UNIVERSIDAD
DE GRANADA

Escuela Internacional de Posgrado

Máster en Estadística Aplicada

TRABAJO DE FIN DE MÁSTER

Simulación de difusiones clásicas y anómalas

Presentado por:
Beatriz Gutiérrez Trillo

Tutor:
María Dolores Ruiz Medina
Departamento de Estadística e Investigación Operativa

Curso académico 2022-2023

Simulación de difusiones clásicas y anómalas

Beatriz Gutiérrez Trillo

Beatriz Gutiérrez Trillo *Simulación de difusiones clásicas y anómalas*.
Trabajo de fin de Máster. Curso académico 2022-2023.

Responsable de tutorización

María Dolores Ruiz Medina
Departamento de Estadística e Investigación Operativa

Máster en Estadística Aplicada

Escuela Internacional de Posgrado

Universidad de Granada

Declaración de originalidad

D./Dña. Beatriz Gutiérrez Trillo

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Máster (TFM), correspondiente al curso académico 2022-2023, es original, entendida esta, en el sentido de que no ha utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 4 de febrero de 2023

Fdo: Beatriz Gutiérrez Trillo

Índice general

Resumen

Los procesos de difusión son ampliamente utilizados fuera del ámbito de las matemáticas y la estadística. Su importancia en campos como la medicina o la economía hace patente la necesidad de estudiar dichos procesos desde una perspectiva probabilística, estadística y numérica o computacional, mediante la simulación de sus trayectorias. Este Trabajo de Fin de Máster proporciona una breve introducción a los conceptos y elementos básicos que intervienen en la teoría de procesos estocásticos dedicada al estudio de los procesos de difusión. Se introduce, en primer lugar, el concepto clásico de proceso de difusión, contemplando su caracterización a partir de los momentos infinitesimales, así como mediante la correspondiente teoría de ecuaciones diferenciales estocásticas. En particular, nos centraremos en este Trabajo Fin de Máster en los métodos usuales para la simulación de dichos procesos, con el fin de mostrar el comportamiento de sus trayectorias a pequeña y gran escala, mediante representaciones gráficas de las trayectorias generadas. Adicionalmente, se introducen clases más amplias de procesos de difusión, tales como los procesos de difusión anómala, que se hayan caracterizados mediante ecuaciones pseudodiferenciales fraccionarias estocásticas. En concreto, nos centraremos en el análisis de dichas difusiones sobre la esfera, resumiendo algunos de los resultados más recientes en la literatura científica sobre este tipo de procesos estocásticos. Implementaremos asimismo la simulación de sus trayectorias sobre la esfera a lo largo del tiempo. Finalmente, se proporciona un breve resumen sobre las líneas de investigación abiertas en este campo y áreas relacionadas, con especial enfásis en las aplicaciones sobre modelización mediante funciones aleatorias en otras áreas aplicadas. Se resumen, a continuación, por capítulos, los principales contenidos de este Trabajo de Fin de Máster.

El Capítulo 1 introduce las definiciones y elementos básicos de la teoría de procesos estocásticos, en particular, en el contexto de procesos de Markov con espacio de parámetros y espacio de estados continuos. Se proporciona asimismo su caracterización mediante las probabilidades de transición. Dentro de dicha familia se encuentran los procesos de difusión con trayectorias continuas casi seguramente, caracterizados por los dos primeros momentos infinitesimales. En este Capítulo, adicionalmente, se muestra también su definición como solución de ciertos modelos de ecuaciones diferenciales estocásticas no lineales, cuyas innovaciones aleatorias se construyen a partir del movimiento Browniano vectorial. Como casos especiales se consideran los procesos de difusión más básicos y ampliamente aplicados en otras áreas, tales como el proceso Browniano, el proceso Browniano geométrico y el proceso de Ornstein-Uhlenbeck. Para la introducción de los modelos de difusión con saltos, se define también el proceso de Poisson compuesto. Adicionalmente, se estudia la relación entre el proceso Browniano y el proceso Browniano geométrico, así como, en general, se analizan las principales propiedades y aplicaciones de estos procesos de difusión. Para todas estas familias de procesos y, en particular, para los ejemplos mencionados, se implementarán en el Capítulo 2 algunos de los métodos más frecuentes de simulación.

