2 = 9) = \text{máx}\{S_2 - K, 1/3 \cdot b_3 + 2/3 \cdot c_3\} = \text{máx}\{1/2, 5/6\} = 5/6$, lo que implica que se debería

seguir, porque es mejor esperar 5/6 que ejercer consiguiendo 1/2 en $n = 2$.

En el momento inicial $n = 1$, $S_1 = 10$, $U_1 = \text{máx}\{S_1 - K, 1/3 \cdot U_2(S_2 = 12) + 2/3 \cdot U_2(S_2 = 9)\} = \text{máx}\{3/2, 1/3 \cdot 3.5 + 1/2 \cdot 5/6\} = \text{máx}\{3/2, 19/12\} = 19/12$, lo que implica que se debería seguir en este momento inicial, ya que es mejor esperar 19/12 que ejercer consiguiendo 3/2 en $n = 1$.

1.7.5. Economía con múltiples activos con riesgo

Considerando una economía con s activos con riesgo, $S_t = (S_{1t}, \ldots, S_{st})$ en un momento de tiempo t, se asume que cada activo i produce un dividendo δ_i, y la volatilidad de cada activo i es σ_i, ambos supuestos constantes. Los retornos de los activos siguen una distribución multivariante lognormal. Es un caso que corresponde al caso multivariante del modelo de un solo activo de Black-Scholes-Merton. La tasa de riesgo libre es r también considerada constante. Entonces se puede escribir los precios de los activos de la siguiente forma:

Sea $\log S_{it}$, $1 \leq i \leq s$, entonces el vector $(\log S_{1t}, \cdots, \log S_{st})$ está normalmente distribuido con media

$$\mu_t = \left(\log S_{10} + \left(r - \delta_1 - \frac{1}{2}\sigma_1^2 \right) t, \ldots, \log S_{s0} + \left(r - \delta_s - \frac{1}{2}\sigma_s^2 \right) t \right)$$

y matriz de covarianzas,

$$\Sigma_t = (\sigma_{ij}) = \rho_{ij}\sigma_i\sigma_j t$$

El valor actual del precio de una opción en este mercado de muchos activos, es igual al valor esperado bajo la medida de probabilidad Q de riesgo neutro 1.6.10. Para el ejemplo de una opción europea con función de pago $p(S_T) = p(S_{1T}, \ldots, S_{sT})$, que depende del precio terminal de los activos $(S_{1T}, \ldots, S_{sT})$, el valor actual de una opción es:

$$V = e^{-rT} E_Q[p(S_T)] \tag{1.16}$$

$$= e^{-rT} \int p(s) f(s) ds, \tag{1.17}$$

$$\tag{1.18}$$

donde $f(s)$ es una función s-variada lognormal, con función de densidad,

$$f(s) = \frac{1}{\sqrt{|\sum_T| (2\pi)^s} s_1 \cdots s_s} \exp\left(-\frac{1}{2} (\log s - \mu_T) \Sigma_T^{-1} (\log s - \mu_T) \right)$$

con $\log s = (\log s_1, \ldots, \log s_s)$ y $\mu_T = (\mu_{1t}, \ldots, \mu_{st})'$. La complejidad del problema dependerá de la forma de la función de pago $p(S_T)$.

1.7.6. Opciones Exóticas

Las opciones exóticas son mucho más complejas que las *Vanilla*, y por tanto su valoración es más difícil. En la mayoría de los casos, no se conocen fórmulas analíticas para la valoración, e incluso conocidas en algunos casos, como las LookBack, son fórmulas intratables. La posibilidad de la valoración mediante MC o QMC de este tipo de opciones, son la razón de ser del estudio actual.

Opciones LookBack discretas

Son opciones cuya función de pago busca el máximo o mínimo de la cotización de un único subyacente, según sea *Call* o *Put*, de un conjunto finito de momentos de monitorización. Su función de pago tiene la forma siguiente:

$$p = p(S_0, \ldots, S_m) = \text{máx}\{\text{máx}\{S_0, S_1, \cdots, S_m\} - K, 0\}$$

donde,

- K precio del ejercicio.
- S_0 es el precio actual.
- $S_1, \cdots, S_m$, son precios futuros de monitorización

Desafortunadamente, aún siendo conocida su expresión analítica, el tiempo de evaluación crece muy rápidamente en complejidad analítica y computacional, con el número de momentos de monitorizados (dimensión del problema) del subyacente, siendo ineficiente como mecanismo de valoración. En el capítulo 5, se verán aplicaciones de tipo QMC sobre esta clase de opciones, discutiendo su validez y adecuación.

Spread options

Son una clase de opciones formadas por una combinación lineal de subyacentes correlacionados entre sí. En el capítulo 5, se intentará conseguir su valor objetivo para el caso particular de tan sólo dos precios subyacentes correlacionados.

La función de pago para las Spread Options con dos subyacentes tiene la forma:

$$p = p(S_{T1}, S_{T2}) = \text{máx}[w_2 S_{2T} - w_1 S_{1T} - K, 0],$$

siendo,

- S_{1T} y S_{2T} las componentes de la variable aleatoria S_T.
- w_1, w_2 son los pesos de cada componente.
- K el precio del ejercicio.

La distribución de probabilidad de S_T es log-normal multivariada de coeficiente de correlación ρ.

Margrabe Formula

La **fórmula de Margrabe** es una fórmula para valorar opciones. Estas se aplican a opciones que permiten intercambiar dos activo con riesgo en el momento de maduración. Suponer que se tienen S_1, y S_2 los precios de dos activos con riesgo en un momento t, y que cada uno posee un dividendo δ_1 y δ_2. La opción C que se desee valorar, da el derecho, y no la obligación, a intercambiar el segundo activo por el primero en el momento de maduración T. Es decir, la función de pago es $C(T) = \text{máx}\{S_1(T) - S_2(T), 0\}$.

El modelo de Margrabe, asume que los precios siguen un modelo de tipo GBM [15]. Las volatilidades de estos movimientos no se supondrán necesariamente constantes, pero sí se supondrá constante la volatilidad de S_1/S_2, σ. Si la volatilidad de S_i es σ_i, entonces $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2\rho}$, donde ρ es el coeficiente de correlación entre los movimientos Brownianos de S_i's.

El precio correcto para una opción de éste tipo en el momento $t = 0$ es de la forma:

$$\exp(-\delta_1 T)S_1(0)N(d_1) - \exp(-\delta_2 T)S_2(0)N(d_2)$$

donde $N(\cdot)$ denota la función de distribución de la normal estándar,

$$d_1 = (\ln(S_1(0)/S_2(0)) + \frac{(\delta_1) - \delta_2 + \sigma^2/T)}{\sigma\sqrt{T}}$$

Para demostrar la fórmula, ésta puede reducirse a una situación en la que pueda aplicarse Black-Scholes. Basta considerar ambos activos en unidades de S_2, así el valor del primer activo será S_1/S_2 y del segundo 1.

En el actual trabajo, se utiliza esta fórmula para contrastarla con las simulaciones QMC de *Spread Options*, y que pasan a ser del tipo de Margrabe con las condiciones de un precio de ejercicio $K = 0$ y pesos asociados $w_1 = w_2 = 1$.

1.7.7. Las griegas de una opción

Las llamadas **griegas** de una opción financiera, son las variables aleatorias que miden la sensibilidad de su *Fair Price* respecto de la variabilidad de diferentes parámetros y/o variables independientes, como pueden ser, el precio subyacente, la volatilidad, el tipo de interés o el periodo de maduración, entre otros.
Los casos de las griegas deducidas de la primera y segunda derivación parcial del valor del instrumento respecto al precio subyacente, se llaman Δ y Γ:

$$\Delta = \frac{\partial V}{\partial S}, \tag{1.19}$$

$$\Gamma = \frac{\partial^2 V}{\partial S^2} \tag{1.20}$$

$$\tag{1.21}$$

[15] Ver ecuación 1.7

Existe una multitud diferente de griegas, por ejemplo, respecto de σ, r y T:

$$\nu = \frac{\partial V}{\partial \sigma}$$
$$\rho = \frac{\partial V}{\partial r}$$
$$\Theta = -\frac{\partial V}{\partial T}$$

El cálculo de esta griegas es posible conseguirlo de dos formas diferentes, una primera mediante el método de simulación, y una segunda mediante cálculo directo de estimadores. Se verá más adelante en el capítulo 4, aplicaciones prácticas de estos dos procedimientos.

En primer lugar, se trata de recalcular el valor de la opción en un precio ligeramente diferente. El método se basa en calcular la diferencia finita de la estimación del precio $\hat{V}$, evaluado en dos precios diferentes $S_0 + h$ y S_0, siendo h pequeño en relación al precio:

$$\Delta = \frac{\hat{V}(S_0 + h) - \hat{V}(S_0)}{h} \tag{1.22}$$

Una mejor estimación podría hacerse usándose la fórmula,

$$\Delta = \frac{\hat{V}(S_0 - 2h) - 8\hat{V}(S_0 - h) + 8\hat{V}(S_0 + h) - \hat{V}(S_0 + 2h)}{12h}$$

debida a Shammas (1995).
Análogamente, la segunda derivada podría estimarse usando,

$$\Gamma = \frac{-\hat{V}(S_0 - 2h) + 16\hat{V}(S_0 - h) - 30\hat{V}(S_0) + 16\hat{V}(S_0 + h) - \hat{V}(S_0 + 2h)}{12h^2}$$

El segundo procedimiento se basa en desarrollar expresiones analíticas de estimadores insesgados, a partir de las derivadas parciales del *Fair Price* respecto de sus parámetros. En el caso de utilizar MC, se tendrán dos metodologías de deducción, una primera *Pathwise* basada en la relación entre el pago de la opción y el tipo de interés, y una segunda metodología, *Likelilhood Ratio* basada en la densidad de probabilidad del precio subyacente y el parámetro de interés.

El primer procedimiento tiene más coste computacional que el segundo, pero es más sencillo conceptualmente. Desde [2], se puede listar algunos estimadores directos de tipo *Pathwise* para las griegas Δ y ρ de una opción Europea:

$$\Delta(\frac{\partial V}{\partial S_0}) : e^{-\delta T} 1_{\{S_T \geq K\}} \frac{S_T}{S_0}$$
$$\rho(\frac{\partial V}{\partial r}) : KTe^{rT} 1_{\{S_T \geq K\}},$$

En algunos casos bastará hacer simulaciones de la variable aleatoria S_T con distribución log-normal para evaluar los estimadores. En otros, será necesario simular la trayectoria en ciertos momentos futuros discretos del precio del subyacente S_t, para realizar entre otros casos, promedios, calcular máximos o mínimos, etc.

Capítulo 2

Métodos de Monte Carlo.

En este capítulo se va a introducir el método de MC, su definición, propiedades, usos y aplicaciones, y la generación de muestras aleatorias. En la primera sección se da la definición formal y las primeras propiedades de MC, como su tasa de convergencia y el intervalo de confianza de su estimación de la varianza del estimador MC. Además se ilustra con varios ejemplos básicos, aplicaciones a problemas, con una varianza grande, y otro con multidimensional. En la segunda sección se hablará de la simulación de variables aleatorias; la generación de muestras con distribuciones normal y log-normal; generación de sucesiones de variables aleatorias, en particular, el movimiento geométrico Browniano.

2.1. Introducción al método de Monte Carlo

Los métodos de Monte Carlo (MC) realizan una evaluación numérica sobre un conjunto de puntos aleatorios dentro del dominio de definición de las expresiones o funciones analíticas del problema tratado. Son llamados de muestreo o en inglés *sampling methods* [1], debido a que utilizan de muestras aleatorias independientes e idénticamente distribuidas (i.i.d.). Las muestras pueden ser extraídas directamente desde experimentos científico o datos socio económicos. Otra forma habitual, es utilizando generadores de número pseudo-aleatorios en un ordenador.

Los MC se apoyan en la ley fuerte de los grandes números [2] para calcular una esperanza. Considerar una variable aleatoria X, suponiendo que es posible generar una secuencia de muestras independientes $X_1, \cdots, X_n, \cdots$, de forma indefinida siguiendo las distribución de X. Aplicando la ley de los grandes números, se puede asegurar que si se tiene una

[1] http://es.wikipedia.org/wiki/Muestreo_en_estadística

[2] **Ley fuerte de los grandes números**: Si $X_1, X_2, \cdots$ es una sucesión de variables aleatorias independientes e idénticamente distribuidas cumpliendo que $E[|X_i|] < \infty$, con valor esperado μ, entonces

$$\mathbb{P}\left(\lim_{n\to\infty} \overline{X}_n = \mu\right) = 1,$$

es decir, el promedio de las variables aleatorias converge a μ casi seguramente.

función f, cumpliendo que $E(|f(X)|) < \infty$, entonces

$$E[f(X)] = \lim_{n\to+\infty} \frac{1}{n}\sum_{k=1}^{n} f(X_k) \tag{2.1}$$

Tasa de convergencia: Sea una muestra aleatoria $X_1, \cdots, X_n$ siguiendo una distribución X. Se define el error ϵ_n de un MC,

$$\epsilon_n = E(X) - \frac{1}{n}(X_1 + \cdots + X_n),$$

que puede ser estimada usando el teorema central del límite, [3] el cual, da una distribución asintótica de $\sqrt{n}\epsilon_n$ para tamaños de n grandes.

Intervalos de confianza: Desde el teorema central del límite se sigue que dados $c_1 < c_2$,

$$\lim_{n\to+\infty} P\left(\frac{\sigma}{\sqrt{n}}c_1 \leq \epsilon_n \leq \frac{\sigma}{\sqrt{n}}c_2\right) = \int_{c_1}^{c_2} e^{\frac{-x^2}{2}}\frac{dx}{\sqrt{2\pi}} \tag{2.2}$$

En la práctica, para una cantidad de muestras, n suficientemente grande, se puede aplicar que ϵ_n es una distribución Gausiana con media 0 y varianza $\frac{\sigma^2}{n}$.

Debido a que el soporte de una distribución Gausiana es $\mathbb{R}$, no es posible acotar el error utilizando 2.2. De todas formas,la regla 2.2, permitirá definir un intervalo de confianza aproximado. A continuación, observando que si G es Gausiana estándar,

$$P(|G| \leq 1.96) \approx 0.95$$

,y se tiene por tanto que con una probabilidad del 0.95, y para n suficientemente grande que,

$$|\epsilon_n| \leq 1.96\frac{\sigma}{\sqrt{n}}. \tag{2.3}$$

Estimador de la varianza: Desde el párrafo anterior, es un requisito indispensable la estimación de la varianza. Para ello, sea X una variable aleatoria, y sea $(X_1, \cdots, X_n)$ una muestra de la misma siguiendo la distribución de X. Si el estimador $E(X)$ de MC se denota por $\bar{X}_n$, entonces la estimación de la varianza vendrá dado por la ecuación:

$$\bar{\sigma}_n^2 = \frac{n}{n-1}\sum_{i=1}^{n}(X_i - \bar{X}_n)^2 = \frac{n}{n-1}\left(\frac{1}{n}\sum_{i=1}^{n}X_i^2 - \bar{X}_n^2\right). \tag{2.4}$$

Se puede probar que si $E(X) < +\infty$, entonces $\lim_{n\to+\infty}\bar{\sigma}_n^2 = \sigma^2$ casi seguramente, y $E(\bar{\sigma}_n^2) = \sigma^2$. Esto permite crear los intervalos de confianza anteriores, reemplazando σ

[3] **Teorema Central del Límite**: Dados $(X_i, i \geq 1)$ una secuencia de variables i.i.d tales que $E(X_1^2) < +\infty$. Si σ^2 es la varianza de X_1, entonces $\frac{\sqrt{n}}{\sigma}\epsilon_n$ converge en distribución a G, siendo G una distribución normal con media 0 y varianza 1.

con $\bar{\sigma}_n$. Por tanto, con una probabilidad aproximada del 0.95,

$$E(X) \in \left[\bar{X}_n - \frac{1.96\bar{\sigma}_n}{\sqrt{n}}, \bar{X}_n + \frac{1.96\bar{\sigma}_n}{\sqrt{n}}\right]$$

Así de esta forma, se tiene un estimar el error para los métodos MC.

2.1.1. Ejemplo 1: Cálculo de $E(e^{\beta G})$

Sea G la variable aleatoria con distribución normal de media 0 y varianza 1, y sea $Y_\beta = e^{\beta G}$. Se quiere calcular $E(Y_\beta)$ utilizando MC, y siendo $\beta \in \{2, 4, 6, 8, 10\}$ una constante. Antes, se obtendrá una estimación del intervalo de confianza descrito en la ecuación 2.3.
Para calcular el intervalo de confianza, se requiere calcular una estimación de la varianza 2.4, aplicando el método MC mediante el script *Octave*,

- **esperanza_familia_exponencial_gausianas_Nmu0sig1_MC.m** A.1.1

Se obtienen los resultados del cuadro 2.1 para los valores de $\frac{\bar{\sigma}_n}{\sqrt{n}}$ según el número de muestras y los valores del parámetro β.

Teóricamente, se puede calcular $E(Y_\beta)$ de la siguiente manera:

$$\begin{aligned} E(e^{\beta G}) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{\beta X} e^{-\frac{x^2}{2}} dx \\ &= e^{\beta^2/2} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{(x-\beta)^2}{2}} dx = e^{\beta^2/2} \end{aligned}$$

Muestras	$\beta = 2$	$\beta = 4$	$\beta = 6$	$\beta = 8$	$\beta = 10$
100	240.67	8.2429e+06	3.0337e+11	1.1500e+16	4.4367e+20
1000	168.90	2.2526e+08	4.0561e+14	7.5296e+20	1.4041e+27
10000	25.529	4.6751e+07	1.6919e+14	1.8490e+20	2.8695e+27
100000	14.572	2.5459e+08	1.1311e+16	5.4873e+23	2.7431e+31
500000	5.6060	2.2584e+07	2.3695e+14	2.9782e+21	4.0107e+28

Cuadro 2.1: $\frac{\bar{\sigma}_n}{\sqrt{n}}$ en función del número de muestras y del parámetro β.

Observando las columnas, solamente se puede aplicar con garantía MC para el caso en que $\beta = 2$, ya que en el resto, no parece que haya convergencia para el intervalo de confianza. Los errores absolutos (diferencia absoluta entre el valor teórico y el calculado por MC) se pueden ver en el cuadro 2.2.

Muestras	$\beta = 2$	$\beta = 4$	$\beta = 6$	$\beta = 8$	$\beta = 10$
100	2.5910e+00	1.6657e+03	6.5427e+07	7.8963e+13	5.1847e+21
1000	1.9537e+00	1.7139e+02	6.2783e+07	7.8959e+13	5.1847e+21
10000	6.7247e-01	1.6335e+03	6.4206e+07	7.8961e+13	5.1847e+21
100000	5.6146e-02	5.7614e+02	5.8376e+07	7.8922e+13	5.1847e+21
500000	3.6579e-02	1.1541e+03	6.2915e+07	7.8956e+13	5.1847e+21

Cuadro 2.2: Error absoluto en función del número de muestras y el parámetro β.

2.1.2. Ejemplo 2: Calcular la integral de un problema en N dimensiones.

En este ejemplo se probará MC para un problema de alta dimensión como es el cálculo de volúmenes, que pasará a ser un problema de probabilidad, es decir, proporción de puntos que caen a un lado u otro de la esfera N-dimensional. Para el caso de una esfera de dimensión N, es bien conocida su fórmula,

$$V_N = \frac{\pi^{N/2} R^N}{\Gamma(\frac{N}{2}+1)}$$

donde N es la dimensión del espacio contenedor y Γ la función Gamma. Si $N = 2$, quedará el área de la circunferencia $V_2 = \pi R^2$. Es importante observar, como el volumen de la esfera tiende a cero cuando la dimensión tiende a ∞:

$$\begin{aligned}\lim_{N\to+\infty} V_N &= \lim_{N\to+\infty} \frac{\pi^{N/2} R^N}{\Gamma(\frac{N}{2}+1)} \\ &= \lim_{N\to+\infty} \frac{\pi^{N/2} R^N}{N!} (Stirling) \\ &= \lim_{N\to+\infty} \frac{a^N}{\sqrt{2\pi N}(\frac{N}{e})^N} = 0\end{aligned}$$

Suponer que no se conociera la fórmula, una primera aproximación es utilizar los métodos de cuadratura sobre los puntos de la esfera, pero su principal problema es su orden de convergencia, es muy lento, como se analizará un poco más adelante. Los métodos MC pueden ayudar en el problema debido a que se basan en proporciones, siendo bastante más satisfactorios en su orden de convergencia en dimensiones medias y altas, independiente de la dimensión del espacio.
Para aplicar MC al problema de aproximar el volumen de una esfera de dimensión N, se considera la variable aleatoria X con distribución uniforme en $[0, R]^N$, y la función $f : [0, R]^N \to [0, R]$. Esta función se define por partes, según el valor de la muestra de X, esté dentro, o fuera de la misma, midiendo la normal 2 del vector $(x_1, \ldots, x_N)$, es decir:

$$f(x_1, \ldots, x_N) = \begin{cases} (2R)^N & , \| (x_1, \ldots, x_N) \|_2 < R \\ 0 & , \| (x_1, \ldots, x_N) \|_2 \geq R \end{cases} \tag{2.5}$$

siendo, $\| x \|_2 = \sqrt{x_1^1 + \cdots + x_N^2}$, con, $x \in \mathbb{R}^N$,. Es obvio que $E[|f(X)|] < \infty$, por lo tanto, es posible insertar $f(X)$ en la ecuación 2.1 de forma lícita, lo que permite resolver el problema del volumen mediante MC, sin más que generar muestras aleatorias distribuidas según la variable aleatoria X.
En este trabajo se han implementado simulaciones para este problema, cuyos resultados pueden verse en el cuadro 2.3. Los detalles de la implementación del algoritmos se muestra en los scripts *Octave* siguientes:

- **test_hiperesfera_volumen.m** A.1.2

que depende de las funciones *Octave*,

- **f_hiperesfera_volumen.m** A.2.1
- **f_hipercubo_dentro_radio.m** A.2.2
- **f_hipercubo_punto_aleatorio.m** A.2.3

Dimensión	Volumen Teórico	Volumen MC	Error Absoluto	Tiempo de Cálculo
4	4.9348	4.9505	0.0157	11.373
7	4.7248	4.6325	0.0923	14.649
10	2.5502	2.9973	0.4471	18.316
13	0.91063	0.9975	0.0868	21.225

Cuadro 2.3: Volumen de la esfera teórico y calculado mediante MC para diferentes dimensiones.

En los resultados del cuadro 2.3, se puede ver como el tiempo de cálculo es creciente debido a que el coste de generar muestras aleatorias, se incrementa con la dimensión. Además, se comprueba empíricamente la tendencia hacia 0 del volumen de la esfera con la dimensión según el límite de V_N cuando N tiende a infinito. Se observa en el cuadro 2.1, que incluso en altas dimensiones, y a pesar de la naturaleza del problema, visto en el límite anterior, MC obtiene aproximaciones razonablemente buenas para todas las dimensiones examinadas. En particular, para el caso de dimensión 13, se han necesitado 50000 muestras para obtener apenas 6 puntos en el interior de la esfera. Esto fuerza en la práctica, incrementar el número de muestras para obtener resultados suficientemente buenos, y por ende incrementa el tiempo de cómputo. Esto es un ejemplo de cómo no siempre es cierto que MC sea totalmente independiente de la dimensión del problema.

2.2. Métodos de Cuadratura y sus inconvenientes

La razón fundamental para el uso de MC a favor de los métodos de cuadratura, es debido al efecto llamado *curse of dimensionality*, que sufren todos estos últimos. Este

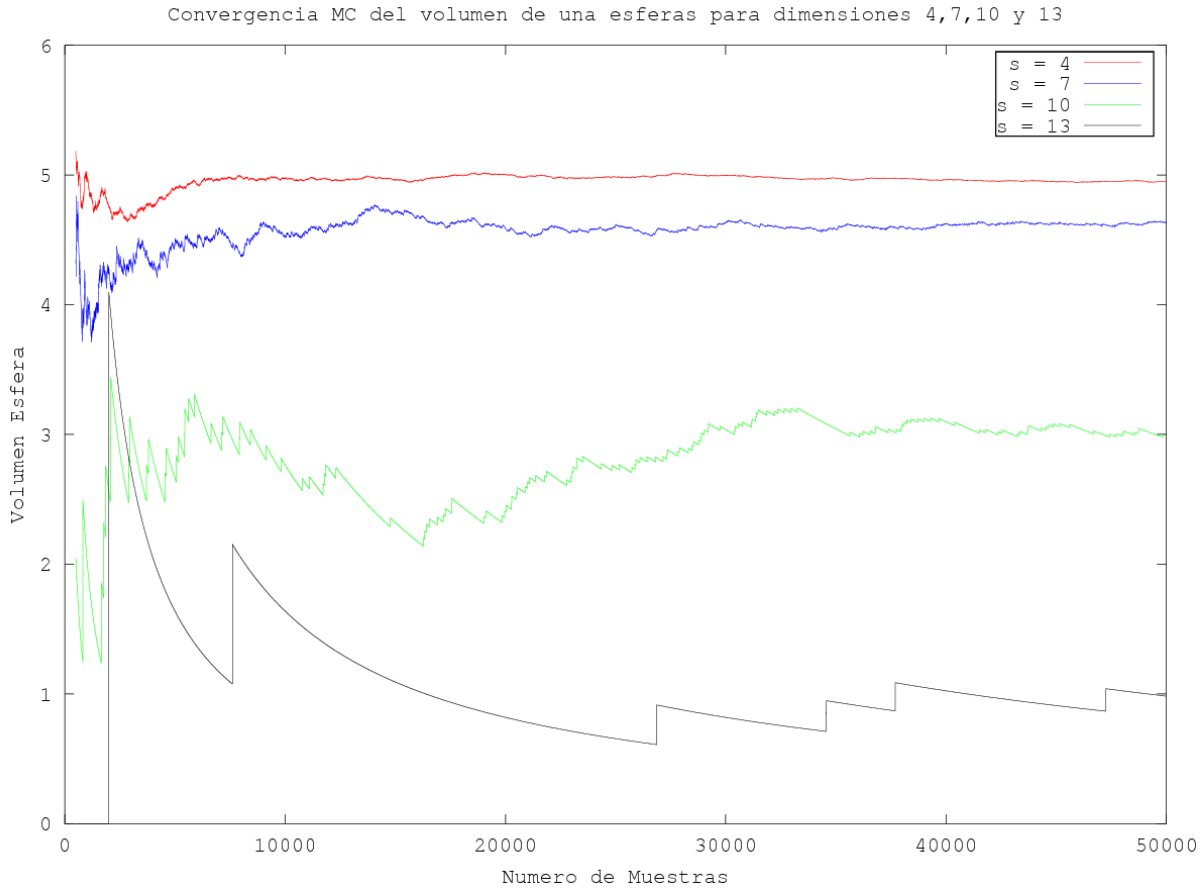

Figura 2.1: Evolución del MC en función de las muestras para el cálculo del volumen de una esfera en diferentes dimensiones.

nombre es debido al inevitable crecimiento exponencial del conjunto de puntos de integración utilizados por las cuadraturas En el caso del ejemplo del volumen de un cuerpo geométrico, concretamente el caso de la esfera, con tan apenas dimensión 10 un método de cuadratura puede resulta inviable. En efecto, se puede demostrar empíricamente que suponiendo una red centrada para el método de cuadratura candidato, basta con elegir, apenas 5 divisiones por dimensión del retículo, para obtener una red de integración de $5^{10} \approx 10^7$ puntos. Es un número elevado en general, y en particular, para el ejemplo de la esfera y por su naturaleza degenerada, pueden incluso ser insuficientes para obtener una buena aproximación.

Se puede probar, que el error cometido por una cuadratura es $O(N^{-1/s})$, siendo s la dimensión del problema. Como se puede ver en la figura 2.2) resulta extremadamente lenta para valores de s moderados. Para minimizar este problema de convergencia, MC permiten acotar el error absoluto por la cantidad $\frac{\sigma(f)}{\sqrt{N}} = O(N^{-1/2})$, siendo $\sigma(f) = \sqrt{Var(f)}$. Se concluye que para problemas de dimensiones moderadas, $s = 3, 4, ...$, siempre suponiendo

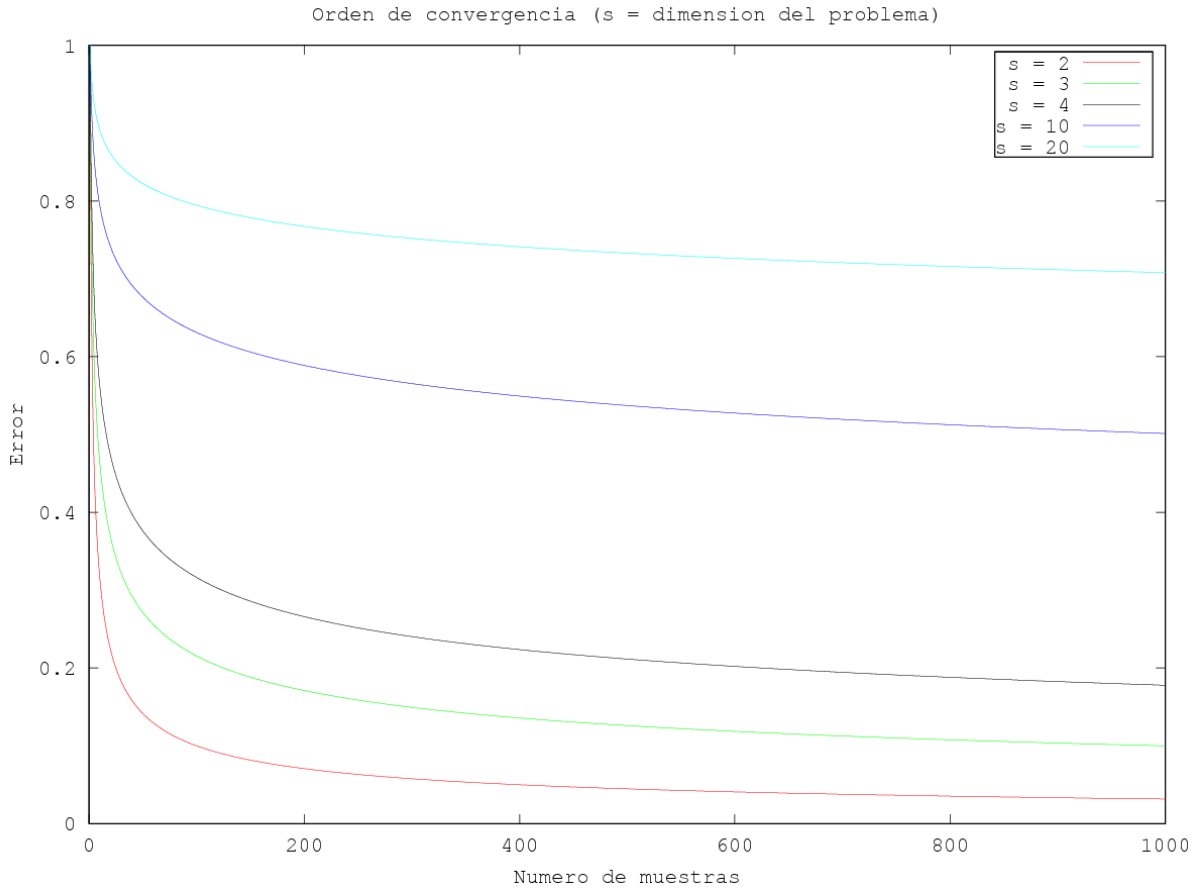

Figura 2.2: Comparativa entre órdenes convergencia $N^{-1/s}$ del error en los métodos de cuadratura.

funciones con una varianza acotada (siendo uniformemente continuas), los métodos de MC evitan el problema de la convergencia extremadamente lenta del error, cuando la dimensión del problema crece. Por tanto, una aproximación con MC, en el caso de dimensión moderada, es mejor que una regla de cuadratura. Adicionalmente, en el capítulo 3 se estudiará como mejorar el orden de convergencia de los MC, mediante redes de integración de baja discrepancia.

2.3. Simulación de variables aleatorias

2.3.1. Generación de muestras aleatorias Normal y Log-Normal

El mecanismo de simulación más conocido para muestrear una variable aleatoria, es el método de la función inversa. Este método, trata de generar muestras a partir de la evaluación de la distribución uniforme aplicadas sobre la inversa de la función de dis-

tribución de la variable aleatoria específica a muestrear. Una función de distribución es por definición, una función monótona no decreciente, por lo tanto, para conseguir que la inversa sea única, será necesario definir la inversa mediante: $F^{-1}(u) = \text{ínf}\{y|F(y) \leq u\}$.

Para el caso de una función normal, dada la función de distribución de la normal estándar y un valor aleatorio uniforme $y \in (0, 1)$,

$$y = F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{x^2}{2}} dx$$

encontrar, x tal que $F(x) = y$.

Existe un algoritmo para calcular x a partir de $F(x)$, el algoritmo de **Box-Muller** [4]. El método permite generar pares de números aleatorios independientes con distribución normal estándar, a partir dos números aleatorios uniformemente distribuidos. El algoritmo se basa en las siguientes fórmulas: suponer U_1, U_2, variables aleatorias independientes en $(0, 1]$. Sean las variables,

$$X = R\cos(\Theta) = \sqrt{-2\log U_1}\cos(2\pi U_2),$$
$$Y = R\sin(\Theta) = \sqrt{-2\log U_1}\sin(2\pi U_2),$$

entonces X y Y son variables aleatorias independientes con una distribución normal estándar. La demostración se basa en que un sistema cartesiano bidimensional, donde las coordenadas X e Y son dadas por dos variables aleatorias independientes y distribuidas normalmente, las variables aleatorias para R^2 y Θ, en las coordenadas polares correspondientes, también son independientes y poseen las expresiones siguientes,

$$R^2 = -2\log U_1 \tag{2.6}$$
$$\Theta = 2\pi U_2. \tag{2.7}$$
$$\tag{2.8}$$

En efecto, si X e Y son variables aleatorias de tipo normal estándar e independientes; realizando el cambio de variable de Jacobiano,

$$J = \begin{vmatrix} \frac{\partial X}{R} & \frac{\partial X}{\Theta} \\ \frac{\partial Y}{R} & \frac{\partial Y}{\Theta} \end{vmatrix} = R, \tag{2.9}$$

siendo F la función de distribución conjunta del par de variables (X, Y), entonces se puede

[4] Ref.: es.wikipedia.org/wiki/Método_de_Box-Muller

expresar,

$$\begin{aligned} F(X,Y) &= \frac{1}{(2\pi)^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\frac{x^2+y^2}{2}}dxdy \\ &= \frac{1}{(2\pi)^2}\int_{0}^{\infty}\int_{0}^{2\pi} Re^{-\frac{R^2}{2}}dRd\Theta(c.v.: r = R^2, dr = 2RdR) \\ &= \frac{1}{(2\pi)^2}\int_{0}^{2\pi} d\Theta \int_{0}^{2\pi}\frac{1}{2}e^{-\frac{r}{2}}dr \\ &= G(\Theta, r) \end{aligned}$$

Obteniendo, $(\Theta, r) = (U(0, 2\pi), \mathscr{E}(\frac{1}{2}))$, es decir, que Θ tiene una distribución uniforme $(0, 2\pi)$, y $r = R^2$ una distribución exponencial de parámetro $\lambda = 1/2$, ambas independientes. Además, estas variables aleatorias por ser de tipo uniforme y exponencial, pueden ponerse en función de U_1 y U_2 como se expresa en 2.6. Así, en resumen, se puede escribir el algoritmo de generación de una variable aleatoria con distribución normal como sigue:

1. Generar (U_1, U_2) dos muestra aleatorias independientes uniformes en $(0, 1]$.
2. Calcular $X = \sqrt{-2\log(U_1)}\cos(2\pi U_2)$
3. Devolver X.

La variable retornada por los pasos anteriores, X poseerá la distribución Gausiana normal estándar. Para calcular una variable aleatoria normal Gausiana con media m y varianza σ, basta hacer $G = m + \sigma X$, donde X es Gausiana normal estándar. Para reproducir el algoritmo de Box-Muller basta aplicar la función *Octave* siguiente,

- **f_box_muller.m** A.2.4

Una variables aleatoria Log-Normal, está relacionada con la distribución normal $X = N(\mu, \sigma)$ a través de la expresión, $Y = e^X$. De esta forma, se observa, que mientras X, toma valores en $\mathbb{R}$, la variables Y tomará valores en $(0, \infty)$. En el actual trabajo, se utilizan muestras aleatorias de tipo Y, para los casos de simulaciones con precios económicos. El algoritmo para la generación de una variable log-normal quedará:

1. Generar una aleatoria normal X con Box-Muller
2. Calcular la expresión $Y = e^X$
3. Devolver Y.

La variable aleatoria Y posee la distribución log-normal estándar con la misma media y varianza que X.

2.3.2. Generación de un Movimiento Browniano Geométrico

Existen multitud de métodos diferentes para la discretización de una ecuación diferencial estocástica, algunos de ellos muy sofisticados[5]. En este trabajo se ha utilizado el *método de aproximación de Euler* para la discretización de los problemas relacionados con la valoración de opciones.

Considerar $(W_t)_{t\geq 0}$ un movimiento Browniano estándar, y sean las funciones continuas deterministas siguientes $b : \mathbb{R} \to \mathbb{R}$ y $\sigma : \mathbb{R} \to \mathbb{R}$. Se supone para las funciones b y σ cumplen condiciones suficientes de regularidad para la existencia de única solución para la ecuación diferencial estocástica siguiente:

$$\begin{aligned} X_0 &= x \\ dX_t &= b(X_t)dt + \sigma(X_t)dW_t \end{aligned}$$

Si la discretización en tiempo se hace con una retícula fija Δt, entonces se puede construir una proceso en tiempo discreto $(S_n)_{n\geq 0}$ aproximando la solución de la ecuación diferencial estocástica en el tiempo $n\Delta t$,

$$\begin{aligned} S_0 &= x \\ S_{n+1} - S_n &= b(S_n)\Delta t + \sigma(S_n)(W_{(n+1)\Delta t} - W_{n\Delta t}). \end{aligned}$$

Si $X_t^n = S_{[t/\Delta t]}$, el proceso $(X_t^n)_{t\geq 0}$, aproxima $(X_t)_{t\geq 0}$ en el sentido siguiente:

Teorema 6. *Para cada $T > 0$,*

$$E\left(\sup_{t\leq T} |X_t^n - X_t|^2\right) \leq C_T \Delta t,$$

siendo C_T una constante dependiente de T.

La distribución de $(W_{(n+1)\Delta t} - W_{n\Delta t})_{n\geq 0}$ es la distribución de una secuencia de variables aleatorias normales e independientes. Es posible, sustituir $G_n\sqrt{\Delta t}$ por $(W_{(n+1)\Delta t} - W_{n\Delta t})$, donde $(G_n)_{n\geq 0}$ es una secuencia de variables aleatorias normales estándar e independientes. En éste caso, la aproximación $(S_n')_{n\geq 0}$ es en éste caso definida por:

$$S_0' = x \tag{2.10}$$

$$S_{n+1}' = S_n' + \Delta t b(S_n') + \sigma(S_n')G_n\sqrt{\Delta t}, \tag{2.11}$$

$$\tag{2.12}$$

[5]Pardoux and Talay (1985), Kloeden and Platen (1992).

Aplicación al modelo de Black-Sholes

En el caso del modelo de Black-Sholes, se simula la solución de la ecuación siguiente:

$$\begin{aligned} X_0 &= x \\ dX_t &= X_t r dt + \sigma X_t dW_t \end{aligned}$$

donde $(W_t)_{t\geq 0}$ es un movimiento Browniano y las funciones $b(X_t) = rX_t$ y $\sigma(X_t) = \sigma X_t$.

Existen dos métodos disponible para conseguir las aproximaciones a la solución de la ecuación diferencial estocástica. El primero consiste en utilizar la aproximación de Euler:

$$S_0 = x \tag{2.13}$$

$$S_{n+1} = S_n(1 + r\Delta t + \sigma G_n\sqrt{\Delta t}), \tag{2.14}$$

$$\tag{2.15}$$

simulando X_t por $X_t = S_{[t/\Delta t]}$.
Un segundo método consisten en utilizar la expresión explícita de la solución a la ecuación,

$$X_t = x \exp\left(rt - \frac{\sigma^2}{2}t + \sigma W_t\right)$$

simulando el BM por el método anterior 2.10. En este caso por $\sqrt{\Delta t}\sum_{i=1}^{n} G_i$, obteniendo,

$$S_n = x \exp\left((r - \frac{\sigma^2}{2})n\Delta t + \sigma\sqrt{\Delta t}\sum_{i=1}^{n} G_i\right) \tag{2.16}$$

Aproximando X_t por $X_t = S_{[t/\Delta t]}$.

Desde que un BM puede tomar valores negativos, no servirá como modelo para simular los precios económicos en los problemas financieros, para ello, habrá que utilizará su versión geométrica, GBM la cual tiene únicamente valores positivos. Para su simulación, vistas anteriormente en las ecuaciones 2.13 y 2.16. En este trabajo se han implementado los algoritmos anteriores en el scripts *Octave* siguiente,

- **test_geometric_brownian_motion.m** A.1.3

que depende de la función *Octave*

- **f_geometric_brownian_motion.m** A.2.5

En la figura 2.3 se presentan 100 simulaciones de GBM generados con la función anterior, con los siguientes parámetros:

- Precio inicial, $S_0 = 10$.

- Periodo, $T = 2$, en años.
- Iteraciones, $n = 1000$.
- Tipo de interés, $r = 0.1$.
- Dividendo, $\delta = 0.01$.
- Volatilidad, $\sigma = 0.4$.

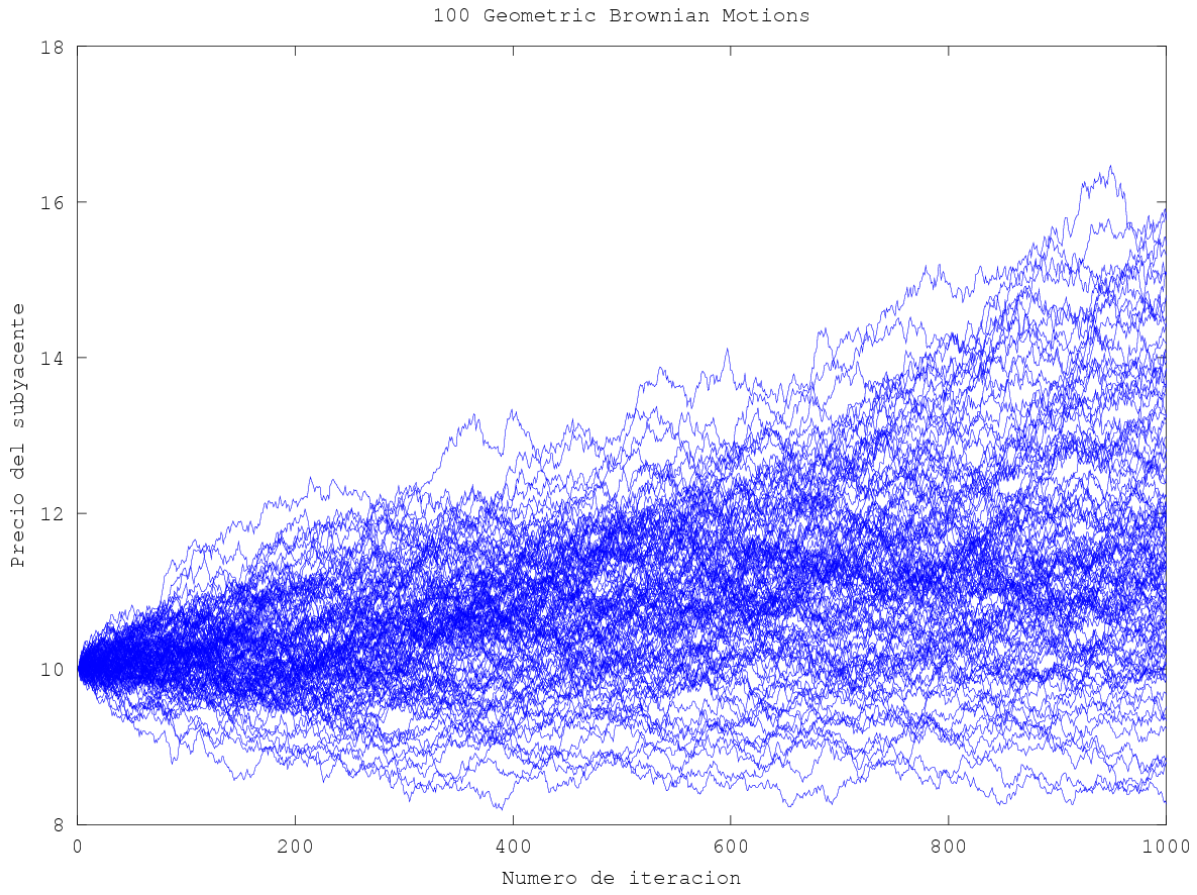

Figura 2.3: Simulación de 100 movimientos geométricos Brownianos, ($T = 2, n = 1000, r = 0.1, \delta = 0.01, \sigma = 0.4, S_0 = 10$)

Una propiedad interesante de un GBM $(B_t)_{t\geq 0}$, para problemas financieros, es la siguientes; dados $0 \leq s \leq t$ instantes de tiempo, el *coeficiente de autocorrelación* entre los valores B_s y B_t de un GBM será:

$$\rho = \frac{\text{mín}\{s,t\}}{\sqrt{st}}$$

Aplicando la fórmula anterior a diferentes instantes de tiempo uniformemente distribuidos, $t \in \{1, 2, 3, 4, 5\}$, se obtiene la siguiente matriz de auto-correlaciones:

$$\begin{pmatrix} 1 & 0.70710 & 0.57735 & 0.50000 & 0.44721 \\ 0.70710 & 1 & 0.81649 & 0.70710 & 0.63245 \\ 0.57735 & 0.81649 & 1 & 0.86602 & 0.77459 \\ 0.50000 & 0.70710 & 0.86602 & 1 & 0.89442 \\ 0.44721 & 0.63245 & 0.77459 & 0.89442 & 1 \end{pmatrix}$$

Capítulo 3

Métodos de Quasi-Monte Carlo.

En este capítulo se introducen los métodos QMC, definidos para una función de prueba f, y un conjunto de puntos $x_1, \ldots, x_N$ quasi-aleatorios, mediante el estimador siguiente,

$$\hat{\theta}^{QMC} = \frac{1}{N} \sum_{i=1}^{N} f(x_i),$$

Un conjuntos de puntos quasi-aleatorios estarán formados por una red de punto uniformemente distribuida en mayor medida que una distribución uniforme, sin *clusters*, acumulaciones o solapamientos, siendo su número N, el mínimo posible para alcanzar la precisión deseada. El estimador es igual que un MC, pero se diferencia en que QMC está utilizando secuencias de naturaleza determinista. Por tanto, su aplicación precisa de la generación de conjuntos quasi-aleatorios, que son en principio diferentes a los pseudo-aleatorios, teniendo la propiedad de baja discrepancia, *Low discrepancy sequences* (LDS). Además se introducirán varios tipos de secuencias LDS, comenzando por el caso básico de *Van der Corput* en una dimensión, y derivadas en dimensiones superiores a uno, deducidas a partir de ésta última, como pueden ser las secuencias de *Halton* y *Faure*. También se presenta una breve introducción a las secuencias de Sobol, que tendrán una forma de construirse diferente. Además, otro tipo de LDS, las generadas por *Lattice Rules*, estarán bien diferenciadas de las anteriores, siendo aplicadas a los problemas de prueba.
[1]

Los QMC proporcionan una convergencia más rápida y de mayor precisión que su versión aleatoria MC, como se verá en los ejemplos posteriores. Se puede demostrar,

[1] La **discrepancia** de un conjunto $P = \{x_1, \ldots, x_n\}$ se define como

$$D_N(P) = \sup_{B \in J} \left| \frac{A(B;P)}{N} - \lambda_s(B) \right|$$

, donde λ_s es la medida s-dimensional medida de Legesgue. $A(B;P)$ es el número de puntos $P \in B$, y J es un conjuntos s-dimensional formado por cajas de $\prod_{i=1}^{s}[a_i, b_i) = \{x \in R^s : a : i \leq x_i \leq b_i\}$ siendo $0 \leq a_i \leq b_i \leq 1$. Se define además la **discrepancia estrella** $D_N^*(P)$, análogamente, pero tomando el supremo sobre J^* conjunto de intervalos de la forma $\prod_{i=1}^{s}[0, u_i)$ con u_i en $[0, 1)$.

desde la desigualdad de *Koksma-Hlawka* [2] ,que el error absoluto cometido por un QMC, está acotado por el producto la variación total de la función del problema, y la medida de la discrepancia del conjunto de punto quasi-aleatorios. Por tanto, en general, los métodos QMC, siempre que se trabajen con funciones suficientemente *suaves*, es decir de variación acotada y relativamente pequeña, se comportarán mejor que su versión aleatorizada MC. En este capítulo se considerarán integrales múltiples definidas en el hipercubo $[0,1)^s$. En contrates a la eficiencia alcanzada por un MC, $O(\frac{1}{\sqrt{(N)}})$,gracias a **Koksma-Hlawka**, este permite a los QMC acotar su cota del error utilizando la discrepancia de los LDS donde se aplican. En la practica, es posible encontrar para LDS s-dimensionales [3] ,una discrepancia $D_N^*(x_1,\ldots,x_N) \leq C\frac{(\log N)^s}{N}$ con C constante.

3.1. Construcción de secuencias de baja discrepancia.

Intuitivamente, el concepto de secuencias de baja discrepancia, se refiere a la evitación de acumulaciones o huecos en la secuencia de puntos construida. Se busca una distribución lo más uniforme posible, obteniendo una distribución de puntos equidistantes entre sí llenando la máxima cantidad de espacio de trabajo. Con esta idea, se trata de encontrar puntos que maximicen la evitación de unos sobre otros, ya que, tener puntos solapados tan solo empeorará el proceso de convergencia de QMC al valor teórico en general.

3.1.1. Secuencias de Van der Corput

En 1935, JG van der Corput publicó una secuencia de baja discrepancia sobre el intervalo $[0,1]$, la cual, rellena los huecos del mismo, evitando las aglomeraciones de puntos.
La construcción de dicha secuencia se basa en transformar los números naturales $1,2,3,\ldots$, a su representación en base b fija elegida, extrayendo los coeficientes de la expansión polinómica resultante. A continuación, con los coeficientes obtenidos, se hace una reflexión especular respecto del cero entero de sus dígitos, convirtiendo esta último representación, de nuevo a su expresión decimal. Reiterando el proceso para cada número natural, se genera la secuencia de Van der Corput. Una propiedad de los elementos de una serie de Van der Corput, es que para cualquier base b, estos forman una secuencia densa [4] en el intervalo $[0,1]$.
Para construir la secuencia, suponiendo que $a_j(n)$ son los dígitos invertidos de n en base b, y para cada n se puede representar su expansión polinómica de potencia m

[2]Desigualdad de **Koksma-Hlawka**: Sea $\bar{I}^s = [0,1]^s$, y sea f una función de variación acotada en I^s, para todos $x_1,\ldots,x_n \in I^s$,

$$\left|\frac{1}{N}\sum_{i=1}^{N} f(x_i) - \int_{\bar{I}^2} f(u)du\right| \leq V(f)D_N^*(x_1,\ldots,x_n)$$

[3]Alcanzado para Van der Corput, Halton y Sobol.

[4]Se dice que $A \subset B$ es **denso** en B, cuando para todo $y \in B$, existe una sucesión $\{x_n\} \in A$ tal que $\lim_{n\to\infty} x_n = y$

$(a_j(n) = 0, j > m)$ de la siguiente fórmula:

$$n = \sum_{j=0}^{m} a_j(n) b^j$$

Así, el número correspondiente en la secuencia de Van der Corput en base b será

$$\phi_b(n) = \sum_{j=0}^{m} a_j(n) b^{-j-1}$$

El proceso de detalla más formalmente en el algoritmo generador en la siguiente subsección.

3.1.2. Algoritmo generador de una secuencia de Van der Corput

Como ejemplo ilustrativo, si se elige como base $b = 2$, el número $4 = 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$, es 100. Su reflexión de los dígitos será 0.001, que corresponde en decimal al número $\frac{1}{8} = 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-3} = 0.125$.

A continuación se presenta el algoritmo que transformar un número natural en uno de la secuencia de Van der Corput:

Algoritmo 1 Algoritmo de Van der Corput

Entrada: Base b, número n

Salida: n-ésimo término de la secuencia de Van der Corput c

$n_0 \leftarrow n$
$c \leftarrow 0$
$i_b \leftarrow 1/b$
mientras $n_0 > 0$ **hacer**
  $n_1 \leftarrow [n_0/b]$
  $i \leftarrow n_0 - n_1 b$
  $c \leftarrow c + i_b i$
  $i_b \leftarrow i_b/b$
  $n_0 \leftarrow n_1$
fin mientras
devolver c

Se ha implementado el algoritmo anterior, en una función *Octave*

- **f_van_der_corput.m** A.2.7

Ejecutando la función anteriores, para las bases 2 y 3, se obtienen las secuencia Van der Corput siguientes:

- 2: $1/2, 1/4, 3/4, 1/8, 5/8, \ldots$
- 3: $1/3, 2/3, 1/9, 4/9, 7/9, \ldots$

3.1.3. Secuencia Van der Corput Híbrida

Existe la posibilidad de generar conjuntos de puntos de baja discrepancia de dimensión s de forma sencilla a partir de una secuencia de Van der Corput. El procedimiento pasa por permutar la s-tupla de los s primeros términos de la secuencia de Van der Corput. Seguidamente se aplica una permutación aleatoria a la tupla de salida, para obtener una segunda, para después iterando el procedimiento, hasta obtener una matriz Nxs, donde cada una de sus entradas estará evaluada en $[0,1)^s$. Es posible generar ésta secuencia con las funciones *Octave* siguientes,

- **f_van_der_corput_ndim.m** A.2.9 que depende de,
  - **f_permuta_aleatoria.m** A.2.8
  - **f_van_der_corput.m** A.2.7

3.1.4. Secuencia de Halton

Las secuencias de Halton fueron introducidas en 1960, y generalizan la noción de van der Corput a varias dimensiones. Se construyen usando como bases para cada dimensión números coprimos entre sí. El conjunto generado en el caso para dimensión 2, utilizando como bases, 2 y 3, son los pares de puntos del plano,

- $(1/2, 1/3), (1/4, 2/3), (3/4, 1/9), \ldots,$

los cuales, son forma un conjunto denso en el cuadrado $[0,1)^2$. Ahora, utilizando el script *Octave* siguiente,

- **test_van_der_corput.m**, A.1.4

se ha generado un conjunto denotado, Halton-(2,3), presentado en la figura 3.1. Es posible generar conjunto en dimensiones mayores que dos, sin más que escogiendo como bases para las nuevas dimensiones, números coprimos con las base anteriores. Reiterando el proceso, permite generar conjuntos quasi-aleatorios en dimensiones arbitrariamente grandes. La pregunta que se plantea, es, ¿son mejores que las muestras aleatorias uniformes?, es decir, para un problema tratado cualquiera, ¿utilizar QMC(Halton) permite una convergencia hacia el valor teórico en un menor número de iteraciones?, es una cuestión que en principio se puede verificar con experimentos numéricos a problemas concretos, pero que en general será difícil de responder. A continuación, se explicarán varios ejemplos de aplicaciones de QMC(Halton), contratándolo contra MC crudo (sin control de varianza).

3.1.5. Cálculo del volumen de una s-esfera mediante QMC(Halton)

Análogamente como se hizo en la subsección 2.1.2, aquí se va a resolver la integral asociada al volumen de una esfera mediante QMC utilizando una secuencia de Halton. El problema se ha resuelto en dimensiones $4, 7, 10, 13$. Para resolverlo se han generado secuencia de Halton en las dimensiones propuestas. Para cada caso ha requerido un vector con tanto coprimos como su dimensión. Ejecutando el script *Octave*,

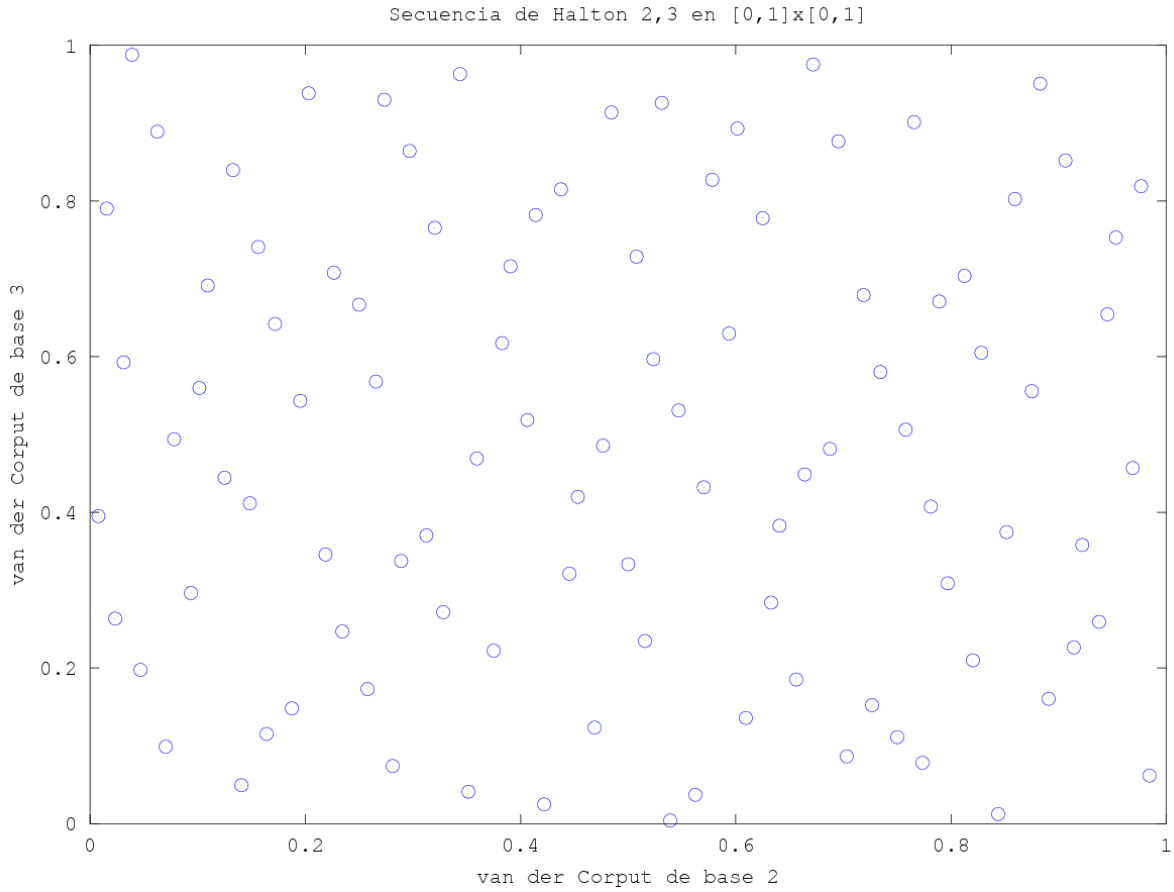

Figura 3.1: Secuencia de Halton 2,3 sobre el cuadrado $[0,1]^2$ formado por 100 muestras.

- **test_hiperesfera_volumen_QMC.m**, A.1.5

basado en las funciones *Octave* siguientes

- **f_hipercubo_punto_aleatorio.m** A.2.3
- **f_hiperesfera_dentro_radio.m** A.2.2
- **f_hiperesfera_volumen.m** A.2.1
- **f_halton.m**, A.2.10

se pueden ver los resultado obtenidos en la figura 3.2. La convergencia utilizando QMC(Halton), en azúl, es claramente superior y más directa, sin variaciones respecto al valor teórico, que utilizando MC crudo, en rojo. De la misma forma que en 2.1.2 se han utilizado $N = 50000$ muestras aleatoria y quasi-aleatorias.Los primeros resultados obtenidos del volumen para la 4-esfera han sido lo siguientes:

- Teórico : 4.9348.
- Estimado mediante MC: 4.9507.
- Estimado mediante QMC(Halton-2,3): 4.9348.

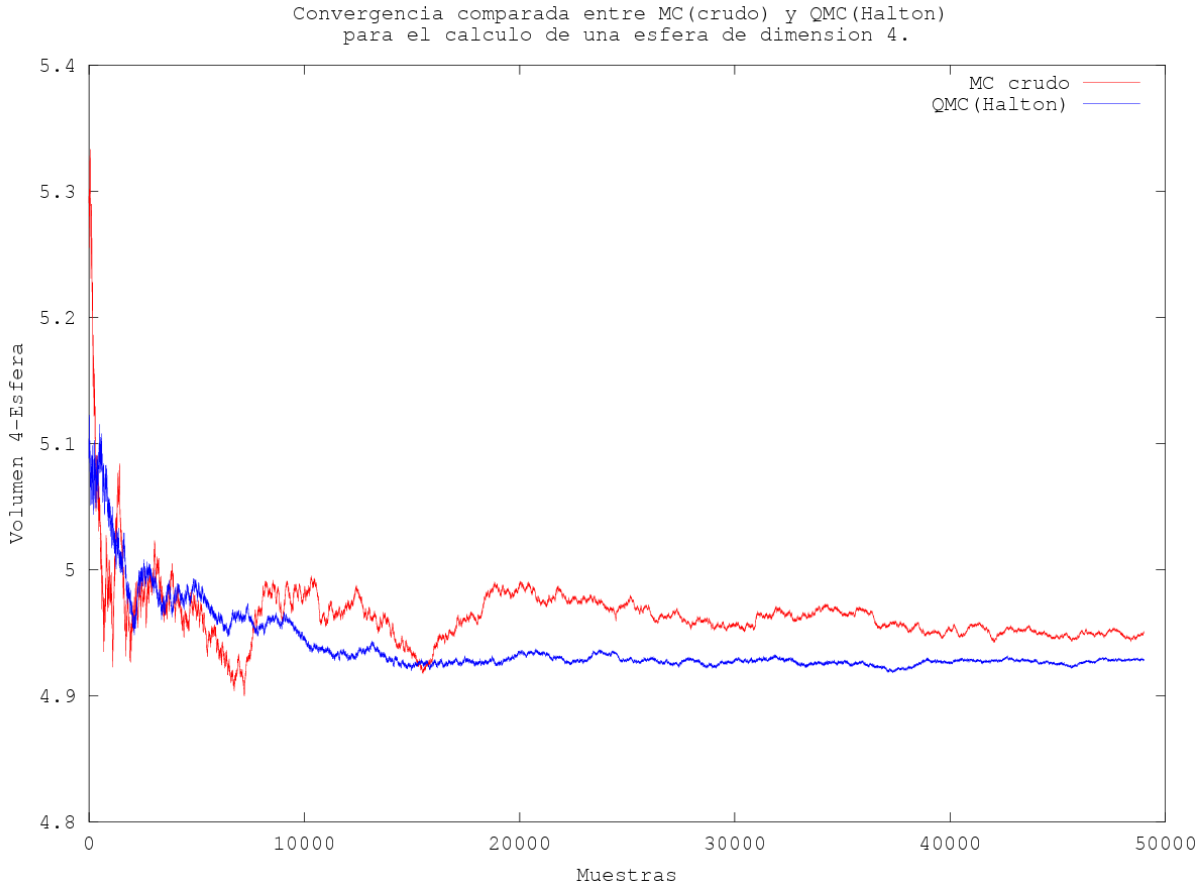

Figura 3.2: Comparativa de la convergencia de MC y QMC(Halton-2,3) para calcular el volumen de una 4-Esfera.

Repitiendo el experimento para dimensiones $7, 10, 13$, se obtienen resultados menos satisfactorios. Los resultados se resumen en el cuadro 3.1. Los conjuntos de coprimos utilizados para generar las secuencias de Halton para las diferentes dimensiones s han sido:

- $s = 4 : \{2, 3, 5, 7\}$
- $s = 7 : \{2, 3, 5, 7, 11, 13, 17\}$
- $s = 10 : \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29\}$

Dimensión	Teórico	MC	QMC(Halton)
4	4.9348	4.9507	4.9348
7	4.7248	4.6822	4.6925
10	2.5502	2.7034	2.6010
13	0.91063	0.98304	0.81920

Cuadro 3.1: Volumen de una s-esfera: teórico vs. calculado mediante MC crudo y QMC(Halton) para dimensiones $4, 7, 10, 13$.

- $s = 13 : \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41\}$

Se observa en el cuadro 3.1 que los resultados en dimensiones $3, 7, 10$ son mejores para las secuencias de Halton que para MC crudo, pero en dimensión 13, los resultados de QMC(Halton) dejan de ser significativamente mejores que en MC crudo.

3.1.6. Cálculo de integral s-dimensional mediante QMC(Halton)

Otro ejemplo para una integral en el que todas sus variables son igual de importantes, será utilizado para comparar MC contra QMC(Halton), examinando las posibles mejoras producidas por este último respecto a su versión aleatoria. Según ciertos resultados teóricos [5], las aplicaciones de QMC en integrales isotrópicas [6] de dimensión d son mejores respecto a MC. La expresión de la integral multidimensional a evaluar propuesta en este ejemplo será la siguiente:

$$\int_{[0,1)^s} \sin\left(\pi \sum_{i=1}^{s} x_i\right) d\bar{x} \tag{3.1}$$

donde, s es la dimensión del problema de integración[7].

Para obtener los resultados del cuadro 3.2, se ha ejecutando el script *Octave* siguiente:

- **test_integra_seno_n_dim.m** A.1.6

que depende de las funciones *Octave* siguientes,

- **f_halton.m** A.2.10

Los resultados del cuadro 3.2 muestran cómo es mejor la precisión alcanzada por QMC(Halton) respecto MC crudo. Por contra, la generación de los puntos de Halton, es mucho más lenta que su versión aleatorizada, siendo aproximadamente, entre 7 y 10 veces más costosa respecto de los puntos pseudo-aleatorios. Un inconveniente de las secuencias de Halton se pone en evidencia cuando se examinan los puntos generados para dimensiones altas, es decir, con ciclo de primo alto. Se puede ver en la figura 3.4 la degradación y la correlación entre dimensiones incluso con un número alto de muestras.

[5] Papageorgious,A y Traub,JF (1997), (Faster evaluation of mutidimensional integrals)

[6] Todas las variables tienen la misma importancia

[7] $\int_{[0,1)} sen(\pi \sum_{i=1}^{s} x_i) d\bar{x} = 0$ para s par.

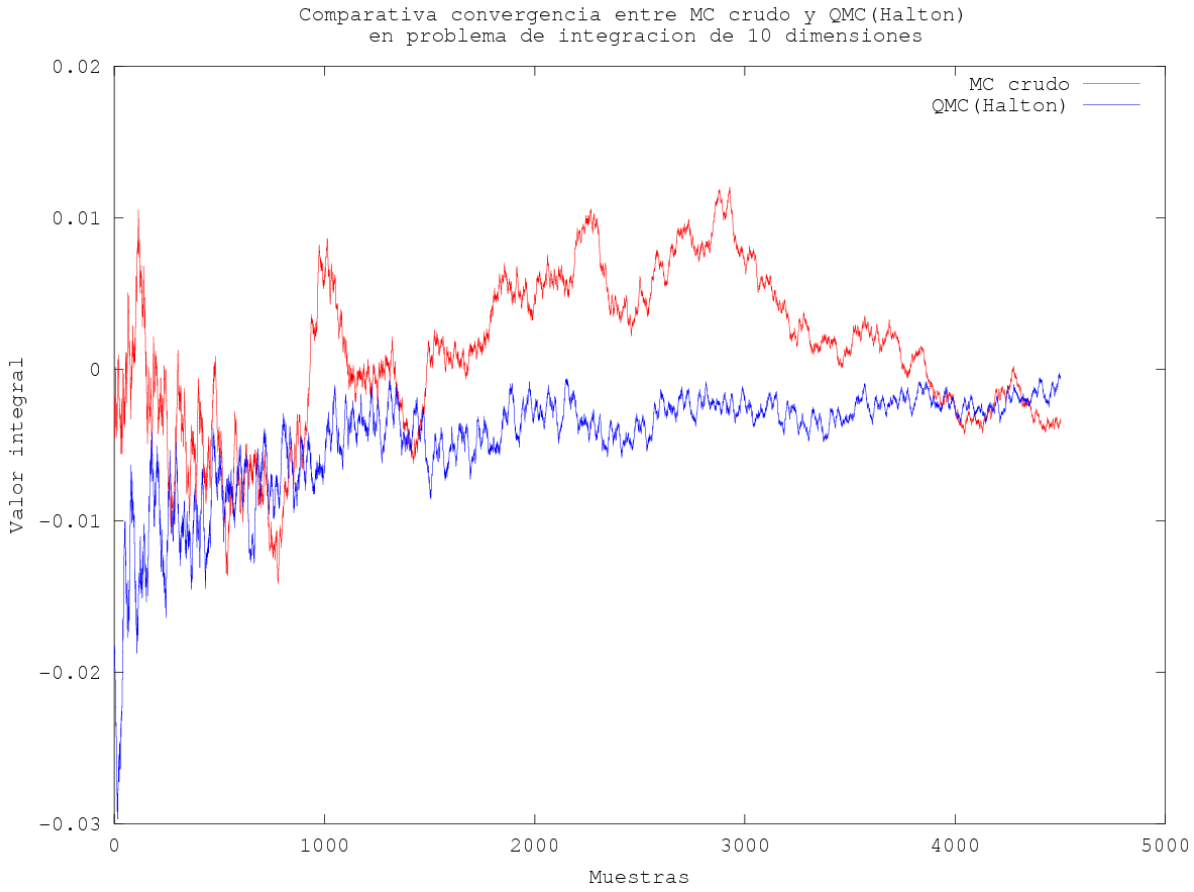

Figura 3.3: Convergencia comparada de MC y QMC(Halton) para 3.1

Dimensión	MC crudo	Tiempo MC	QMC(Halton)	Tiempo QMC(Halton)
4	0.0023619	3.6860	-3.0711e-04	33.666
10	-0.0019537	6.7611	-9.0648e-04	60.126
20	0.0018404	12.107	-0.0029891	97.362
40	-0.0044966	25.463	-9.9525e-04	201.23
100	-9.3560e-04	83.857	6.6134e-04	553.45

Cuadro 3.2: Resultados de la integración MC crudo y QMC(Halton) de Integral de 3.1 en diferentes dimensiones, junto al tiempo de computo estimado.

3.1.7. Secuencias de Faure

Estas secuencias serán similares a las de Halton en cuanto parten de Van der Corput, pero utilizarán únicamente una única base b en todo el proceso de generación, para después realizar una permutación aleatoria de los coeficientes en base b de cada término

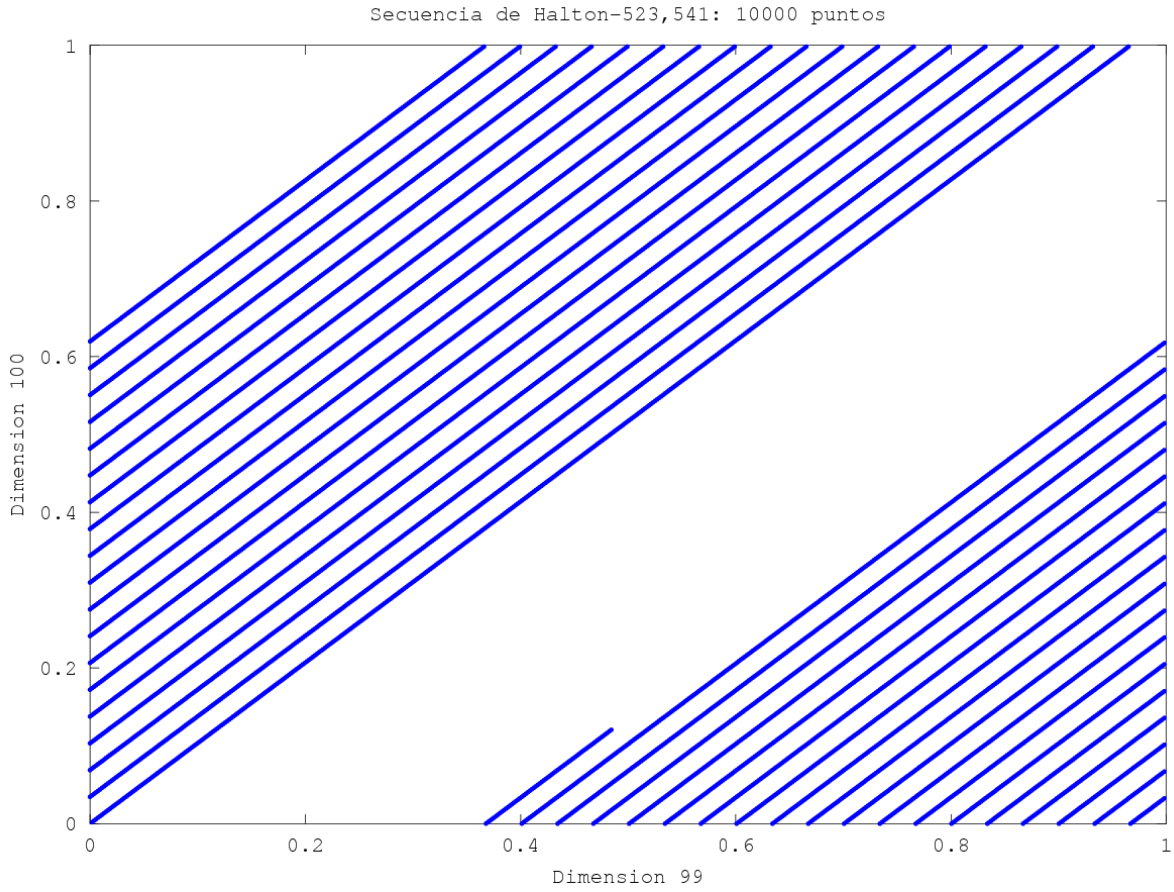

Figura 3.4: Degradación y correlación entre dimensiones de la secuencia de Halton.

de la secuencia de Van der Corput. Este método combina la teoría de secuencias de baja discrepancia y la teoría combinatoria de reordenación. Se tomará como base generadora de la secuencia, el mínimo número primo superior o igual a la dimensión del problema tratado. Faure trabaja con ciclos largos de primos para problemas de alta dimensión, lo que implica un alto coste computacional. De todas formas, esto último no es un inconveniente con respeto a una secuencia de Halton, ya que para una dimensión, dada, por ejemplo 60, el número primo siguiente a 60 es 61, en cambio para una secuencia de Halton, utilizará los 60 primeros primos, llegando a contener el primo 281. Para cada nueva dimensión, Faure hace una reordenación de los coeficientes de la dimensión inmediatamente anterior, evitando el problema de la correlación en dimensiones altas con Halton. El algoritmo de generación de la secuencia, en primer lugar, utiliza la ecuación de Van der Corput 3.1.1 con una base fija para cada término de la secuencia, y para todas las coordenadas. Dado el término n-ésimo de la secuencia de d-Faure, para obtener sus coordenadas $i = 1, \ldots, d$, se aplicarán sobre los coeficientes de la descomposición polinómica en base b de Van der Corput, $a_j(n)$, la siguiente permutación de forma recursivamente

para todas las dimensiones.

$$a_i^d(n) = \sum_{j \geq i}^{m} \binom{j}{i} a_j^{d-1}(n) \mod b,$$

siendo, b la base elegida, n el número de la secuencia, y d la dimensión tratada. De esta forma, la dimensión $d-1$, genera la dimensión d, hasta obtener un vector de la dimensión deseada.

3.1.8. Secuencia de Sobol

Las secuencias de Sobol, al igual que las de Faure, se basan en el mismo procedimiento de reordenación en cada dimensión. Sobol usa siempre la base 2 para todas las dimensiones, esto da ventajas respeto a Faure. Pero la tarea de reordenar es mucho más compleja, basada en vectores directores satisfaciendo una relación de recurrencia [8] utilizando polinomios primitivos irreducibles en el campos de Galois $G(2)$. Sobol sigue manteniendo la propiedad de rellenar el hueco en su siguiente términos, pero por contra tiende a la repetición de sus términos.

[8] http://en.wikipedia.org/wiki/Sobol_sequence

3.2. Reglas de Lattice

En ésta sección se van a describir las secuencias LDS basadas en *Lattice Rules* (LR) [9] a diferencia de las basadas en la secuencia de Van der Corput. Se aplicarán en varios problemas de prueba comparando los resultados obtenidos con MC crudo. Este método teóricamente es especialmente bueno para las integrales de funciones suaves, pero es posible aplicarlas sobre cualquier tipo de integrando. Una LR se define para un número fijo de puntos, aspecto que puede no ser ventajoso, ya que no se conoce a priori el número de puntos necesario para alcanzar cierta precisión deseada para un problema.

De forma general, se define una Lattice de integración s-dimensional, $\mathcal{L}$, como el subconjunto de $\mathbb{R}^s$ cerrado bajo suma y resta conteniendo a $\mathbb{Z}^s$ como subconjunto. Si el conjunto de puntos de integración P con un número de puntos N, es igual a $\mathcal{L} \cap \mathbb{R}^s$, entones el resultado se conoce como **Lattice Rule** de orden N.

Se pueden establecer la clasificación [10] de las *Lattice Rules*, de tal forma que para cada s-dimensional Lattice, existe un único entero r, $1 \leq r \leq s$, y enteros positivos $n_1, \ldots, n_r$, donde n_{i+1} es divisor de n_i para $i = 1, \ldots, r-1$, y $n_r > 1$, tal que los nodos de P se deducen desde,

$$\left\{ \frac{k_1}{n_1} z_1 + \cdots + \frac{k_r}{n_r} z_r \right\} \tag{3.2}$$

con $0 \leq k_i < n_i$ para $1 \leq i \leq r$, y $z_1, \ldots, z_r \in \mathbb{Z}^s$ son vectores de enteros, definiendo

$$\{z\} = (\{z_1\}, \ldots, \{z_s\}) \in [0,1)^s,$$

siendo $\{z\} = z(\mod 1)$. El resto de módulo 1 viene a decir, tomar la parte fraccionaria del número $xmod1 = x - [x]$, aplicando la operación separadamente a cada coordenada del vector.

Los puntos definidos por 3.2 son todos distintos con $N = n_1 \cdots n_r$. El valor de r más pequeño satisfaciendo la expresión 3.2 se conoce como el rango de la *Lattice Rule*. Los valores enteros $n_1, \ldots, n_r$ se conocen como invariantes. De esta forma, se puede definir el siguiente estimador θ basado en una *Lattice Rule* de N puntos:

$$\hat{\theta}^{LR} = \frac{1}{N} \sum_{j_r=0}^{n_r-1} \cdots \sum_{j_1=0}^{n_1-1} f\left(\left\{ \frac{j_1}{n_1} z_1 + \cdots + \frac{j_r}{n_r} z_r \right\} \right)$$

Como caso particular, en este trabajo se considerarán las *Lattice Rules* de rango-1 de N puntos en dimensión s, conocidas como *Good Lattice Points* (G.L.P.):

$$\{ \frac{k}{n} z \mod 1, k = 0, \ldots, n-1 \}, \tag{3.3}$$

[9] Introducidas por Sloan and Kachoyan (1987).
[10] Resultado teórico de Sloan and Lyness (1989)

con $z \in \mathbb{Z}^s$. Una forma de minimizar el espacio de búsqueda de buenas G.L.P.[11], a vectores z a los de la forma,

$$z(l) = (1, l, l^2 mod N, \cdots, l^{s-1} mod N), 1 \leq l < N \tag{3.4}$$

De esta forma, se restringe a un total de $N-1$ posibles elecciones, siendo mucho menor que en el caso general, N^s.

3.2.1. Aplicación de una *Lattice Rule* de rango 1.

A continuación, se volverá sobre el ejemplo de la s-esfera, aplicando QMC con una *Lattice Rule* de rango 1, del tipo 3.3. Se utilizará como vector generador 3.4. Para este tipo de *Lattice* se requiere buscar el parámetro más adecuado, elegido dentro del conjunto $\{l | 1 \leq l < N\}$, siendo N el número de puntos.

Caso 1: Encontrando el parámetro l visualizando los datos por pares de coordenadas.

Como posible procedimiento heurístico de obtención del parámetro l adecuado, se pueden seguir los siguientes pasos:

- *Paso 1:* Fijar la dimensión del problema s.
- *Paso 2:* Determinar el número N de puntos según la precisión deseada.
- *Paso 3:* Elegir en principio un l coprimo con N,entorno a 1/3 de N.
- *Paso 4:* Generar una *Lattice Rule* de rango 1 con parámetros s,N y l.
- *Paso 5:* Explorar los $\frac{s \cdot (s-1)}{2}$ pares de variables del conjunto generado.
- *Paso 6:* Si el conjuntos de integración muestra correlaciones fuertes o acoples entre pares de variables, descartar el conjunto volviendo al *Paso 3*, y eligiendo otro valor para el parámetro l.

Se reitera el proceso, hasta encontrar algún parámetro que muestre todos sus pares uniformemente distribuidos en cada par de coordenadas.
En el cada de la esfera de dimensión $s = 4$, es necesario generar una *Lattice Rule* de dimensión $s = 3$, evaluando la función para $s = 4$ siguiente,

$$f(x_1, \ldots, x_{s-1}) = \begin{cases} \sqrt{R^2 - \sum_{i=1}^{s-1} x_i} & , \| (x_1, \ldots, x_{s-1}) \| < R^2 \\ 0 & , \| (x_1, \ldots, x_{s-1}) \| \geq R^2 \end{cases} \tag{3.5}$$

[11] Consideradas por Korobov (1959)

Como caso particular, se ha fijado el número de puntos en $N = 500$, buscando el parámetro l, de la forma descrita en el párrafo anterior. Se ha encontrado como parámetro satisfactorio para el vector generador, un valor de $l = 131$. El aspecto de ésta LR en particular, según un examen de los pares de variables, se muestra en la figura 3.5. Se presenta una matriz de imágenes simétrica respecto de la diagonal. Los resultados numé-

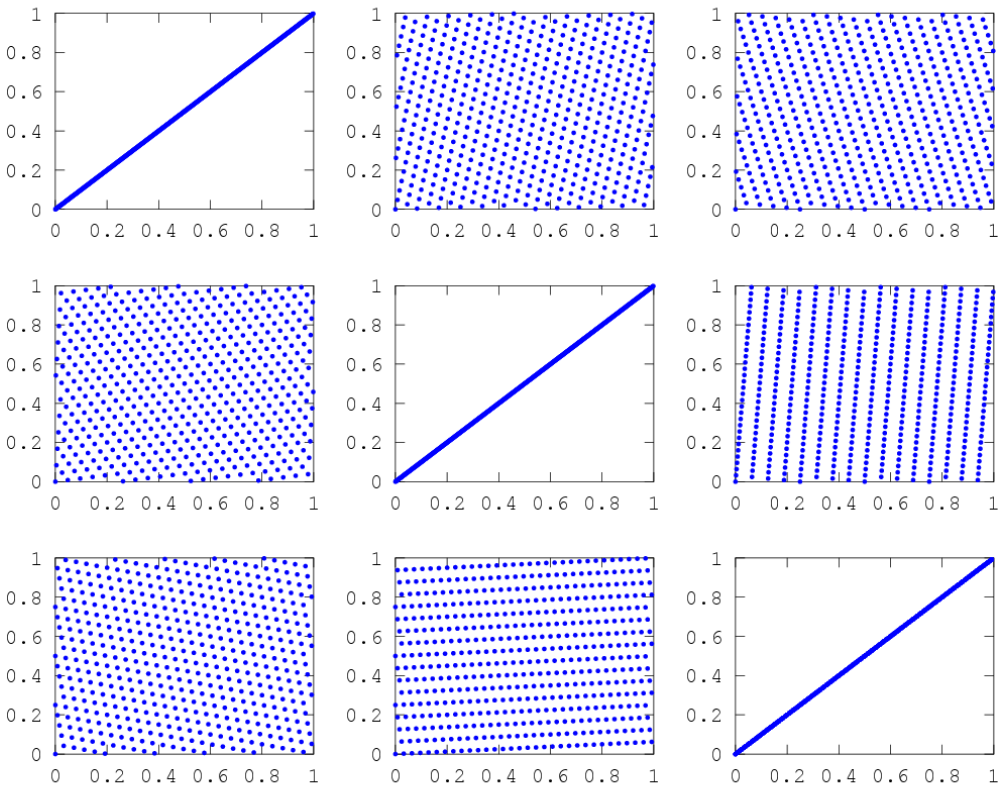

Figura 3.5: Visualización de los pares de coordenadas de una *Lattice Rule* de rango 1, con $N = 500$ de dimensión 3 y parámetro $l = 131$.

ricos para volumen utilizando QMC(*Lattice Rule*) se han obtenido ejecutado la función y script *Octave* siguientes;

- **test_n_esfera_vol_qmc_lattice_rule.m** A.1.8

que depende de la función *Octave*

- **f_hiperesfera_volumen.m** A.2.1
- **f_hipercubo_punto_aleatorio.m** A.2.3
- **f_hiperesfera_dentro_radi.m** A.2.2

- **f_glp_ndim.m** A.2.11

Se puede observar en el cuadro 3.3, como un QMC(*Lattice Rule*) mejora los resultados respecto a MC crudo, el cual tiene una gran varianza entorno al valor objetivo teórico. La convergencia hacia el resultado teórico ha seguido el comportamiento mostrado en 3.6.

Metodo	Valor
MC crudo	5.24800
QMC (*Lattice Rule*)	5.05962
Teórico	4.93480

Cuadro 3.3: Cálculo del volumen de una 4-esfera mediante una *Lattice Rule* de rango 1 de dimensión 3, número de puntos $N = 500$ y parámetro $l = 131$, contra MC crudo.

Caso 2: Evitando redes de integración *Lattice Rules* degeneradas. Búsqueda del parámetro l utilizando correlaciones entre pares de coordenadas.

Ahora, manteniendo el número de puntos $N = 500$, como el caso anterior, se tratará de buscar un parámetro l mediante un procedimiento más sistemático. La idea pasa por encontrar para cada parámetro posible del rango $1 \leq l < N$ de una *Lattice Rule* de rango 1, calcular la correlación entre cada par de coordenadas, que serán un número total de $\frac{s \cdot (s-1)}{2}$ correlaciones. La elección será aquella l, que haga mínima la correlación máxima entre pares de coordenadas. Se ha escrito un script que busca l con un algoritmo del orden $O(N^2 \cdot s)$, con s la dimensión. Ejecutando la función *Octave* siguientes

- **f_mincorr_param_glp_r1.m** A.2.12

se puede observar en el cuadro 3.4 la existencia de una mejora del cálculo respecto al caso 1 anterior. Este método no garantiza el mejor valor para una integral, pero sí uno suficientemente bueno, evitando *Lattices Rules* de integración degeneradas.

Metodo	Valor
MC crudo	5.3760
QMC (*Lattice Rule*)	4.9624
Teórico	4.93480

Cuadro 3.4: Cálculo del volumen de una 4-esfera mediante una *Lattice Rule* de rango 1 de dimensión 3, número de puntos $N = 500$ y parámetro $l = 115$, contra MC crudo.