El Capítulo 2 se centra en la simulación de las difusiones clásicas. Concretamente, se proporciona en código MatLab el algoritmo de simulación de los procesos de difusión introducidos en el Capítulo 1, haciendo uso de funciones propias de este lenguaje de programación. Específicamente, se introducen los esquemas de aproximación numérica más frecuentes que

intervienen en el diseño de estos algoritmos de simulación, tales como la aproximación de Euler, el Esquema de Milstein y el método basado en los incrementos, así como algunas técnicas alternativas basadas en la expansión de Karhunen-Lóeve. Finalmente, se extienden los algoritmos de simulación previamente implementados al contexto de los procesos de difusión con saltos. Mediante dichos algoritmos se generan las trayectorias de esta familia extendida de difusiones. De igual forma, se obtienen previamente las trayectorias generadas para los ejemplos de procesos de difusión clásicos analizados. Su representación gráfica permite replicar el comportamiento de sistemas o modelos aleatorios dirigidos por dichos procesos, tal es el caso de los modelos de mercado. Cabe destacar que se ha realizado una descripción exhaustiva y detallada de cada uno de los elementos que intervienen en la sintaxis del lenguaje MatLab, utilizado en la implementación de los diferentes pasos de los algoritmos de simulación referidos.

El Capítulo 3 introduce los elementos básicos que intervienen en la caracterización de campos aleatorios isotrópicos sobre la esfera, así como en el análisis de sus propiedades. Concretamente, se centra en la derivación de sus estructuras de correlación lineal, contemplando el caso de campos aleatorios espacio temporales. En particular, se analizan dos tipos de ecuaciones estocásticas fraccionarias sobre la esfera. Se describen además subordinadores y operadores fraccionarios que intervienen en la derivación y demostración de tres teoremas, que reflejan dicha caracterización y propiedades. En este capítulo se simulan, a través de código MatLab, las trayectorias de la solución obtenida para los modelos de ecuaciones de evolución pseudodiferenciales fraccionarias estudiados sobre la esfera, que reflejan la evolución sobre dicha variedad de los comportamientos anómalos modelizados. En particular, se prueba, en el Teorema 1, la propiedad de memoria larga de la familia de procesos de difusión anómala considerada. Como aportación de este Trabajo de Fin de Máster, en el Capítulo 3 se introduce y genera una extensión de la familia de modelos pseudodiferenciales fraccionarios sobre la esfera, anteriormente analizada, al contexto de campos aleatorios esféricos multifraccionarios, implementándose, de nuevo, dichas simulaciones en lenguaje MatLab. La evolución temporal de los patrones espaciales generados sobre la esfera, en ambos modelos (fraccionarios y multifraccionarios), ilustran la persistencia en el tiempo de los patrones esféricos observados. Por tanto, se puede apreciar la propiedad de memoria larga o persistencia temporal del comportamiento sobre la esfera de estas familias de procesos. Destaca, en particular, el caso de procesos multifraccionarios sobre la esfera, donde interaccionan por escalas esféricas los diferentes niveles de persistencia en el tiempo. Este aspecto, ilustrado mediante los gráficos resultantes de las simulaciones implementadas, es especialmente interesante y novedoso, pues permite modelizar de forma más flexible el rango de dependencia temporal dependiendo de la escala espacial, en nuestro caso, dependiendo de la escala esférica.

Finalmente, el Capítulo 4 expone algunas aplicaciones en distintos ámbitos, como Astrofísica y Medio Ambiente, poniendo de manifiesto la importancia de los procesos de difusión en su concepción clásica (caracterizados mediante operadores locales), así como en su concepción más avanzada y flexible, como es el caso de los modelos no locales pseudodiferenciales fraccionarios estudiados sobre la esfera. En particular, se subraya la relevancia del proceso movimiento Browniano, así como de sus versiones extendidas (no locales), tales como el movimiento Browniano fraccionario y multifraccionario, en multitud de investigaciones desarrolladas en diferentes campos científicos, entre los que se encuentran, por ejemplo, los ya mencionados (Astrofísica y Medio Ambiente), así como Geofísica, Medicina y Finanzas.