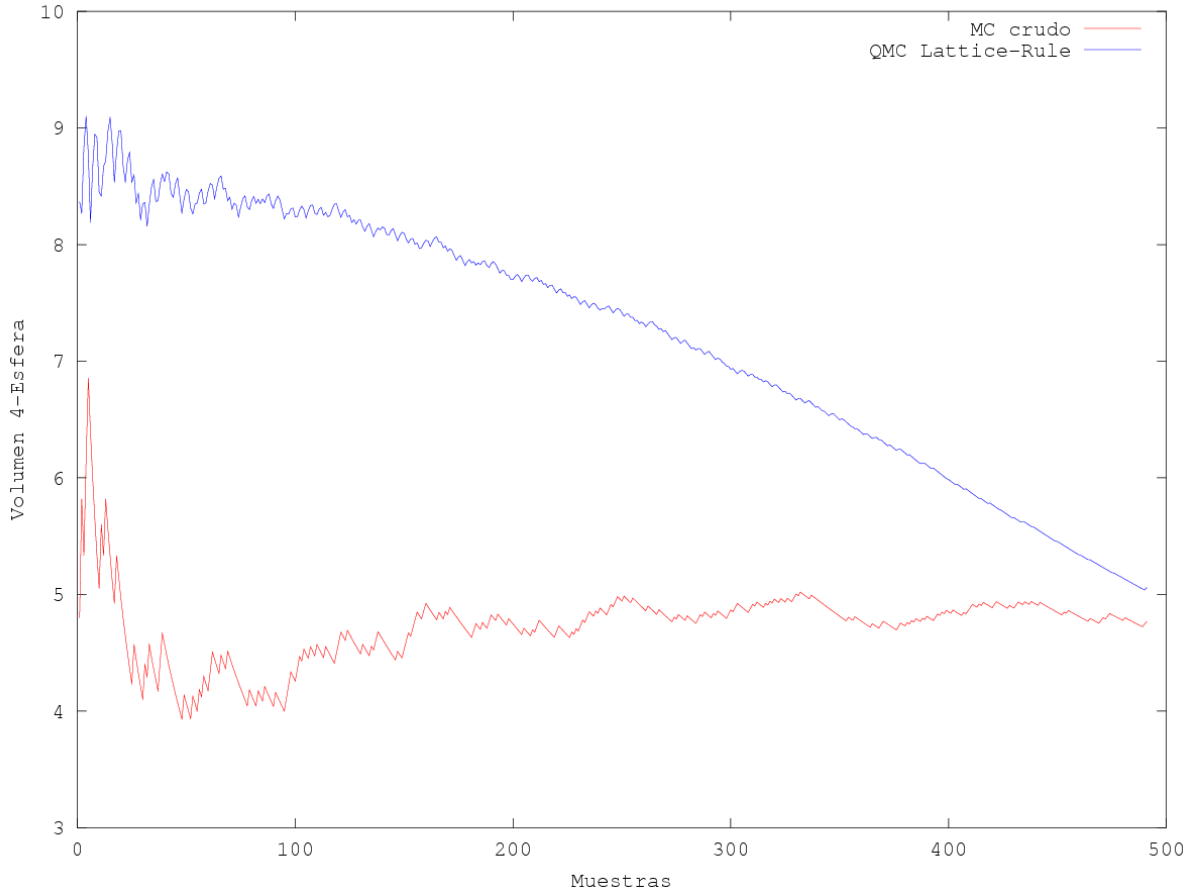

Figura 3.6: Comparativa de la convergencia del cálculo del volumen de una 4-esfera utilizando un QMC con *Lattice Rule* de rango 1 de dimensión 3, número de puntos $N = 500$ y parámetro $l = 131$ contra un MC crudo.

3.2.2. *Lattice Rule* para funciones periódicas

Añadiendo la hipótesis de periodicidad al integrando f de periodo uno en $\mathbb{R}^s$, es posible aplicar la teoría de las series de Fourier, llegando a una cota del error específica [12], con f una clase de funciones de s variables continuas periódicas con un comportamiento suave, $\mathscr{E}_\alpha^s(C)$,

$$\left| \frac{1}{N} \sum_{k=1}^{N} f(x_k) - \int_{[0,1)^s} f(u)du \right| \leq C \cdot \sum_{h \neq 0, h \in \mathcal{L}^\perp} r(h)^{-\alpha}$$

siendo:

- $\alpha > 1$ fijo, y C constante,
- $\mathcal{L}$ una Lattice de integración en $[0,1)^s$,

[12] Sloan and Joe 1995

- $r(h) = \prod_{i=1}^{s} \text{máx}(1, |h_i|)$,
- $\mathcal{L}^{\perp} = \{h \in \mathbb{R}^s : h \cdot x \in \mathbb{Z}, \forall x \in \mathcal{L}\}$, dual de una Lattice $\mathcal{L}$.

En el caso particular de *Lattice Rule*s de rango 1, la expresión anterior se reduce a:

$$\left| \frac{1}{N} \sum_{k=1}^{N} f(x_k) - \int_{[0,1)^s} f(u)du \right| \leq C \cdot \sum_{z \cdot h = 0 (\mod N)} r(h)^{-\alpha} = C \cdot P_\alpha(z, N) \tag{3.6}$$

siendo $\mathcal{L}^{\perp} = \{h \in \mathbb{Z}^s | h \cdot x = 0 (\mod N)\}$. Adicionalmente, se puede establecer [13] que para cada $s > 1$ y $\alpha > 1$, y cada primo N, la existencia de una función $\beta(s, \alpha)$ tal que,

$$P_\alpha(z, N) = O(N^{-\alpha} (\log N)^{\beta(s,\alpha)}) \tag{3.7}$$

Las secuencias de puntos $z = z(N)$ que satisfacen 3.7 se llaman *Good Lattice Points* (G.L.P.) y las componentes de z son los coeficientes óptimos. Cuando N es primo[14], $\beta(s, \alpha) = \alpha(s - 1)$.. Específicamente para el caso $s = 2$,[15], se puede demostrar que se alcanza la cota inferior en $\beta(s, \alpha) = s - 1$. De esta forma, la cota del error desde 3.6 y 3.7 quedará,

$$\left| \frac{1}{N} \sum_{k=1}^{N} f(x_k) - \int_{[0,1)^s} f(u)du \right| = O\left(\frac{(\log N)^{\beta(s,\alpha)}}{N^\alpha} \right)$$

Ahora, en el caso particular de $s = 2$, es posible construir un algoritmo para crear una *Lattice Rule* que alcance la cota del error anterior, utilizando como coeficientes óptimos $z = (1, F_{m-1})$, los número de Fibonacci F_m,

$$F_1 = 1, F_2 = 1, F_m = F_{m-1} + F_{m-2}, m \geq 3$$

Se elegirá como número de puntos $N = F_m$. De esta forma, la fórmula particular para la generación de los g.l.p en dimensión 2 será:

$$U_k = (\{\frac{k}{N}\}, \{\frac{kF_{m-1}}{N}\}), K = 1, \ldots, N, N = F_m, m \geq 3 \tag{3.8}$$

Se ha implementado la función *Octave* siguiente,

- **f_glp.m** A.2.26

para generar una GLP con $N = F_m$ en dimensión $s = 2$:

[13] Korobov(1959)
[14] Bahvalov(1959)
[15] Sargin(1963)

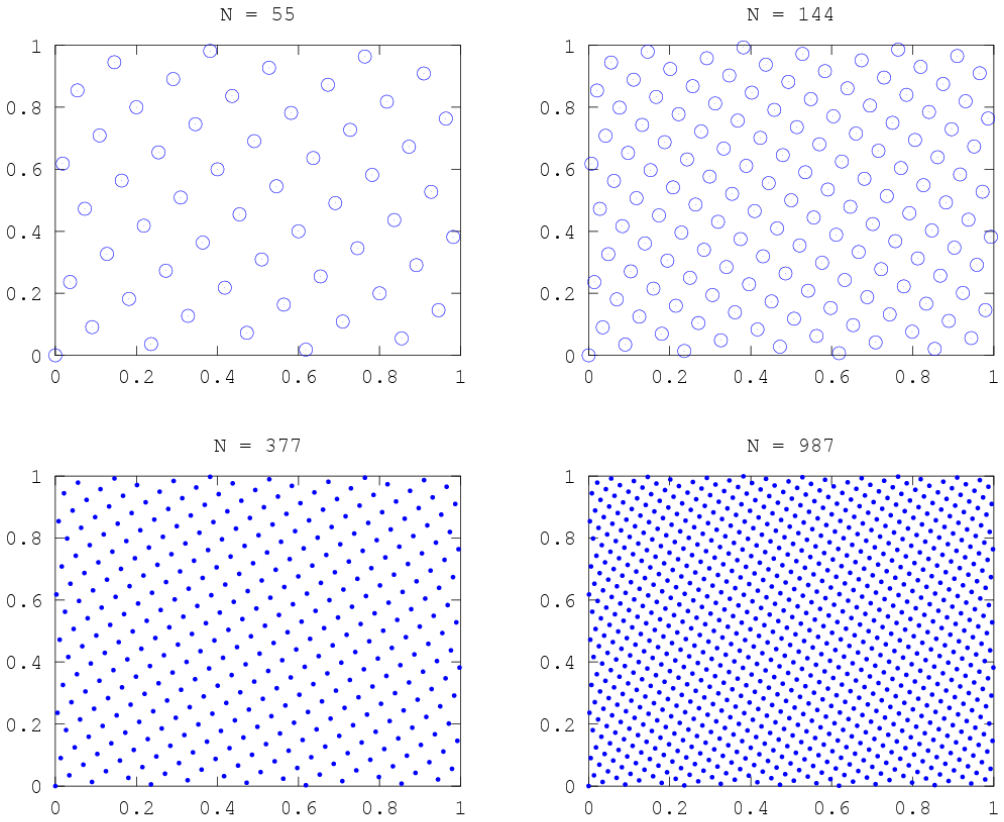

Figura 3.7: Good Lattice Points generadas con los números de Fibonacci $N \in \{55, 144, 377, 987\}$

Ejemplo: Aplicación de una *Lattice Rule* rango 1 para una función periódica.

Volviendo sobre el ejemplo de la integral de la función sin(), se necesita una función periódica de periodo 1 en $\mathbb{R}^s$, es decir, $\forall x \in \mathbb{R}^s, f(x+1) = f(x)$. Sea la función de prueba periódica siguiente,

$$\sin\left(2 \cdot \pi \sum_{i=1}^{s} x_i\right) \tag{3.9}$$

es una función periódica, en efecto, dado $x \in \mathbb{R}^s$, sumando $\overline{1} \in \mathbb{R}^s$,

$$
\begin{aligned}
f(x+1) &= \sin\left(2\pi \cdot \sum_{i=1}^{s}(x_i+1)\right) \\
&= \sin\left(2\cdot\pi\cdot\sum_{i=1}^{s}(x_i) + 2\cdot s\cdot\pi\right) \\
&= \sin\left(2\cdot\pi\cdot\sum_{i=1}^{s} x_i\right) + \sin(2\cdot s\cdot\pi) \\
&= \sin\left(2\cdot\pi\cdot\sum_{i=1}^{s} x_i\right) = f(x)
\end{aligned}
$$

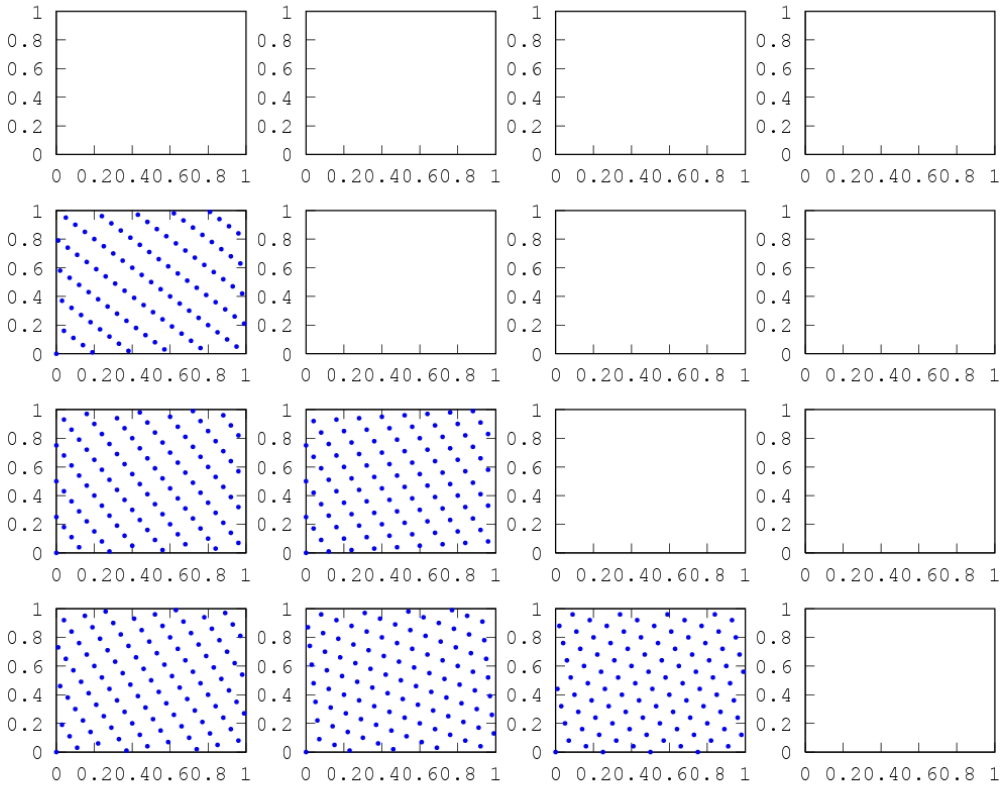

Figura 3.8: Visualización de las coordenadas por pares de una *Lattice Rule* de rango 1, de dimensión 4, con un número de puntos $N = 100$ y parámetro $l = 19$

A continuación, ejecutando el script *Octave* para dimensión $s = 4$, con una *Lattice Rule* rango 1 con $N = 100$, y parámetro $l = 19$,

- **test_integra_seno_ndim_qmc_lattice.m** A.1.7

se integra la función 3.9 sobre $[0,1)^4$, consiguiéndose los valores siguientes del cuadro 3.5

Metodo	Valor
MC crudo	0.022523
QMC (*Lattice Rule*)	$-1.5210 \cdot 10^{-16}$.
Teórico	0.0

Cuadro 3.5: Cálculo de la integral 3.9 mediante una *Lattice Rule* de rango 1 de dimensión $s = 4$, número de puntos $N = 100$ y parámetro $l = 19$, contra MC crudo.

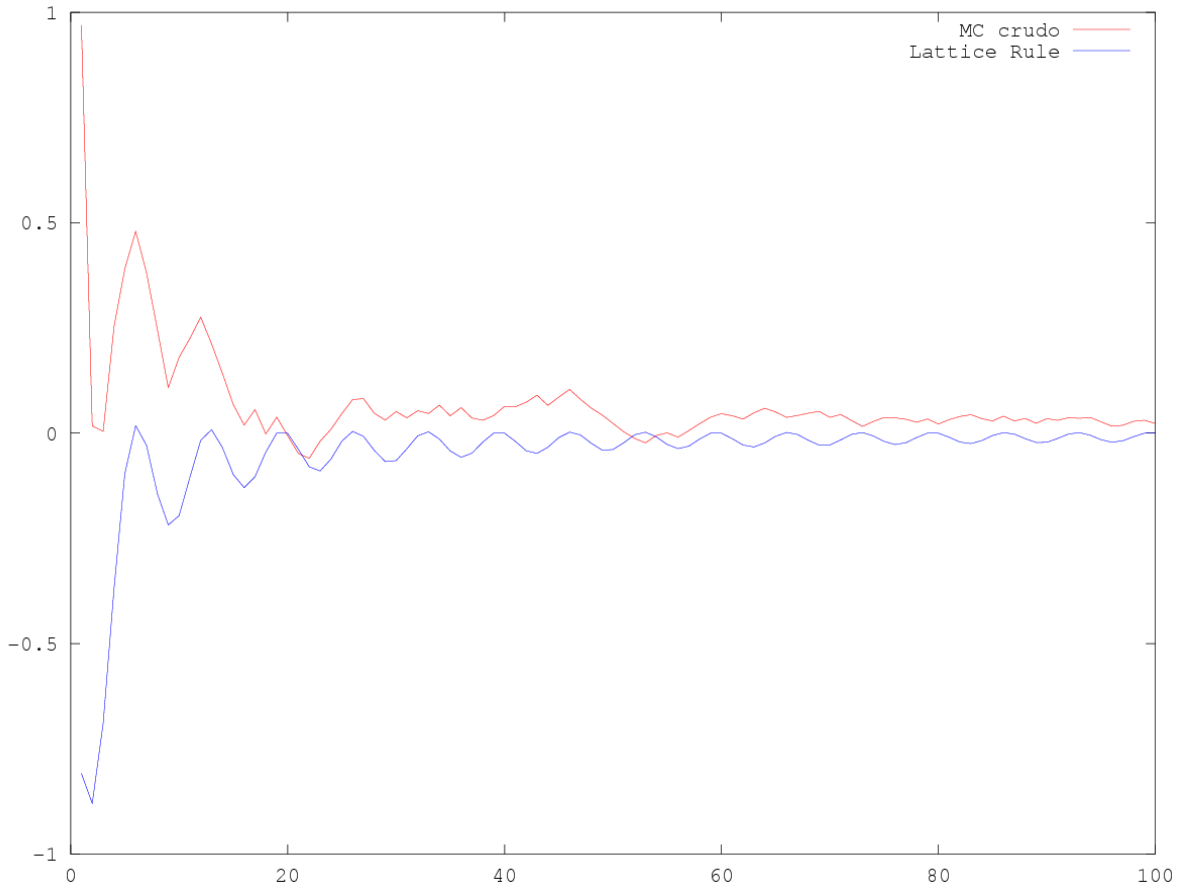

Figura 3.9: Comparativa de convergencia de la integración de una función periódica en el espacio de integración, para MC crudo y una *Lattice Rule* de rango 1 de dimensión 4 con $N = 100$ y parámetro $l = 19$.

Se puede observar un resultado prácticamente exacto con la red *Lattice Rule* con sólo $N = 100$ puntos, sin embargo, con MC crudo, apenas se ha conseguido el primer decimal exacto. Para obtener un buen parámetro de la red de integración *Lattice Rule*, se ha procedido por inspección de pares de coordenadas, viendo conveniente utilizar $l = 19$, por la distribución de puntos conseguido por éste. Se puede ver el aspecto de lar red en la figura 3.8
Por último, se observa la naturaleza determinista de la red de integración en la figura 3.9, además para una *Lattice Rule*, se puede deducir de la figura, que todos los puntos de la red de integración son en mayor medida necesarios, para alcanzar el valor exacto, en comparación a un MC, cuyo valor "oscila" estadísticamente entorno al valor objetivo teórico.

3.2.3. Construcción de una G.L.P.

Los resultados teóricos de 3.2.2 indican que sobre una clase de funciones $\mathscr{E}_\alpha^s(C)$ suaves, es posible obtener una *Lattice Rule* de rango 1, con un coeficiente óptimo z, y por lo tanto, alcanzar la cota del error 3.7. En el caso de dimensión 2, se vio que es posible construir una G.L.P. de forma óptima mediante sucesiones de Fibonacci. Ahora, para el caso de multidimensional, se tienen teoremas [16] que garantizan la existencia de una G.L.P para todo punto N de la red de integración, pero sin quedar determinada de una forma tan sencilla como en el caso de dimensión 2.
Para encontrar el coeficiente óptimo z de una G.L.P., habrá que ponerse en el peor caso de función dentro del espacio de funciones $\mathscr{E}_\alpha^s(C)$. Así, una G.L.P. que minimice la cota del error para la peor función f_α del espacio, también lo hará para una arbitraria $f \in \mathscr{E}_\alpha^s(C)$. Suponer que se tiene la función f_α siguiente,

$$f_\alpha(u) = \sum_{h \in \mathbb{Z}^s} \frac{\exp 2\pi i < h \cdot u >}{\text{máx}\, 1, |h|^\alpha}$$

Entonces, f_α es la peor función en $\mathscr{E}_\alpha^s(1)$, y se cumple la identidad,

$$\int_{[0,1)^s} f_\alpha(u) du = 1,$$

En efecto, desde que,

$$f_\alpha(u) = \prod_{i=1}^{s} F_\alpha(u_i). \tag{3.10}$$

[16] Niederreiter(1978): Teoremas de existencia de G.L.P. para todo tamaño N de puntos.

se puede escribir,

$$\begin{aligned} f_\alpha(u) &= \sum_{h\in\mathbb{Z}^s} \frac{1}{h(u)^\alpha} e^{2\pi i<h,u>} \\ &= \sum_{h\in\mathbb{Z}^s} \left(\left(\prod_{i=1}^{s} \text{máx}\,(1,|h_i|) \right)^{-\alpha} \exp\left(2\pi i(h_1u_1 + \ldots + h_su_s) \right) \right) \\ &= \sum_{h\in\mathbb{Z}^s} \left(\frac{\exp(2\pi i h_1u_1)\cdots\exp(2\pi i h_su_s)}{\text{máx}\,(1,|h_1|)^\alpha \cdots \text{máx}\,(1,|h_s|)^\alpha} \right) \\ &= \sum_{h_1\in\mathbb{Z}} \left(\frac{exp(2\pi i h_1u_1)}{\text{máx}\,(1,|h_1|)^\alpha} \right) \cdots \sum_{h_s\in\mathbb{Z}} \left(\frac{exp(2\pi i h_su_s)}{\text{máx}\,(1,|h_s|)^\alpha} \right) = \prod_{i=1}^{s} F_\alpha(u_i) \end{aligned}$$

Llamando F_α a,

$$F_\alpha(x) := \sum_{h=-\infty}^{\infty} \frac{exp(2\pi i h x)}{\text{máx}\,(1,|h|)^\alpha}$$

es suficiente probar que $\int_0^1 F_\alpha(x)dx = 1$.

$$\int_0^1 F_\alpha(x)dx = \underbrace{\sum_{\substack{n\in\mathbb{Z}\\ n\neq 0}} \frac{1}{\text{máx}(1,|n|)^\alpha} \int_0^1 e^{2\pi i n x}dx}_{A} + \int_0^1 1dx = A + 1.$$

Ahora, faltar ver que $A = 0$, emparejando términos en n y $-n$,

$$\begin{aligned} A &= \sum_{\substack{n\in\mathbb{Z}\\ n\neq 0}} \frac{1}{\text{máx}(1,|n|)^\alpha} \int_0^1 e^{2\pi i n x}dx = \sum_{n\in\mathbb{N}} \frac{1}{|n|^\alpha} \int_0^1 \left(e^{2\pi i n x} + e^{-2\pi i n x} \right) dx \\ &= \sum_{n\in\mathbb{N}} \frac{2}{|n|^\alpha} \int_0^1 \cos(2\pi n x)dx = 0. \end{aligned}$$

. De esta forma,

$$\left| \frac{1}{N} \sum_{k=1}^{N} f_\alpha\left(\frac{k}{N} z \right) - 1 \right| = \hat{P}_\alpha(z, N)$$

Lo que implica para cualquier $f \in \mathscr{E}_\alpha^s(C)$,

$$\left| \frac{1}{N} \sum_{k=1}^{N} f\left(\frac{k}{N} z \right) - \int [0,1)^s f(u)du \right| \leq C\hat{P}_\alpha(z, N)$$

obteniéndose la igualdad con $f = Cf_\alpha$. Esto permite encontrar una G.L.P. minimizando en z, $\hat{P}_\alpha(z, N)$. Directamente es difícil sacar un algoritmo de minimización, así que habrá que facilitar los cálculos de alguna forma.

Para facilitar el trabajo del algoritmo de minimización, hay que tener en cuenta que desde que el parámetro de regularidad α suele ser un número par, F_α se puede expresar explícitamente en términos de los polinomios de Bernoulli. En efecto, para el caso $\alpha = 2$, desde que el polinomio de Bernoulli $B_1(u) = u - \frac{1}{2}$, y aplicando series de Fourier a este polinomio, se obtiene, para $u \in [0, 1)$,

$$B_1(u) = u - \frac{1}{2} = \sum_{n>0} \frac{(-1)^{n+1}}{\pi n} sin(2\pi n(u - 1/2),$$

Desde la propiedad para los polinomios de Bernoulli, $B_n^{'} = nB_{n-1}$, e integrando, se obtiene que

$$\begin{aligned} B_2(u) &= -2\sum_{n>0} \frac{(-1)^{n+1}}{2\pi^2 n^2} cos(2\pi n(u - 1/2)) \\ &= \sum_{n>0} \frac{cos(2\pi nu)}{\pi^2 n^2} \end{aligned}$$

Despejando, $\pi^2 B_2(u) = \sum_{n>0} \frac{cos(2\pi nu)}{n^2}$, y como $F_2(u) = 1 + 2\sum_{n>0} \frac{cos(2\pi nu)}{n^2}$, y $B_2(u) = u^2 - u + \frac{1}{6}$ se obtiene,

$$F_2(u) = 1 + 2\pi^2(u^2 - u + \frac{1}{6}) = F_2(1 - u),$$

Análogamente, para $\alpha = 4$,

$$F_4(u) = 1 + \frac{\pi^4}{45}(1 - 30u^2(1 - u)^2) = F_4(1 - u),$$

Ahora, como $z \in \mathbb{Z}^s$, con cada coordenada $0 \leq z_i < N$, se tienen N^s posibilidades para encontrar una G.L.P., siendo demasiadas para valores grandes de N. Existe una restricción a los vectores $z \in \mathbb{Z}^s$ del tipo, lo que 3.4 [17]. permite buscar en un conjunto de $N - 1$ posibilidades. Todavía se puede restringir más el espacio, teniendo en cuenta que $u \in [0, 1), F_\alpha(u) = F_\alpha(1 - u)$, por lo tanto, como 3.10, $f_\alpha(\frac{z(l)}{N}k) = f_\alpha(\frac{z(l)}{N}(N - k)$, se divide en dos el espacio, teniendo que considerar tan sólo $z(l)$ para $1 \leq l \leq \lfloor N/2 \rfloor$. Se tiene que resolver el problema de minimización siguiente,

$$\min_{1 \leq l \leq \lfloor N/2 \rfloor} \hat{P}_\alpha(z, N) = \min_{1 \leq l \leq \lfloor N/2 \rfloor} \frac{1}{N} \sum_{k=1}^{N} f_\alpha(\frac{z(l)}{N}k)$$

Algunos valores de l óptimos[3, Sección 4] se presentan en el cuadro 3.6.

[17] Korobov(1959): Consideró restricciones al espacio de búsqueda de G.L.P.

N / Dim.	2	3	4	5
512	189	123	107	151
1024	399	173	493	363
1536	447	375	369	297

Cuadro 3.6: Valores óptimos para l en $z = (1, l, l^2 \mod N, \cdots, l^{s.1} \mod N)$ para estructura de *Lattice Points*. Se presentan las dimensiones en columnas(Dim,) y los puntos en filas(N).

3.2.4. Estimación del Error Estándar para QMC

Surge la pregunta de cómo calcular el error estándar con un método determinista como es QMC. Cranley y Paterson (1976) propusieron el mecanismo de producir un desplazamiento aleatorio de la rejilla de puntos, implicando diferentes estimaciones de la función objetivo. El desplazamiento se hace de forma aleatoria, es decir, se genera un vector aleatorio v en el espacio de trabajo $[0,1)^s$, se aplica a todos los puntos de la rejilla por igual, y finalmente se aplica módulo uno a los puntos desplazados, volviendo al espacio de partida. De esta manera se tiene una red de puntos $P = x_0, \ldots, x_{N-1}$, que será la anterior P, desplazada un vector v, $P + v$. Repitiendo el proceso para m muestras, se generan $v_1, \ldots, v_m$ vectores aleatorios $i.i.d$ obteniendo $P + v_i$ redes de puntos produciendo θ_i estimaciones de la función a estimar.

$$\hat{\theta}^{g.l.p.} = \frac{1}{m} \sum_{i=1}^{m} \theta_i$$

con una estimación del error estándar,

$$\sqrt{\frac{m-1}{m} \sum_{i=1}^{m} (\theta_i - \theta^{g.l.p.})^2}$$

De esta forma, se obtendrá un estimador insesgado de la media muestral de θ,

3.2.5. Periodización del integrando

Como se explicó en la subsección 3.2.2, es necesario tener un integrando periódico de periodo uno en el espacio $\mathbb{R}^s$, algo que en principio, parece muy restrictivo, ya que en general no se tendrá tal suerte. Por ello, en esta subsección, se van a tratar algunas técnicas de periodización para convertir las funciones del integrando en periódicas.

Una forma trivial de conseguir que el integrando sea periódico, es la siguiente, dada una función f definida en $x \in [0,1]$, se puede tomar la función

$$g(x) = \frac{1}{2}(f(x) + f(1-x)$$

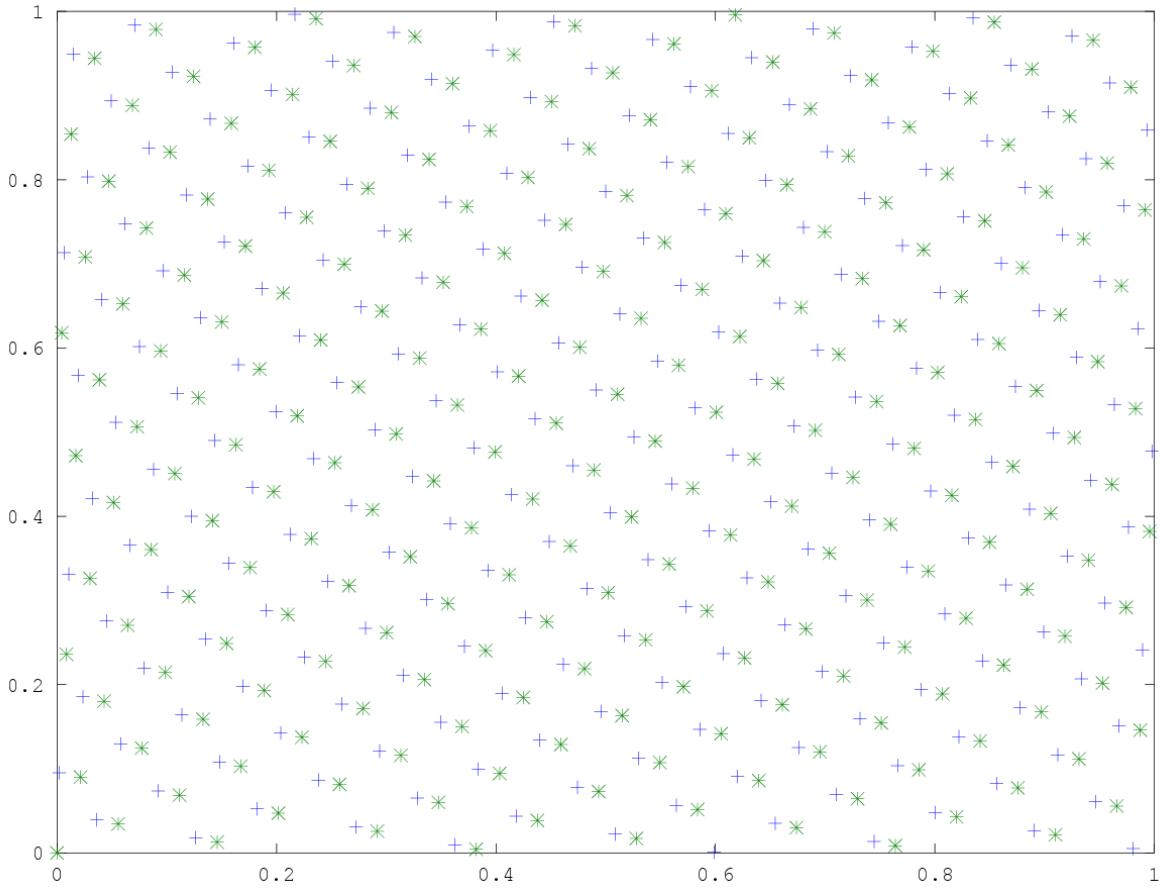

Figura 3.10: G.L.P. con N=233, junto a su desplazamiento aleatorio.

extendiéndose a la recta real de una forma continua y periódica de periodo 1. Se cumple trivialmente que las integrales siguientes son iguales:

$$\int_0^1 g(x)dx = \int_0^1 f(x)dx$$

El método anterior, puede ser extendido a funciones de varias variables, pero carece de diferenciabilidad en los puntos finales de los intervalos $[0,1]$. Así que éste método es utilizado en el caso de integrando que no sean suaves.

Se puede definir operadores de periodización de una forma más general. Sea $\psi(x) : [0,1] \to [0,1]$ una función creciente con derivadas continuas de orden $\alpha+1$. Suponer además que $\psi(0) = 0, \psi(1) = 1, \psi^{(n)}(0) = \psi^{(n)}(1) = 0, \forall, n = 1, \ldots, \alpha+1$. Entonces se llama a ψ una **transformación de periodización** de orden α.

Definición 12. *Sea función* $f : I^s \to \mathbb{R}$ *con* I^s *el cubo unidad* $[0,1)^s$. *Sean los coeficientes*

de Fourier $c(m_1,\ldots,m_s)$ *de la función* f*:*

$$c(m_1,\ldots,m_s) = \int_{I^S} f(x_1,\ldots,x_s)\exp\left(-2\pi i(m_1x_1+\ldots+m_sx_s)\right)dx_1\cdots dx_s$$
$$c(0,\ldots,0) = \int_{I^S} f(x_1,\ldots,x_s)dx_1\cdots dx_s$$

Se define la clase de funciones siguiente, $f \in \mathscr{E}_\alpha^s(C)$ *sí y sólo sí*

$$|c(m_1,\ldots,m_s)| \leq C|(\overline{m}_1,\ldots,\overline{m}_s)^{-\alpha}|,$$

siendo $\overline{m} = \text{máx}\{1,|m|\}$*, con* C *y* α *constantes.*

Una transformación de periodización de orden α, convierte una función con derivadas parciales continuas de orden α en una función periódica con las misma propiedades de diferenciabilidad además de pertenecer a la clase de funciones $\mathscr{E}_\alpha^s(C)$.

En el caso de integración sobre I^s de una función f no periódica con derivadas parciales continuas de orden α, considerar el cambio de variable siguiente:

$$\phi(u) = f(\psi(u_1),\ldots,\psi(u_s))\psi'(u_1)\cdots\psi'(u_s) \tag{3.11}$$

siendo ψ una función suave creciente, que mapea $[0,1)$ en $[0,1)$ y hace $\psi^{(j)}(0) = \psi^{(j)}(1) = 0$ para $1 \leq j \leq \alpha$. Entonces, el valor de las integrales siguientes coincide:

$$\int_{I^s} f(x_1,\ldots,x_s)dx_1\cdots dx_s = \int_{I^s} f(\psi(u_1),\ldots,\psi(u_s))\psi'(u_1)\cdots\psi'(u_s)du_1\cdots du_s$$

A continuación, se verán ejemplos concretos de transformaciones de periodización, que utilizan como cambio de variable ϕ, funciones algebraicas,(*polinómicas-m*) o funciones trigonométricas (*seno^m-transformaciones*).

Las transformaciones algebraicas con grado $m = 2,3,4$, y las utilizadas en este trabajo se muestran a continuación:

$$\psi_2(t) = 3t^2 - 2t \tag{3.12}$$
$$\psi_3(t) = 10t^3 - 15t^4 + 6t^5 \tag{3.13}$$
$$\psi_4(t) = 35t^4 - 84t^5 + 70t^6 - 20t^7 \tag{3.14}$$
$$\tag{3.15}$$

por otro lado, las primeras cuatro transformaciones trigonométricas con grado $m =$

$1, 2, 3, 4$, listadas a continuación

$$\xi_1(t) = \frac{1}{2}(1 - \cos(\pi t)) \tag{3.16}$$

$$\xi_2(t) = \frac{1}{2\pi}(2\pi t - \sin(2\pi t)) \tag{3.17}$$

$$\xi_3(t) = \frac{1}{16}(8 - 9\cos(\pi t) + \cos(3\pi t)) \tag{3.18}$$

$$\xi_4(t) = \frac{1}{12\pi}(12\pi t - 8\sin(2\pi t) + \sin(4\pi t)) \tag{3.19}$$

$$\tag{3.20}$$

Para calcular las transformaciones, se han implementado dos funciones *Octave* que aplican estas transformaciones sobre cada coordenada de los integrandos, permitiendo seleccionar la transformación ψ a utilizar, que en este caso se han fijado en las nombradas de tipo polinómica o trigonométricas, según entradas del usuario,

- **f_peridifica.m** A.2.33
- **f_peridifica_deriv.m** A.2.34

Los efectos de aplicar las transformaciones anteriores a una red G.L.P, producen un desplazamiento de sus puntos hacia las fronteras del recinto de integración $[0, 1)^s$.

EL efecto anterior, puede ser una fuente de problemas, como se verá en una aplicación más adelante, a la hora de manejarse con la función de distribución inversa de la normal estándar. La razón es que los valores cercanos a 0 por la derecha o 1 por la izquierda, son llevados por las aproximaciones de Φ^{-1}, (en el caso de *Octave* es *stdnormal_inv*), a infinito, produciendo símbolos NaN [18]. Para evitar tal producción indeseable de símbolos NaN, se ha de modificar la función *Octave stdnormal_inv* a otra función definida por el usuario, definida por partes, y acotando la evaluación para los valores extremos del dominio transformado.

Ejemplo: Volumen de una s-esfera mediante *Lattice Rule* con periodización.

Volviendo al ejemplo de la s-esfera, en ésta parte, como paso previo a aplicar QMC a la función del problema, se aplicará una transformación previa de periodización a la *Lattice Rule* y a la función del integrando 3.5.
A partir del caso 1 de la subsección 3.2.1, donde se tenían los parámetros, dimensión del problema $s = 3$, número de puntos $N = 500$ y parámetro de la red $l = 131$, se han aplican las transformaciones algebraicas 3.12 y trigonométricas 3.16. Para obtener los resultados de la integración, se ha ejecutado el script *Octave* siguiente:

- Script: **test_n_esfera_qmc_lr_perio.m**

Los números arrojados han mejorado los resultados en los casos de periodización.

[18] Símbolo utilizado por los paquetes de software pare las indeterminaciones, literalmente en inglés, NaN = "Not a Number"

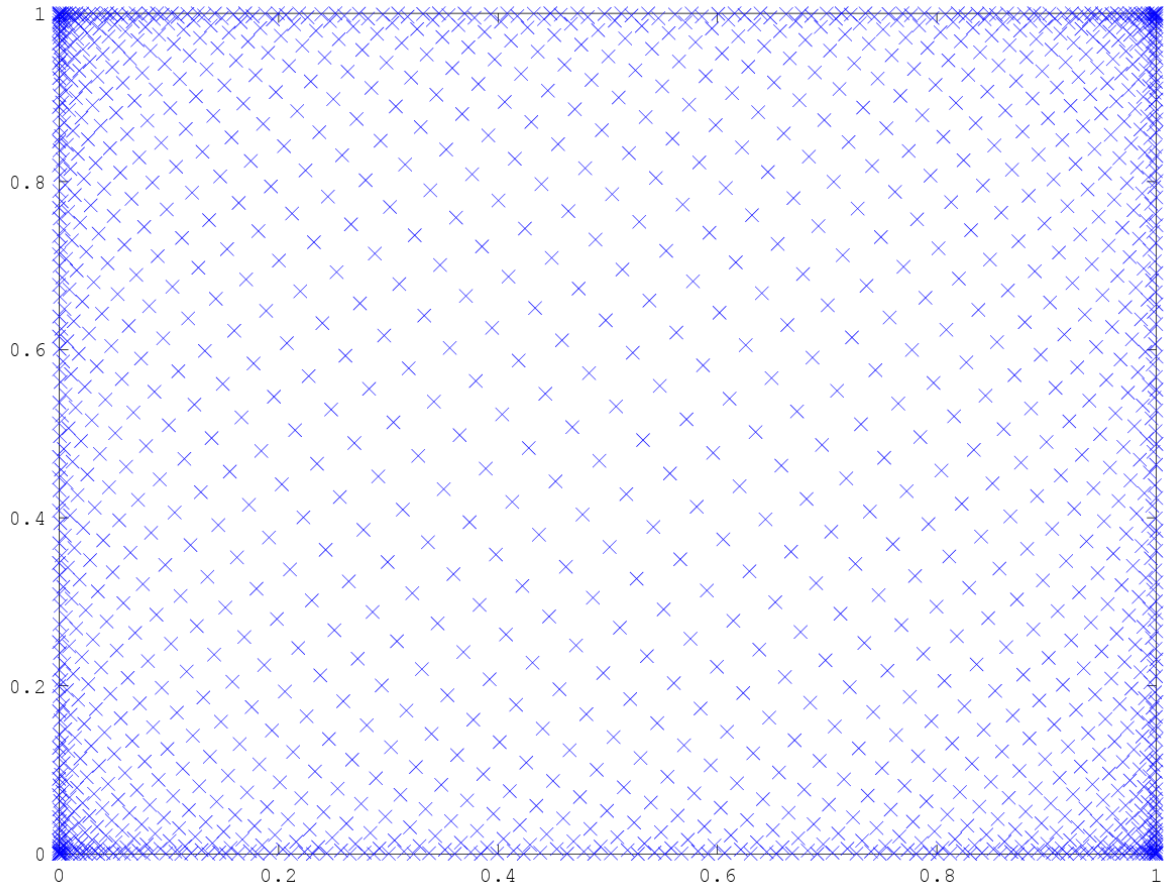

Figura 3.11: G.L.P. con $N = 1597$, sometidos a una transformación de periodización polinómica $\psi_4(t)$

Metodo	Valor (Error Estándar)
MC	5.0280 (0.57334)
QMC(*Lattice Rule* Sin Trans.)	5.0596
QMC(*Lattice Rule*-Periodiza. ψ_2)	4.9301
QMC(*Lattice Rule*-Periodiza. ψ_3)	4.9388
QMC(*Lattice Rule*-Periodiza. ψ_4)	4.9436
QMC(*Lattice Rule*-Periodiza. ξ_1)	4.9198
QMC(*Lattice Rule*-Periodiza. ξ_2)	4.9405
QMC(*Lattice Rule*-Periodiza. ξ_3)	4.9548
QMC(*Lattice Rule*-Periodiza. ξ_4)	4.9961
Teórico	4.93480

Cuadro 3.7: Cálculo del volumen de una 4-esfera mediante MC (8 intentos), QMC(*Lattice Rule*) de rango 1 de dimensión 3, número de puntos $N = 500$ y parámetro $l = 131$, según la transformación de periodización ϕ.

Capítulo 4

Simulaciones de Monte Carlo

En este capítulo se realizan simulaciones MC para los problemas de opciones financieras del capítulo 1.7. Además se aproximan precios y sensibilidades, para las opciones de tipo Call Europeas y Call Asiáticas. En la sección 4.1, se trabajará sobre una aplicación inicial aplicada al problema de las opciones Call Europeas, a modo de pruebas del método MC, examinando la relación del error en función del número de muestras y la volatilidad. En la sección 4.2, se introducen los estimadores insesgados para las sensibilidades o griegas de una opción, y en particular para las Call Europeas. Se expondrán dos puntos de vista para los estimadores, *pathwise* y *likelihood ratio*. Después, en la sección 4.3, se hace una introducción teórica sobre las condiciones necesarias para que un estimador de los tratados, sea insesgado. En la sección 4.4, se presentan los dos tipos de estimadores insesgados nombrados, añadiendo alguna aplicación práctica con MC. Adicionalmente, en la sección 4.5, se hace una breve introducción a una técnica de reducción de varianza y su aplicación práctica. Se podrán observar las mejoras que aportan estas técnicas a los problemas de valoración de opciones.

Las simulaciones y cálculos realizados utilizando los métodos MC han partido de la referencia [2], donde se habla principalmente sobre estimadores insesgados de las griegas o sensibilidades de opciones europeas y asiáticas, entre otras. El artículo de Broadie [2], presenta dos tipos de metodología para deducir estimadores de esta guisa, los de tipo *likelihood ratio* y los de tipo *pathwise-derivative*. Adicionalmente, en el mismo artículo nombrado, se realizan aproximaciones indirectas de las griegas mediante resimulación. Por otro lado, también se utilizan para todos los casos, técnicas de reducción de varianza, presentando las mejoras de los resultados numéricos. Se detallarán los algoritmos de los estimadores de [2] para el cálculo de las griegas, utilizando simulaciones MC, y cuyos resultados, servirán de comparación con otros métodos analizados en este trabajo, es decir, tanto contra los métodos exactos, como contra los métodos QMC, siendo éstos últimos detallados en el capítulo 3. Como se explicó en la sección 2.1, para aplicar una simulación MC sobre una función de una variable aleatoria, será necesario generar muestras aleatorias de las variables implicadas en las expresiones de los problemas. El tipo de muestras aleatorias serán creadas a partir de una distribución log-normal (para ver más detalles ver la sección 2.3).

4.1. Valoración del precio de una Opción Europea usando simulaciones MC

Típicamente, para calcular el precio de una opción Call Europea, se utiliza la fórmula de Black-Scholes, (ver sección 1.7.4). Como mecanismo de contraste para los algoritmos estudiados, en esta sección, se estimará el precio de la opción mediante simulaciones MC, examinando la precisión alcanzada y el ajuste estadístico respecto a su modelo teórico exacto, dependiendo del número de muestras y la volatilidad σ.

Para estimar el precio p de una opción Europea se considera un subyacente S_t que paga dividendos cumpliendo una difusión lognormal. En particular, se asume que el precio de riesgo nulo cumple la ecuación diferencial estocástica,

$$\frac{dS_t}{S_t} = (r - \delta)dt + \sigma dB_t, \tag{4.1}$$

siendo B_t un movimiento Browniano estandar, r es el tipo de interés, δ es la tasa de dividendo, y σ el parámetro de la volatilidad. Así, bajo la medida de riesgo nulo, la variable aleatoria $\log(S_T/S_0)$ seguirá una distribución normal de media $(r - \delta - \sigma^2/2)$ y varianza $\sigma^2 T$. Resolviendo la ecuación (4.1), el precio subyacente en el momento T de maduración, tendrá la forma siguiente,

$$S_T = S_0 e^{(r-\delta-\sigma^2/2)T+\sigma\sqrt{T}Z}, \tag{4.2}$$

siendo S_0, el precio del subyacente en el instante cero y Z la variable aleatoria normal estándar. Esta solución es un mecanismo para muestrear la variable aleatoria, S_T a partir de la variable aleatoria normal estandar Z. Para el caso del problema de la opción Europea, si el precio del strike es K, y el instante de maduración es T, el valor de la opción Call Europea vendrá dada por,

$$\begin{aligned} p &= E[e^{-rT}\,\text{máx}(S_T - K, 0)] \\ &= e^{-rT}E[\text{máx}(S_T - K, 0)] \end{aligned}$$

siendo $E[.]$ el operador esperanza sobre una variable aleatoria.

Ahora, se calculará el valor de la esperanza anterior, mediante el promedio de todas las muestras de la variable aleatoria $max(S_T - K, 0)$. Desde la solución vista en (4.2), se conoce la distribución seguida por la variable S_T, con lo que se permite generar las muestras aleatorias necesarias, y por tanto aplicar el método MC.

Algoritmo de Monte Carlo: Opción Call Europea
Se presenta a continuación, el pseudocódigo del algoritmo MC crudo:
Siendo las variables listadas a continuación los conceptos:

- S_0 : Precio del subyacente en el momento inicial.
- K : Precio de ejercicio.

Algoritmo 2 MC para una una opción Call Europea

Entrada: S_0,K,r,δ,σ,T,N

Salida: Precio $\hat{C}_n$ de una Opción Call Europea.

1: **para** $i = 1$ hasta N **hacer**

2: Generar muestra aleatoria normal estándar Z_i.

3: $S_T \leftarrow S_0 \exp\left((r - \delta - \frac{1}{2}\sigma^2)T + \sigma\sqrt{T}Z_i\right)$

4: $C_i \leftarrow e^{-rT}\,\text{máx}\{S_T - K, 0\}$

5: **fin para**

6: $\hat{C}_n \leftarrow \frac{1}{N}\sum_{j=1}^{N} C_i$

- r : Tipo de interés.
- δ : Dividendo del subyacente.
- σ : Volatilidad del subyacente.
- T : Periodo de maduración de la opción en años.
- N : Número de muestras normales estándar

Las condiciones de la opción Europea elegidas se han fijado en el cuadro 4.1.

K	S_0	r	δ	σ	T
90	100	0.01	0.05	0.1-0.4	1

Cuadro 4.1: K Precio Strike, S_0 Precio Inicial, r Tipo de Interés, δ Dividendo, σ Volatilidad, T Periodo de maduración en años

Ejecutando el script *Octave* siguiente,

- *test_ black_ scholes_ monte_ carlo.m* A.1.9

se han presentado los resultados en el cuadro 4.2, conteniendo los errores estándar del método MC para el precio de una opción Call Europea; según valores del parámetro de volatilidad $\sigma \in \{0.1, 0.2, 0.3, 0.4\}$; número de muestras consideradas $n = 10.000, 50.000, 100.000$. Los cálculos han utilizado para cada entrada del cuadro 15 muestreos independientes.

4.2. Metodologías para deducir estimadores insesgados para las griegas de Opciones Europeas

En esta sección, se tratará de deducir estimadores insesgados para las griegas de opciones Europeas. Como ya se ha nombrando, en la referencia [2] , se proponen dos metodologías para deducir estimadores de tipo, *pathwise* y *likelihood ratio*, la primera técnica,

	Valores de σ			
Muestras	0.1	0.2	0.3	0.4
10.000	0.27456	0.41879	0.69942	1.72259
50.000	0.14191	0.27064	0.35176	0.47488
100.000	0.076584	0.255489	0.298856	0.369766

Cuadro 4.2: Errores estándar del método MC para las estimaciones del precio de una opción Call Europea, donde n es el número de muestras y *sigma* la volatilidad.

se basa en la relación del pago que se obtiene al ejercer la opción, con el tipo de interés, y la segunda, con la relación entre la distribución de probabilidad del subyacente y el tipo de interés. Por lo tanto, de esta manera, los estimadores obtenidos por ambos métodos, no tendrán en general las mismas expresiones analíticas, lo que implicará diferencias numéricas en sus aproximaciones a los parámetros de las griegas a ser estimadas.

Para ilustrar la diferencia, se tratará el caso del tipo de griega *Vega* de una opción Call Europea. El estimador obtenido con la metodología *pathwise* para *Vega* ($\frac{\partial P}{\partial \sigma}$), siendo P el precio de la opción, se puede deducir desde que la variable aleatoria S_T se puede expresar como,

$$S_T = S_0 e^{(r-\delta-\sigma^2/2)T+\sigma\sqrt{T}Z}$$

ahora, derivando respecto del parámetro σ, se obtiene,

$$\frac{\partial S_T}{\partial \sigma} = S_T(-\sigma T + \sqrt{T}Z) \tag{4.3}$$

$$= \frac{S_T}{\sigma}[\log(S_T/S_0) - (r - \delta + \frac{1}{2}\sigma^2)T] \tag{4.4}$$

$$\tag{4.5}$$

Por otro lado, como $P = e^{-rT}\,\text{máx}(S_T - K, 0)$, derivando respecto de S_T

$$\frac{\partial P}{\partial S_T} = e^{-rT} 1_{\{S_T \geq K\}} \tag{4.6}$$

siendo $1_{\{.\}}$, la función indicadora del evento entre llaves. Ahora como $\frac{\partial S_T}{\partial \sigma} = \frac{\partial P}{\partial S_T}\frac{\partial S_T}{\partial \sigma}$, combinando los resultados de (4.3) y (4.6) se obtiene,

$$\frac{\partial P}{\partial \sigma} = \frac{\partial P}{\partial S_T}\frac{\partial S_T}{\partial \sigma}$$

$$= e^{-rT} 1_{\{S_T \geq K\}} \frac{S_T}{\sigma}[\log(S_T/S_0) - (r - \delta + \frac{1}{2}\sigma^2)T]$$

es un estimador *pathwise* para Vega, siendo sencillo de evaluar respecto a S_T.

En cambio, el estimador obtenido con la metodología "likelihood ratio" para *Vega*, es diferente, centrándose en la distribución de probabilidad asociada al precio subyacente.

Se deducirá a partir de la siguiente integral,

$$P = E[e^{-rT}\,\text{máx}(S_T - K.0)] = \int_0^\infty e^{-rT}\,\text{máx}(x-K,0)g(x)dx$$

siendo $g(x)$ la función de densidad de la variable aleatoria S_T, que típicamente será de tipo lognormal. Ahora, derivando bajo el signo integral respecto de σ, siempre suponiendo las condiciones suficientes de regularidad para el integrando, se obtienen las expresiones,

$$\begin{aligned}
\frac{\partial P}{\partial \sigma} &= \int_0^\infty e^{-rT}\,\text{máx}(x-K,0)\frac{\partial g(x)}{\partial \sigma}dx \\
&= \int_0^\infty e^{-rT}\,\text{máx}(x-K,0)\frac{\partial g(x)}{\partial \sigma}\frac{1}{g(x)}g(x)dx \\
&= \int_0^\infty e^{-rT}\,\text{máx}(x-K,0)\frac{\partial \log(g(x))}{\partial \sigma}g(x)dx \\
&= E\left[e^{-rT}\,\text{máx}(x-K,0)\frac{\partial \log(g(x))}{\partial \sigma}\right]
\end{aligned}$$

Así, el estimador *likekihood ratio* para *Vega* será,

$$e^{-rT}\,\text{máx}(x-K,0)\frac{\partial \log(g(x))}{\partial \sigma}$$

y que efectivamente, es bien diferente del encontrado en

4.3. Condiciones generales para estimadores insesgados

Considerando el proceso estocástico X_n, que puede representar los precios de un subyacente, y suponiendo que $f(X)$ es una función pago realizado por la opción, con f función real de variable real, $X = (X_1, \ldots, X_T)$ y T el momento de maduración, entonces el precio de una opción será, $p = E(f(X))$.

Ahora, suponer que las variables $X_n|n \geq 0$, dependen de un parámetro escalar θ sobre un intervalo Θ, es decir, $X_n = X_n(\theta)$. Se tiene la siguiente proposición:

Proposición:

Proposición 3. *Para todo $\theta \in \Theta$ intervalo, existe $\frac{\partial p(\theta)}{\partial \theta}$ y es igual a $E[\frac{f(X)}{\theta}]$, siempre que se cumplan (A1)-(A4).*

- *(A1): En cada $\theta \in \Theta$,*

$$X_n^{'}(\theta) = \lim_h \frac{X_n(\theta+h) - X_n(\theta)}{h}$$

- *(A2): Si D_f denota el conjunto de puntos diferenciable de f, entonces*

$$P(X(\theta) \in D_f) = 1, paratodo\theta \in \Theta$$

- *(A3): Existe una constante k_f tal que*

$$|f(x) - f(y)| \leq k_f|x - y|,$$

para todos los vectores x, y en el dominio de f.

Con la proposición 3, se aseguran estimadores insesgados [1] para las derivadas del precio respecto de sus parámetros independientes. La demostración puede verse en el artículo [2, Apéndice A].

4.4. Estimadores en Opciones Europeas

Ahora, con el apoyo teórico de la sección anterior 4.3, se puede asegurar, como caso particular, que los estimadores para las griegas de las opciones europeas son insesgados . En esta sección, se replicarán los resultados numéricos obtenidos en el artículo [2, Sección 4]. Para ello, se han desarrollados algoritmos que permiten evaluar las expresiones anteriormente deducidas.

Como se explicó en 4.2, los estimadores de tipo *Likelihood Ratio* tienen expresiones de la forma: Estas fórmulas se han codificado en varios ficheros *Octave*, conteniendo funciones de los parámetros de las opciones Call Europeas. La lista de ficheros *Octave* con extensión .m son los siguientes:

Las expresiones para los estimadores insesgados de las griegas de una opción Call Europea se han deducido de forma análoga a como se hizo en 4.2. El listado de expresiones se detalla a continuación según los dos casos de metodologías:

Caso *Pathwise*:

$$\begin{aligned}
Delta(\frac{\partial p}{\partial S_0}) &= e^{-rT}1_{\{S_T \geq K\}}\frac{S_T}{S_0} \\
Vega(\frac{\partial p}{\partial \sigma}) &= e^{-rT}1_{\{S_T \geq K\}}\frac{S_T}{\sigma}[\log(S_T/S_0) - (r - \delta + \frac{1}{2}\sigma^2)T] \\
Gamma(\frac{\partial^2 p}{\partial S_0^2}) &= e^{-\delta T}\frac{n(d_1(K))}{S_0\sigma\sqrt{T}} \\
Rho(\frac{\partial p}{\partial r}) &= KTe^{-rT}1_{\{\S_T \geq K\}}, \\
Theta(-\frac{\partial p}{\partial T}) &= re^{-rT}\,\text{máx}(S_T - K, 0) - 1_{\{S_T \geq K\}}e^{-rT}\frac{S_T}{2T}(\log(S_T/S_0) + (r - \delta - \frac{1}{2}\sigma^2)T)
\end{aligned}$$

Siendo $d_1(x) = [\log(S_0/x) + (r - \delta + \frac{1}{2}\sigma^2)T]/(\sigma\sqrt{T})$ y $n(x)$ la función de densidad de la normal estandar. Se observa que el estimador para *Gamma* es exacto en este caso, por lo tanto no requerirá de simulación MC.

[1] Se denomina sesgo de un estimador a la diferencia entre el valor esperado o esperanza del estimador y el verdadero valor del parámetro a estimar. Un estimador se dirá *insesgado* cuando su sesgo es nulo.

Caso *Likelihood Ratio*:

$$
\begin{aligned}
Delta(\frac{\partial p}{\partial S_0}) &= e^{-rT}\,\text{máx}(S_T-K,0)\frac{1}{S_0\sigma^2 T}(\log(S_T/S_0)-(r-\delta-\frac{1}{2}\sigma^2)T),\\
Vega(\frac{\partial p}{\partial \sigma}) &= e^{-rT}\,\text{máx}(S_T-K,0)\bigg(-d\frac{\partial d}{\partial \sigma}-\frac{1}{\sigma}\bigg),\\
Gamma(\frac{\partial^2 p}{\partial S_0^2}) &= e^{-rT}\,\text{máx}(S_T-K,0)\frac{d^2-d\sigma\sqrt{T}-1}{S_0^2\sigma^2 T},\\
Rho(\frac{\partial p}{\partial r}) &= e^{-rT}\,\text{máx}(S_T-K,0)\bigg(-T+\frac{d\sqrt{T}}{\sigma}\bigg)\\
Theta(-\frac{\partial p}{\partial T}) &= e^{-rT}\,\text{máx}(S_T-K,0)\bigg(r+d\frac{\partial d}{\partial T}+\frac{1}{2T}\bigg),
\end{aligned}
$$

Siendo

$$
\begin{aligned}
d=d(S_T) &= [\log(S_T/S_0)+(r-\delta-\frac{1}{2}\sigma^2)T]/(\sigma\sqrt{T}),\\
\frac{\partial d}{\partial \sigma} &= [\log(S_0/S_T)+(r-\delta+\frac{1}{2}\sigma^2)T]/(\sigma^2\sqrt{T}),\\
\frac{\partial d}{\partial T} &= [-\log(S_T/S_0)-(r-\delta-\frac{1}{2}\sigma^2)T]/(2\sigma T^{3/2}),
\end{aligned}
$$

Todas las expresiones anteriores, excepto *Gamma* para *Pathwise* por ser exacta, se han codificado en varios ficheros *Octave* conteniendo funciones de los parámetros de las opciones Call Europeas. La lista de ficheros *Octave* con extensión .m son los siguientes según casos:

- Caso *Pathwise*:
  - Delta: **european_call_estimator_pathwise_delta.m**
  - Vega : **european_call_estimador_pathwise_vega.m**
  - Rho : **european_call_estimador_pathwise_rho.m**
  - Theta: **european_call_estimador_pathwise_theta.m**
- Caso *Likelihood Ratio*:
  - Delta: **european_call_estimador_likelihood_delta.m**
  - Vega : **european_call_estimador_likelihood_vega.m**
  - Gamma : **european_call_estimador_likelihood_gamma.m**
  - Rho : **european_call_estimador_likelihood_rho.m**

- Theta : **european_call_estimador_likelihood_theta.m**

Las funciones anteriores se han agregado en un script *Octave* de prueba o test, que fija ciertas condiciones iniciales elegidas por el usuario. El script contiene un bucle que evalúa las funciones reiteradamente con algunas variación en los parámetros, en este caso en el precio de salida S_0. Las condiciones iniciales se han elegido como las del artículo [2], para evaluar la corrección de los algoritmos. En principio, se ha buscado replicar los resultados numéricos, con vistas a ser reutilizados en otras condiciones. Este test se puede ver en detalle en el anexo A.1.10:

- **test_european_call_estimadores_pathwise.m** A.1.10 que depende de las funciones,
  - **european_call_estimador_pathwise_delta.m** A.2.15
  - **european_call_estimador_pathwise_vega.m** A.2.16
  - **european_call_estimador_pathwise_rho.m** A.2.17
  - **european_call_estimador_pathwise_theta.m** A.2.18
- **test_european_call_estimadores_likelihood.m** A.1.11 que depende de las funciones,
  - **european_call_estimador_likelihood_delta.m** A.2.19
  - **european_call_estimador_likelihood_vega.m** A.2.20
  - **european_call_estimador_likelihood_gamma.m** A.2.21
  - **european_call_estimador_likelihood_rho.m** A.2.22
  - **european_call_estimador_likelihood_theta.m** A.2.23

Se han hecho pruebas en tres casos particulares de $S_0 \in \{90, 100, 110\}$ para las simulaciones de estimadores de las griegas de opciones Call Europeas. El cuadro 4.3 resume las condiciones iniciales utilizadas en la prueba.

K	r	δ	σ	T
100	0.1	0.03	0.25	0.2

Cuadro 4.3: K Precio Strike, r Tipo de Interés, δ Dividendo, σ Volatilidad, T Periodo de maduración en años.

Todas las simulaciones del cuadro 4.4 se han basado en 10.000 muestras aleatorias de S_T en el momento de maduración.

	Precio inicial del subyacente S_0					
	90	(Error Est.)	100	(Error Est.)	110	(Error Est.)
PW-Delta	0.21857	0.0045405	0.56321	0.0054030	0.84058	0.0040362
PW-Vega	11.657	0.26790	17.011	0.29461	10.918	0.38977
PW-Rho	3.6993	0.076708	10.261	0.097917	16.058	0.075466
PW-Theta	-8.5449	0.19106	-14.073	0.20283	-12.079	0.24542
LR-Delta	0.21974	0.0074755	0.56645	0.013036	0.81929	0.017407
LR-Vega	11.253	0.57064	17.555	1.1578	10.509	1.6014
LR-Gamma	0.027786	0.0014090	0.035110	0.0023155	0.017371	0.0026469
LR-Rho	3.7097	0.12788	10.308	0.24730	15.593	0.36452
LR-Theta	-8.3761	0.43160	-14.427	0.80016	-11.661	1.1113

Cuadro 4.4: Estimaciones de tipo *Pathwise*(PW) y *Likelihood Ratio*(LR) junto a sus *errores estándar* de las griegas de una Call Europea.

4.5. Técnicas de reducción de varianza

La convergencia de los métodos MC, desde 2.1, es del orden de $\frac{\sigma}{\sqrt{n}}$, siendo σ la varianza de la variable aleatoria y n el número de muestras o simulaciones. Por lo tanto, salta a la vista, que un estimador de tipo MC, dependerá de su varianza fuertemente, siendo deseable reducir ésta a la mínima posible, para mejorar la convergencia del método.

De esta manera, para mejorar la estimación de una simulación MC de una variable aleatoria X, se tratará de encontrar otra variable aleatoria Y, relacionada con la primera X, de tal forma que $E(X) = E(Y) + C$, siendo $\sigma(Y) < \sigma(X)$, y C una constante.

En el caso de los problemas para opciones financieras, para calcular un estimador con menor varianza se procederá de la siguiente manera:

Sea D un estimador insesgado para una sensibilidad de una opción, talque $E(D) = d$, el verdadero valor del parámetro estimado. Sea S_T el precio final simulado del subyacente.

Desde que $E(S_T) = e^{(r-\delta)T}S_0$, siendo r el tipo de interés, δ el dividendo, T momento de maduración de la opción, y S_0 precio inicial, un estimador insesgado de una griega de la opción es:

$$D' = D + \beta(S_T - e^{(r-\delta)T}S_0), \tag{4.7}$$

Siendo β un parámetro constante En efecto, tomando esperanzas en la ecuación anterior:

$$\begin{aligned} E[D'] &= E[D + \beta(S_T - e^{(r-\delta)T}S_0)] \\ &= E[D] + \beta E[S_T] - \beta E[e^{(r-\delta)T}S_0] \\ &= d + \beta e^{(r-\delta)T}S_0 - \beta e^{(r-\delta)T}S_0 \\ &= d \end{aligned}$$

Para encontrar β, tomando varianzas en la ecuación (4.7):

$$\begin{aligned} Var[D'] &= Var[D + \beta(S_T - e^{(r-\delta)T}S_0)] \\ &= Var[D] + Var[\beta(S_T - e^{(r-\delta)T}S_0] + Cov[D, \beta(S_T - e^{(r-\delta)T}S_0)] \\ &= Var[D] + \beta^2 Var[S_T] + 2\beta Cov[D, S_T] \end{aligned}$$

Derivando parcialmente respecto de β e igualando a 0,

$$\frac{\partial}{\partial\beta} Var[D'] = 2\beta Var[S_T] + 2Cov[D, S_T] = 0$$

Se obtiene el valor β que produce la varianza mínima para el nuevo estimador D',

$$\beta^* = -\frac{Cov[D, S_T]}{Var[S_T]}$$

Ahora, para estimar β^*, se usarán las estimación muestral de $Cov[D, S_T]$ y de $Var[S_T]$. De esta forma, si N es el tamaño muestral, S_T^i y D_i son evaluaciones de la muestra subyacente, $\bar{D} = \frac{1}{N}\sum_{i=1}^{N} D_i$ y $\bar{S_T} = \frac{1}{N}\sum_{i=1}^{N} S_T^i$, la estimación del valor de β^* será:

$$\hat{\beta}^* = -\frac{\frac{1}{N}\sum_{i=1}^{N}(D_i - \bar{D})(S_T^i - \bar{S_T})}{\frac{1}{N}\sum_{i=1}^{N}(S_T^i - \bar{S_T})^2}$$

De esta manera, aplicando las técnicas anteriores, para el caso de una opción Call Europea, teniendo las mismas condiciones de 4.3, se observa como mejoran significativamente los errores estandar en todos los casos estudiados, comparando los cuadros 4.4 y 4.5 de los resultados obtenidos para el caso con y sin control de varianza. Igualmente a 4.4, los experimentos del cuadro 4.5, se han basado en 10.000 muestras aleatorias de S_T en el momento de maduración. Para obtener los resultados del cuadro 4.5, basta aplicar el script *Octave* siguiente

- **test_european_call_estimador_pathwise_con_control.m** A.1.12

que depende de las funciones *Octave*

- **european_call_estimador_pathwise_delta.m** A.2.15
- **european_call_estimador_pathwise_vega.m** A.2.16
- **european_call_estimador_pathwise_rho.m** A.2.17
- **european_call_estimador_pathwise_theta.m** A.2.18

y el script *Octave*

- **test_european_call_estimador_likelihood_withcontrol_delta.m** A.1.13

que depende de las funciones *Octave*

- **european_call_estimador_likelihood_delta.m** A.2.19
- **european_call_estimador_likelihood_vega.m** A.2.20
- **european_call_estimador_likelihood_gamma.m** A.2.21
- **european_call_estimador_likelihood_rho.m** A.2.22
- **european_call_estimador_likelihood_theta.m** A.2.23

	Precio inicial del subyacente S_0					
	90	(Error Est.)	100	(Error Est.)	110	(Error Est.)
PW-CV-Delta	0.21918	0.0030934	0.56564	0.0029729	0.84404	0.0024939
PW-CV-Vega	11.719	0.17555	17.416	0.15647	11.332	0.21924
PW-CV-Rho	3.7077	0.053257	10.290	0.060316	16.108	0.057494
PW-CV-Theta	-8.6403	0.12460	-14.469	0.091254	-12.664	0.11999
LR-CV-Delta	0.21622	0.0060786	0.56955	0.0085929	0.83972	0.010393
LR-CV-Vega	11.358	0.55004	17.727	1.0222	11.620	1.4100
LR-CV-Gamma	0.028045	0.0013581	0.035454	0.0020444	0.019207	0.0023307
LR-CV-Rho	3.6521	0.10466	10.366	0.16654	16.011	0.22574
LR-CV-Theta	-8.3413	0.37796	-14.554	0.69198	-12.497	0.95307

Cuadro 4.5: Estimaciones con técnicas de control de varianza de tipo *Pathwise*(PW-CV) y *Likelihood Ratio*(LR-CV) junto a sus *errores estándar* de las griegas de una Call Europea.

En la figura 4.1 se muestra una comparativa de la evolución de la convergencia con el control de varianza.

4.6. Estimación por resimulación para las griegas

Otros mecanismos diferentes a los métodos directos, para calcular las griegas, es la resimulación, a costa de incrementar el coste computacional y sesgar el estimador en algunos casos. El estimador de resimulación es un cociente incremental, existiendo diferentes formas. El caso de una diferencia finita simple, será un estimador sesgado,

$$e^{-rT}\frac{f(g(S,\alpha),K)-f(g(S,\alpha+h),K)}{h}$$

siendo,

- r tipo de interés,
- T periodo de maduración,

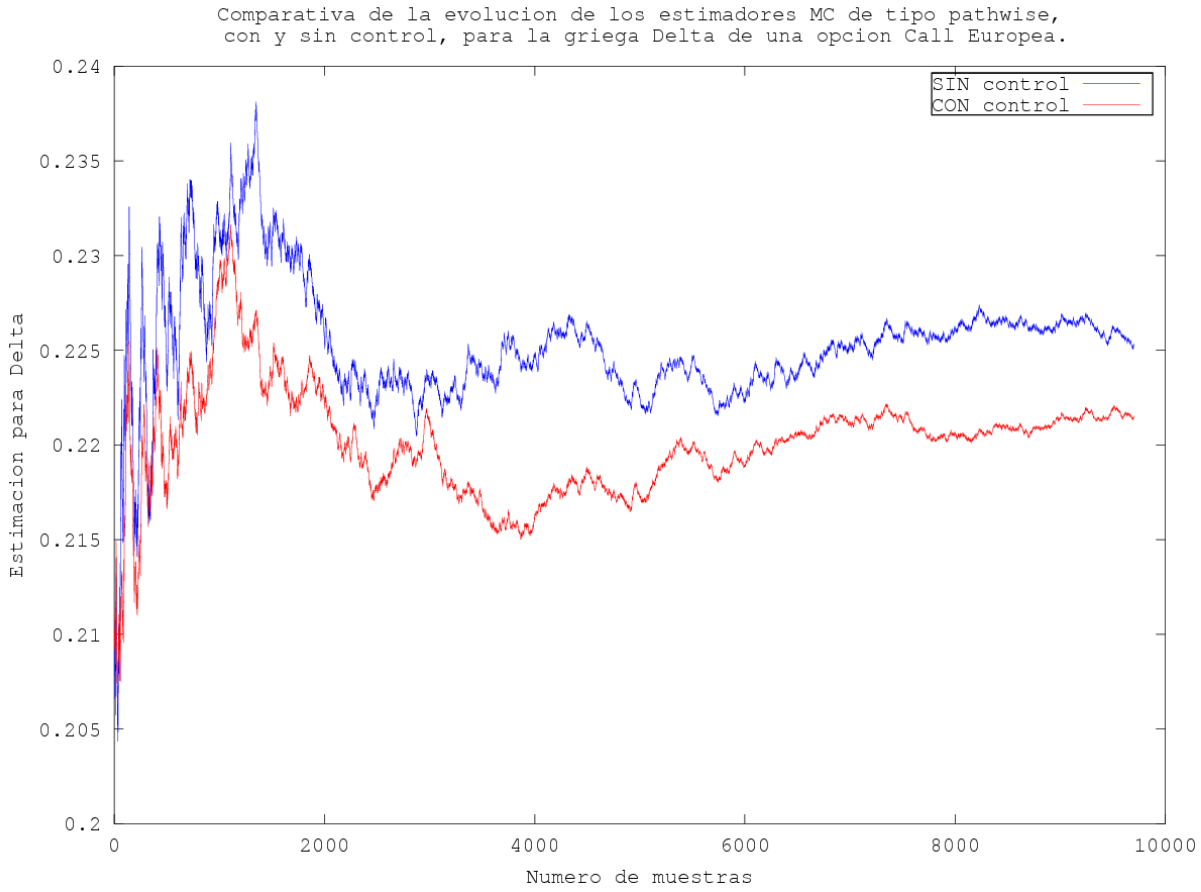

Figura 4.1: Comparativa de la evolución entre dos estimadores de tipo Pathwise, con y sin control de varianza, para la griega Delta, de una opción Call Europea.

- α el parámetro a estimar,
- K el precio del strike,
- f la función de pago,
  - Caso de opción Europea $f(S_T) = \text{máx}(S_T(\alpha) - K, 0)$,
  - Caso Asiática, $f(\overline{S}) = \text{máx}(\overline{S}(\alpha) - K, 0)$.
- g función de los precios S_t.
- h valor pequeño perturbador del parámetro α,

Además, también se podrían utilizar un cociente incremental, que fuera insesgado, pero requeriría más gasto computacional todavía, evaluando la función en los puntos simétricos

$\alpha - h$ y $\alpha + h$.

$$e^{-rT}\frac{f(g(S,\alpha-h),K)-f(g(S,\alpha+h),K)}{2h}$$

4.7. Estimadores en Opciones Asiáticas

En esta sección se exponen resultados numéricos de las aplicaciones desarrolladas, para calcular los estimadores de las griegas para las opciones Asiáticas. A diferencia de las opciones Europeas, este tipo de opciones dependerán además de todo el trayecto del subyacente S_t, además del momento de maduración. Tienen como función de pago la siguiente:

$$P(T) = \text{máx}(\overline{S} - K, 0),$$

siendo,

- $\overline{S} = \frac{1}{m}\sum_{i=1}^{m} S_{t_i}$ el promedio de los precios para un conjunto de puntos $0 \leq t_1 < \ldots < t_n \leq T$ en el intervalo de $[0, T]$.
- K el strike price,
- T el momento de maduración,

Ahora, de la misma forma a como desarrolló en la sección de 4.4, es posible deducir estimadores para las griegas de una opción asiática desde dos puntos de vista o metodologías de trabajo, *pathwise* y *likelihood*. Con las opciones asiática, los métodos MC, tendrán más sentido, porque para estas, no se conocen expresiones cerradas analíticas, ni para su precio, ni para sus sensibilidades. En el artículo de referencia [2, Apéndice C], se detallan las expresiones para los estimadores de sus sensibilidades o griegas.
Para trabajar con las opciones asiáticas, se han creado varias funciones y scripts en *Octave*. Primero, una función para generar movimientos geométricos Brownianos, segundo, una función del estimador para el precio de una opción asiática, tercero, una función para el estimador *pathwise* para *Delta*, y por último, un script recopilando todos los anteriores para calcular los siguientes conceptos:

- Precio de una opción Call Asiática.
- Precio de una opción Call Asiática usando Resimulación.
- Delta usando Pathwise.
- Delta usando Pathwise con Control de Varianza.
- Delta usando Resimulación.
- Delta usando Resimulación con Control de Varianza.

Los ficheros de extensión .m utilizados para realizar cálculos sobre parámetros de opciones asiáticas han sido las siguientes funciones *Octave*:

- **f_geometric_brownian_motion.m** A.2.5
- **asian_call_estimator_aritmetic_average_price.m** A.2.24
- **asian_call_estimator_pathwise_delta.m** A.2.25

y el siguiente script *Octave*

- **test_asian_call_estimators.m** A.1.14

Se presenta a continuación, el pseudocódigo del algoritmo para calcular el precio de una opción asiática mediante el MC crudo:

Algoritmo 3 MC para una una opción asiática

Entrada: S_0,K,r,δ,σ,T,m,n,N, $0 < t_0 \leq t_1 < \ldots < t_m = T$

Salida: Precio $\hat{C}_n$ de una Opción Call Asiática.

1: **para** $i = 1$ hasta N **hacer**
2: **para** $j = 1$ hasta m **hacer**
3: Generar muestra aleatoria normal estandar Z_{ij}.
4: $S_i(t_j) \leftarrow S_i(t_{j-1}) \exp\left((r - \delta - \frac{1}{2}\sigma^2)(t_j - t_{j-1}) + \sigma\sqrt{t_j - t_{j-1}} Z_{ij}\right)$
5: **fin para**
6: $\overline{S} \leftarrow \frac{1}{m}\sum_{j=1}^{m} S_i(t_j)$
7: $C_i \leftarrow e^{-rT}\,\text{máx}\{\overline{S} - K, 0\}$
8: **fin para**
9: $\hat{C}_n \leftarrow \frac{1}{N}\sum_{j=1}^{N} C_i$

Siendo las variables listadas a continuación los conceptos:

- S_0 : Precio del subyacente.
- K : Strike Price.
- r : Tipo de interés.
- δ : Dividendo.
- σ : Volatilidad.
- T : Periodo en años.
- m : Últimos días a precio de cierre.
- n : Número pasos para la discretización del GBM
- N : Número de GBM generados

Finalmente, los resultados numéricos obtenidos ejecutando los algoritmos anteriores se resumen en el cuadro 4.6. Las condiciones de la simulación están dadas en el cuadro 4.3 sobre 10.000 muestras, y que en este caso cada muestra será un GBM.

	Precio inicial del subyacente S_0					
	90	(Error Est.)	100	(Error Est.)	110	(Error Est.)
P	0.73037	0.024655	4.2615	0.061693	11.704	0.094493
P-CV	0.74908	0.019962	4.3098	0.033893	11.607	0.037105
Delta-R	0.16658	0.0040416	0.55183	0.0053274	0.87194	0.0036087
Delta-R-CV	0.16994	0.0030966	0.55572	0.0033452	0.86912	0.0025776
Delta-PW	0.16658	0.0040416	0.55183	0.0053274	0.87194	0.0036087
Delta-PW-CV	0.16994	0.0030966	0.55572	0.0033452	0.86912	0.0025776

Cuadro 4.6: Estimaciones según el precio de salida S_0 para el precio (P) de Opciones Call Asiáticas y de su parámetro Delta ($\frac{\partial p}{\partial S_0}$) mediante resimulación (R) o con *Pathwise*(PW) añadiendo o no control de varianza(CV).

Capítulo 5

Simulaciones Quasi-Monte Carlo

En este capítulo se presentan las simulaciones QMC para resolver los problemas de valoración de precios y griegas 1.7.7 de algunos tipos de opciones financieras sin expresiones analíticas conocidas, o bien, aún siendo conocidas, que sean intratables analíticamente o computacionalmente.

En la sección 5.2 se presentan soluciones para el problema de valoración de opciones de tipo Look-Back discretas. Para el caso de una opción Call, las opciones están diseñadas para buscar el máximo (o mínimo para una Put) del precio de ciertos momentos monitorizados y separados uniformemente desde un precio presente, hasta el momento de la maduración. Su cálculo se trata mediante secuencias quasi-aleatorias de tipo Halton y Lattice Rule, considerando además transformaciones de periodización para las últimas.

En la sección 5.3, se presentan mecanismos para transformar las expresiones analíticas del problema de valoración del precio y de las griegas de una Spread Options 1.7.6, para a continuación, utilizar MC crudo y QMC con *Lattice Rules* de dimensión 2. Se ha visto desde la sección 3.2.2, que una *Lattice Rule* de dimensión 2, se puede obtener de forma óptima a través de las sucesiones de Fibonacci. El aspecto trabajado es la implementación de las manipulaciones analíticas antes de aplicar QMC, experimentando como con muy pocos puntos. es posible obtener resultados significativamente mejores con respecto a MC e incluso QMC con aplicación directa.

5.1. Inversa de la normal estándar

La distribución de la normal estándar es una función no lineal para la cual no existe una inversa Φ^{-1} con una expresión analítica cerrada. La distribución normal es una función continua, monótona creciente, e indefinidamente diferenciable, que lleva los puntos del intervalo $(0, 1)$ a la recta real $\mathbb{R}$. Se necesita una implementación eficiente del algoritmo de la inversa para los problemas que utilicen valores muy extremos del dominio de la función inversa $(0, 1)$, en otro caso la inversión de la normal estándar puede resultar

una fuente de problemas. Con frecuencia, se pueden generar valores NaN con una mala implementación. Existen versiones de esta función inversa más precisas que pueden arrojar mejores resultados, evitando los indeseables NaN. Para ello, se ha programado la inversión en una función [1] *Octave* siguiente:

- **stdnormal_inv_acklam.m** A.2.32

5.2. Valoración de Lookback Options

En esta sección se presentan simulaciones para la valoración de opciones de tipo Lookback discretas 1.7.6 con momentos de monitorización discretos. Son un tipo de opciones, las cuales, poseen una fórmula analítica, pero es intratable computacionalmente, lo que fuerza el uso de métodos numéricos, en particular QMC. Las secuencias LDS aplicadas sobre ellas, son de tipo Halton, Van der Corput Híbrido y *Lattice Rule* rango 1. Además para estas últimas se aplican diferentes procesos de periodización 3.2.5.
El algoritmo para su cálculo, es análogo al visto en las opciones asiáticas de la sección 3, pero en lugar de utilizar una matriz de muestras normales estándar (Z_{ij}), se usa una transformación inversa normal estándar de todas las entradas de la matriz, es decir, $\Phi^{-1}(u_{ij})$, siendo (u_{ij}), una matriz de muestras contendidas en $[0,1)^s$, con $0 \leq i \leq N$, $0 \leq j \leq s$, s la dimensión y N el número de puntos del conjunto con la propiedad LDS. La función de pago a diferencia de la opción asiática, para una Call, es de la forma:

$$\text{máx}\{\text{máx}\{S_0, S_1, \cdots, S_s\}\} - K, 0\}$$

siendo, $S_i, 0 \leq i \leq s\}$, precios de monitorización del subyacente uniformemente distribuidos en el periodo $[0,T]$ y K el precio del ejercicio. en otras palabras, se está buscando la diferencia máxima entre el valor del subyacente en un conjunto finitos de momentos antes de la maduración y el precio fijo de ejercicio K asociado.
A continuación se el pseudocódigo del algoritmo que calcular el precio de una Lookback mediante QMC:
Siendo las variables listadas a continuación los conceptos:

- S_0 : Precio del subyacente.
- K : Precio del ejercicio.
- r : Tipo de interés.
- δ : Dividendo entregado por el subyacente.
- σ : Volatilidad del subyacente.
- T : Periodo de maduración en años.
- m : Número de monitorizaciones del subyacente.

[1] Jonh.Acklam: http://home.online.no/ pjacklam/notes/invnorm/

Algoritmo 4 QMC para una opción LookBack discreta

Entrada: S_0,K,r,δ,σ,T,m,LDS,N.

Salida: Precio $\hat{C}_n$ de una LookBack Call.

1: **para** $i = 1$ hasta N **hacer**
2: **para** $j = 1$ hasta m **hacer**
3: $u_{ij} \leftarrow (LDS)_{ij}$.
4: $w_{ij} \leftarrow \phi^{-1}(u_{ij})$.
5: $S_i(j) \leftarrow S_i(j) \exp\left((r - \delta - \frac{1}{2}\sigma^2)(\frac{T}{m}) + \sigma\sqrt{\frac{T}{m}}w_{ij}\right)$.
6: **fin para**
7: $C_i \leftarrow e^{-rT}\,\text{máx}\{\text{máx}\{S_0, S_i(1), \cdots, S_i(m)\}\} - K, 0\}$.
8: **fin para**
9: $\hat{C}_n \leftarrow \frac{1}{N}\sum_{j=1}^{N} C_i$.

- LDS : Secuencia de baja discrepancia en $[0.1)^s$.
- N : Número de puntos de LDS.

Se han elegido ciertos parámetros fijos para todas las simulaciones de valoración realizadas para las opciones de tipo Lookback, teniendo en cuenta la referencia [1], para conseguir resultados similares.

- $S_0 = 100$
- $K \in \{100, 110, 120\}$
- $r = 0.1$
- $\delta = 0$
- $\sigma = 0.2$
- $T = 5$

Se presentan simulaciones partiendo del conjuntos de condiciones iniciales listadas anteriormente, y utilizando tamaños para la red de integración de $N = 1024$ y $N = 4096$. Cada caso se está ejecutado 10 veces de forma independientes, con objeto de estimar el error estándar. Para la ejecución que utilice G.L.P, los coeficientes óptimos correspondientes se pueden ver en el cuadro 3.6. Con las condiciones descritas, para obtener sus resultados numéricos, se ha ejecutando el siguientes script *Octave*,

- **test_lookback_option.m** A.1.15

El cual tienen dependencias con las siguiente funciones *Octave*,

- **lookback_option.m**, A.2.35

- **lookback_option_periodiza.m**, A.2.36
- **f_van_der_corput_ndim.m** A.2.9
- **f_gbm_explicit.m** A.2.6
- **f_random_shift_ndim_set.m** A.2.28

Las funciones anteriores contiene el algoritmo para calcular el valor de una opción lookback, además de la posibilidad de añadir periodización, y diferentes herramientas para generar secuencias LDS.

Caso 1: Número de puntos para LDS $N = 1024$

La garantía de los resultados MC o QMC pasa por aplicar una secuencia uniformemente distribuida. Para comprobar esto, se ha ejecutado el script *Octave* siguiente,

- **grafica_muldim.m**, A.1.16

mostrando el aspecto problemático de las secuencias GLP, cuando son sometidas a una transformación inversa de normal estándar. Se puede observar cómo la transformación destruye las propiedades de una GLP en la figura 5.1, degenerando la red de integración.

	$K = 100$	$K = 110$	$K = 120$
MC crudo	47.085(0.48246)	42.0902(0.3034)	36.898(0.5096)
QMC(Van der Corput-Hibrido)	47.160(0.16665)	41.9071(0.0978)	36.937(0.1600)
QMC(Halton)	47.333(0.06982)	41.8593(0.0346)	36.838(0.0604)
QMC(GLP)	47.127(0.1163)	41.8922(0.0206)	36.847(0.1368)
QMC(GLP+ψ_2)	47.127(0.4077)	42.0250(0.2084)	36.961(0.3584)
QMC(GLP+ψ_3)	46.011(1.4681)	44.5180(1.0930)	37.980(0.8775)
QMC(GLP+ξ_1)	47.283(0.51610)	41.5630(0.6294)	36.405(0.4801)
QMC(GLP+ξ_2)	45.968(1.9823)	41.6850(1.7443)	35.262(1.2277)

Cuadro 5.1: Valoraciones y error estándar de una opción Call Lookback con número de muestras $N = 1024$ y volatilidad $\sigma = 0.2$ según método de resoluciónm, siendo ψ_i y ξ_i son funciones de periodización 3.2.5

Debido a la distorsión generada por el algoritmo que calcula la inversión de la normal estándar sobre los conjuntos GLP, las estimaciones de los errores estándar de QMC basados en GLP, son demasiado grandes, resultando inútiles como método de valoración.

Caso 2: Número de puntos para LDS $N = 4096$
Debido a que ahora con un número mayor de puntos, se cubre más área de integración, ver la figura 5.2, siendo antes un conjunto degenerado, los métodos QMC basados en GLP, superan con creces al resto, como se ve en cuadro comparativo 5.2.