1 Teoría básica de procesos de difusión

La presencia de correlaciones en el tiempo en las componentes aleatorias que definen los procesos aleatorios supone una mayor dificultad en el diseño de métodos de generación, por lo que se suelen realizar simplificaciones sobre la estructura de dichos procesos. Algunos ejemplos de estas simplificaciones son considerar una discretización del parámetro temporal o considerar estructuras de dependencia débil, como las asociadas a modelos o procesos de Markov (como las Cadenas de Markov).

Algunos procesos aleatorios elementales son los recorridos aleatorios, el movimiento Browniano y el proceso de Poisson. Bajo ciertas condiciones, el movimiento Browniano admite una definición como proceso límite de un recorrido aleatorio. Este proceso, al igual que el proceso de Poisson, posee incrementos independientes, lo que los hace muy interesante a su fácil implementación e interpretación.

Un recorrido aleatorio es una formalización matemática de una trayectoria definida mediante una secuencia de saltos aleatorios. Los recorridos aleatorios se construyen a partir de secuencias de variables aleatorias independientes y constituyen un modelo de proceso estocástico ampliamente utilizado en el análisis y representación de sistemas, tales como la trayectoria descrita por una molécula en un líquido o en un gas o el precio de un stock con fluctuaciones, en diversas áreas aplicadas como Ciencias de la Computación, Física, Ecología y Economía.

Por otra parte, los recorridos aleatorios permiten aproximar asintóticamente diversos procesos, como el movimiento Browniano o procesos tipo Lévy. Además, se hallan relacionados con los procesos de difusión y constituyen una herramienta fundamental en el estudio de procesos de Markov.

Existe una gran variedad de modelos de recorridos aleatorios. De hecho, un recorrido aleatorio puede asociarse a un grafo, una línea, un plano, o bien, a espacios de dimensión superior. En el caso más sencillo, un recorrido aleatorio se suele definir sobre los números enteros.

1.1. Definición y elementos básicos en la teoría de procesos

1.1.1. Recorrido aleatorio simple

Sea S_n, $n \in \mathbb{N}$, un recorrido aleatorio simple, siendo $S_0 = 0$, es decir, que comienza en cero. En cada paso, el recorrido realiza un salto unitario ± 1 (ascendente o descendente) con igual probabilidad. Formalmente, se construye a partir de una secuencia de variables aleatorias independientes $Z_1, Z_2, \ldots, Z_k, \ldots$, cada una de las cuales toma el valor 1 con probabilidad $1/2$ y el valor -1 con probabilidad $1/2$. Se considera entonces la variable aleatoria suma S_n definida mediante la ecuación

$$S_n = \sum_{j=1}^{n} Z_j.$$

La secuencia $\{S_n,\ n \in \mathbb{N}\}$ recibe el nombre de recorrido aleatorio simple. Este concepto

admite una formulación general, en términos de una probabilidad $p \in (0,1)$ de tomar el valor 1 (ascenso) y una probabilidad $1-p$ de tomar el valor -1 (descenso) para cada una de las variables aleatorias Z_i, $i \in \mathbb{N}$, de la secuencia involucrada en la definición de un recorrido aleatorio $\{S_n,\ n \in \mathbb{N}\}$. Se define de forma análoga el recorrido aleatorio simple considerando, en lugar de saltos unitarios, saltos (ascendentes o descendentes) de tamaño aleatorio definidos mediante una distribución de probabilidad centrada.

1.1.2. Proceso de Markov

La propiedad básica que caracteriza a los procesos de Markov es el análogo probabilístico a una propiedad muy frecuente en los sistemas físicos. Si se conoce el estado de un sistema en un determinado instante, éste determina el comportamiento futuro del sistema, independientemente del comportamiento anterior. Esto es lo que se conoce como *Principio de independencia entre el pasado y el futuro cuando el presente es conocido* [16].