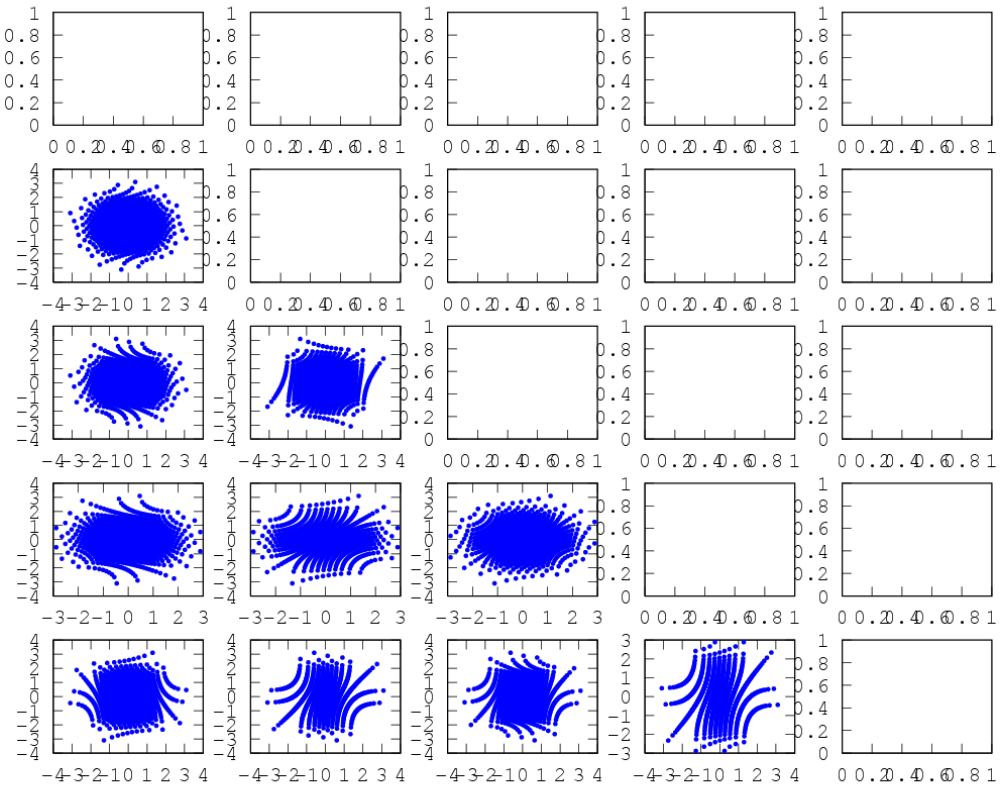

Figura 5.1: Aspecto de los pares de coordenadas transformadas por la inversa de la normal estándar Φ^{-1} de una GLP, en dimensión $s = 5$, número de puntos $N = 1024$.

5.3. Valoración de Spread Options

En esta sección se presentan simulaciones de las Spread Options introducidas en la sección 1.7.6, tanto aproximaciones a su precio como a sus valores griegas Δ 1.19 y Γ 1.19. Se consideran apenas dos subyacentes, dimensión del espacio $s = 2$, en este tipo de opción. Se tratan dos casos, con precio de ejercicio $K = 0$, en el cual se puede aplicar la fórmula de Margrabe, vista en la sección 1.7.6, que precisa pesos $w_1 = w_2 = 1$, y el menos restrictivo, con iguales pesos pero con precio de ejercicio $K \neq 0$. A partir de ahora, todas las Spread Options tratadas en esta sección tendrán las condiciones específicas nombradas.

Desde la referencia [1], se recomienda una reformulación del problema, donde se prometen mejores resultados que con una aplicación directa. Para ello, se precisan cambios de variable que convierten la integral múltiple de un recinto no acotado, en otro reciento de definición acotado, es decir, convierte $[0, \infty)^2$ en el hipercubo de $[0, 1)^2$.

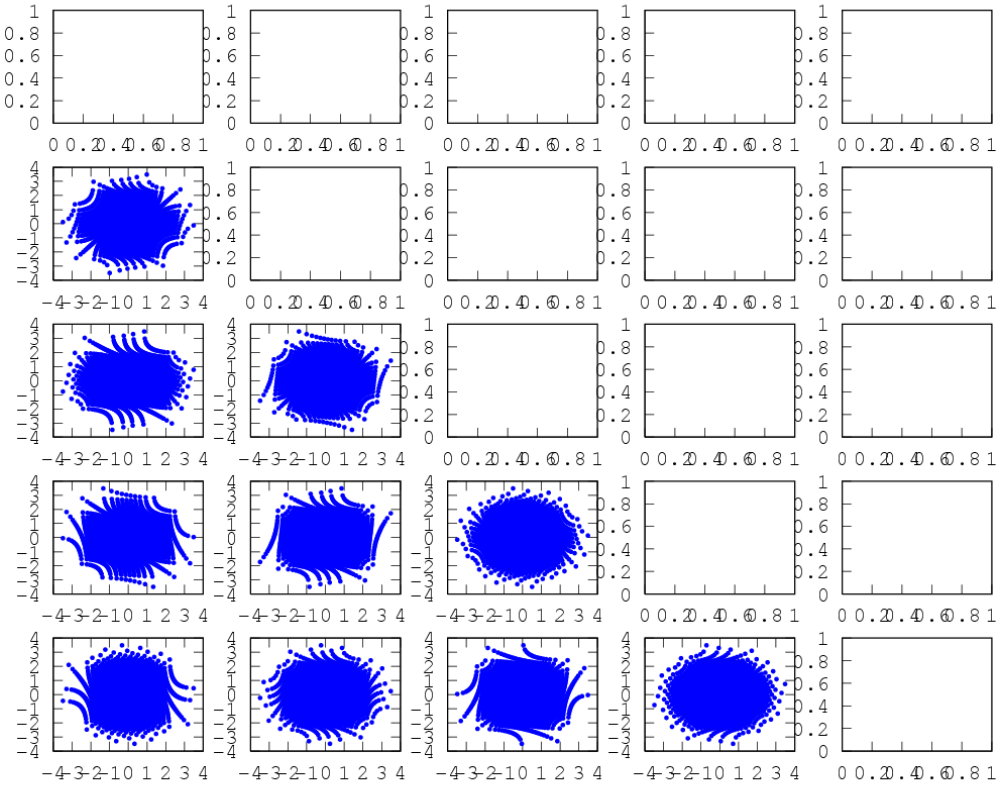

Figura 5.2: Aspecto de los pares de coordenadas transformadas por la inversa de la normal estándar Φ^{-1} de una GLP, en dimensión $s = 5$, número de puntos $N = 4096$.

Considerando la función de pago, $p = p(S_{T1}, S_{T2}) = \max[w_2 S_{2T} - w_1 S_{1T} - K, 0]$, vista en la subsección 1.7.6, y sustituyéndola en la ecuación 1.16 de la subsección 1.7.5

$$\begin{aligned} \hat{V} &= e^{-rT} E_Q[\max\{w_2 S_{2T} - w_1 S_{1T} - K, 0\}] \\ &= e^{-rT} \int_0^\infty \int_0^\infty \max[w_2 s_2 - w_1 s_1 - K, 0] f(s) ds \end{aligned}$$

La anterior ecuación se puede reducir al dominio de definición $u = (u_1, u_2) \in [0, 1)^2$,

$$\hat{V} = e^{-rT} \int_{[0,1)^2} h^*(u) du \tag{5.1}$$

	$K = 100$	$K = 110$	$K = 120$
MC crudo	47.4480(0.3216)	42.0902(0.3034)	36.876(0.1561)
QMC(Van der Corput-Hibrido)	47.3370(0.0581)	41.9071(0.0978)	36.886(0.0690)
QMC(Halton)	47.2830(0.0220)	41.8593(0.0346)	36.830(0.0375)
QMC(GLP)	47.3160(0.0162)	41.8922(0.0206)	36.851(0.0333)
QMC(GLP+ψ_2)	47.3090(0.0099)	41.8856(0.0107)	36.846(0.0098)
QMC(GLP+ψ_3)	47.3310(0.0072)	41.9000(0.0058)	36.847(0.0057)
QMC(GLP+ξ_1)	47.3140(0.0098)	41.8790(0.0131)	36.848(0.0112)
QMC(GLP+ξ_2)	47.3330(0.0064)	41.9000(0.0064)	36.853(0.0059)

Cuadro 5.2: Valoraciones y error estándar de una opción Call Lookback con número de muestras $N = 4096$ y volatilidad $\sigma = 0.2$ según método de resolución, siendo ψ_i y ξ_i funciones de periodización 3.2.5

donde h^* se deduce [2] desde las siguientes:

$$\begin{aligned}
h^*(u) &= h^*(u_1, u_2) = (1 - d_2(u))h(u) \\
h(u) &= h_2(u) - h_1(u) - K \\
h_1(u) &= w_1 e^{h_3(u)} \\
h_2(u) &= w_2 e^{h_4(u)} \\
h_3(u) &= c_{11}\Phi^{-1}(u_1) + \mu_{1T} \\
h_4(u) &= c_{21}\Phi^{-1}(u_1) + c_{22}h_5(u) + \mu_{2T} \\
h_5(u) &= \Phi^{-1}(d_2(u) + u_2(1 - d_2(u))) \\
d_2(u) &= \Phi(g(u)) \\
g(u) &= \frac{1}{c_{22}}(\log(h_1(u) + K) - \log(w_2) - \mu_{2T} - c_{21}\Phi^{-1}(u_1))
\end{aligned}$$

donde c_{ij} son las entradas de una descomposición de Cholesky C triangular inferior tal que $C \cdot C' = \Sigma_T$. De esta forma, la integral doble anterior puede aproximarse mediante un método QMC aplicado sobre $[0,1)^2$. Suponiendo $u_n = (u_{n1}, u_{n2}) \in [0,1)^2$, es el término n-ésimo de una *Lattice Rule*, el estimador insesgado QMC es:

$$\hat{V} = \frac{e^{-rT}}{N} \sum_{n=1}^{N} h^*(u_{n1}, u_{n2})$$

Adicionalmente, habrá que conseguir una transformación de periodización para poder aplicar ciertos teoremas [3] de convergencia sobre *Lattice Rules*, que permitan acotar el

[2] h^* se deduce para una Spread Options en el apéndice A de la referencia [1], tanto el precio como las griegas Δ y Γ

[3] Ver el teorema de convergencia de funciones periódicas en [1, 2.3.1]

error eficientemente, teniendo validez para familias de funciones periódica de periodo 1, en $[0,1)^s$. Es por esta razón, la necesidad de conseguir una función periódica en el integrando. Para ello, basta hacer una transformación de periodización ψ[4], a la función del integrando h^*, obteniendo la siguiente fórmula,

$$\hat{V} = \frac{e^{-rT}}{N} \sum_{n=1}^{N} h^*(\psi(u_{n1}), \psi(u_{n2}))\psi'(u_{n1})\psi'(u_{n2}) \tag{5.2}$$

Para esta última ecuación, se ha implementado un algoritmo en *Octave* que calcula el precio y las griegas de las Spread Options.

5.3.1. Raiz del error cuadrático medio relativo: RMSE

Una forma habitual de comprobar la bondad de los métodos estadísticos, es calcular las diferencias entre el valor teórico (siempre que se disponga) y el aproximado. En este trabajo, una diferencia cuadrática, será utilizada para estimar el error cometido en las aproximaciones de la valoración de las Spread Options. El error se medirá en base al número obtenido del cálculo de la raíz del error cuadrático medio relativo al valor teórico, llamado con las siglas en inglés RMSE. Se define con la siguiente expresión:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\frac{\hat{V}_i - V_i}{V_i} \right)^2} \tag{5.3}$$

siendo $\hat{V}_i$ la estimación i-ésima y V_i el verdadero valor teórico.

5.3.2. Valoración de una Spread Option con precio de ejercicio fijo igual a cero.

En el caso particular de considerar un precio de ejercicio igual a cero ($K = 0$), es posible utilizar la fórmula de Margrabe 1.7.6, permitiendo obtener un valor exacto para la opción. Es un caso especial que va a permitir evaluar la fiabilidad de los algoritmos implementados, para su posterior extensión al caso menos restrictivo con $K \neq 0$. De de esta forma, una vez realizada una primera aproximación con el caso particular, será posible hacer una extrapolación a otro conjunto de condiciones iniciales más amplio.

Aplicando el script *Octave* siguiente,

- **test_spread_option_perio_k_nulo.m** A.1.17

que depende de las funciones *Octave* siguientes:

- **f_glp.m** A.2.26

[4] Ver sección 3.2.5

- **margrabe_formula.m** A.2.29
- **spread_option_periodifica.m** A.2.37
  - **hstar.m** A.2.31, depende de:
    - **stdnormal_inv_acklam.m** A.2.32
    - **f_periodifica.m** A.2.33
    - **f_periodifica_deriv.m** A.2.34
- **f_error.m** A.2.14

Se han obtenido los resultados presentados en el cuadro 5.3. Se puede observar como una transformación por periodización del integrando mejora los resultados a igual número de muestras N, implicando una mejora en la convergencia hacia el precio teórico.

N	pol-3	pol-4	seno-2	seno-3	seno-4	No-periodiza
13	4.72100 %	4.72000 %	2.84600 %	7.38700 %	11.67000 %	29.54000 %
21	0.97190	1.63900	0.67090	1.49000	1.69400	21.11000
34	0.32510	0.20990	0.14870	0.18480	0.19380	16.20000
55	0.076640	0.02362	0.03044	0.01428	0.01815	11.17000
89	0.01943	0.00667	0.00812	0.00650	0.00884	8.17700
144	0.00442	0.00136	0.00296	0.00179	0.00243	5.47500
233	0.00161	0.00018	0.00079	0.00014	0.00010	3.87400
377	0.00030	0.00006	0.00018	0.00003	0.00002	2.53900

Cuadro 5.3: Aplicaciones de QMC+GLP con periodización al problema de las Spread Option con $K = 0$, mostrando una comparativa del %RMSE para diferentes tipos de funciones de periodización.

5.3.3. Valoración de una Spread Option con precio de ejercicio arbitrario

En el caso general, se considera un precio de ejercicio cualquiera distinto de cero. En los experimentos numéricos realizados se ha utilizado, sin pérdida de generalidad, y a modo de ejemplo, el precio de ejercicio $K = 1$, sin tener preferencia por ningún otro valor específico de este parámetro. Como en este caso, al no disponer de una fórmula analítica exacta, como en el caso $K = 0$, se hace necesario aplicar métodos numéricos, y en particular, simulaciones QMC. Para continuar con el mismo procedimiento de cálculo del error relativo al teórico, se ha escogido como referencia teórica de contraste, el valor obtenido de la aplicación de un método QMC con una GLP de tamaño muestral suficientemente grande, por ejemplo, $N = 121.393$.

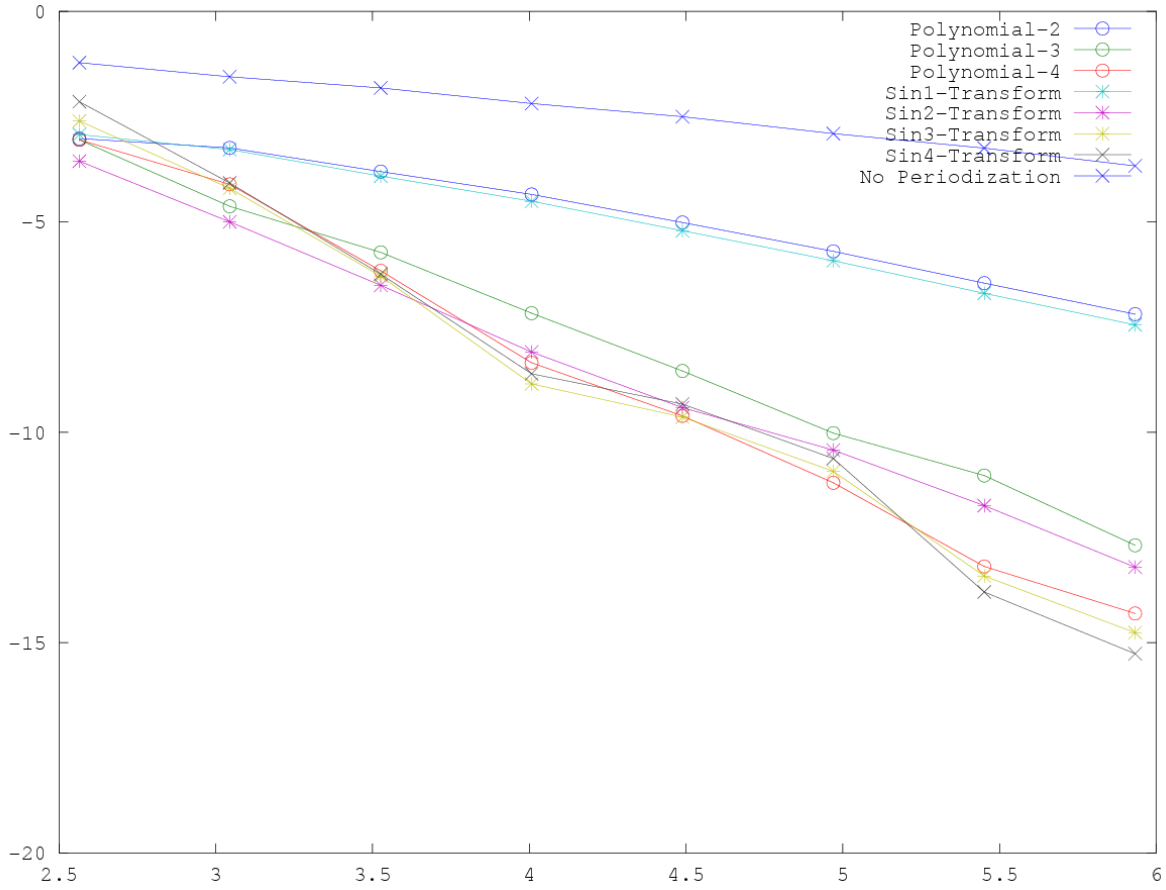

Figura 5.3: Comparativa del $\log(RMSE)$ de las aproximaciones para el valor de las Spread Options con $K = 0$, utilizando QMC+GLP con varias funciones de periodización.

La aplicación de QMC se ha aplicado con diferentes tamaños de GLP y modos de periodización de forma análoga a la sección anterior. Se ha hecho la elección del siguiente conjunto de parámetros de las Spread Options a aproximar,

- $S_{10} \in \{92, 104\}$
- $\sigma_1 \in \{0.1, 0.3\}$
- $T \in \{1/12, 1, 5\}$ en años.
- $\rho_{12} = \{-0.5, 0.5\}$

que hacen un total de 24 posibles Spread Options diferentes. El objetivo de los experimentos presentados ha sido calcular la cantidad RMSE de un vector de opciones contrastadas con sus correspondientes valores teóricos, en su aplicación de QMC+GPL con periodización. Así, ejecutando el script *Octave* siguiente:

- **test_spread_option_k_dist_zero.m** A.1.18

que depende de las funciones *Octave* siguientes:

- **f_glp.m** A.2.26
- **spread_option.m** A.2.30, depende de:
  - **hstar.m** A.2.31, depende de:
- **spread_option_periodifica.m** A.2.37 , depende de:
  - **hstar.m** A.2.31, depende de:
  - **f_periodifica.m** A.2.33
  - **f_periodifica_deriv.m** A.2.34
- **f_error.m** A.2.14

se han obtenido los resultados presentados en el cuadro 5.4. De forma análoga, al caso específico donde $K = 0$, los resultados mejoran cuando se incrementa el tamaño de la red de integración GLP utilizada, mejorando sustancialmente los resultados aplicando los métodos junto a transformaciones de funciones de periodización, polinómicas de grados 3 y 4, y trigonométricas de grado $1, 2, 3$ y 4.

N	polinom.-2	polinom.-3	polinom.-4	seno-1	seno-2	seno-3	seno-4
13	4.0923e-02	1.8964e-02	2.1681e-02	3.8334e-02	1.4110e-02	3.0045e-02	5.9776e-02
21	2.6966e-02	2.7762e-03	3.6301e-03	2.3536e-02	2.1459e-03	3.9520e-03	9.1040e-03
55	1.2627e-02	1.4106e-03	2.1855e-04	1.0594e-02	8.2784e-04	3.2380e-04	5.5077e-04
89	6.8408e-03	2.2913e-04	1.1019e-04	5.6033e-03	1.5563e-04	1.1965e-04	1.5766e-04
144	3.0939e-03	1.3438e-04	6.1275e-05	2.4662e-03	9.8536e-05	6.5176e-05	7.4618e-05
233	1.5561e-03	5.8926e-05	5.4749e-05	1.2353e-03	5.1514e-05	5.4696e-05	5.3464e-05

Cuadro 5.4: Aplicaciones QMC+GLP con periodización al problema de las Spread Option con $K = 1$, mostrando una comparativa del RMSE para diferentes tipos de funciones de periodización.

5.3.4. Aproximación de las griegas de una Spread Opción mediante el método directo

La aproximación directa de las griegas de una opción se puede calcular mediante la misma metodología de la sección 5.3. Como se ha visto anteriormente, en primer lugar, es necesario reformular el problema de integración determinando la expresión de la función h^* del integrando definida sobre el espacio $[0, 1)^2$.

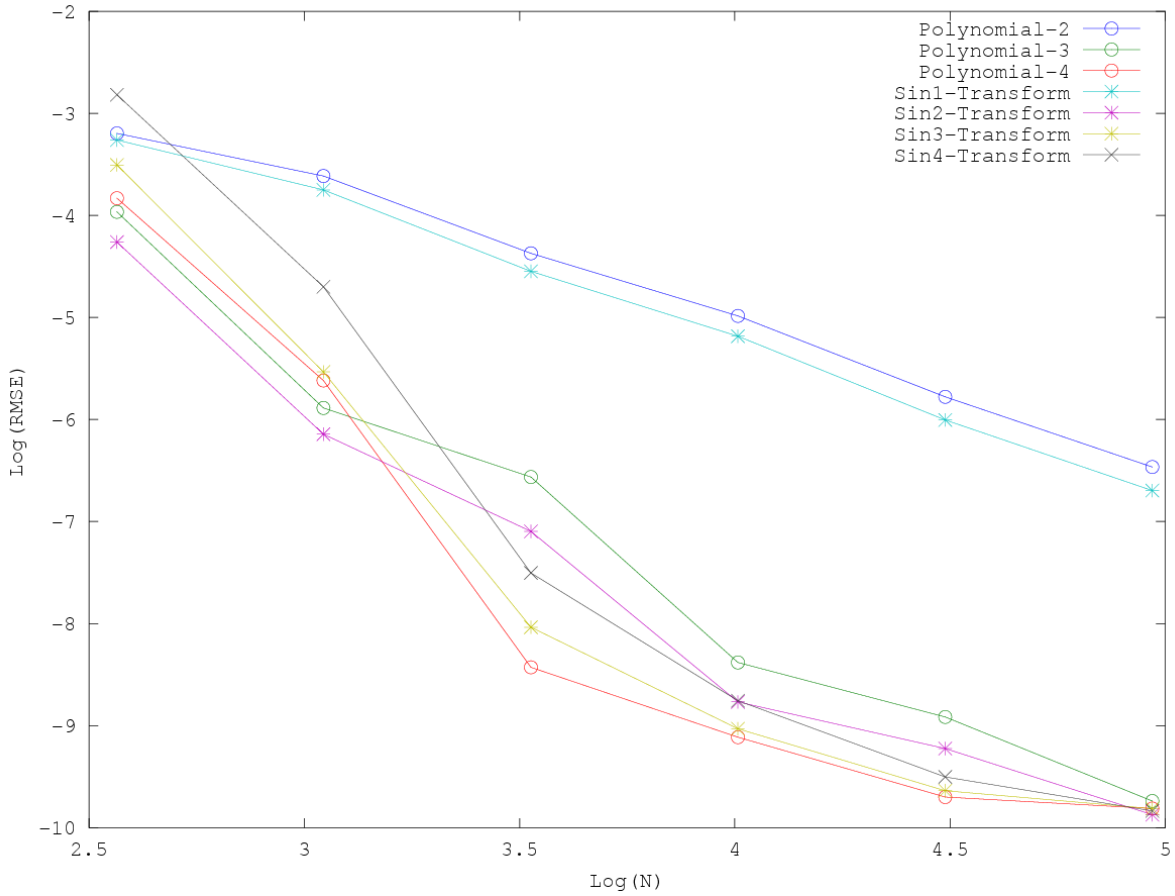

Figura 5.4: Comparativa del $\log(RMSE)$ de las aproximaciones para el valor de las Spread Options con $K \neq 0$, utilizando QMC+GLP con varias funciones de periodización.

Para el caso de la griega Δ_i, se tendrá que derivar bajo el signo integral respecto del precio del subyacente, $i = 1, 2$, la integral de la ecuación 5.1,

$$\begin{aligned}
\Delta_i &= \frac{\partial V}{\partial S_i} \\
&= e^{-rT} \frac{\partial}{\partial S_i} \left\{ \int_{[0,1)^2} h^*(u) du \right\} \\
&= e^{-rT} \frac{\partial}{\partial S_i} \left\{ \int_{[0,1)^2} (1 - d_2) h(u) du \right\} \\
&= e^{-rT} \int_{[0,1)^2} \frac{\partial}{\partial S_i} \{(1 - d_2) h(u) du\} \\
&= e^{-rT} \int_{[0,1)^2} \left[(1 - d_2) \frac{\partial h}{\partial S_i} - \frac{\partial^2 d_2}{\partial S_i^2} h(u) \right] du
\end{aligned}$$

Por lo tanto,

$$\Delta_i = e^{-rT} \int_{[0,1)^2} \left[(1 - d_2) \frac{\partial h}{\partial S_i} - \frac{\partial^2 d_2}{\partial S_i^2} h(u) \right] du \tag{5.4}$$

Análogamente, para Γ_i se tiene la siguiente expresión:

$$\Gamma_i = \frac{\partial^2 V}{\partial S_i^2} = e^{-rT} \int_{[0,1)^2} \left[(1 - d_2) \frac{\partial^2 h}{\partial S_i^2} - 2 \frac{\partial h}{\partial S_i} \frac{\partial d_2}{\partial S_i} - \frac{\partial^2 d_2}{\partial S_i^2} h(u) \right] du \tag{5.5}$$

El cálculo detallado de las derivadas parciales contenidas en los integrandos anteriores, con objeto de obtener un método directos para las griegas nombradas puede consultarse en la referencia [1, apéndice B].

Utilizando las expresiones 5.4 y 5.5 para las griegas Δ y Γ respectivamente, se han desarrollado scripts en *Octave*, con objeto de realizar experimentos numéricos. En principio se han elegido unas condiciones iniciales para las opciones extraídas desde la referencia [1], con vistas a contrastar sus resultados:

- $\sigma_1 = 0.3$
- $\sigma_2 = 0.2$
- $\delta_1 = \delta_2 = 0.05$
- $\rho = 0.5$
- $r = 0.05$
- $T = 5$ años
- $K = 4$
- $S_1 \in \{96, 100, 104\}$
- $S_2 = 100$

que describen 3 Spread Options diferentes. En los experimentos numéricos de esta subsección, se ha elegido un tamaño de red de integración GLP de $N = 233$ puntos (tamaño igualmente elegido en la referencia [1]). A continuación, se presentan las referencias al script *Octave* y todas las funciones de las que depende:

- ***test_spread_option_delta_gamma.m*** A.1.19

que depende de las siguientes funciones *Octave*,

- **f_glp.m** A.2.26
- **f_random_shift.m** A.2.27
- **spread_option_delta1_perio.m** A.2.39, que depende de:

    - **f_periodifica.m** A.2.33
    - **f_periodifica_deriv.m** A.2.34
    - **hstar_delta1.m** A.2.38, que depende de:
        - **stdnormal_inv_acklam.m** A.2.32
- **spread_option_delta2_perio.m** A.2.41, que depende de:
    - **f_periodifica.m** A.2.33
    - **f_periodifica_deriv.m** A.2.34
    - **hstar_delta2.m** A.2.40, que depende de:
        - **stdnormal_inv_acklam.m** A.2.32
- **spread_option_gamma1_perio.m** A.2.43, que depende de:
    - **f_periodifica.m** A.2.33
    - **f_periodifica_deriv.m** A.2.34
    - **hstar_gamma1.m** A.2.42, que depende de:
        - **stdnormal_inv_acklam.m** A.2.32
- **spread_option_gamma2_perio.m** A.2.45, que depende de:
    - **f_periodifica.m** A.2.33
    - **f_periodifica_deriv.m** A.2.34
    - **hstar_gamma2.m** A.2.44, que depende de:
        - **stdnormal_inv_acklam.m** A.2.32

Los modelos de periodización elegidos para las transformaciones de los integrandos han sido de tipo algebraico 3.12 y trigonométrico 3.16. A modo de contraste, también se han realizado cálculos de las griegas sin transformación de periodización.

- Polinómico-m ,$m \in \{2, 3, 4\}$
- Seno-m ,$m \in \{1, 2, 3, 4\}$
- No periodización

Debido a que la naturaleza del método QMC es determinista, se precisa de un mecanismos de aproximación del error estándar especial 3.2.4. El mecanismo precisa desplazar la red de puntos de integración, de forma aleatoriamente para simular cómo un pequeño desplazamiento, afecta al resultado, proporcionando un vector de resultados o aproximaciones, sobre el cual, es posible calcular su error estándar. De esta forma, cada resultado particular obtenido de esta sección, mostrado en el cuadro 5.3.4, ha sido recalculado 10 veces, siendo cada uno de ellos, calculado en una red desplazada de la GLP original. Además, para comparar la precisión alcanzada por cada métodos, se detallan en el cuadro 5.3.4 se muestran los errores estándar cometidos para cada caso.

$N = 233$								
Griega	poly-2	poly-3	poly-4	seno-1	seno-2	seno-3	seno-4	No perio.
$S_1 = 96$								
Δ_1	-0.29836	-0.29835	-0.29835	-0.29835	-0.29835	-0.29835	-0.29835	-0.29877
Δ_2	0.47940	0.47942	0.47942	0.47941	0.47942	0.47942	0.47942	0.47952
Γ_1	0.00519	0.00519	0.00519	0.00519	0.00519	0.00519	0.00519	0.00521
Γ_2	0.00521	0.00521	0.00521	0.00521	0.00521	0.00521	0.00521	0.00517
$S_1 = 100$								
Δ_1	-0.27820	-0.27819	-0.27820	-0.27820	-0.27820	-0.27820	-0.27820	-0.27774
Δ_2	0.45887	0.45883	0.45883	0.45887	0.45883	0.45883	0.45883	0.45971
Γ_1	0.00487	0.00487	0.00487	0.00487	0.00487	0.00487	0.00487	0.00485
Γ_2	0.00530	0.00530	0.00530	0.00530	0.00530	0.00530	0.00530	0.00531
$S_1 = 104$								
Δ_1	-0.25929	-0.25930	-0.25930	-0.25931	-0.25930	-0.25930	-0.25930	-0.25976
Δ_2	0.43875	0.43875	0.43875	0.43874	0.43875	0.43875	0.43875	0.43832
Γ_1	0.00457	0.00457	0.00457	0.00457	0.00457	0.00457	0.00457	0.00458
Γ_2	0.00536	0.00537	0.00537	0.00536	0.00537	0.00537	0.00537	0.00534

Cuadro 5.5: Valores obtenidos para los valores de las griegas Δ y Γ de una Spread Option, con 3 precios diferentes en su primer subyacente $S_1 \in \{96, 100, 104\}$, calculados mediante QMC con Lattice Rule de $N = 233$ y diferentes tipos de periodización.

5.3.5. Aproximación de griegas mediante resimulación

Como se hablaba anteriormente, la aproximación de las griegas también se puede calcular de forma indirecta mediante diferencias finitas de la valoración. Esto implica ejecutar el problema de valoración en dos o más valores del parámetro respecto al cual se calcula la griega o sensibilidad. Por ejemplo, en el caso de aplicar el método respecto de las griegas Δ y Γ, se necesitará calcular la valoración de una opción en varios precios del subyacente S_t diferentes y próximos entre sí. Ejemplos de fórmulas de diferencias finitas pueden verse en la sección 1.7.7.

En concreto, en ésta subsección se ha trabajado la implementación para la griega Δ de una Spread Option, utilizando para ello la diferencia finita 1.22. De forma análoga a las subsecciones anteriores, las condiciones elegidas para las simulaciones numéricas de resimulación han sido las siguientes:

- $w_1 = w_2 = 1$ pesos de los subyacentes de la Spread,
- $\sigma_1 = 0.3$ volatilidad del primer subyacente,
- $\sigma_2 = 0.2$ volatilidad del segundo subyacente,
- $\delta_1 = \delta_2 = 0.05$ dividendos de los subyacentes,

$N = 233$								
Griega	poly-2	poly-3	poly-4	seno-1	seno-2	seno-3	seno-4	No perio.
$S_1 = 96$								
Δ_1	1.18e-04	4.54e-06	1.93e-07	1.39e-04	2.90e-06	3.31e-07	1.61e-07	8.43e-03
Δ_2	1.49e-04	4.72e-06	5.26e-07	1.22e-04	2.78e-06	2.80e-07	2.31e-07	1.26e-02
Γ_1	3.03e-06	1.76e-07	4.02e-08	2.38e-06	1.53e-07	2.86e-08	2.53e-08	2.02e-04
Γ_2	4.05e-06	1.61e-07	5.18e-08	3.72e-06	1.49e-07	4.50e-08	3.81e-08	1.34e-04
$S_1 = 100$								
Δ_1	9.17e-05	2.85e-06	2.72e-07	7.75e-05	2.20e-06	2.59e-07	1.67e-07	3.68e-03
Δ_2	1.27e-04	6.48e-06	3.40e-07	1.28e-04	4.30e-06	2.72e-07	1.58e-07	1.18e-02
Γ_1	2.99e-06	1.36e-07	3.64e-08	3.55e-06	8.55e-08	2.03e-08	1.31e-08	1.03e-04
Γ_2	2.07e-06	1.82e-07	4.50e-08	2.29e-06	1.62e-07	2.20e-08	3.28e-08	1.97e-04
$S_1 = 104$								
Δ_1	1.05e-04	4.37e-06	2.48e-07	7.50e-05	3.55e-06	1.52e-07	1.06e-07	8.21e-03
Δ_2	1.19e-04	6.44e-06	2.70e-07	1.33e-04	2.53e-06	3.41e-07	1.23e-07	7.98e-03
Γ_1	1.49e-06	1.11e-07	2.30e-08	2.35e-06	9.94e-08	1.93e-08	1.32e-08	1.71e-04
Γ_2	5.31e-06	1.64e-07	3.83e-08	4.17e-06	9.29e-08	2.07e-08	2.47e-08	1.81e-04

Cuadro 5.6: Errores estándar para los valores de las griegas Δ y Γ de una Spread Option, con 3 precios diferentes en su primer subyacente $S_1 \in \{96, 100, 104\}$, calculados mediante QMC con Lattice Rule de $N = 233$ y diferentes tipos de periodización.

- $\rho = 0.5$ correlación entre los subyacentes
- $r = 0.05$ tasa de interés
- $T = 5$ años
- $K = 4$ precio de ejercicio,
- $S_1 = 96$ precio del primer subyacente
- $S_2 = 100$ precio del segundo subyacente

Las funciones de periodización utilizadas han sido de tipo trigonométrico $seno-2$, con un valor de desplazamiento $h = 10^{-7}$. En el cuadro **??** se presentan los resultados obtenidos ejecutando el siguiente script *Octave*,

- ***test_spread_option_delta_resimula.m*** A.1.20

que depende de las siguientes funciones *Octave*:

- **f_glp.m** A.2.26
- **spread_option_delta1_perio.m** A.2.39

- **spread_option_periodifica.m** A.2.37

obteniendo los siguientes resultados Ha calculado utilizando los dos métodos, el directo, y el resimulado, obteniendo el valor para Δ_1 en el caso de $N(glp) = 4181$ de -0.29835 para ambos métodos. De hecho el orden de error estándar en 10^{-7}. Los tiempos de ejecución para el método directo y el de resimulación, según el número de muestras del tamaño de la red g.l.p ha sido:

Muestras	Método directo	Resimulacion
55	0.089684	0.114960
89	0.090279	0.167633
144	0.146385	0.274467
233	0.231804	0.439796
377	0.376160	0.713089
610	0.700569	1.420158
987	1.077370	2.218933
1597	1.715614	3.462105
2584	2.779121	5.200577
4181	4.208344	8.010368

Cuadro 5.7: Tiempos de la resimulación de las sensibilidades

Se observa en el cuadro 5.3.5 como el tiempo de aplicar una resimulación es doble con respecto a su correspondiente métodoe directo para todos los tamaños muestrales. Por tanto, en éste caso son preferibles los métodos directos en tiempos de computación a los de resimulación, a pesar la dificultad de un pretratamiento analítico de los directos.

Capítulo 6

Conclusiones

Después de haber estudiado los métodos MC y QMC, desde un punto de vista teórico y práctico, habiendo implementado soluciones ejemplo y problemas de los mercados financieros, se han llegado a las siguientes conclusiones:

- Aplicar e implementar los estimadores MC ha resultado sencillo comparado con las soluciones teóricas que trata, permitiendo resolver integrales complicadas en sencillos algoritmos numéricos.
- El ejemplo 2.1.1 para MC resuelto en primera instancia, ha mostrado las precauciones necesarias a tomar sobre el intervalo de confianza del resultado de MC. Este intervalo, depende del comportamiento de la varianza, $\frac{\sigma_n}{\sqrt{n}}$, lo que implica su examen detallado, para aplicar con garantía MC. En general, es una buena práctica, asegurarse que la longitud del intervalo de confianza para la estimación MC disminuye con las muestras. En la práctica, esto se cumplirá en la mayoría de los casos.
- Se ha utilizado con éxito MC en problemas de alta dimensión, manteniendo una precisión razonable.
- El error estándar para los estimadores MC de opciones Europeas crece con la volatilidad.
- Los estimadores MC para las griegas de opciones, de tipo *pathwise* y *likelihood ratio*, en los casos estudiados, han arrojado buenos resultados con pequeños errores estándar, siendo mejores los de tipo *pathwise* respecto los del tipo *likelihood ratio*.
- Los estimadores MC con control de varianza han mejorado los resultados en general.
- Los estimadores MC de tipo pathwise con control de varianza, para la griega Δ de una opción asiática, han resultados mejores que la simple resimulación.

En cuanto a los métodos QMC y las redes de integración LDS,

- La teoría que sostiene a las redes LDS, es muy compleja en general, trayendo resultados de la teoría de números a aplicaciones prácticas de QMC.

- Una vez obtenidos los mecanismos de obtención de secuencias, su generación es sencilla en los casos de Van der Corput, Halton y Lattice Rules.

- Las secuencias LDS pueden tener problemas de correlación y dependencias entre pares de dimensiones, lo que implica una degeneración en la red de integración, existiendo una amplia teoría al respecto.

- Las redes Lattice Rules, han resultado extraordinarias en comparación MC crudo, pudiendo ser muy potentes en orden de convergiendo, si se usan junto a mecanismos de periodización de los integrandos, desde los resultados teóricos basados en series de Fourier.

- Los métodos de QMC han resultado ser válidos para problemas de integración en baja dimensión, y en particular para los problemas de valoración de opciones financieras, en baja dimensión, en función del problema considerado y de la red de integración LDS.

- En baja dimensión, QMC ha resultado ser superior a MC, para todos los tipos de redes de integración LDS, tanto en precisión como en tiempo de cómputo.

- Se ha visto que para aplicar QMC de forma eficaz en los problemas de valoración para Lookback y Spread options, es necesario aplicar un pretratamiento analítico para evitar la degeneración de las propiedades de baja discrepancia de la red de punto de integración. En el caso de las Lookback, se puede ver claramente como la aplicación directa utilizando *Lattice Rules* de rango, 5.2, es ineficiente con pocos puntos en la red, siendo los resultados peores incluso que un MC crudo.

- QMC puede mejorar en gran medida su convergencia, si a las funciones del problema tratado, se les aplica un pretratamiento analítico conveniente, siempre que se esté trabajando con redes de integración de tipo *Lattice Rules* de rango 1.

Desde el punto de vista de la metodología de trabajo,

- La combinación de herramientas software sobre el sistema operativo *Linux*, es decir, el lenguaje interpretado de código abierto *Octave* , el editor textos avanzado *Emacs* y el procesador de texto LaTeX, han potenciado la escritura del trabajo actual y el de una librería de código anexada al actual documento para resolver los problemas planteados y resueltos con MC y QMC.

- Gracias a las referencias cruzadas de LaTeX, todos los nombres de scripts y funciones *Octave* han sido indexados al presente documento, permitiendo reproducir completamente los cálculos de este trabajo con mayor facilidad.

- La implementación en *Octave* de scripts y funciones ha resultado ágil la codificación de cada algoritmo, por contra, sus ejecuciones han resultado lentas en casos donde se han utilizado multitud parámetros o casuísticas. Esto les hace perder respecto a los lenguajes compilados.

- Desde el punto de vista del lenguaje, ha resultado sencillo implementar los algoritmos utilizando *Octave* en comparación a los lenguajes compilados, tanto en modo interactivo como en procesamiento por lotes. El modo interactivo ha permitido analizar los resultados rápidamente, permitiendo introducir cambios en las funciones para su depuración.

- Además hay que decir, que se ha hecho una rápida exportación desde la salida de los programas, al procesador de textos LaTeXa través del editor de textos avanzado *Emacs*, que ha permitido maquetar tablas y cuadros con mucha eficacia, ayudado de las expresiones regulares disponibles *Emacs*, para hacer búsquedas y reemplazamientos.

Trabajos futuros:

- Reformular analíticamente el problema de Lookback Options como se hizo en las Spread Options, comprobando cómo las funciones de periodización mejoran el rendimiento en precisión para *Lattice Rules* con pocos puntos.
- Programación de un algoritmo eficiente para encontrar los parámetros óptimos de *Lattice Rules* de rango 1 para dimensiones entre 2 y 8, con un número de puntos entre 10 y 10000.
- Desarrollar una batería de test con funciones de prueba para evaluar una *Lattice Rule* de rango1.
- Desarrollar algoritmos MC y QMC con tecnologías de procesamiento paralelo en C++. Los métodos basados en MC o QMC, son fácilmente paralelizables, pudiéndose repartir el trabajo entre varios procesadores.
- Refactorizar el código fuente en Octave, para utilizar estructuras y objetos.

Apéndice A

Código Fuente Octave

A.1. Scripts

A.1.1. Test MC para $E(e^{\beta G})$

Nombre Script: **esperanza_familia_exponencial_gausianas_Nmu0sig1_MC.m**

```
##########################################
#
# AUTOR : Ruben Colomina Citoler
#
##########################################
#
# INPUT
# Calculo mediante MC de esperanza de una
# familia de variables aleatorias
# exponencial de normal estandar
##########################################

# Condiciones iniciales
betas=[2,4,6,8,10];
len=length(betas);

# Numero de muestras MC
n=100000;

# variable para las muestras
muestras=zeros(n,len);

# variable para las muestras al cuadrado
muestras_cuadrados=zeros(n,len);

# algoritmo
for i=1:n
  G=stdnormal_rnd(1);
  for j=1:len
     X=exp(G*betas(j));
     Y=power(X,2);
     muestras(i,j)=X;
     muestras_cuadrados(i,j)=Y;
```

```
  endfor
endfor

# calcular el estimador MC para cada caso de beta: mean(Xi)
estimadorMC=mean(muestras);

# resultados teoricos
resultadosExactos=exp(power(betas(1),2)/2);
for i=2:len
  resultadosExactos=[resultadosExactos,exp(power(betas(i),2)/2)];
endfor

# calculando los errores
mc=estimadorMC
teo=resultadosExactos
errorAbsoluto=abs(mc-teo)

# Intervalos de confianza: Cantidad de muestras para obtener una precision
# prefijada con una probabilidade del 95  [media_muestral-1.96*sigma_muestrao/sqrt(n),
# media_muestral+1.96*sigma_muestrao/sqrt(n)]. A partir de los intervalos de
# confianza, se calculara el valor de n, para que el resulatado se encuentre
# con un margen de confianza en el rango deseado prefijado.
# calculamos el estimador
# varianzas empiricas de la muestra: sigma_n= n/(n-1) * (1/n*mean(Yi) - mean(Xi)^2)
estimadorMC2=mean(muestras_cuadrados)
varianzas_muestrales=zeros(len,1);
for i=1:len
   varianzas_muestrales(i)=n/(n-1)*(estimadorMC2(i)-estimadorMC(i)^2);
endfor

# Mostrar los resultados
for i=1:len
  varianzas_muestrales(i)/sqrt(n)
endfor
```

A.1.2. Test MC cálculo volumen de S^{n-1}

Nombre Script: test_hiperesfera_volumen.m

```
################################################
#
# AUTOR : Ruben Colomina Citoler
#
# FECHA ULT. MODIF. :
#
# LICENCIA
#
# DESCRIPCION :
# Test para calcular el volumen exacto de una n-espfera#
# DEPENDENCIAS :
#
# INPUTS
# dim : dimension del espacio
# R   : radio de la esfera
# OUTPUT
# Volumen exacto Vn= pi^n/2 * R^n / gamma(n/2+1)
################################################
```

```
19
20 # Parametros iniciales
21 dim=10;
22 R=1;
23 muestramax=50000;
24
25 # Reserva de memoria
26 est_vol=zeros(muestramax,1);
27 errores_mc=zeros(muestramax,1);
28
29
30 elapse_time=0;
31 t0=clock();
32 num_puntos_dentro_esfera=0;
33
34 for i=1:muestramax
35         punto_aleat=f_hipercubo_punto_aleatorio(dim,R);
36         if(f_hiperesfera_dentro_radio(R,punto_aleat)==true);
37           num_puntos_dentro_esfera+=1;
38         endif
39         est_vol(i)=(num_puntos_dentro_esfera/i)*R*power(2,dim);;
40 endfor
41
42 # Mostrar el resultado
43 elapse_time=etime(clock(),t0)
44 vol_teo=f_hiperesfera_volumen(dim,R)
45
46 # Graficar el resultado
47 x=500:muestramax;
48 est_vol1=est_vol(500:muestramax);
49 plot(x,est_vol1,'k');
```

A.1.3. Test Movimiento Geométrico Browniano

Nombre función: *test_geometric_brownian_motion.m*

```
1  ##########################################
2  #
3  # Autor    : Ruben Colomina Citoler
4  #
5  ##########################################
6  # GEOMETRIC BROWNIAN MOTION
7  #
8  # INPUT
9  # Parametros del GBM
10 # N : numero de GBM
11 # T : periodo de generacion
12 # Z : vector aleatorio normalde n muestras
13 # rate : tipo de interes
14 # delta: dividentos
15 # sigma: volatilidad
16 # S0 : precio inicial
17 # Algoritmo de discretizacion : Aproximation de Euler
18 # S(i+1) = S(i) * ( 1 + (rate - delta) * DT + sigma * sqrt(DT) * Z)
19 # OUTPUT : vector con un GBM
20 #Generacion de N movientos brownianos geometricos
21 # condiciones iniciales para el GBM a generar
```

```
T=2;
n=1000;
rate=0.1;
delta=0.01;
sigma=0.4;
S0=10;
# generamos N=10 GBM
N=100;
var=eye(N,2);
for i=1:N
  bm=f_geometric_brownian_motion(T,n,rate,delta,sigma,S0);
  var(i,1)=bm(N/2);
  var(i,2)=bm(N);
  hold on
  plot(bm,'r');
endfor
corr(var(:,1),var(:,2))
```

A.1.4. Generador de una secuencia de Halton-2,3 en $[0,1]^2$

Nombre Script: test_van_der_corput.m

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# Generador de secuencias de Halton y Van der Corput
# INPUT
# b : base
# N : tamango de la secuencia
# OUTPUT
# Secuencia de Vander Corput en [0,1]x[0,1]
# Secuencia uniforme bidimensional en [0,1]x[0,1]
baseX=2;
baseY=3;
N=100;
vdc=zeros(N,2);
vur=zeros(N,2);
#Secuencia bidimensional
for i=1:N
  x=f_van_der_corput(baseX,i);
  y=f_van_der_corput(baseY,i);
  vdc(i,1)=x;
  vdc(i,2)=y;
  vur(i,1)=rand;
  vur(i,2)=rand;
endfor
```

A.1.5. Test QMC(Halton) para calcular el volumen de una esfera multidimensional

Nombre Script: test_hiperespera_volumen_QMC.m

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################

# Calcula el volumen de una n-esfera mediante QMC y MC
# haciendo una compartiva segun la dimension
# INPUTS
# dim : dimension del espacio
# R : radio de la esferea
# nuestramax : Numero de muestras
# OUTPUT
# Volumen n-esfera utilizando MC y QMC
# Volumen n-esfera exacta:
# Formula exacta Vn= pi^n/2 * R^n / gamma(n/2+1)
dim=10;
R=1;
muestramax=50000;
#Reservar memoria
volumen_estimado=zeros(muestramax,1);
errores_mc=zeros(muestramax,1);
### Estimacion mediante MC
elapse_time=0;
t0=clock();
num_puntos_dentro_esfera=0;
for i=1:muestramax
        punto_aleat=f_hipercubo_punto_aleatorio(dim,R);
        if(f_hiperesfera_dentro_radio(R,punto_aleat)==true);
          num_puntos_dentro_esfera+=1;
        endif
        est_vol_mc(i)=(num_puntos_dentro_esfera/i)*R*power(2,dim);;
endfor
elapse_time=etime(clock(),t0)
volumen_estimado_mc=est_vol_mc(muestramax)
#Volumen exacto teorico
volumen_teorico=f_hiperesfera_volumen(dim,R)
### Estimacion mediante QMC(Halton)
if(dim==4)
  v=[2,3,5,7];
else if(dim==7)
  v=[2,3,5,7,11,13,17];
else if(dim==10)
  v=[2,3,5,7,11,13,17,19,23,29];
else if(dim==13)
  v=[2,3,5,7,11,13,17,19,23,29,31,37,41];
endif
endif
endif
endif
u=f_halton(v,muestramax);
num_puntos_dentro_esfera=0;
for i=1:muestramax
   punto_halton=u(i,:);
        if(f_hiperesfera_dentro_radio(R,punto_halton)==true);
          num_puntos_dentro_esfera+=1;
        endif
        est_vol_halton(i)=(num_puntos_dentro_esfera/i)*R*power(2,dim);;
endfor
volulem_estimado_halton=est_vol_halton(muestramax)
plot(est_vol_mc(1000:muestramax),'r',est_vol_halton(1000:muestramax),'b');
```

A.1.6. Test MC y QMC(Halton) de la integral de una función trigonométrica multidimensional

Nombre Script: test_integra_seno_n_dim.m

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# Calcular la integral seno (X1+...Xs)
# sobre [0,1)^s
# usando metodo MC crudo y QMC(Halton)
# INPUTS
# dim : dimension del espacio
# N : numero de puntos
# l : parametro de lattice rule rango 1
# OUTPUT
# Integral de funcion sen() por MC y QMC

dim=4; #dimension
muestramax=5000;
volumen_estimado=zeros(muestramax,1);
elapse_time=0;
t0=clock();
sumatorio=zeros(muestramax,1);

for i=1:muestramax
   sumaparcial=0;
   for s=1:dim
       sumaparcial+=rand;
        endfor
        sumaparcial*=pi;
   if(i==1)
      sumatorio(i)=sin(sumaparcial);
   else
      sumatorio(i)=sumatorio(i-1)+sin(sumaparcial);
   endif
        est_vol_mc(i)=(sumatorio(i)/i);
endfor
elapse_time=etime(clock(),t0)

### Estimacion mediante QMC(Halton)
if(dim==4)
  v=[2,3,5,7];
else if(dim==7)
  v=[2,3,5,7,11,13,17];
else if(dim==10)
  v=[2,3,5,7,11,13,17,19,23,29];
else if(dim==13)
  v=[2,3,5,7,11,13,17,19,23,29,31,37,41];
endif
endif
endif
endif
punto_halton=f_halton(v,muestramax);
sumatorio=zeros(muestramax,1);
for i=1:muestramax
   sumaparcial=sum(punto_halton(i,:));
        sumaparcial*=pi;
   if(i==1)
      sumatorio(i)=sin(sumaparcial);
```

```
   else
      sumatorio(i)=sumatorio(i-1)+sin(sumaparcial);
   endif
        est_vol_halton(i)=(sumatorio(i)/i);

endfor
volumen_halton=est_vol_halton(muestramax)

plot(est_vol_mc(500:muestramax),'r',est_vol_halton(500:muestramax),'b');
```

A.1.7. Test MC y QMC+LR de la integral de una función trigonométrica multidimensional

Nombre Script: test_integra_seno_ndim_qmc_lattice.m

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################

# Calcular la integral seno (X+...Xs)
# sobre [0,1)^s
# usando metodo MC crudo y QMC(Lattice Rule Periodizacion)
# INPUTS
# dim : dimension del espacio
# N : numero de puntos
# l : parametro de lattice rule rango 1
# OUTPUT
# Integral de funcion sen() por MC y QMC
dim=4;
N=100;
l=19;
punto_lattice=zeros(N,dim);
punto_lattice=f_glp_ndim(N,dim,l);
### Estimacion mediante MC crudo
elapse_time=0;
t0=clock();
sumatorio=zeros(N,1);
for i=1:N
   #Funcion sin[pi*(x1+..+xs)]
   sumaparcial=0;
   for s=1:dim
     sumaparcial+=rand;
        endfor
        sumaparcial*=2*pi;
   if(i==1)
      sumatorio(i)=sin(sumaparcial);
   else
      sumatorio(i)=sumatorio(i-1)+sin(sumaparcial);
   endif
   #Aproximacion parcial
        est_vol_mc(i)=(sumatorio(i)/i);
endfor
elapse_time=etime(clock(),t0)
volumen_mc=est_vol_mc(N)
### Estimacion mediante QMC(Lattice)
sumatorio=zeros(N,1);
```

```
for i=1:N
   #Funcion sin(pi*(sum(xi))
   sumaparcial=sum(punto_lattice(i,:));
        sumaparcial*=2*pi;
   if(i==1)
      sumatorio(i)=sin(sumaparcial);
   else
      sumatorio(i)=sumatorio(i-1)+sin(sumaparcial);
   endif
   #Aproximacion parcial
        est_vol_lattice(i)=(sumatorio(i)/i);
endfor
#Aproximacion total
volumen_qmc_lattice=est_vol_lattice(N)
#Grafica de la convergencia de MC y QMC
plot(est_vol_mc(1:N),'r',est_vol_lattice(1:N),'b');
```

A.1.8. Test MC y QMC+LR para el volumen de una N-esfera

Nombre Script: test_integra_seno_ndim_qmc_lattice.m

```
##########################################
# Autor     : Ruben Colomina Citoler
##########################################
#
# Calcula el volumen de una n-esfera usando metodo MC y QMC(Lattice Rule)
# INPUTS
# dim : dimension del espacio
# R : radio de la esferea
# nuestramax : Numero de muestras
# Formula exacta Vn= pi^n/2 * R^n / gamma(n/2+1)

# Dimension del problema
dim=4;
# Radio
R=1;
muestramax=500
# parametro para el vector generador z de QMC
param=115;

#Reserva de memoria
volumen_estimado=zeros(muestramax,1);
errores_mc=zeros(muestramax,1);

### Estimacion mediante MC
elapse_time=0;
t0=clock();
num_puntos_dentro_esfera=0;
for i=1:muestramax
        punto_aleat=f_hipercubo_punto_aleatorio(dim,R);
        if(f_hiperesfera_dentro_radio(R,punto_aleat)==true);
          num_puntos_dentro_esfera+=1;
        endif
        est_vol_mc(i)=(num_puntos_dentro_esfera/i)*R*power(2,dim);;
endfor
elapse_time=etime(clock(),t0)
volumen_estimado_mc=est_vol_mc(muestramax)
```

```
37
38 ### Estimacion mediante QMC(Lattice Rule)
39 numpuntos=muestramax;
40 vol_qmc_lr=zeros(numpuntos,1);
41
42 t1=clock();
43 glp=f_glp_ndim(numpuntos,dim-1,param);
44
45 suma=0;
46 for k=1:numpuntos
47
48   vector_glp=glp(k,:);
49
50   normv=norm(vector_glp)*norm(vector_glp);
51   if(normv>1)
52         result=0;
53   else
54         result=sqrt(R*R-normv);
55   endif
56
57   suma+=result;
58   vol_qmc_lr(k)=power(2,dim)*suma/k;
59
60 endfor
61 volumen_qmc_lattice_rule=vol_qmc_lr(numpuntos)
62
63
64 #Volumen exacto teorico
65 volumen_teorico=f_hiperesfera_volumen(dim,R)
66
67 #Graficando el resultado
68 #subplot(1,1,1);
69 plot(est_vol_mc(10:muestramax),'r',vol_qmc_lr(10:muestramax),'b');
```

A.1.9. Test MC Estimador Valoración Opción Europea mediante Black-Sholes

Nombre Script: test_black_scholes_monte_carlo.m

```
1  ##########################################
2  #
3  # Autor : Ruben Colomina Citoler
4  #
5  ##########################################
6  # Simulacion MC del modelo de Black Scholes
7
8  # INPUTS
9  # S0     : precio inicial
10 # K      : strike price
11 # rate   : tipo interes
12 # delta  : dividendo
13 # sigma  : volatilidad
14 # T      : periodo
15
16 S0=100;
17 K=90;
```

```
rate=0.01;
delta=0.05;
sigma=0.3;
T=1;

#Variable aleatoria ST LOG-NORMAL
# ST = S0*exp{(rate-delta-sigma*sigma)/2)*T+sigma*sqrt(T)*G}

#En el modelo de Black-Scholes, el precio de la opcion es
# p = E [exp(-rate * T) * max(ST-K,0)]

valor=0;
error_estandart=0;

numsigmas=4; #numero sigmas probados
sigma=0;

itera=15; #numero de estimaciones
vector_estimador=zeros(itera,numsigmas);
for sig=1:numsigmas
  sigma=sigma+0.1;
  alfa=((rate-delta-sigma*sigma)/2)*T;
  beta=sigma*sqrt(T);
  for j=1:itera
    #Simular muestras de var. i.i.d gausianas
    vector_gaussiano;
    for i=1:n
      ST=S0*exp(alfa+beta*vgauss(i));
      valor=valor+max(ST-K,0);
    endfor
    valor=exp(-rate*T)*valor/n;
    #Almacenar la estimacion optenida en cada iteracion
    vector_estimador(j,sig)=valor;
  endfor
endfor
errores_standard=1:numsigmas;
for sig=1:numsigmas
    errores_standard(sig)=error_standard(vector_estimador(:,sig));
endfor
```

A.1.10. Test MC Estimadores Pathwise para una Opción Europea

Nombre Script: test_european_call_estimadores_pathwise.m

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################

# Test Griegas Opcion Call Europe usando estimadores de tipo pathwise
#INPUTS
# S0    : precio subyacente
# K     : strike price
# rate  : tipo interes
# delta : dividendo
# sigma : volatilidad
```

```
# T     : periodo
#OUTPUT
# Delta : d(precio)/d(S0)
# Vega  : d(precio)/d(sigma)
# Rho   : d(precio)/d(rate)
# Theta : -d(precio)/d(T)

# Condiciones iniciales
S0=90;
K=100;
rate=0.1;
delta=0.03;
sigma=0.25;
T=0.2;
# Tamagno de la muestra aleatoria
N=10000;
# Generamos N muestras aleatorias de Z distribucion normal estandar
Z=stdnormal_rnd(N,1);
# Aplicamos las muestras al estimador pathwise de delta

S0=[90,100,110];
for k=1:length(S0)
  # Calculamos ST lognormal variable aleatoria desde Z
  ST=lognormal_random_ST(S0(k),rate,delta,sigma,T,Z);
  #DELTA
  DELTA=zeros(N,1);
  for i=1:N
        DELTA(i)=european_call_estimador_pathwise_delta(S0(k),ST(i),K,rate,T);
  endfor
  Delta=mean(DELTA)
  StandardErrorMeanDelta=standard_error(DELTA)
  #VEGA
  VEGA=zeros(N,1);
  for i=1:N
        VEGA(i)=european_call_estimador_pathwise_vega(S0(k),ST(i),K,rate,delta,sigma,T);
  endfor
  Vega=mean(VEGA)
  StandardErrorMeanVega=standard_error(VEGA)
  #RHO
  RHO=zeros(N,1);
  for i=1:N
        RHO(i)=european_call_estimador_pathwise_rho(S0(k),ST(i),K,rate,delta,sigma,T);
  endfor
  Rho=mean(RHO)
  StandardErrorMeanRho=standard_error(RHO)
  #THETA
  THETA=zeros(N,1);
  for i=1:N
        THETA(i)=european_call_estimador_pathwise_theta(S0(k),ST(i),K,rate,delta,sigma,T
            );
  endfor
  Theta=mean(THETA)
  StandardErrorMeanTheta=standard_error(THETA)
endfor
```

A.1.11. Test MC Estimadores Likelihood para una Opción Europea

Nombre Script: test_european_call_estimadores_likelihood.m

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# Test estimador opcion call europea usando likelihood ratio

# INPUTS
# S0    : precio subyacente
# K     : strike price
# rate  : tipo interes
# delta : dividendo
# sigma : volatilidad
# T     : periodo

#Condiciones iniciales de la simulacion
S0 = 110;
K = 100;
rate = 0.1;
delta = 0.03;
sigma = 0.25;
T = 0.2;

# Tamanho muestral
N=10000;

# Generars N muestras aleatorias de Z distribucion normal estandar
Z=stdnormal_rnd(N,1);
#DELTAwithControl=eye(N,1);

# Calcular ST lognormal var. aleatoria desde Z
ST=lognormal_random_ST(S0,rate,delta,sigma,T,Z);

# ESTIMACION DELTA
DELTA=eye(N,1);
AUXPLOT1=eye(N,1);

for i=1:N
         DELTA(i)=european_call_estimador_likelihood_delta(S0,ST(i),K,rate,delta,sigma,T)
             ;
    AUXPLOT1(i)=mean(DELTA(1:i));
endfor

EstimacionLikelihoodDelta=mean(DELTA);
StandardErrorMeanEstimacionDelta=standard_error(DELTA);

EstimacionLikelihoodDelta
StandardErrorMeanEstimacionDelta

# ESTIMACION VEGA
VEGA=eye(N,1);
AUXPLOT2=eye(N,1);

for i=1:N
         VEGA(i)=european_call_estimador_likelihood_vega(S0,ST(i),K,rate,delta,sigma,T);
    AUXPLOT2(i)=mean(VEGA(1:i));
endfor
```

```
55
56 EstimacionLikelihoodVega=mean(VEGA);
57 StandardErrorMeanEstimacionVega=standard_error(VEGA);
58
59 EstimacionLikelihoodVega
60 StandardErrorMeanEstimacionVega
61
62 # ESTIMACION GAMMA
63 GAMMA=eye(N,1);
64 AUXPLOT1=eye(N,1);
65
66 for i=1:N
67          GAMMA(i)=european_call_estimador_likelihood_gamma(S0,ST(i),K,rate,delta,sigma,T)
                ;
68     AUXPLOT1(i)=mean(GAMMA(1:i));
69 endfor
70
71 EstimacionLikelihoodGamma=mean(GAMMA);
72 StandardErrorMeanEstimacionGamma=standard_error(GAMMA);
73
74 EstimacionLikelihoodGamma
75 StandardErrorMeanEstimacionGamma
76
77 # ESTIMACION RHO
78 RHO=eye(N,1);
79 AUXPLOT1=eye(N,1);
80
81 for i=1:N
82          RHO(i)=european_call_estimador_likelihood_rho(S0,ST(i),K,rate,delta,sigma,T);
83     AUXPLOT1(i)=mean(RHO(1:i));
84 endfor
85
86 EstimacionLikelihoodRho=mean(RHO);
87 StandardErrorMeanEstimacionRho=standard_error(RHO);
88
89 EstimacionLikelihoodRho
90 StandardErrorMeanEstimacionRho
91
92 # ESTIMACION THETA
93 THETA=eye(N,1);
94 AUXPLOT1=eye(N,1);
95
96 for i=1:N
97          THETA(i)=european_call_estimador_likelihood_theta(S0,ST(i),K,rate,delta,sigma,T)
                ;
98     AUXPLOT1(i)=mean(THETA(1:i));
99 endfor
100
101 EstimacionLikelihoodTheta=mean(THETA);
102 StandardErrorMeanEstimacionTheta=standard_error(THETA);
103
104 EstimacionLikelihoodTheta
105 StandardErrorMeanEstimacionTheta
```

A.1.12. Test MC Estimador Pathwise con control de varianza para una Opción Call Europea

Nombre Script: test_european_call_estimador_pathwise_con_control.m

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# Test del estimador pathwise con control de varianza para una opcion call Europea

#INPUTS
# S0    : precio subyacente
# K     : strike price
# rate  : tipo interes
# delta : dividendo
# sigma : volatilidad
# T     : periodo

#OUTPUT
# Delta : d(precio)/d(S0)
# Vega  : d(precio)/d(sigma)
# Rho   : d(precio)/d(rate)
# Theta : -d(precio)/d(T)

# Condiciones iniciales de la simulacion
#S0=[90,100,110];
S0=110;
K=100;
rate=0.1;
delta=0.03;
sigma=0.25;
T=0.2;

# Tamanho de la muestra aleatoria
N=10000;

#Reserva de memoria
ST=eye(N,1);
DELTA=eye(N,1);
DELTAwithControl=eye(N,1);
VEGA=eye(N,1);
VEGAwithControl=eye(N,1);
RHO=eye(N,1);
RHOwithControl=eye(N,1);
THETA=eye(N,1);
THETAwithControl=eye(N,1);

# Generacion de N muestras aleatorias para una distribucion normal estandar Z
Z=stdnormal_rnd(N,1);

# Vector ST lognormal desde Z
ST=lognormal_random_ST(S0,rate,delta,sigma,T,Z);

###DELTA###
AUXPLOT1=eye(N,1);
AUXPLOT2=eye(N,1);
for i=1:N
         DELTA(i)=european_call_estimador_pathwise_delta(S0,ST(i),K,rate,T);
    AUXPLOT1(i)=mean(DELTA(1:i));
endfor
```

```
EstDelta=mean(DELTA)
stderrMeanEstDelta=standard_error(DELTA)
# calculamos beta para el estimador
beta = - cov(DELTA,ST)/var(ST)
for i=1:N
    DELTAwithControl(i)=DELTA(i)+ beta * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
        AUXPLOT2(i)=mean(DELTAwithControl(1:i));
endfor
EstDeltaWithControl=mean(DELTAwithControl)
stderrMeanEstDeltaWithControl=standard_error(DELTAwithControl)
#EstDeltaWithError=mean(DELTAWITHERROR)


###VEGA###
for i=1:N
        VEGA(i)=european_call_estimador_pathwise_vega(S0,ST(i),K,rate,delta,sigma,T);
endfor
beta = - cov(VEGA,ST)/var(ST)
EstVega=mean(VEGA)
stderrMeanEstVega=standard_error(VEGA)
for i=1:N
    VEGAwithControl(i)=VEGA(i)+ beta * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
endfor
EstVegaWithControl=mean(VEGAwithControl)
stderrMeanEstVegaWithControl=standard_error(VEGAwithControl)

###RHO###
for i=1:N
        RHO(i)=european_call_estimador_pathwise_rho(S0,ST(i),K,rate,delta,sigma,T);
endfor
beta = - cov(RHO,ST)/var(ST)
EstRho=mean(RHO)
stderrMeanEstRho=standard_error(RHO)
for i=1:N
    RHOwithControl(i)=RHO(i)+ beta * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
endfor
EstRhoWithControl=mean(RHOwithControl)
stderrMeanEstRhoWithControl=standard_error(RHOwithControl)

###THETA##
for i=1:N
        THETA(i)=european_call_estimador_pathwise_theta(S0,ST(i),K,rate,delta,sigma,T);
endfor
beta = - cov(THETA,ST)/var(ST)
EstTheta=mean(THETA)
stderrMeanEstTheta=standard_error(THETA)
for i=1:N
    THETAwithControl(i)=THETA(i)+ beta * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
endfor
EstThetaWithControl=mean(THETAwithControl)
stderrMeanEstThetaWithControl=standard_error(THETAwithControl)
```

A.1.13. Test MC Estimador Pathwise con control de varianza para una Opción Call Europea

Nombre Script: test_european_call_estimador_likelihood_withcontrol_delta.m

```
##########################################
# Autor     : Ruben Colomina Citoler
##########################################
# Test del estimador de una opcion call europea usando likelihood con control de varianza
#INPUTS
# S0     : precio subyacente
# K      : strike price
# rate   : tipo interes
# delta  : dividendo
# sigma  : volatilidad
# T      : periodo

#OUTPUT
# Delta  : d(precio)/d(S0)
# Vega   : d(precio)/d(sigma)
# Rho    : d(precio)/d(rate)
# Theta  : -d(precio)/d(T)

# Condiciones iniciales de la simulacion
#S0=[90,100,110];

S0=90;
K=100;
rate=0.1;
delta=0.03;
sigma=0.25;
T=0.2;
# Tamanhoo de la muestra aleatoria
N=10000;

# Generamos N muestras aleatorias de Z distribucion normal estandar
Z=stdnormal_rnd(N,1);
DELTA=eye(N,1);
DELTAwithControl=eye(N,1);
VEGA=eye(N,1);
VEGAwithControl=eye(N,1);
RHO=eye(N,1);
RHOwithControl=eye(N,1);
THETA=eye(N,1);
THETAwithControl=eye(N,1);
ST=eye(N,1);

# Calculamos ST lognormal variable aleatoria desde Z
ST=lognormal_random_ST(S0,rate,delta,sigma,T,Z);

###DELTA###
AUXPLOT1=eye(N,1);
AUXPLOT2=eye(N,1);
for i=1:N
         DELTA(i)=european_call_estimador_likelihood_delta(S0,ST(i),K,rate,delta,sigma,T)
             ;
    AUXPLOT1(i)=mean(DELTA(1:i));
endfor
EstDelta=mean(DELTA)
stderrMeanEstDelta=standard_error(DELTA)
# calculamos beta para el estimador
beta = - cov(DELTA,ST)/var(ST)
for i=1:N
    DELTAwithControl(i)=DELTA(i)+ beta * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
         AUXPLOT2(i)=mean(DELTAwithControl(1:i));
```

```
endfor
EstDeltaWithControl=mean(DELTAwithControl)
stderrMeanEstDeltaWithControl=standard_error(DELTAwithControl)

###VEGA###
for i=1:N
        VEGA(i)=european_call_estimador_likelihood_vega(S0,ST(i),K,rate,delta,sigma,T);
endfor
# calculamos beta para el estimador
beta = - cov(VEGA,ST)/var(ST)
EstVega=mean(VEGA)
stderrMeanEstVega=standard_error(VEGA)
for i=1:N
    VEGAwithControl(i)=VEGA(i)+ beta * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
endfor
EstVegaWithControl=mean(VEGAwithControl)
stderrMeanEstVegaWithControl=standard_error(VEGAwithControl)

###GAMMA###
for i=1:N
        GAMMA(i)=european_call_estimador_likelihood_gamma(S0,ST(i),K,rate,delta,sigma,T)
            ;
endfor
# calculamos beta para el estimador
beta = - cov(GAMMA,ST)/var(ST)
EstGamma=mean(GAMMA)
stderrMeanEstGamma=standard_error(GAMMA)
for i=1:N
    GAMMAwithControl(i)=GAMMA(i)+ beta * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
endfor
EstGammaWithControl=mean(GAMMAwithControl)
stderrMeanEstGammaWithControl=standard_error(GAMMAwithControl)


###RHO###
for i=1:N
        RHO(i)=european_call_estimador_likelihood_rho(S0,ST(i),K,rate,delta,sigma,T);
endfor
# calculamos beta para el estimador
beta = - cov(RHO,ST)/var(ST)
EstRho=mean(RHO)
stderrMeanEstRho=standard_error(RHO)
for i=1:N
    RHOwithControl(i)=RHO(i)+ beta * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
endfor
EstRhoWithControl=mean(RHOwithControl)
stderrMeanEstRhoWithControl=standard_error(RHOwithControl)

###THETA###
for i=1:N
        THETA(i)=european_call_estimador_likelihood_theta(S0,ST(i),K,rate,delta,sigma,T)
            ;
endfor
# calculamos beta para el estimador
beta = - cov(THETA,ST)/var(ST)
EstTheta=mean(THETA)
stderrMeanEstTheta=standard_error(THETA)
for i=1:N
    THETAwithControl(i)=THETA(i)+ beta * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
endfor
EstThetaWithControl=mean(THETAwithControl)
```

```
119 stderrMeanEstThetaWithControl=standard_error(THETAwithControl)
```

A.1.14. Test MC estimador de una Opción Call Asiática

Nombre Script: test_asian_call_estimadors.m

```
1  ##########################################
2  # Autor    : Ruben Colomina Citoler
3  ##########################################
4  # Test estimadores para una Opcion Call Asiatica
5  # INPUTS
6  # S0     : precio subyacente
7  # K      : strike price
8  # rate   : tipo interes
9  # delta  : dividendo
10 # sigma  : volatilidad
11 # T      : periodo en agnos
12 # m      : ultimos dias a precio de cierre
13 # n      : numero paso discretizacion GBM
14 # h      : paso de la resimulacion
15 # a      : agno en dias
16 # N      : numero de GBM generados
17 #OUTPUT
18 # Precio opcion call asiatica
19 # Precio opcion call asiatica con Resimulacion
20 # Delta Pathwise
21 # Delta Pathwise con Control
22 # Delta Resimulacon
23 # Delta Resimulacion con Contro
24 # Condiciones iniciales de la simulacion
25 #S0=[90,100,110];
26
27 S0=110
28 K=100;
29 rate=0.1;
30 delta=0.03;
31 sigma=0.25;
32 T=0.2;
33 m=30;
34 n=200;
35 h=0.0001;
36 a=365.25;
37 N=10000;
38 # Reserva de memoria
39 ST=zeros(N,1);
40 Z=zeros(N,1);
41 ASIAN=zeros(N,1);
42 ASIANwithControl=zeros(N,1);
43 DELTA=zeros(N,1);
44 DELTAwithControl=zeros(N,1);
45 ST=zeros(1,m);
46 STresimula=zeros(1,m);
47 ASIAN=zeros(N,1);
48 ASIANpertur=zeros(N,1);
49 DeltaResimula=zeros(N,1);
50 DeltaResimulaControl=zeros(N,1);
51 # Generacion de N de caminos GBM: N filas y m columnas
```

```
#matriz N GBMs n iteraciones
MatrizGBM  = zeros(N,n);
#matriz N GBMs n iteraciones
MatrizGBMresimula = zeros(N,n);
#matriz N GBMs m ultimos dias precios a cierre
MatrizGBM2 = zeros(N,m);
#matriz N GBMs m ultimos dias precios a cierre
MatrizGBM2resimula = zeros(N,m);
#######GENERACION DE N GBM######
t0=clock();
for i=1:N
   Z=stdnormal_rnd(n,1);
   MatrizGBM(i,:)=
     f_geometric_brownian_motion(T,Z,rate,delta,sigma,S0);
   MatrizGBMresimula(i,:)=
    f_geometric_brownian_motion(T,Z,rate,delta,sigma,S0+h);
endfor
elapse_time_t0=etime(clock(),t0)
#####EXTRAER LOS ULTIMOS M DIAS DEL GBM####
t1=clock();
for i=1:N
 # Para cada GBM nos quedamos con los ultimos m dias
  for j=1:m
   #convertir indice de GBM a dias
    d=floor(n*(T*a-j+1)/(T*a));
    MatrizGBM2(i,m-j+1)=MatrizGBM(i,d);
    MatrizGBM2resimula(i,m-j+1)=MatrizGBMresimula(i,d);
  endfor
endfor
elapse_time_t1=etime(clock(),t1)
##########################################################
###PRECIO OPCION CALL ASIATICA CON MEDIA ARITMETICA####
t0=clock();
for i=1:N
  GBM=MatrizGBM2(i,:);
  ASIAN(i)=
    asian_call_estimator_aritmetic_average_price(S0,GBM,K,rate,T);
endfor
EstASIANPriceCallAsian=mean(ASIAN)
stderrPriceCallAsian=standard_error(ASIAN)
printf("-----\n");
#####################################################################
###PRECIO OPCION CALL ASIATICA CON MEDIA ARITMETICA CON CONTROL####
ST=MatrizGBM2(:,30);
beta = - cov(ASIAN,ST)/var(ST);
for i=1:N
    ASIANwithControl(i) =
      ASIAN(i) + beta * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
         #AUXPLOT2(i)=mean(DELTAwithControl(1:i));
endfor
EstASIANPriceWithControl=mean(ASIANwithControl)
stderrEstASIANPriceWithControl=standard_error(ASIANwithControl)
printf("-----\n");
##############################
###DELTA PATHWISE MC CRUDO####
for i=1:N
    GBM=MatrizGBM2(i,:);
         DELTA(i)=asian_call_estimator_pathwise_delta(S0,GBM,K,rate,T);
endfor
EstDeltaPW = mean(DELTA)
stderrMeanEstDeltaPW = standard_error(DELTA)
```

```
printf("-----\n");

############################
###DELTA PW WITH CONTROL###
# calculamos beta para el estimador
beta = - cov(DELTA,ST)/var(ST);
for i=1:N
    DELTAwithControl(i) =
     DELTA(i) + beta * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
endfor
EstDeltaPWwithControl=mean(DELTAwithControl)
stderrMeanEstDeltaPWwithControl=standard_error(DELTAwithControl)
printf("-----\n");
############################
###DELTA CON RESIMULACION###
for i=1:N
    GBM=MatrizGBM2(i,:);
    GBMpertur=MatrizGBM2resimula(i,:);
    ASIAN(i)=
        asian_call_estimator_aritmetic_average_price(S0,GBM,K,rate,T);
    ASIANpertur(i)=
        asian_call_estimator_aritmetic_average_price(S0,GBMpertur,K,rate,T);
    DeltaResimula(i)=(ASIANpertur(i)-ASIAN(i))/h;
endfor

EstDeltaWithRersimula=mean(DeltaResimula)
stderrDeltaWithResimula=standard_error(DeltaResimula)
printf("-----\n");
########################################
###DELTA CON RESIMULACION Y CONTROL###
STresimula=MatrizGBM2resimula(:,30);
beta = - cov(DeltaResimula,STresimula)/var(STresimula);
for i=1:N
  DeltaResimulaControl(i)=DeltaResimula(i) + beta
      * ( ST(i) - exp ( ( rate - delta ) * T ) * S0 );
endfor
EstDeltaWithRersimulaControl=mean(DeltaResimulaControl)
stderrDeltaWithResimulaControl=standard_error(DeltaResimulaControl)
printf("-----\n");

elapse_time_t1=etime(clock(),t0)
```

A.1.15. Test QMC Lookback option

Nombre Script: test_lookback_option.m

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
#
# TEST LOOKBACK OPTIONS
# INPUTS
# N : numero de puntos de la red de integracion
# param : parametro de Lattice Rule rango 1
# v_halton : vector de primos para suc. halton
# Precios de una Lookback Call Option basado en
# Monte Carlo, GLP y GLP+Periodizacion
```

```
12
13 #Parametros de lso metodos
14 N=1024;
15 param=363;
16 #N=512;#1024;
17 #param=151;#363;
18
19 v_halton=[2,3,5,7,11];
20 pruebas=10;
21 dim=5;
22
23 param_perio=3;
24
25 #Parametros de la opcion
26 S0=100;
27 K=120
28 r=0.1;
29 delta=0;
30 sigma=0.2;
31 T=5;
32
33 # Test
34 mc=0;
35 qmchib=0;
36 qmchalton=0;
37 qmcglp=1;
38 qmcglpper=1;
39
40 #MONTE CARLO CRUDO
41 if(mc==true)
42 vec_val_lb=zeros(pruebas,1);
43 for i=1:pruebas
44    set_puntos=rand(N,dim);
45    vec_val_lb(i)=lookback_option(S0,K,r,delta,sigma,T,set_puntos);
46 endfor
47 valor_lookback_mc=mean(vec_val_lb)
48 desviacion_estandard=standard_error(vec_val_lb)
49 printf("------------------\n");
50 endif
51
52 #QMC - HIBRIDO USANDO VAN DER CORPUT N DIM
53 if(qmchib==true)
54 set_puntos=f_van_der_corput_ndim(N,dim);
55 for i=1:pruebas
56   set_puntos_desplazados=f_random_shift_ndim_set(set_puntos);
57   valor_lookback_qmc_hibrido_vandercorput(i)=lookback_option(S0,K,r,delta,sigma,T,
       set_puntos_desplazados);
58 endfor
59 val_lookback_qmc_hibrido_vandercorput=mean(valor_lookback_qmc_hibrido_vandercorput)
60 est_error_val_lookback_qmc_hibrido_vandercorput=standard_error(
     valor_lookback_qmc_hibrido_vandercorput)
61 printf("------------------\n");
62 endif
63
64 #QMC - HALTON
65 if(qmchalton==true)
66 set_puntos=f_halton(v_halton,N);
67 for i=1:pruebas
68   set_puntos_desplazados=f_random_shift_ndim_set(set_puntos);
69   valor_lookback_qmc_halton(i)=lookback_option(S0,K,r,delta,sigma,T,
       set_puntos_desplazados);
```

```
endfor
val_lookback_qmc_halton=mean(valor_lookback_qmc_halton)
est_error_val_lookback_qmc_halton=standard_error(valor_lookback_qmc_halton)
printf("------------------\n");
endif

#QMC - GOOD LATTICE POINTS
if(qmcglp==true)
set_puntos=f_glp_ndim(N,dim,param);
for i=1:pruebas
  set_puntos_desplazados=f_random_shift_ndim_set(set_puntos);
  valor_lookback_qmc_lr(i)=lookback_option(S0,K,r,delta,sigma,T,set_puntos_desplazados);
endfor
val_lookback_qmc_lr=mean(valor_lookback_qmc_lr)
est_error_val_lookback_qmc_lr=standard_error(valor_lookback_qmc_lr)
printf("------------------\n");
endif

#QMC - GOOD LATTICE POINTS (Periodizacion)
if(qmcglpper==true)
vparam=[1,2,4,5];
for param_perio=1:length(vparam)

  param_perio=1

  set_puntos=f_glp_ndim(N,dim,param);
  for i=1:pruebas
         set_puntos_desplazados=f_random_shift_ndim_set(set_puntos);
         valor_lookback_qmc_lr_perio(i)=lookback_option_periodiza(S0,K,r,delta,sigma,T,
             set_puntos_desplazados,param_perio);
  endfor
  val_lookback_qmc_lr_perio=mean(valor_lookback_qmc_lr_perio)
  est_error_val_lookback_qmc_lr_perio=standard_error(valor_lookback_qmc_lr_perio)
  endif
endfor

#QMC - GOOD LATTICE POINTS (Periodizacion)
param_perio=2
if(qmcglpper==true)
set_puntos=f_glp_ndim(N,dim,param);
for i=1:pruebas
  set_puntos_desplazados=f_random_shift_ndim_set(set_puntos);
  valor_lookback_qmc_lr_perio(i)=lookback_option_periodiza(S0,K,r,delta,sigma,T,
      set_puntos_desplazados,param_perio);
endfor
val_lookback_qmc_lr_perio=mean(valor_lookback_qmc_lr_perio)
est_error_val_lookback_qmc_lr_perio=standard_error(valor_lookback_qmc_lr_perio)
endif

#QMC - GOOD LATTICE POINTS (Periodizacion)
if(qmcglpper==true)
param_perio=4
set_puntos=f_glp_ndim(N,dim,param);
for i=1:pruebas
  set_puntos_desplazados=f_random_shift_ndim_set(set_puntos);
  valor_lookback_qmc_lr_perio(i)=lookback_option_periodiza(S0,K,r,delta,sigma,T,
      set_puntos_desplazados,param_perio);
endfor
val_lookback_qmc_lr_perio=mean(valor_lookback_qmc_lr_perio)
est_error_val_lookback_qmc_lr_perio=standard_error(valor_lookback_qmc_lr_perio)
endif
```

```
128
129 #QMC - GOOD LATTICE POINTS (Periodizacion)
130 if(qmcglpper==true)
131 param_perio=5
132 set_puntos=f_glp_ndim(N,dim,param);
133 for i=1:pruebas
134   set_puntos_desplazados=f_random_shift_ndim_set(set_puntos);
135   valor_lookback_qmc_lr_perio(i)=lookback_option_periodiza(S0,K,r,delta,sigma,T,
        set_puntos_desplazados,param_perio);
136 endfor
137 val_lookback_qmc_lr_perio=mean(valor_lookback_qmc_lr_perio)
138 est_error_val_lookback_qmc_lr_perio=standard_error(valor_lookback_qmc_lr_perio)
139 endif
140
141 #Resultados
142 #mSalida=zeros(5,2);
143
144 #mSalida(1,1)=valor_lookback_mc;
145 #mSalida(1,2)=desviacion_estandard;
146
147 #mSalida(2,1)=val_lookback_qmc_hibrido_vandercorput;
148 #mSalida(2,2)=est_error_val_lookback_qmc_hibrido_vandercorput;
149
150 #mSalida(3,1)=val_lookback_qmc_halton;
151 #mSalida(3,2)=est_error_val_lookback_qmc_halton;
152
153 #mSalida(4,1)=val_lookback_qmc_lr;
154 #mSalida(4,2)=est_error_val_lookback_qmc_lr;
155
156 #mSalida(5,1)=val_lookback_qmc_lr_perio;
157 #mSalida(5,2)=est_error_val_lookback_qmc_lr_perio;
```

A.1.16. Script graficador multidimensional

Nombre Script: grafica_muldim.m

```
1  ##########################################
2  # Autor    : Ruben Colomina Citoler
3  ##########################################
4
5  #Dimension del problema
6  dim=5;
7
8  # parametros GLP RANGO1
9  paramglpr1=[[512,151];[1024,363];[1536,297];[4096,661]];
10
11 numparam=2
12 N=paramglpr1(numparam,1);
13 param=paramglpr1(numparam,2);
14
15 glp=f_glp_ndim(N,dim,param);
16
17 #Periodiza los datos
18 parametro_perio=8;
19
20 datos_perio=zeros(N,dim);
21
```

```
for i=1:N
   datos_perio(i,:)=f_periodifica(glp(i,:),parametro_perio);
endfor

glp_modif=datos_perio;

#Barajeo de los datos
#for j=1:dim
#    permuta=f_permuta_aleatoria(N);
#    for i=1:N
#       array(i,j)=glp(permuta(i),j);
#    endfor
#endfor
#glp=array;

# LDS HIBRIDO USANDO VAN DER CORPUT
#glp=f_van_der_corput_ndim(N,dim);

# HALTON
#glp=f_halton([2,3,5,7,11],N);

#INVERSION POR OCTAVE
glp_modif=stdnormal_inv(glp);

#INVERSION DE ACKLAM
#glp_modif=eye(N,5);
#for i=1:N-1
#       for j=1:dim
#       val=stdnormal_inv_acklam(glp(i,j));
#                glp_modif(i,j)=val;
#   endfor
#endfor

# MULTIPLOT
subplot(dim,dim,1);
index=1;
for i=1:dim
          for j=1:dim
                   subplot(dim,dim,index);
                   index=index+1;
        if(i>j)
                  plot(glp_modif(:,i),glp_modif(:,j),'.');
        endif
          endfor
endfor
```

A.1.17. Test QMC del precio de Spread Option con precio de ejercicio nulo

Nombre Script: test_spread_option_perio_k_nulo.m

```
##############################################
# Autor     : Ruben Colomina Citoler
#
# CALCULA APROXIMACIONES DEL VALOR DE UNA SPREAD OPTIONS
# PARA CASO K IDENTICAMENTE CERO
```

```
#
########################################
# INPUTS
# N : numero grande QMC referencia
# conjunto de parametros, S10, sigma1, T, ro
# S10 : precio inicial primer activo
# sigma1 : volatilidad primer activo
# T : perido de vida
# ro : coeficiente de correlacion entre S1 y S2
# rate : tipo de interes
# sigma2 : volatilidad segundo activo
# S10 : precio inicial primer activo
# S20 : precio inicial segundo activo
# delta1 : dividendos para el primer subyacente
# numopciones : numero de opciones a simular

# OUTPUTS
# Valoraciones de Spread Option para diferentes valores
# de tam. muestral y forma de periodizacion

#Primera pata
S10=100;
delta1=0.05;
sigma1=0.3;

# entorno: Se fijan a precio de strike y pesos
# precio del ejercicio no modificable
K=0;
# pesos de la spread option no modificable
w1=w2=1;

#Segunda pata del Spread: generadas aleatoriamente
numopciones=50;
S20=1:numopciones;
T=1:numopciones;
sigma2=1:numopciones;
delta2=1:numopciones;
ro12=1:numopciones;
rate=1:numopciones;

#aleatorizar la segunda pata
for j=1:numopciones
          vS20(j)=unifrnd(50,130);
     vT(j)=unifrnd(0.5,1);
     vsigma2(j)=unifrnd(0.1,0.5);
     vdelta2(j)=unifrnd(0.01,0.1);
     vro12(j)=unifrnd(-0.8,0.8);
     vrate(j)=unifrnd(0.01,0.15);
endfor


#Numero de pruebas N=fibo(m)
iter=8;   #numero de pruebas para diferentes tamanhos de glp
m=6;      #comenzamos con fibo(6) y terminamos con fibo(6+iter)
models=8; #modelos numericos

error=zeros(iter,models);
vector_N=zeros(1,iter);
for k=1:iter
  m=m+1;
  glp=f_glp(m);
```

```
67  #calculamos un paquete de opciones
68  errorsum=zeros(1,models);
69  mc=zeros(1,models);
70  for numero=1:numopciones
71
72     #colocamos los parametros a la segunda pata del spread
73          S20=vS20(numero);
74     T=vT(numero);
75          sigma2=vsigma2(numero);
76          delta2=vdelta2(numero);
77     ro12=vro12(numero);
78     rate=vrate(numero);
79
80     #Calculo de valor con modelo teorico
81     mf=margrabe_formula(S10,S20,delta1,delta2,sigma1,sigma2,ro12,T);
82
83     #calculo con metodo numerico: Modelo poly-2
84          num_param_perio=1;
85     mc1=spread_option_periodifica(glp,w1,w2,rate,K,S10,S20,delta1,delta2,sigma1,sigma2,
           ro12,T,num_param_perio);
86     errorsum(num_param_perio)=errorsum(num_param_perio)+f_error(mc1,mf);
87
88     #calculo con metodo numerico: Modelo poly-3
89          num_param_perio=2;
90     mc2=spread_option_periodifica(glp,w1,w2,rate,K,S10,S20,delta1,delta2,sigma1,sigma2,
           ro12,T,num_param_perio);
91     errorsum(num_param_perio)=errorsum(num_param_perio)+f_error(mc2,mf);
92
93     #calculo con metodo numerico: Modelo poly-4
94          num_param_perio=3;
95     mc3=spread_option_periodifica(glp,w1,w2,rate,K,S10,S20,delta1,delta2,sigma1,sigma2,
           ro12,T,num_param_perio);
96     errorsum(num_param_perio)=errorsum(num_param_perio)+f_error(mc3,mf);
97
98     #calculo con metodo numerico: Modelo sin 1-transform
99          num_param_perio=4;
100    mc4=spread_option_periodifica(glp,w1,w2,rate,K,S10,S20,delta1,delta2,sigma1,sigma2,
           ro12,T,num_param_perio);
101    errorsum(num_param_perio)=errorsum(num_param_perio)+f_error(mc4,mf);
102
103    #calculo con metodo numerico: Model sin 2-transform
104         num_param_perio=5;
105    mc5=spread_option_periodifica(glp,w1,w2,rate,K,S10,S20,delta1,delta2,sigma1,sigma2,
           ro12,T,num_param_perio);
106    errorsum(num_param_perio)=errorsum(num_param_perio)+f_error(mc5,mf);
107
108    #calculo con metodo numerico: Modelo sin 3-transform
109         num_param_perio=6;
110    mc6=spread_option_periodifica(glp,w1,w2,rate,K,S10,S20,delta1,delta2,sigma1,sigma2,
           ro12,T,num_param_perio);
111    errorsum(num_param_perio)=errorsum(num_param_perio)+f_error(mc6,mf);
112
113    #calculo con metodo numerico: Modelo sin 4-transform
114         num_param_perio=7;
115    mc7=spread_option_periodifica(glp,w1,w2,rate,K,S10,S20,delta1,delta2,sigma1,sigma2,
           ro12,T,num_param_perio);
116    errorsum(num_param_perio)=errorsum(num_param_perio)+f_error(mc7,mf);
117
118    #calculo con metodo numerico: No periodificado
119         num_param_perio=8;
120    mc8=spread_option_periodifica(glp,w1,w2,rate,K,S10,S20,delta1,delta2,sigma1,sigma2,
```

```
         ro12,T,num_param_perio);
121      errorsum(num_param_perio)=errorsum(num_param_perio)+f_error(mc8,mf);
122
123    endfor
124
125    error(k,1)=sqrt(errorsum(1)/numopciones)*100; #*100 para expresar en tanto por ciento
126         error(k,2)=sqrt(errorsum(2)/numopciones)*100;
127         error(k,3)=sqrt(errorsum(3)/numopciones)*100;
128         error(k,4)=sqrt(errorsum(4)/numopciones)*100;
129         error(k,5)=sqrt(errorsum(5)/numopciones)*100;
130         error(k,6)=sqrt(errorsum(6)/numopciones)*100;
131         error(k,7)=sqrt(errorsum(7)/numopciones)*100;
132    error(k,8)=sqrt(errorsum(8)/numopciones)*100;
133
134    vector_N(k)=fibo(m);
135 endfor
136
137 plot(log(vector_N),log(error(:,1)),"-o;Polynomial-2;",
138      log(vector_N),log(error(:,2)),"-o;Polynomial-3;",
139      log(vector_N),log(error(:,3)),"-o;Polynomial-4;",
140      log(vector_N),log(error(:,4)),"-*;Sin1-Transform;",
141      log(vector_N),log(error(:,5)),"-*;Sin2-Transform;",
142      log(vector_N),log(error(:,6)),"-*;Sin3-Transform;",
143      log(vector_N),log(error(:,7)),"-x;Sin4-Transform;",
144      log(vector_N),log(error(:,8)),"-x;No Periodization;");
145
146 plot(log(vector_N),log(error(:,2)),"-o;Polynomial-3;",
147      log(vector_N),log(error(:,3)),"-o;Polynomial-4;",
148      log(vector_N),log(error(:,6)),"-*;Sin3-Transform;",
149      log(vector_N),log(error(:,7)),"-x;Sin4-Transform;",
150      log(vector_N),log(error(:,8)),"-x;No Periodization;");
151
152 xlabel('Log(N)');
153 ylabel('Log(RMSE)');
```

A.1.18. Test QMC del precio de Spread Option con precio de ejercicio distinto de cero

Nombre Script: test_spread_option_k_dist_zero.m

```
##############################################
# Autor    : Ruben Colomina Citoler
# PRECIO DE SPREAD OPTIONS PARA EL CASO GENERAL K distinto de CERO
##############################################
# INPUTS
# w1,w2 : pesos
# K : precio de ejercicio
# N : numero grande QMC referencia
# conjunto de parametros, S10, sigma1, T, ro
# S10 : precio inicial primer activo
# sigma1 : volatilidad primer activo
# T : perido de vida
# ro : coeficiente de correlacion entre S1 y S2
# rate : tipo de interes
# sigma2 : volatilidad segundo activo
# S20 : precio inicial segundo activo

# OUTPUTS
# Valoraciones de Spread Option para diferentes valores
# de tam. muestral y forma de periodizacion

# pesos de la spread option
w1=w2=1;

# relajar condicion K distinto de zero
K=1;

# numero de muestras QMC referencia teorica
N=121393;

# conjuntos 1 de parametros: 144 casos
#vector_S10=[92,96,100,104];    # 4 casos
#vector_sigma1=[0.1,0.2,0.3]    # 3 casos
#vector_T=[7/364,30/365,1,5];   # 4 casos : 1 semana, 1 mes, 1 anho, 5 anhos
#vector_ro=[-0.5,0,0.5];        # 3 casos

# conjunto 2 de parametros: 16 casos
#vector_S10=[92,104];       # 2 casos
#vector_sigma1=[0.3]    # 2 casos
#vector_T=[1/12];       # 2 casos : 1 mes, 1 anho
#vector_ro=[-0.5];      # 2 casos

# conjunto 3 de parametros: 24 casos
vector_S10=[92,104];       # 2 casos
vector_sigma1=[0.1,0.3]    # 2 casos
vector_T=[1/12,1,5];       # 3 casos : 1 mes, 1 anho, 5 anhos
vector_ro=[-0.5,0.5];      # 2 casos

# parametros fijos
rate=0.1;
sigma2=0.2;
S20=100;
delta1=delta2=0.05;

# glp dimension 2 fibo. tamanho 121393 puntos
```

```
m=26;
glp_big=f_glp(m);

################################################################################

numero_casos=length(vector_S10)*length(vector_sigma1)*length(vector_T)*length(vector_ro);

# arrancar reloj t0
t0=clock();

# reservar memoria
qmc_n_grande=zeros(numero_casos,1);
num_tam_glp=10;
num_param_perio=7;

valor_opciones=zeros(num_tam_glp,num_param_perio,numero_casos); # (tamanho glp, tipo
    periodiza, casos)

# bucle para calcular valor teoricos y aproximados
conta=1;
for i=1:length(vector_S10)
  for j=1:length(vector_sigma1)
          for k=1:length(vector_T)
                 for l=1:length(vector_ro)
                     S10=vector_S10(i);
                          sigma1=vector_sigma1(j);
                          T=vector_T(k);
                          ro12=vector_ro(l);

                          ### CALCULAR LOS VALORES DE REFERENCIA TEORICAS PARA EL CONJUNTO
                               DE PARAMETROS
                          qmc_n_grande(conta)=spread_option(glp_big,w1,w2,rate,K,S10,S20,
                              delta1,delta2,sigma1,sigma2,ro12,T);
                          ###

                          ### CALCULAR UN VALOR APROXIMADOS SEGUN TAM. MUESTRA Y TIPO
                              PERIODIZACION
                     for n=1:num_tam_glp
                            glp=f_glp(n+6);
                            for param_perio=1:num_param_perio
                                    qmc_n=spread_option_periodifica(glp,w1,w2,rate,K,S10,
                                        S20,delta1,delta2,sigma1,sigma2,ro12,T,param_perio)
                                        ;
                                    param_perio;
                                    log_long_glp=log(length(glp));

                                    valor_opciones(n,param_perio,conta)=qmc_n;

                                   endfor
                          endfor
                          #######
                          conta++;
                 endfor
          endfor
  endfor
endfor

# tomar tiempo t0
elapse_time_t0=etime(clock(),t0)

### CALCULAR RMSE PARA POR COLUMNAS DE MATRIZ valor_opciones
```

```
errores=zeros(num_tam_glp,num_param_perio);
for i=1:num_tam_glp
  for j=1:num_param_perio
              sumerror=0;
                        for k=1:numero_casos
                           sumerror+=f_error(valor_opciones(i,j,k),qmc_n_grande(k));
                        endfor
              errores(i,j)=log(sqrt(sumerror/numero_casos));
  endfor
endfor

### PINTANDO GRAFICAS DEL ERROR
x=log(length(f_glp(7)));
for i=1:9
   x=[x,log(length(f_glp(7+i)))];
endfor
aux=8;
plot(x(1:aux),errores(1:aux,1),"-o;Polynomial-2;",
     x(1:aux),errores(1:aux,2),"-o;Polynomial-3;",
     x(1:aux),errores(1:aux,3),"-o;Polynomial-4;",
     x(1:aux),errores(1:aux,4),"-*;Sin1-Transform;",
     x(1:aux),errores(1:aux,5),"-*;Sin2-Transform;",
     x(1:aux),errores(1:aux,6),"-*;Sin3-Transform;",
     x(1:aux),errores(1:aux,7),"-x;Sin4-Transform;");
xlabel('Log(N)');
ylabel('Log(RMSE)');
```

A.1.19. Test QMC para aproximar las griegas Delta y Gamma de una Spread Option

Nombre Script: test_spread_option_delta_gamma.m

```
##########################################
# Autor    : Ruben Colomina Citoler
# APROXIMACIONES DEL VALOR DE LAS GRIEGAS PARA SPREAD OPTIONS
##########################################
# TEST SPREAD OPTIONS:  Calculo de griegas Delta y Gamma
# INPUTS
# w1 y w2 : pesos de cada subyacente considerados positivos
# rate : tipo de interes
# delta1 y delta2 : dividendos para los subyacente 1 y 2
# S01 y S02 precios de los subyacente en el instante 0
# sigma1 y sigma2 son las volatilidades de los subyacentes
# ro12 en [-1,1] correlacion entre los subyacentes
# T tiempo de madurez de las opciones
# K precio del strike del spread
# param : tipo de periodizacion aplicada
# OUTPUTS
# Valor de la griega delta1, delta2, gamma1 y gamma2 para una Spread Option

#Parametros fijos
w1=w2=1;
S20=100;
delta1=0.05;
delta2=0.05;
sigma1=0.3;
```

```
25 sigma2=0.2;
26 ro12=0.5;
27 rate=0.05;
28 T=5;
29 K=4;
30
31 printf("PARAMETROS DE LA SIMULACION\n");
32 printf("w1: %.2f \nw2: %.2f \n",w1,w2)
33 printf("S20: %.2f \n",S20);
34 printf("delta1: %.2f \ndelta2: %.2f \n",delta1,delta2);
35 printf("sigma1: %.2f \nsigma2: %.2f \n",sigma1,sigma2);
36 printf("ro12: %.2f \n",ro12);
37 printf("rate: %.2f \n",rate);
38 printf("T: %.2f \nK: %.2f \n",T,K);
39
40 # casos de periodizacion
41 # (1,poly-2),(2,poly-3),(3,poly-4),(4,sin-1),(5,sin-2),(6,sin-3),(7,sin-4),(8,noperio)
42 array_param_perio=1:8; # poly-3, sin-2, sin-3
43
44 # casos de precios para S10
45 array_S10 = [95,96,97,98,99,100,101,102,103,104];
46 DELTA1    = zeros(length(array_S10),length(array_param_perio));
47 DELTA2    = zeros(length(array_S10),length(array_param_perio));
48 GAMMA1    = zeros(length(array_S10),length(array_param_perio));
49 GAMMA2    = zeros(length(array_S10),length(array_param_perio));
50
51 errorstandardDELTA1    = zeros(length(array_S10),length(array_param_perio));
52 errorstandardDELTA2    = zeros(length(array_S10),length(array_param_perio));
53 errorstandardGAMMA1    = zeros(length(array_S10),length(array_param_perio));
54 errorstandardGAMMA2    = zeros(length(array_S10),length(array_param_perio));
55
56
57 ##############################
58 # CASOS N=55(m=10),233(m=13) #
59 ##############################
60 array_fibo=[10,13]; #casos para glp fibonaci
61
62 for tamfibo=1:length(array_fibo)
63 # creamos el conjunto de puntos glp
64   m=array_fibo(tamfibo);
65
66   printf("-----------------------------------\n");
67   printf("NUMERO PUNTOS good lattice points: %d\n",length(glp));
68   printf("-----------------------------------\n");
69
70   for precioS10=1:length(array_S10)
71     S10=array_S10(precioS10);
72     printf("----------\n");
73          printf("Precio S10: %d\n",S10);
74     printf("----------\n");
75     for perio_param=1:length(array_param_perio)
76       printf("-------------\n");
77            printf("Periodizacion: %d\n",perio_param);
78       printf("-------------\n");
79
80       parametro_periodizacion=array_param_perio(perio_param);
81
82       glp=f_glp(m);
83       muestras=10;
84                 teta=zeros(muestras,1);
85
```

```
86      #Se generan desplazamiento del gpl para estimar el error standard
87                elapse_time_delta1=0;
88                sumestimaciones=0;
89      for k=1:muestras
90                    glpshift=f_random_shift_set(glp);
91                         t0=clock();
92                teta(k)=spread_option_delta1_perio(glpshift,w1,w2,rate,K,S10,S20,delta1,
                      delta2,sigma1,sigma2,ro12,T,parametro_periodizacion);
93                         elapse_time_delta1+=etime(clock(),t0)/muestras;
94                         sumestimaciones+=teta(k);
95                endfor
96
97                DELTA1(precioS10,perio_param)=sumestimaciones/muestras;
98                errorstandardDELTA1(precioS10,perio_param)=log(error_standard(teta));
99
100     #Se generan desplazamiento del gpl para estimar el error standard
101               elapse_time_gamma1=0;
102               sumestimaciones=0;
103     for k=1:muestras
104                   glpshift=f_random_shift_set(glp);
105                        t0=clock();
106               teta(k)=spread_option_gamma1_perio(glpshift,w1,w2,rate,K,S10,S20,delta1,
                      delta2,sigma1,sigma2,ro12,T,parametro_periodizacion);
107                        elapse_time_gamma1+=etime(clock(),t0)/muestras;
108                        sumestimaciones+=teta(k);
109               endfor
110
111               GAMMA1(precioS10,perio_param)=sumestimaciones/muestras;
112               errorstandardGAMMA1(precioS10,perio_param)=log(error_standard(teta));
113
114     #Se generan desplazamiento del gpl para estimar el error standard
115               elapse_time_delta2=0;
116               sumestimaciones=0;
117     for k=1:muestras
118                   glpshift=f_random_shift_set(glp);
119                        t0=clock();
120               teta(k)=spread_option_delta2_perio(glpshift,w1,w2,rate,K,S10,S20,delta1,
                      delta2,sigma1,sigma2,ro12,T,parametro_periodizacion);
121                        elapse_time_delta2+=etime(clock(),t0)/muestras;
122                        sumestimaciones+=teta(k);
123               endfor
124
125               DELTA2(precioS10,perio_param)=sumestimaciones/muestras;
126               errorstandardDELTA2(precioS10,perio_param)=log(error_standard(teta));
127
128     #Se generan desplazamiento del gpl para estimar el error standard
129               elapse_time_gamma2=0;
130               sumestimaciones=0;
131     for k=1:muestras
132                   glpshift=f_random_shift_set(glp);
133                        t0=clock();
134               teta(k)=spread_option_gamma2_perio(glpshift,w1,w2,rate,K,S10,S20,delta1,
                      delta2,sigma1,sigma2,ro12,T,parametro_periodizacion);
135                        elapse_time_gamma2+=etime(clock(),t0)/muestras;
136                        sumestimaciones+=teta(k);
137               endfor
138
139               GAMMA2(precioS10,perio_param)=sumestimaciones/muestras;
140               errorstandardGAMMA2(precioS10,perio_param)=log(error_standard(teta));
141
142
```

```
    endfor
  endfor
endfor

printf("simulacion terminada con exito :) \n");
```

A.1.20. Test QMC aproximación de las griegas Delta de una Spread Option mediante resimulación

Nombre Script: test_spread_option_delta_resimula.m

```
##########################################
#
# Autor    : Ruben Colomina Citoler
#
# APROXIMACIONES DELTA DE UNA SPREAD OPTIONS
# MEDIANTE RESIMULACION
# TIEMPOS DE CALCULO
#
##########################################
#
# INPUTS
# w1 y w2 : pesos de cada subyacente considerados positivos
# rate : tipo de interes
# delta1 y delta2 : dividendos para los subyacente 1 y 2
# S01 y S02 : precios de los subyacente en el instante 0
# sigma1 y sigma2 : volatilidades de los subyacentes
# ro12 en [-1,1] : correlacion entre los subyacentes
# T tiempo de madurez de las opciones
# K : precio del strike del spread
# param : tipo de periodizacion aplicada
# OUTPUTS
# Valor de las griega delta1 de forma directa e indirecta
# Tiempos de simulacion de ambos metodos

#Parametros fijos
w1=w2=1;
S20=100;
delta1=0.05;
delta2=0.05;
sigma1=0.3;
sigma2=0.2;
ro12=0.5;
rate=0.05;
T=5;
K=4;

# periodizacion sin-2
param=2;
# N=55
m=10;
glp=f_glp(m);
# Precio
S10=96;
# Variacion del precio
h=0.0000001;  #10e-7
```

```
# Iteraciones glp
iter=10;  # N=55,89,144,233,377,610,987,1597,2584,4181

elapse_time=zeros(iter,2);
lenglp=zeros(iter,1);

for i=1:iter
  #generar GLM con m
  glp=f_glp(m);
  lenglp(i)=length(glp);
  m+=1;
  #METODO DIRECTO
  t0=clock();
  delta_met_dir=spread_option_delta1_perio(glp,w1,w2,rate,K,S10,S20,delta1,delta2,sigma1,
      sigma2,ro12,T,param);
  elapse_time(i,1)=etime(clock(),t0);
  #METODO DE RESIMULACION
  t0=clock();
  valor           =spread_option_periodifica(glp,w1,w2,rate,K,S10,S20,delta1,delta2,
      sigma1,sigma2,ro12,T,param);
  valor_perturbado=spread_option_periodifica(glp,w1,w2,rate,K,S10+h,S20,delta1,delta2,
      sigma1,sigma2,ro12,T,param);
  delta_met_resim=(valor_perturbado-valor)/h;
  elapse_time(i,2)=etime(clock(),t0);
  diff=delta_met_resim-delta_met_dir
endfor
```

A.2. Funciones

A.2.1. Fórmula del volumen de S^{n-1}

Nombre función: *f_hiperesfera_volumen.m*

```
##############################################
# Autor     : Ruben Colomina Citoler
##############################################
# INPUT
# n dimension del espacio
# R radio
# OUTPUT: volumen exacto de la hiperesfera
function f_retorno=f_hiperesfera_volumen(n,R)
        f_retorno = power(pi,n/2)*power(R,n)/gamma(n/2+1);
endfunction
```

A.2.2. Determina si $x \in S^{n-1}$

Nombre función: *f_hiperesfera_dentro_radio.m*

```
##############################################
# Autor     : Ruben Colomina Citoler
##############################################
```

```
# INPUT
# R radio
# x punto espacio de dimension n, n-componentes
# OUTPUT: true = dentro, false = fuera
function f_retorno=f_hiperesfera_dentro_radio(R,x)
   if((norm(x))<R)
          f_retorno=true;
        else
          f_retorno=false;
   endif
endfunction
```

A.2.3. Generador de puntos aleatorios $x \in [0, R)^n$

Nombre función: *f_hipercubo_punto_aleatorio.m*

```
########################################
# Autor    : Ruben Colomina Citoler
########################################
# Generador de puntos aleatorios en cubo centrado en cero
# y radio R de dimension n
# INPUTS
# n dimension del espacio
# R radio hipercubo
# OUTPUT: punto aleatorio en el hipercubo de dimension
n y radio R function f_retorno=f_hipercubo_punto_aleatorio(n,R)
   x=zeros(n,1);
   for i=1:n
          x(i)=rand*R;
        endfor
        f_retorno=x;
endfunction
```

A.2.4. Generación de distribución normal mediante Box-Muller

Nombre de Función: *f_box_muller.m*

```
########################################
# Autor    : Ruben Colomina Citoler
########################################
# INPUT
# mu : Media
# sigma : Varianza
# OUTPUT
# Valor aleatorio distribucion normal de media mu
# y varianza sigma
function f_return=f_box_muller(mu,sigma)
  # Generar variables aleatorias uniformes independientes
  U1 = rand();
  U2 = rand();
  # Aplicar el algoritmo de Box-Muller
  Z = sqrt( -2 * log(U1) ) * cos ( 2 * pi * U2 );
  # Dilatar y desplazar con sigma y mu la normal estandar
```

```
  f_return = sigma * Z + mu;
endfunction
```

A.2.5. Generador de Moviento Geométrico Browniano

Nombre función: *f_geometric_brownian_motion.m*

```
##########################################
#
# Autor    : Ruben Colomina Citoler
#
##########################################
# GENERADOR DE UN GEOMETRIC BROWNIAN MOTION
#
# INPUT
# Parametros del GBM
# T : periodo de generacion
# Z : vector aleatorio normalde n muestras
# rate : tipo de interes
# delta: dividentos
# sigma: volatilidad
# S0 : precio inicial
# Algoritmo de discretizacion : Aproximation de Euler
# S(i+1) = S(i) * ( 1 + (rate - delta) * DT + sigma * sqrt(DT) * Z)
# OUTPUT : vector con un GBM
function f_return = f_geometric_brownian_motion(T,Z,rate,delta,sigma,S0)
  n=length(Z);
  # Dt : diferencial de tiempo
  Delta_t=T/n;
  # reservamos memoria
  GEOBROWMOT=zeros(n,1);
  # Algoritmo de discretizacion : Aproximation de Euler
  # S(i+1) = S(i) * ( 1 + (rate - delta) * DT + sigma * sqrt(DT) * Z)
  GEOBROWMOT(1)=S0;
  for i=2:n
    S0 = S0 * ( 1 + (rate - delta) * Delta_t + sigma * sqrt(Delta_t) * Z(i));
    GEOBROWMOT(i) = S0;
  endfor
  f_return=GEOBROWMOT;
endfunction
```

A.2.6. Momento explítico de un Movimiento Geométrico Browniano

Nombre función: *f_gbm_explicit.m*

```
##########################################
#
# Autor    : Ruben Colomina Citoler
#
##########################################
# MOMENTO EXPLICITO DE UN GEOMETRIC BROWNIAN MOTION
```

```
# Iteracion n-esima de un Geometric Brownian Motion
# INPUT : parametros del GBM
# S0 : Precio inicial
# r  : Tipo de interes
# sigma : volatilidad
# delta : dividendo subyacente
# n : iteracion requerida (n<=long(Z))
# T : periodo del GBM
# Z : vector normal gausiano

# Algoritmo de discretizacion: Formula explicita
# S(n) = S0 * exp( r-delta-0.5*sigma2)nT/m+
#            sigma*Raiz(T/m)*suma(Z1+...Zn)

# OUTPUT : Momento n-esima del GBM definido por vector Z
function f_return=f_gbm_explicit(S0,r,sigma,delta,n,T,Z)
   m=length(Z);
   if(n>m)
          printf("Iteracion fuera de rango\n");
   endif
   paso=T/m;
   sumaZ=0;
        if(n==0)
          f_return=S0;
   else
          for i=1:n
            sumaZ=Z(i);
     endfor
     aux1 = exp (( r - delta - 0.5 * sigma * sigma) * n * paso);
     aux2 = exp ( sigma * sqrt(paso) * sumaZ );
          f_return = S0 * aux1 * aux2;
       endif
endfunction
```

A.2.7. Transformar un número natural en su correspondiente Van der Corput de base b

Nombre función: *f_van_der_corput.m*

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# Funcion para transformar un numero natural en
# su correspondintente de la secuencia de Van der Corput
# INPUT
# b : base
# n : numero
# OUTPUT
# c : numero n transformado en la secuencia
function f_return=f_van_der_corput(b,n)
n0=n;
c=0;
ib=1/b;
while(n0>0)
  n1=floor(n0/b);
  i=n0-n1*b;
```

```
  c=c+ib*i;
  ib=ib/b;
  n0=n1;
endwhile
f_return=c;
endfunction
```

A.2.8. Permutación de orden n

Nombre función: *f_permuta_aleatoria.m*

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# INPUTS
# n : orden de la permutacion
# OUTPUT
# vector de tam. n con permutacion aleatoria
function f_return=f_permuta_aleatoria(n)
  x=1;
  for i=2:n
        x=[x,i];
  endfor
  for i=1:n-1
        aleat=floor(rand*(n-i+1))+1;
        y(i)=x(aleat);
        if(aleat==1)
              x=x(2:n-i+1);
    else if(aleat==n-i+1)
              x=x(1:n-i);
    else
          x=[x(1:aleat-1),x(aleat+1:n-i+1)];
    endif
    endif
  endfor
  y(n)=x;
  f_return=y;
endfunction
```

A.2.9. Genera LDS Van der Corput multidimensional

Nombre función: *f_van_der_corput_ndim.m*

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# INPUTS
# N   : numero de puntos
# dim : dimension
# OUTPUTS
# Array multidimensional LDS van der corput
function f_return=f_van_der_corput_ndim(N,dim)
  for i=1:N
```

```
        secuenciaN(i)=f_van_der_corput(2,i);
  endfor
  secuenciaN;
  array=zeros(N,dim);
  for j=1:dim
        permuta=f_permuta_aleatoria(N);
        for i=1:N
              array(i,j)=secuenciaN(permuta(i));
    endfor
  endfor
  f_return=array;
endfunction
```

A.2.10. Generador de una secuencia de Halton con N puntos en $[0,1)^s$

Nombre función: *f_halton.m*

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# Generador de secuencias de Halton
# INPUT
# v : Vector de coprimos
# N : tamagno de la secuencia
# OUTPUT
# Secuencia de Halton de N en [0,1]^s
function f_return=f_halton(v,N)
u=zeros(N,length(v));
for i=1:N
  for j=1:length(v)
        u(i,j)=f_van_der_corput(v(j),i);
  endfor
endfor
f_return=u;
endfunction
```

A.2.11. Generador de una Lattice Rule de rango 1

Nombre función: *f_glp_ndim.m*

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# Genera una red de integracion usando una Lattile Rule
# de Fibonacci
# Vector de generacion (1,l,l2 mod N,
#INPUTS:
# m :Numero de puntos de la red de integracion
# d :Dimension de la red
# l :Parametro libre 1<=l<N
#OUTPUT:
```

```
# Array d-dimensional de m puntos
function f_return=f_glp_ndim(m,dim,l)
  glp=zeros(m,dim);
  z=zeros(dim,1);
  z=[1,l];
  for i=3:dim
         z=[z,fmod(power(l,i),m-1)];
  endfor
  for i=1:m
         for j=1:dim
                x=i*z(j)/m;
                glp(i,j)=fmod(i*z(j)/m,1);
         endfor
  endfor
  f_return=glp;
endfunction
```

A.2.12. Método de elección del parámetro de una Lattice Rule de rango 1

Nombre función: *f_mincorr_param_glp_r1.m*

```
##########################################
# Autor     : Ruben Colomina Citoler
##########################################
# Metodo para elegir un candidado de parametro
# para una Lattice Rule de rango 1
# INPUT
# N : numero de puntos
# dim : dimension
# Complejidad del orden N^2*dim
# OUTPUT
# Red con correlacion minima entre pares
function f_return=f_mincorr_param_glp_r1(N,dim)
  parametro=0;
  correlacion_antigua=0.2;
   for k=2:N-1
         tam=dim*(dim-1)/2;
         correlacion=zeros(tam,1);
    index=0;
    punto_lattice=f_glp_ndim(N,dim,k);
         for i=1:(dim-1)
                for j=(i+1):dim
                  index=index+1;
                  correlacion(index)=corr(punto_lattice(:,i),punto_lattice(:,j));
                endfor
                correlacion_nueva=max(abs(correlacion));
         endfor
         if(correlacion_nueva<correlacion_antigua)
                parametro=k;
      correlacion_antigua=correlacion_nueva;
    endif
  endfor
  f_return=parametro;
endfunction
```

A.2.13. Modelo exacto de Black-Scholes Opción Call Europea

Nombre función: *f_european_call_black_scholes.m*

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# Black-Scholes para Opcion Call Europea
# INPUTS
# S0     : precio inicial
# K      : strike price
# rate   : tipo interes
# delta  : dividendo
# sigma  : volatilidad
# T      : periodo
# OUTPUT
# precio de opcion call europea
function f_return=f_european_call_black_scholes(S0,K,rate,delta,sigma,T)
  d1 = ( log ( S0 / K ) +
       (rate - delta + 0.5 * sigma * sigma) * T ) / (sigma * sqrt(T));
  d2 = d1 - sigma * sqrt(T);
  f_return = S0 * exp ( - delta * T ) * stdnormal_cdf(d1)
       - exp ( - rate * T ) * K * stdnormal_cdf(d2);
endfunction
```

A.2.14. Error cuadrático de una estimación respecto a su valor teórico

Nombre función: *f_error.m*

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# INPUTS
# estimado : Valor de una estimacion
# teorico  : Valor teorico
# OUTPUT
# Error cuadratico relativo al valor teorico
function f_ret=f_error(estimado,teorico)
          f_ret=power(((teorico-estimado)/teorico),2);
endfunction
```

A.2.15. Estimador Pathwise Delta Opción Call Europea

Nombre función: *european_call_estimator_pathwise_delta.m*

```
##########################################
#
# AUTOR : Ruben Colomina Citoler
#
##########################################

```

```
# Estimador Delta pathwise de una opcion Call Europea
# INPUTS
# S0     : precio subyacente
# ST     : precio final en mome
# K      : strike price
# rate   : tipo interes
# T      : periodo
# OUTPUT
# Delta d(precio)/dS0
function f_return=european_call_estimator_pathwise_delta(S0,ST,K,rate,T)
  if( ST >= K )
    f_return = exp(-rate*T) * ST / S0;
  else
         f_return = ST * 0;
  endif
endfunction
```

A.2.16. Estimador Pathwise Vega Opción Call Europea

Nombre función: *european_call_estimator_pathwise_vega.m*

```
##########################################
#
# AUTOR : Ruben Colomina Citoler
#
##########################################

# Estimador de Vega para una opcion call europea usando pathwise
# INPUTS
# S0     : precio subyacente
# ST     : precio final en T
# K      : strike price
# rate   : tipo interes
# delta  : dividendo
# sigma  : volatilidad
# T      : periodo
# OUTPUT
# Vega d(precio)/d(sigma) (ST): Depende de la muestra
function f_return=european_call_estimador_pathwise_vega(S0,ST,K,rate,delta,sigma,T)
  sigma2=sigma*sigma;
  aux = ( ST / sigma) * ( log ( ST / S0 ) - ( rate - delta + 0.5 * sigma2 ) * T );
  if( ST >= K )
    f_return = exp( - rate * T ) * aux;
  else
         f_return = 0;
  endif
endfunction
```

A.2.17. Estimador Pathwise Rho Opción Call Europea

Nombre función: *european_call_estimator_pathwise_rho.m*

```
##########################################
```

```
#
# AUTOR : Ruben Colomina Citoler
#
########################################

# Estimador pathwise para Rho de una opcion call europea
# INPUTS
# S0     : precio inicial subyacente
# ST     : precio final subyacente
# K      : strike price
# rate   : tipo interes
# delta  : dividendo
# sigma  : volatilidad
# T      : periodo de maduracion
# OUTPUT
# Rho d(precio)/d(rate) (ST): Depende de la muestra
function f_return=european_call_estimador_pathwise_rho(S0,ST,K,rate,delta,sigma,T)
  if(ST>=K)
         f_return = K * T * exp ( - rate * T );
  else
    f_return = 0;
  endif
endfunction
```

A.2.18. Estimador Pathwise Theta Opción Call Europea

Nombre función: *european_call_estimator_pathwise_theta.m*

```
########################################
#
# AUTOR : Ruben Colomina Citoler
#
########################################

# Estimador pathwise para Theta de una Opcion call Europea
# INPUTS
# S0     : precio inicial subyacente
# ST     : precio final subyacente
# K      : strike price
# rate   : tipo interes
# delta  : dividendo
# sigma  : volatilidad
# T      : periodo de maduracion
# OUTPUT
# Theta (-d(precio)/d(T)) (ST): Depende de la muestra
function f_return=european_call_estimador_pathwise_theta(S0,ST,K,rate,delta,sigma,T)
  if(ST>=K)
         unoSTK=1;
  else
         unoSTK=0;
  endif
  sigma2 = sigma * sigma;
  aux1 = rate * exp ( - rate * T ) * max( ST - K, 0 );
  aux2 = exp ( -rate * T ) * ST / ( 2 * T ) * ( log(ST/S0) + (rate - delta - 0.5 * sigma2
      ) * T );
  f_return = aux1 - unoSTK * aux2;
endfunction
```

A.2.19. Estimador Likelihood Delta Opción Call Europea

Nombre función: *european_call_estimator_likehood_delta.m*

```
##########################################
#
# AUTOR : Ruben Colomina Citoler
#
##########################################

# Estimador likelihood ratio de Delta para una opcion call europea
# INPUTS
# S0    : precio subyacente en instante cero
# ST    : precio subyacente a final de periodo
# K     : strike price
# rate  : tipo interes
# delta : dividendo
# sigma : volatilidad
# T     : periodo
# OUTPUT
# Delta d(precio)/d(S0)
function f_return=european_call_estimador_likelihood_delta(S0,ST,K,rate,delta,sigma,T)
  aux1 = exp( - rate * T ) * max( ST - K, 0);
  aux2 = log( ST / S0 ) - ( rate - delta - 0.5 * sigma * sigma ) * T;
  f_return = aux1 / ( S0 * sigma * sigma * T ) * aux2;
endfunction
```

A.2.20. Estimador Likelihood Vega Opción Call Europea

Nombre función: *european_call_estimator_likelihood_vega.m*

```
##########################################
#
# AUTOR : Ruben Colomina Citoler
#
##########################################

# Estimador likelihood ratio de Vega para una opcion call europea
# INPUTS
# S0    : precio subyacente en instante cero
# ST    : precio subyacente a final de periodo
# K     : strike price
# rate  : tipo interes
# delta : dividendo
# sigma : volatilidad
# T     : periodo
# OUTPUT
# Delta d(precio)/d(sigma)

function f_return=european_call_estimador_likelihood_vega(S0,ST,K,rate,delta,sigma,T)
  aux1 = exp( - rate * T ) * max( ST - K, 0);
```

```
  aux2 = log( ST / S0 ) - ( rate - delta - 0.5 * sigma * sigma ) * T;
  d = aux2 / (sigma * sqrt(T));
  parcial_d_sigma = ( log( S0 / ST ) + ( rate - delta + 0.5 * sigma * sigma ) * T ) / (
      sigma * sigma * sqrt(T));
  f_return = aux1 * ( - d * parcial_d_sigma - 1 / sigma );
endfunction
```

A.2.21. Estimador Likelihood Gamma Opción Call Europea

Nombre función: *european_call_estimator_likelihood_gamma.m*

```
########################################
#
# AUTOR : Ruben Colomina Citoler
#
########################################

# Estimador likelihood ratio de Gamma para una opcion call europea
# INPUTS
# S0     : precio subyacente en instante cero
# ST     : precio subyacente a final de periodo
# K      : strike price
# rate   : tipo interes
# delta  : dividendo
# sigma  : volatilidad
# T      : periodo
# OUTPUT
# Delta d2(precio)/d2(S0)
function f_return=european_call_estimador_likelihood_gamma(S0,ST,K,rate,delta,sigma,T)
  aux1 = exp( - rate * T ) * max( ST - K, 0);
  aux2 = log( ST / S0 ) - ( rate - delta - 0.5 * sigma * sigma ) * T;
  d = aux2 / (sigma * sqrt(T));
  f_return = aux1 * ( d * d - d * sigma * sqrt(T) - 1 ) / ( S0 * S0 * sigma * sigma * T);
endfunction
```

A.2.22. Estimador Likelihood Rho Opción Call Europea

Nombre función: *european_call_estimator_likelihood_rho.m*

```
########################################
#
# AUTOR : Ruben Colomina Citoler
#
########################################

# Estimador likelihood ratio de Rho para una opcion call europea
# INPUTS
# S0     : precio subyacente en instante cero
# ST     : precio subyacente a final de periodo
# K      : strike price
# rate   : tipo interes
# delta  : dividendo
# sigma  : volatilidad
```

```
# T     : periodo
# OUTPUT
# Delta d(precio)/d(rate)
function f_return=european_call_estimador_likelihood_rho(S0,ST,K,rate,delta,sigma,T)
  aux1 = exp( - rate * T ) * max( ST - K, 0);
  aux2 = log( ST / S0 ) - ( rate - delta - 0.5 * sigma * sigma ) * T;
  d = aux2 / (sigma * sqrt(T));
  f_return = aux1 * ( - T + ( d * sqrt(T) ) / sigma );
endfunction
```

A.2.23. Estimador Likelihood Theta Opción Call Europea

Nombre función: *european_call_estimator_likelihood_theta.m*

```
##########################################
#
# AUTOR : Ruben Colomina Citoler
#
##########################################

# Estimador likelihood ratio de Rho para una opcion call europea
# INPUTS
# S0    : precio subyacente en instante cero
# ST    : precio subyacente a final de periodo
# K     : strike price
# rate  : tipo interes
# delta : dividendo
# sigma : volatilidad
# T     : periodo
# OUTPUT
# Delta -d(precio)/d(T)

function f_return=european_call_estimador_likelihood_theta(S0,ST,K,rate,delta,sigma,T)
  aux1 = exp( - rate * T ) * max( ST - K, 0);
  aux2 = log( ST / S0 ) - ( rate - delta - 0.5 * sigma * sigma ) * T;
  d = aux2 / (sigma * sqrt(T));
  parcial_d_T = ( -log( ST / S0 ) - ( rate - delta - 0.5 * sigma * sigma ) * T ) / ( 2 *
      sigma * power(T,3/2));
  f_return = aux1 * ( rate + d * parcial_d_T + 1/ (2*T) );
endfunction
```

A.2.24. Estimador del precio de una Opción Call Asiática

Nombre función: *asian_call_estimator_aritmetic_average_price.m*

```
##########################################
# Autor    : Ruben Colomina Citoler
##########################################
# Estimador de una opcion Call Asiatica
# INPUTS
# S0    : precio subyacente
# S     : vector de precios
# K     : strike price
```

```
# rate  : tipo de interes
# T     : momento de maduracion
# OUTPUT
# Asian Call aritmetic average price
# Precio = E(exp^(-rT)*max(Promedio(S)-K,0)]
function f_return=asian_call_estimator_aritmetic_average_price(S0,S,K,rate,T)
  aux1 = exp ( - rate * T );
  aux2 = max( mean ( S ) - K , 0 );
  f_return = aux1 * aux2;
endfunction
\begin{lstlisting}
```

A.2.25. Estimador Pathwise de Delta de una Opción Call Asiática

Nombre función: *european_call_estimator_pathwise_delta.m*

```
########################################
# Autor    : Ruben Colomina Citoler
########################################
# Estimador Pathwise de Delta de una Opcion Call Asiatica
# INPUTS
# S0     : precio subyacente
# S      : vector de precios a cierre de dia
# K      : strike price
# rate   : tipo de interes
# T      : momento de maduracion
# OUTPUT
# Delta d(precio)/d(S0)
function f_return=asian_call_estimator_pathwise_delta(S0,S,K,rate,T)
  mediaPrecios=mean(S);
  if( mediaPrecios >= K )
    f_return = exp( - rate * T ) * mediaPrecios / S0;
  else
        f_return = 0;
  endif
endfunction
```

A.2.26. Generador de Good Lattice Rule de rango 1 bidimensional óptima de tamaño $N = F_m$

Nombre función: *f_glp.m*

```
########################################
#
# Autor    : Ruben Colomina Citoler
#
########################################
#
# Generador de red de integracion GLP optima (Fibonacci)
# INPUTS
# m : Termino de la sucesion de Fibonacci
# OUTPUT
```

```
# Array bidimensional N=Fm termino m-esimo Fibonacci

function f_return=f_glp(m)

  fibo=1:m;
  fibo(1)=1;fibo(2)=1;
  for i=3:m
           fibo(i)=fibo(i-1)+fibo(i-2);
  endfor

  N=fibo(m);
  glp=zeros(N,2);

  z=[1,fibo(m-1)];

  for i=1:N
          glp(i,1)=fmod(i*z(1)/N,1);
          glp(i,2)=fmod(i*z(2)/N,1);
  endfor

  f_return=glp;
endfunction
```

A.2.27. Desplazamiento aleatorio de una GLP bidimensional

Nombre función: *f_random_shift.m*

```
##########################################
#
# Autor    : Ruben Colomina Citoler
#
##########################################
#
# DESPLAZAMIENTO ALEATORIO DE ARRAY BIDIMENSIONAL
#
# INPUT
# Array bidimensional en [0,1)^2
# OUTPUT
# Array bidimensional desplazado aleatoriamente en [0,1)^2
#
function f_retorno=f_random_shift_set(glp)

  #crer un vector aleatorio aleatorios en [0,1)^2
  N=length(glp);
  angle=rand;
  vector_rand=[cos(angle),sin(angle)];
  vector_rand/=norm(vector_rand);

  #desplazar el conjunto
  shifted_glp=zeros(length(glp),2);
  shifted_glp(:,1)=glp(:,1)+vector_rand(1);
  shifted_glp(:,2)=glp(:,2)+vector_rand(2);

  #volver al espacio de partida
  shifted_glp(:,1)=fmod(shifted_glp(:,1),1);
  shifted_glp(:,2)=fmod(shifted_glp(:,2),1);

```

```
  f_retorno=shifted_glp;
endfunction
```

A.2.28. Desplazamiento aleatorio de una GLP multidimensional

Nombre función: *f_random_shift_ndim_set.m*

```
##########################################
#
# Autor    : Ruben Colomina Citoler
#
##########################################
#
# DESPLAZAMIENTO ALEATORIO DE ARRAY MULTIDIMENSIONAL
# modulo 1
#
# INPUTS:
# Array multidimensional Nxs en [0,1)^s
# OUTPUTS:
# Array multidimensional desplazado en [0,1)^s
function array_shifted_mod1=f_random_shift_ndim_set(array)

  dim=length(array(1,:));

  #crear vector de desplazamiento aleatorio
  vrand=rand;
  for i=2:dim
        vrand=[vrand,rand];
  endfor

  #desplazar array de entrada y aplicar mod 1
  for i=1:length(array(:,1))
     array_shifted(i,:)=array(i,:)+vrand;
  endfor

  array_shifted_mod1=fmod(array_shifted,1);

endfunction
```

A.2.29. Fórmula de Margrabe

Nombre función: *spread_option.m*

```
##########################################
# Autor    : Ruben Colomina Citoler
#
##########################################
# FORMULAD DE MARGRABE: PRECIO DE UNA SPREAD OPTION CON K=0
#
# INPUTS
# S01,S02 : precios de los subyacente en el instante 0  (numero decimal positivo)
# delta1,delta2 : dividendo subyacente 1 y 2   (numero decimal positivo)
# sigma1,sigma2 : volatilidades de los subyacentes  (numero decimal positivo)
```

```
# ro12 : Correlacion entre subyacente en [-1,1]
# T tiempo de madurez de las opciones (expresar en anhos)
# OUTPUT
# Valoracion de una spread option con k=0

function f_ret=margrabe_formula(S10,S20,delta1,delta2,sigma1,sigma2,ro12,T)

          sigma=sqrt(sigma1*sigma1+sigma2*sigma2-2*sigma1*sigma2*ro12);
     d2=(log(S20/S10)+(delta1-delta2+sigma*sigma/2)*T)/(sigma*sqrt(T));
     d1=d2-sigma*sqrt(T);
     f_ret=exp(-delta2*T)*S20*stdnormal_cdf(d2)-exp(-delta1*T)*S10*stdnormal_cdf(d1);

endfunction
```

A.2.30. Estimador de Spread Option sobre $[0,1)^2$

Nombre función: *spread_ option.m*

```
##########################################
# Autor     : Ruben Colomina Citoler
#
# Dependencias : (Funcion) hstar.m
##########################################
#
#
# INPUT
# set_points : Conjuto de puntos del dominio de integracion [0,1)^2
# w1, w2 : pesos de la spread
# rate : tipo de interes
# K precio de ejercicio
# S10, S20: precios iniciales de los subyacentes
# delta1, delta2 : dividendo de los subyacentes
# sigma1, sigma2 : volatilidades de los subyacentes
# ro12 : coeficiente de correlacion entre los subyacentes
# T : periodo de maduracion
# OUTPUT
# Valor de la Spread Option
function f_retorno = spread_option(set_points,w1,w2,rate,K,S10,S20,delta1,delta2,sigma1,
    sigma2,ro12,T)
          value=0;
     N=length(set_points);
          for i=1:N
                         value=value+hstar(set_points(i,1),set_points(i,2),w1,w2,rate,
                             delta1,delta2,S10,S20,sigma1,sigma2,ro12,T,K);
     endfor
     f_retorno=exp(-rate*T)*value/N;
endfunction
```

A.2.31. Integrando transformado para valorar el precio de una Spread Option mediante QMC

Nombre función: *hstar.m*

```
##########################################
# Autor     : Ruben Colomina Citoler
#
# Dependencias : (Funcion) stdnormal_inv_acklam.m
##########################################
#
# INTEGRANDO TRANSFORMADO DE UNA SPREAD OPTIONS EVALUADO SOBRE [0,1)^2
# INPUTS
# u1 en [0,1) :componente x del n-esimo termino de la serie de baja discrepancia
# u2 en [0,1) :componente y ...
# w1 y w2 : pesos de cada subyacente considerados positivos
# rate : tipo de interes
# delta1 y delta2 : dividendo subyacente 1 y 2
# S01 y S02 precios de los subyacente en el instante 0
# sigma1 y sigma2 son las volatilidades de los subyacentes
# ro12 en [-1,1]
# K precio del strike del spread
# T tiempo de madurez de las opciones
# OUTPUT
# Valor del integrando de una Spread Option en el punto evaluado
function f_ret=hstar(u1,u2,w1,w2,rate,delta1,delta2,S10,S20,sigma1,sigma2,ro12,T,K)

    #Cambio de variable muiT=log(Si0) + (rate-delta(i)-1/2sigma(i)^2)*T
         mu1T=log(S10)+(rate-delta1-1/2*sigma1*sigma1)*T;
    mu2T=log(S20)+(rate-delta2-1/2*sigma2*sigma2)*T;

         #Descomposicion de matriz de covarianzas en descomposicion de Cholesky: A=C*C'
    c11=sigma1*sqrt(T);
    c21=ro12*sigma2*sqrt(T);
    c22=sqrt(1-ro12*ro12)*sigma2*sqrt(T);


    if(u1>0) #criterio de seguridad
              h3=c11*stdnormal_inv_acklam(u1)+mu1T;
                        h1=w1*exp(h3);
         g=(log(h1+K)-log(w2)-mu2T-c21*stdnormal_inv_acklam(u1))/c22;
         d2=stdnormal_cdf(g);
         if((d2+u2*(1-d2))<1) #criterio de seguridad
                                  h5=stdnormal_inv_acklam(d2+u2*(1-d2));
              h4=c21*stdnormal_inv_acklam(u1)+c22*h5+mu2T;
              h2=w2*exp(h4);
              h=h2-h1-K;
              f_ret=(1-d2)*h;
                        else
                                  f_ret=0;
         endif
    else
             f_ret=0;
    endif

endfunction
```

A.2.32. Inversa de la normal estándar (Acklam)

Nombre función: *stdnormal_inv_acklam.m*

```
##########################################
# INVERSA NORMAL ESTANDAR (por ACKLAM)
# INPUTS
# p : percentil de la distribucion normal estandar en (0,1)
# OUTPUS
# Valor de la funcion de distribucion normal estandar
#
function f_return=stdnormal_inv_acklam(p)

   #Coefficients in rational approximations.

   a(1) = -3.969683028665376e1;
   a(2) = 2.209460984245205e2;
   a(3) = -2.759285104469687e2;
   a(4) = 1.383577518672690e2;
   a(5) = -3.066479806614716e1;
   a(6) = 2.506628277459239e0;

   b(1) = -5.447609879822406e+01;
   b(2) =  1.615858368580409e+02;
   b(3) = -1.556989798598866e+02;
   b(4) =  6.680131188771972e+01;
   b(5) = -1.328068155288572e+01;

   c(1) = -7.784894002430293e-03;
   c(2) = -3.223964580411365e-01;
   c(3) = -2.400758277161838e+00;
   c(4) = -2.549732539343734e+00;
   c(5) =  4.374664141464968e+00;
   c(6) =  2.938163982698783e+00;

   d(1) =  7.784695709041462e-03;
   d(2) =  3.224671290700398e-01;
   d(3) =  2.445134137142996e+00;
   d(4) =  3.754408661907416e+00;

  # Define break-points.
   p_low  = 0.02425;
   p_high = 1 - p_low;
  # Rational approximation for lower region.
  if((0 < p) && (p < p_low))
      q = sqrt(-2*log(p));
                aux1= ((((c(1)*q+c(2))*q+c(3))*q+c(4))*q+c(5))*q+c(6);
                aux2= ((((d(1)*q+d(2))*q+d(3))*q+d(4))*q+1);
      x =  aux1 / aux2;
   endif
  # Rational approximation for central region.
   if( (p_low <= p) && (p <= p_high))
      q = p - 0.5;
      r = q*q;
                aux1=(((((a(1)*r+a(2))*r+a(3))*r+a(4))*r+a(5))*r+a(6));
                aux2=(((((b(1)*r+b(2))*r+b(3))*r+b(4))*r+b(5))*r+1);
      x = aux1*q /  aux2;
   endif
  # Rational approximation for upper region.
   if( p_high < p && p < 1)
      q = sqrt(-2*log(1-p));
                aux1=-(((((c(1)*q+c(2))*q+c(3))*q+c(4))*q+c(5))*q+c(6));
                aux2=((((d(1)*q+d(2))*q+d(3))*q+d(4))*q+1);
      x = aux1 / aux2;
   endif
```

```
   f_return=x;
endfunction
```

A.2.33. Transformación de periodización del integrando de tipos polinómica y trigonométrica

Nombre función: *f_periodifica.m*

```
##########################################
#
# Autor    : Ruben Colomina Citoler
#
##########################################
# TRANSFORMACION DE FUNCIONES DE PERIODIZACION POLINOMICAS Y TRIGONOMETRICAS
# INPUTS
# t : variable de la funcion
# parametro : permite elegir entre tipo de periodizacion
# parametro,modelo
# (1,poly-2),(2,poly-3),(3,poly-4),(4,sin-1),(5,sin-2),(6,sin-3),(7,sin-4),(8,noperio)
# OUTPUT
# Valor de la funcion transformada por la funcion elegida por parametro
#
function f_return=f_periodifica(t,parametro)
     #transformaciones polinomicas
          if(parametro==1)
                        f_return= 3 * power(t,2) - 2 * power(t,3);
          else if(parametro==2)
                        f_return= 10 * power(t,3)- 15 * power(t,4) + 6 * power(t,5);
          else if(parametro==3)
                        f_return= 35 * power(t,4)-84 * power(t,5) + 70 * power(t,6) - 20
                            * power(t,7);
     endif
     endif
     endif
     #transformaciones trigonometricas
          if(parametro==4)
                   f_return = 1 / 2 * ( 1 - cos( pi * t ) );
          else if(parametro==5)
                        f_return = ( 1 / ( 2 * pi ) ) * ( 2 * pi * t - sin ( 2 * pi * t )
                             );
          else if(parametro==6)
                        f_return = ( 1 / 16 ) * ( 8 - 9 * cos ( pi * t ) + cos( 3 * pi *
                            t ) );
          else if(parametro==7)
                        f_return = ( 1 / ( 12 * pi ) ) * ( 12 * pi * t - 8 * sin( 2 * pi
                            * t ) + sin ( 4 * pi * t ) );
     endif
     endif
          endif
     endif

     #No periodifica
          if(parametro==8)
                         f_return=t;
     endif

```

```
endfunction
```

A.2.34. Funciones derivadas de las transformaciones de periodización de tipo polinómicas y trigonométricas

Nombre función: *f_periodifica_deriv*

```
##############################################
#
# Autor    : Ruben Colomina Citoler
#
##############################################
# TRANSFORMACION DE FUNCIONES DE PERIODIZACION POLINOMICAS Y TRIGONOMETRICAS
# INPUT
# t : variable de la funcion
# parametro : tipo de transformada elegida elegida entre [1,2,3,4,5,6,7,8]
# parametro,modelo
# (1,poly-2),(2,poly-3),(3,poly-4),(4,sin-1),(5,sin-2),(6,sin-3),(7,sin-4),(8,noperio)
# OUTPUT
# Devuelve la derivada de la transformada de la funcion elegida por parametro
#
function f_return=f_periodifica_deriv(t,parametro)

      #transformaciones polinomicas
           if(parametro==1)
                        f_return=6*power(t,1)-6*power(t,2);
           else if(parametro==2)
                        f_return=30*power(t,2)-60*power(t,3)+30*power(t,4);
           else if(parametro==3)
                        f_return=140*power(t,3)-420*power(t,4)+420*power(t,5)-140*power(t
                            ,6);
                else
      #transformaciones polinomicas
           if(parametro==4)
                        f_return=pi/2*(sin(pi*t));
           else if(parametro==5)
                        f_return=1-cos(2*pi*t);
           else if(parametro==6)
                      f_return=1/16*(9*pi*sin(pi*t)-3*pi*sin(3*pi*t));
                else if(parametro==7)
                      f_return=(1/(12*pi))*(12*pi-16*pi*cos(2*pi*t)+4*pi*cos(4*pi*t));
      #no transfoma
                else if(parametro==8)
                        f_return=1;
      endif
                endif
                endif
                endif
                endif
                endif
                endif
      endif

endfunction
```

A.2.35. Aproximación del precio de una LookBack Options Discreta por QMC

Nombre función: *lookback_ option.m*

```
##############################################
# Autor   : Ruben Colomina Citoler
# Fecha   : 30-08-2013
##############################################
# INPUTS
# S0 : precio de arranque del subyacente
# K  : precio del strike
# r  : tipo de interes
# delta : dividendo del subyacente
# sigma : volatilidad
# T  : periodo de maduracion de la opcion
# set_points : conjunto de puntos de Nxm  en [0,1)^m
# siendo N numero de puntos y m la dimension
#
##############################################
# OUTPUT
# Valoracion de una opcion lookback discreta
##############################################
function f_return=lookback_option(S0,K,r,delta,sigma,T,set_points)
   N=length(set_points(:,1));
        m=length(set_points(1,:));
   v_valor=zeros(N,1);
   for num_punto=1:N-1
     #Condicion para descargar el cero vector de un punto
          vector_precios=zeros(m+1,1);
     vector_precios(1)=S0;

          if(sum(set_points(num_punto,:)>0)==m)

                Z=stdnormal_inv(set_points(num_punto,:));
            for num_periodo=1:m
         aux1 = exp (( r - delta - 0.5 * sigma * sigma) * T/m);
         aux2 = exp ( sigma * sqrt(T/m) * Z(num_periodo));
                        vector_precios(num_periodo+1)=vector_precios(num_periodo)*aux1*
                            aux2;
                endfor

          endif
     # Lookback max(max(S0,S1,...,Sm)-K,0)
     maximo=max(max(S0,max(vector_precios))-K,0);
     v_valor(num_punto)=maximo;
        endfor
   f_return=exp(-r*T)*mean(v_valor);
endfunction
```

A.2.36. Aproximación del precio de una LookBack Options Discreta por QMC y periodización

Nombre función: *lookback_ option_ periodiza.m*

```
1
2  ##############################################
3  # Autor   : Ruben Colomina Citoler
4  # Fecha   : 30-08-2013
5  ##############################################
6  # INPUTS
7  # S0 : precio de arranque del subyacente
8  # K : precio del strike
9  # r : tipo de interes
10 # delta : dividendo del subyacente
11 # sigma : volatilidad
12 # T : periodo de maduracion de la opcion
13 # set_points : conjunto de puntos de Nxm
14 #  siendo N numero de puntos y m la dimension
15 #  en [0,1)^m
16 # param_perio : parametro de periodizacion
17 #
18 ##############################################
19 # OUTPUT
20 # Valoracion de una opcion lookback discreta
21 ##############################################
22 function f_return=lookback_option_periodiza(S0,K,r,delta,sigma,T,set_points,param_perio)
23
24    N=length(set_points(:,1));
25         m=length(set_points(1,:));
26
27    aux1 = exp (( r - delta - 0.5 * sigma * sigma) * T/m);
28
29    #Acota valores en entrada
30    coordmin=0.000000000000000001; #10e-8
31    coordmax=0.999999999999999999; #1-10e-8
32
33    sumaZ=0;
34    sumaMax=0;
35
36    v_valor=zeros(N-1,1);
37    for num_punto=1:N-1
38      #Condicion para descargar el cero vector de un punto
39           vector_precios=zeros(m+1,1);
40      vector_precios(1)=S0;
41      #Coeficiente de periodizacion multiplica al integrando
42      coef_perio_dif=1;
43
44      #Transformar el conjunto a normal estandard
45           #puntos_perio=set_points(num_punto,:);
46           puntos_perio=f_periodifica(set_points(num_punto,:),param_perio);
47
48           #Z=zeros(m,1);
49           #Z=stdnormal_inv(puntos_perio);
50      #Z=stdnormal_inv_modif(puntos_perio);
51
52           #Inversion por Acklam
53           for i=1:m
54
55        coordenada_modif=puntos_perio(i);
56                  if(puntos_perio(i)<coordmin)
57                    coordenada_modif=coordmin;
58                  else if(puntos_perio(i)>coordmax)
59                    coordenada_modif=coordmax;
60              endif
61                  endif
```

```
62              Z(i)=stdnormal_inv_acklam(coordenada_modif);
63           endfor
64
65       #Calcular coeficiente de periodizacion
66           for coord=1:m
67                  valor_coord=set_points(num_punto,coord);
68                  coef_perio_dif=coef_perio_dif*f_periodifica_deriv(valor_coord,
                        param_perio);
69           endfor
70
71       vector_precios(1)=S0;
72           for num_periodo=1:m
73
74                    if(Z(num_periodo)>10000)
75                          Z(num_periodo)
76                      end
77
78           aux2 = exp ( sigma * sqrt(T/m) * Z(num_periodo) );
79
80                      if(aux2>10000)
81                        aux2
82                      end
83
84                      valor=vector_precios(num_periodo)*aux1*aux2;
85
86                      if(valor>10000)
87                        valor
88                      endif
89
90                      vector_precios(num_periodo+1)=valor;
91           endfor
92
93       # Lookback max(max(S0,S1,...,Sm)-K,0)
94       maximo=max(max(S0,max(vector_precios))-K,0);
95       #sumaMax=sumaMax+maximo;
96       v_valor(num_punto)=maximo*coef_perio_dif;
97          endfor
98     #sumaMax
99     #sumaZ
100
101    f_return=exp(-r*T)*mean(v_valor);
102 endfunction
```

A.2.37. Aproximación del precio de una Spread Option por QMC y periocización

Nombre función: *spread_option_periodifica.m*

```
1  ##########################################
2  #
3  # Autor    : Ruben Colomina Citoler
4  #
5  # Dependencias : (Funcion) hstar.m
6  #                (Funcion) f_periodifica.m
7  #                (Funcion) f_periodifica_deriv.m
8  #
```

```
##########################################
#
# INPUTS
# glp : Conjuto de puntos del dominio de integracion [0,1)^2
# w1, w2 : pesos de la spread
# rate : tipo de interes
# K precio de ejercicio
# S10, S20: precios iniciales de los subyacentes
# delta1, delta2 : dividendo de los subyacentes
# sigma1, sigma2 : volatilidades de los subyacentes
# ro12 : coeficiente de correlacion entre los subyacentes
# T : periodo de maduracion
# param : parametro de periodizacion comprendido en [1,2,3,4,5,6,7,8]
# OUTPUTS
# Valor de la Spread Option
#
function f_retorno = spread_option_periodifica(glp,w1,w2,rate,K,S10,S20,delta1,delta2,
    sigma1,sigma2,ro12,T,param)
          value=0;
     N=length(glp);
          for i=1:N
                              u1=f_periodifica(glp(i,1),param);
            u2=f_periodifica(glp(i,2),param);
            value_aux=hstar(u1,u2,w1,w2,rate,delta1,delta2,S10,S20,sigma1,sigma2,ro12,T,K
                );
            value_aux=value_aux*f_periodifica_deriv(glp(i,1),param)*f_periodifica_deriv(
                glp(i,2),param);
            value=value+value_aux;
     endfor
     f_retorno=exp(-rate*T)*value/N;
endfunction
```

A.2.38. Integrando del estimador QMC para calcular la Delta de una Spread Option respecto del primer subyacente

Nombre función: *hstar_ delta1.m*

```
##########################################
#
# Autor    : Ruben Colomina Citoler
#
# Dependencias : (Funcion) stdnormal_inv_acklam
#
##########################################
#
# INTEGRANDO QMC PARA DELTA DEL PRIMER SUBYACENTE DE UNA SPREAD OPTIONS
#
# INPUTS
# u1 en [0,1) :componente x del n-esimo termino de la secuencia de baja discrepancia
# u2 en [0,1) :componente y del n-esimo termino de la secuencia de baja discrepancia
# w1 y w2 : pesos de cada subyacente considerados positivos
# w1 y w2 : pesos de cada subyacente considerados positivos
# rate : tipo de interes
# delta1 y delta2 : dividendos para los subyacente 1 y 2
# S01 y S02 precios de los subyacente en el instante 0
# sigma1 y sigma2 son las volatilidades de los subyacentes
```

```
# ro12 en [-1,1] correlacion entre los subyacentes
# T tiempo de madurez de las opciones
# K precio del strike del spread
# OUTPUTS
# Devuelve el valor del integrando para calcular la griega delta1 de una spread option
#
function f_ret=hstar_delta1(u1,u2,w1,w2,rate,delta1,delta2,S10,S20,sigma1,sigma2,ro12,T,K
    )

    #Cambio de variable muiT=log(Si0) + (r-delta(i)-1/2sigma(i)^2)*T
         mu1T=log(S10)+(rate-delta1-1/2*sigma1*sigma1)*T;
    mu2T=log(S20)+(rate-delta2-1/2*sigma2*sigma2)*T;
         #Descomposicion de matriz de covarianzas en descomposicion de Cholesky: A=C*C'
    c11=sigma1*sqrt(T);
    c21=ro12*sigma2*sqrt(T);
    c22=sqrt(1-ro12*ro12)*sigma2*sqrt(T);

         #Evitar sigularidad con u1=0 para la inversa de la normal
    if(u1>0 and (d2+u2*(1-d2))<1)
              h3=c11*stdnormal_inv_acklam(u1)+mu1T;
                        h1=w1*exp(h3);
         g=(log(h1+K)-log(w2)-mu2T-c21*stdnormal_inv_acklam(u1))/c22;
         d2=stdnormal_cdf(g);
                   h5=stdnormal_inv_acklam(d2+u2*(1-d2));
         h4=c21*stdnormal_inv_acklam(u1)+c22*h5+mu2T;
         h2=w2*exp(h4);
         h=h2-h1-K;
                        %derivadas parciales respecto S1
                        parcial_h_S1=(h1/S10)*((h2/(h1+K))*(1-u2)*exp(0.5*(h5*h5-g*g))-1)
                            ;
                        parcial_g_S1=h1/(c22*(h1+K)*S10);
         parcial_d2_S1=(1/sqrt(2*pi))*exp(-g*g/2)*parcial_g_S1;

         f_ret=(1-d2)*parcial_h_S1-parcial_d2_S1*h;
    else
              f_ret=0;
    endif

endfunction
```

A.2.39. Aproximación por QMC de la griega Delta de una Spread Option respecto de su primer subyacente

Nombre función: *spread_option_delta1_perio.m*

```
########################################
#
# Autor    : Ruben Colomina Citoler
#
# Dependencias : (Funcion) hstar_delat1.m
#                (Funcion) f_periodifica.m
#                (Funcion) f_periodifica_deriv.m
#
########################################
#
# SPREAD OPTIONS: CALCULO DELTA primer subyacente
```

```
#
# INPUTS
# u1 en [0,1) :componente x del n-esimo termino de la serie de baja discrepancia
# u2 en [0,1) :componente y ...
# w1 y w2 : pesos de cada subyacente considerados positivos
# rate : tipo de interes
# delta1 y delta2 : dividendos para los subyacente 1 y 2
# S01 y S02 precios de los subyacente en el instante 0
# sigma1 y sigma2 son las volatilidades de los subyacentes
# ro12 en [-1,1] correlacion entre los subyacentes
# T tiempo de madurez de las opciones
# K precio del strike del spread
# param : tipo de periodizacion aplicada
# OUTPUTS
# Valor de la griega delta1 para una Spread Option
function f_retorno = spread_option_delta1_perio(glp,w1,w2,rate,K,S10,S20,delta1,delta2,
    sigma1,sigma2,ro12,T,param)
          value=0;
     N=length(glp);
          for i=1:N
                              u1=f_periodifica(glp(i,1),param);
            u2=f_periodifica(glp(i,2),param);

            value_aux=hstar_delta1(u1,u2,w1,w2,rate,delta1,delta2,S10,S20,sigma1,sigma2,
                ro12,T,K);
            value_aux=value_aux*f_periodifica_deriv(glp(i,1),param)*f_periodifica_deriv(
                glp(i,2),param);
            value=value+value_aux;
     endfor
     f_retorno=exp(-rate*T)*value/N;
endfunction
```

A.2.40. Integrando del estimador QMC para calcular la Delta de una Spread Option respecto del segundo subyacente.

Nombre función: *hstar_ delta2.m*

```
##########################################
#
# Autor        : Ruben Colomina Citoler
#
# Dependencias : (Funcion) stdnormal_inv_acklam
#
##########################################
#
# INTEGRANDO QMC PARA DELTA SEGUNDO SUBYACENTE DE UNA SPREAD OPTIONS
#
# INPUTS
# u1 en [0,1) :componente x del n-esimo termino de la secuencia de baja discrepancia
# u2 en [0,1) :componente y del n-esimo termino de la secuencia de baja discrepancia
# w1 y w2 : pesos de cada subyacente considerados positivos
# w1 y w2 : pesos de cada subyacente considerados positivos
# rate : tipo de interes
# delta1 y delta2 : dividendos para los subyacente 1 y 2
# S01 y S02 precios de los subyacente en el instante 0
# sigma1 y sigma2 son las volatilidades de los subyacentes
```

```
# ro12 en [-1,1] correlacion entre los subyacentes
# T tiempo de madurez de las opciones
# K precio del strike del spread
# OUTPUTS
# Devuelve el valor del integrando para calcular la griega delta2 de una spread option
#
function f_ret=hstar_delta2(u1,u2,w1,w2,rate,delta1,delta2,S10,S20,sigma1,sigma2,ro12,T,K
    )

    #Cambio de variable muiT=log(Si0) + (r-delta(i)-1/2sigma(i)^2)*T
         mu1T=log(S10)+(rate-delta1-1/2*sigma1*sigma1)*T;
    mu2T=log(S20)+(rate-delta2-1/2*sigma2*sigma2)*T;
         #Descomposicion de matriz de covarianzas en descomposicion de Cholesky: A=C*C'
    c11=sigma1*sqrt(T);
    c21=ro12*sigma2*sqrt(T);
    c22=sqrt(1-ro12*ro12)*sigma2*sqrt(T);

    if(u1>0 and (d2+u2*(1-d2))<1)
              h3=c11*stdnormal_inv_acklam(u1)+mu1T;
                        h1=w1*exp(h3);
         g=(log(h1+K)-log(w2)-mu2T-c21*stdnormal_inv_acklam(u1))/c22;
         d2=stdnormal_cdf(g);
                   h5=stdnormal_inv_acklam(d2+u2*(1-d2));
         h4=c21*stdnormal_inv_acklam(u1)+c22*h5+mu2T;
         h2=w2*exp(h4);
         h=h2-h1-K;

                   %derivadas parciales respecto S2
                   parcial_h_S2=(h2/S20)*(1-(1-u2)*exp(0.5*(h5*h5-g*g)));
         parcial_d2_S2=-1/(c22*sqrt(2*pi)*S20)*exp(-g*g/2);

         f_ret=(1-d2)*parcial_h_S2-parcial_d2_S2*h;
    else
              f_ret=0;
    endif

endfunction
```

A.2.41. Aproximación por QMC de la griega Delta de una Spread Option respecto de su segundo subyacente

Nombre función: *spread_option_delta2_perio.m*

```
########################################
#
# Autor    : Ruben Colomina Citoler
#
# Dependencias : (Funcion) hstar_delta2.m
#                (Funcion) f_periodifica.m
#                (Funcion) f_periodifica_deriv.m
#
########################################
#
# SPREAD OPTIONS: CALCULO DELTA segundo subyacente
#
# INPUTS
```

```
# u1 en [0,1) :componente x del n-esimo termino de la serie de baja discrepancia
# u2 en [0,1) :componente y ...
# w1 y w2 : pesos de cada subyacente considerados positivos
# rate : tipo de interes
# delta1 y delta2 : dividendos para los subyacente 1 y 2
# S01 y S02 : precios de los subyacente en el instante 0
# sigma1 y sigma2 : son las volatilidades de los subyacentes
# ro12 en [-1,1] correlacion entre los subyacentes
# T : tiempo de madurez de las opciones
# K : precio del strike del spread
# param : tipo de periodizacion aplicada
# OUTPUTS
# Valor de la griega delta2 para una Spread Option
#
function f_retorno = spread_option_delta2_perio(glp,w1,w2,rate,K,S10,S20,delta1,delta2,
    sigma1,sigma2,ro12,T,param)
          value=0;
     N=length(glp);
          for i=1:N
                              u1=f_periodifica(glp(i,1),param);
            u2=f_periodifica(glp(i,2),param);

            value_aux=hstar_delta2(u1,u2,w1,w2,rate,delta1,delta2,S10,S20,sigma1,sigma2,
                ro12,T,K);
            value_aux=value_aux*f_periodifica_deriv(glp(i,1),param)*f_periodifica_deriv(
                glp(i,2),param);
            value=value+value_aux;
     endfor
     f_retorno=exp(-rate*T)*value/N;
endfunction
```

A.2.42. Integrando del estimador QMC para calcular la Gamma de una Spread Option respecto del primer subyacente

Nombre función: *hstar_gamma1.m*

```
##########################################
#
# Autor    : Ruben Colomina Citoler
#
# Dependencias : (Funcion) stdnormal_inv_acklam
#
##########################################
#
# INTEGRANDO QMC PARA GAMMA DEL PRIMER SUBYACENTE DE UNA SPREAD OPTIONS
#
# INPUTS
# u1 en [0,1) :componente x del n-esimo termino de la secuencia de baja discrepancia
# u2 en [0,1) :componente y del n-esimo termino de la secuencia de baja discrepancia
# w1 y w2 : pesos de cada subyacente considerados positivos
# w1 y w2 : pesos de cada subyacente considerados positivos
# rate : tipo de interes
# delta1 y delta2 : dividendos para los subyacente 1 y 2
# S01 y S02 precios de los subyacente en el instante 0
# sigma1 y sigma2 son las volatilidades de los subyacentes
# ro12 en [-1,1] correlacion entre los subyacentes
```

```
# T tiempo de madurez de las opciones
# K precio del strike del spread
# OUTPUTS
# Devuelve el valor del integrando para calcular la griega gamma1 de una spread option
#
function f_ret=hstar_gamma1(u1,u2,w1,w2,rate,delta1,delta2,S10,S20,sigma1,sigma2,ro12,T,K
    )

    #Cambio de variable muiT=log(Si0) + (r-delta(i)-1/2sigma(i)^2)*T
         mu1T = log( S10 ) + ( rate - delta1 - 1/2 * sigma1 * sigma1 ) * T;
    mu2T = log( S20 ) + ( rate - delta2 - 1/2 * sigma2 * sigma2 ) * T;
         #Descomposicion de matriz de covarianzas en descomposicion de Cholesky: A=C*C'
    c11 = sigma1 * sqrt(T);
    c21 = ro12 * sigma2 * sqrt(T);
    c22 = sqrt( 1 - ro12 * ro12 ) * sigma2 * sqrt(T);

    if(u1>0 and (d2+u2*(1-d2))<1)
              h3 = c11 * stdnormal_inv_acklam( u1 ) + mu1T;
                        h1 = w1 * exp( h3 );
         g  = ( log ( h1 + K ) - log( w2 ) - mu2T - c21 * stdnormal_inv_acklam( u1 ) ) /
             c22;
         d2 = stdnormal_cdf( g );

                    h5 = stdnormal_inv_acklam( d2 + u2 * ( 1 - d2 ) );
         h4 = c21 * stdnormal_inv_acklam( u1 ) + c22 * h5 + mu2T;
         h2 = w2 * exp(h4);
         h  = h2 - h1 - K;
                         %calculos intermedios
                         a1 = ( h1 / ( ( h1 + K ) * S10 ) );
                         a2 = h5 * h5 - g * g;
                         %derivadas parciales respecto S1
                         dp_h_S1 = ( h1 / S10 ) * ( ( h2 / ( h1 + K ) ) * ( 1 - u2 ) * exp
                             ( 0.5 * a2 ) - 1 );
                         dp_g_S1 = a1 / c22;
         dp_d2_S1= ( 1 / sqrt( 2 * pi ) ) * exp( - g * g / 2 ) * dp_g_S1;
                         %derivadas parciales segundas respecto S1
                         dp_2_h_S1 = h2 * a1 * a1 * ( 1 - u2 ) * exp( 0.5 * a2 ) * ( ( 1 -
                              u2 ) * exp( 0.5 * a2 ) * ( h5 / c22 + 1 ) -g / c22 - 1 );
                         dp_2_g_S1 = - a1 * a1 / c22;
                         dp_2_d2_S1= dp_d2_S1 * ( ( dp_2_g_S1 / dp_g_S1 ) - dp_g_S1 * g );
                         %resultado final para gamma1
         f_ret= ( 1 - d2 ) * dp_2_h_S1 - 2 * dp_h_S1 * dp_d2_S1 - dp_2_d2_S1 * h;
                         %else
                         %        f_ret=0;
         %endif
    else
              f_ret=0;
    endif

endfunction
```

A.2.43. Aproximación por QMC de la griega Gamma de una Spread Option respecto de su primer subyacente

Nombre función: *spread_option_gamma1_perio.m*

```
##########################################
#
# Autor    : Ruben Colomina Citoler
#
# Dependencias : (Funcion) hstar_delat1.m
#                (Funcion) f_periodifica.m
#                (Funcion) f_periodifica_deriv.m
#
##########################################
#
# SPREAD OPTION: CALCULO GAMMA DEL PRIMER SUBYACENTE
# INPUTS
# u1 en [0,1) :componente x del n-esimo termino de la serie de baja discrepancia
# u2 en [0,1) :componente y ...
# w1 y w2 : pesos de cada subyacente considerados positivos
# rate : tipo de interes
# delta1 y delta2 : dividendos para los subyacente 1 y 2
# S01 y S02 precios de los subyacente en el instante 0
# sigma1 y sigma2 son las volatilidades de los subyacentes
# ro12 en [-1,1] correlacion entre los subyacentes
# T tiempo de madurez de las opciones
# K precio del strike del spread
# param : tipo de periodizacion aplicada
# OUTPUTS
# Valor de la griega gamma1 para una Spread Option
#
function f_retorno = spread_option_gamma1_perio(glp,w1,w2,rate,K,S10,S20,delta1,delta2,
    sigma1,sigma2,ro12,T,param)
          value=0;
     N=length(glp);
          for i=1:N
                              u1=f_periodifica(glp(i,1),param);
            u2=f_periodifica(glp(i,2),param);

            value_aux=hstar_gamma1(u1,u2,w1,w2,rate,delta1,delta2,S10,S20,sigma1,sigma2,
                ro12,T,K);
            value_aux=value_aux*f_periodifica_deriv(glp(i,1),param)*f_periodifica_deriv(
                glp(i,2),param);
            value=value+value_aux;
     endfor
     f_retorno=exp(-rate*T)*value/N;
endfunction
```

A.2.44. Integrando del estimador QMC para calcular la Gamma de una Spread Option respecto del segundo subyacente.

Nombre función: *hstar_ gamma2.m*

```
##########################################
#
# Autor    : Ruben Colomina Citoler
#
# Dependencias : (Funcion) stdnormal_inv_acklam
#
##########################################
#
```

```
# INTEGRANDO QMC PARA GAMMA DEL SEGUNDO SUBYACENTE DE UNA SPREAD OPTIONS
#
# INPUTS
# u1 en [0,1) :componente x del n-esimo termino de la secuencia de baja discrepancia
# u2 en [0,1) :componente y del n-esimo termino de la secuencia de baja discrepancia
# w1 y w2 : pesos de cada subyacente considerados positivos
# w1 y w2 : pesos de cada subyacente considerados positivos
# rate : tipo de interes
# delta1 y delta2 : dividendos para los subyacente 1 y 2
# S01 y S02 precios de los subyacente en el instante 0
# sigma1 y sigma2 son las volatilidades de los subyacentes
# ro12 en [-1,1] correlacion entre los subyacentes
# T tiempo de madurez de las opciones
# K precio del strike del spread
# OUTPUTS
# Devuelve el valor del integrando para calcular la griega gamma2 de una spread option
#
function f_ret=hstar_gamma2(u1,u2,w1,w2,rate,delta1,delta2,S10,S20,sigma1,sigma2,ro12,T,K
    )

    #Cambio de variable muiT = log(Si0) + (r-delta(i)-1/2sigma(i)^2)*T
         mu1T = log(S10) + ( rate - delta1 - 1/2 * sigma1 * sigma1 )* T;
    mu2T = log(S20) + ( rate - delta2 - 1/2 * sigma2 * sigma2 )* T;

         #Matriz de covarianzas en descomposicion de Cholesky: A=C*C'
    c11 = sigma1 * sqrt(T);
    c21 = ro12 * sigma2 * sqrt(T);
    c22 = sqrt(1 - ro12 * ro12) * sigma2 * sqrt(T);

    if(u1>0 and (d2+u2*(1-d2))<1)
              h3 = c11 * stdnormal_inv_acklam( u1 ) + mu1T;
                       h1 = w1 * exp( h3 );
         g   = ( log( h1 + K ) - log(w2) - mu2T - c21 * stdnormal_inv_acklam( u1 ) ) / c22
             ;
         d2 = stdnormal_cdf( g );

                   h5 = stdnormal_inv_acklam( d2 + u2 * ( 1 - d2 ) );
         h4 = c21 * stdnormal_inv_acklam( u1 ) + c22 * h5 + mu2T;
         h2 = w2 * exp( h4 );
         h  = h2 - h1 - K;
                        %calculos intermedios
                        a2 = h5 * h5 - g * g;
                   %derivadas parciales primeras respecto S2
                        dp_h5_S2 = - 1 / ( c22 * S20 ) * ( 1 - u2 ) * exp( 0.5 * a2 );
                   dp_h_S2  = h2 * ( c22 * dp_h5_S2 + 1 / S20 );
         dp_d2_S2 = - 1 / ( c22 * sqrt ( 2 * pi ) * S20 ) * exp( - 0.5 * g * g );
                        %derivadas parciales segundas respecto S2
                        dp_2_d2_S2 = 1 / ( c22 * sqrt(2 * pi) * S20 * S20 ) * exp( - 0.5
                            * g * g ) * ( 1 - g / c22 );
                        dp_2_h5_S2 = h5 * dp_h5_S2 * dp_h5_S2 + dp_h5_S2 / dp_d2_S2 *
                            dp_2_d2_S2;
                        dp_2_h_S2  = h2 * ( c22 * dp_2_h5_S2 - 1 / ( S20 * S20 ) ) + (
                            c22 * dp_h5_S2 + 1 / S20 ) * dp_h_S2;
                        %resultado final para gamma2
         f_ret= ( 1 - d2 ) * dp_2_h_S2 - 2 * dp_h_S2 * dp_d2_S2 - dp_2_d2_S2 * h;
                        %else
                        %        f_ret=0;
         %endif
    else
              f_ret=0;
    endif
```

```
65
66 endfunction
```

A.2.45. Aproximación por QMC de la griega Gamma de una Spread Option respecto de su segundo subyacente

Nombre función: *spread_option_gamma2_perio.m*

```
1  ##########################################
2  #
3  # Autor    : Ruben Colomina Citoler
4  #
5  # Dependencias : (Funcion) hstar_delat1.m
6  #                (Funcion) f_periodifica.m
7  #                (Funcion) f_periodifica_deriv.m
8  #
9  ##########################################
10 #
11 # SPREAD OPTIONS: CALCULO GAMMA SEGUNDO SUBYACENTE
12 # INPUTS
13 # u1 en [0,1) :componente x del n-esimo termino de la serie de baja discrepancia
14 # u2 en [0,1) :componente y del n-esimo termino de la serie de baja discrepancia
15 # w1 y w2 : pesos de cada subyacente considerados positivos
16 # rate : tipo de interes
17 # delta1 y delta2 : dividendos para los subyacente 1 y 2
18 # S01 y S02 precios de los subyacente en el instante 0
19 # sigma1 y sigma2 son las volatilidades de los subyacentes
20 # ro12 en [-1,1] correlacion entre los subyacentes
21 # T tiempo de madurez de las opciones
22 # K precio del strike del spread
23 # param : tipo de periodizacion aplicada
24 # OUTPUTS
25 # Valor de la griega Delta2 para una Spread Option
26 #
27 function f_retorno = spread_option_gamma2_perio(glp,w1,w2,rate,K,S10,S20,delta1,delta2,
       sigma1,sigma2,ro12,T,param)
28           value=0;
29        N=length(glp);
30           for i=1:N
31                                  u1 = f_periodifica(glp(i,1),param);
32             u2 = f_periodifica(glp(i,2),param);
33
34             value_aux = hstar_gamma2(u1,u2,w1,w2,rate,delta1,delta2,S10,S20,sigma1,sigma2
                   ,ro12,T,K);
35             value_aux = value_aux * f_periodifica_deriv(glp(i,1),param) *
                   f_periodifica_deriv(glp(i,2),param);
36             value = value + value_aux;
37        endfor
38        f_retorno = exp( - rate * T) * value / N;
39 endfunction
```

Bibliografía

[1] P.P.Boyle,Y.Lai,K.S.Tan: Pricing options using lattice methods (2010), North American Actuarial Journal; Jul 2005; 9,3;2000

[2] Mark Broadie, Paul Glasserman: Estimating Security Price Derivatives Using Simulation (1996)

[3] Seymour Haber: Parameters for Integrating Periodic Functions of Several Variables (1983)

[4] Ricardo Vélez Ibarrola: Introducción al Movimiento Browniano

[5] Josef Dick: Digital Nets and Sequences (2010)

[6] Peter Zinterhof, Peter Schmitzberger: Integrate: A Kernel of FORTRAN and C package PANNUM for High Dimensional Number Theoretic Quadrature

[7] Demien Lamberton, Bernard Lapeyre: Introduction to Stochastic Calculus Applied to Finance (2008)

[8] Gianluca Fuasi, Andrea Roncoroni Implementing Models in Quantitative Finance: Methods and Cases

[9] Jasper Schmidt Hansen: GNU Octave Beninner's Gunide

[10] José María Valiente Cifuentes: Manual de iniciación a GNU Octave

[11] Jonh W.Eaton: GNU Octave Free your Numbers