Sea $\{X_t\}_{t\geq 0}$ un proceso estocástico en tiempo continuo definido sobre el espacio probabilístico $(\Omega, \mathscr{A}, P)$ y con espacio de estados $(E, \mathscr{B}_E)$, $E \subseteq \mathbb{R}$. Se dice que $\{X_t\}_{t\geq 0}$ es un proceso de Markov, si y solamente si

$$\forall s < t,\ \forall B \in \mathscr{B}_E,\ P[X_t \in B | X_u,\ u \leq s] = P[X_t \in B | X_s],\ c.s.[20]$$

Todo proceso estocástico con incrementos independientes es un proceso de Markov [11]. Para $t_0,\ T \in \mathbb{R}$, la función

$$(s,t) \in [t_0,T] \times [t_0,T],\ \ s \leq t,\ \ x \in E,\ \ A \in \mathscr{B}_E$$

$$p(s,x,t,A) := P(X_t \in A | X_s = x) \in (0,1)$$

satisface las siguiente propiedades [11]:

1) $\forall (s,t) \in [t_0,T] \times [t_0,T]$ con $s \leq t$, y $\forall A \in \mathscr{B}_E$ fijos, la función

$$\begin{array}{ccc} (E, \mathscr{B}_E) & \longrightarrow & (\mathbb{R}, \mathscr{B}_{\mathbb{R}}) \\ x \in E & \mapsto & p(s,x,t,A) \end{array}$$

es medible.

2) $\forall (s,t) \in [t_0,T] \times [t_0,T]$, con $s \leq t$ y $\forall x \in E$, la función $A \in \mathscr{B}_E \mapsto p(s,x,t,A)$ es una medida de probabilidad de una variable aleatoria degenerada en $\mathscr{B}_E$ tal que

$$p(s,x,s,A) = \left\{ \begin{array}{ccc} 1 & si & x \in A \\ 0 & si & x \notin A. \end{array} \right.$$

3) Satisface la ecuación de Chapman-Kolmogorov, es decir, $\forall x \in E$, $\forall (s,r,t) \in [t_0,T] \times [t_0,T] \times [t_0,T]$, $s \leq r \leq t$ y $\forall A \in \mathscr{B}_E$

$$p(s,x,t,A) = P(X_t \in A | X_s = x) = \int_{\mathbb{R}} P(X_r \in \{y\} | X_s = x) P(X_t \in A | X_r = y) dy =$$

$$= \int_{\mathbb{R}} p(s,x,r,\{y\}) p(r,y,t,A) dy\ \ c.s.$$

Además,$P(X_t \in A | X_s = x) = \int_A p(s,x,t,\{y\})dy$ [17].

Cualquier función no negativa $p(s,x,t,A)$ definida para $t_0 \leq s \leq t \leq T$, $x \in E$, $A \in \mathscr{B}_E$ satisfaciendo estas tres propiedades es una función de probabilidad de transición de un proceso de Markov [11].

La función de probabilidad de transición $p(s,x,t,A)$ expresa la probabilidad de que una partícula que ha partido del punto x en el instante s esté en el conjunto A en el tiempo t. En 1931, Kolmogorov demostró que, bajo supuestos amplios, esta probabilidad se puede obtener a partir de una determinada ecuación diferencial parabólica, lo cual permitió responder varias preguntas sobre el movimiento Browniano utilizando la teoría de ecuaciones diferenciales [16].

1.1.3. Procesos de difusión

Muchos fenómenos están modelados por procesos de difusión unidimensionales. Un proceso de difusión es un proceso de Markov en tiempo continuo con función de probabilidad de transición $p(s,x,t,A)$ que verifica:

- Tiene trayectorias continuas (casi seguramente).
- $lim_{h \downarrow 0} \ \frac{1}{h} \int_{|x-y|>\varepsilon} p(t,x,t+h,\{y\})dy = 0, \forall \varepsilon > 0, \forall t \geq 0, \forall x \in \mathbb{R}$
  Es decir, en intervalos pequeños de tiempo son improbables saltos grandes, lo que significa que el proceso casi seguramente tiene trayectorias muestrales continuas [17].
- Existen $a(t,x)$, $b(t,x)$ tales que $\forall \varepsilon > 0, \forall t \geq 0, \forall x \in \mathbb{R}$

$$lim_{h \downarrow 0} \ \frac{1}{h} \int_{|x-y|<\varepsilon} (y-x) p(t,x,t+h,\{y\})dy = a(t,x)$$

$$lim_{h \downarrow 0} \ \frac{1}{h} \int_{|x-y|<\varepsilon} (y-x)^2 p(t,x,t+h,\{y\})dy = b(t,x)$$

  donde $a(t,x)$ es el *coeficiente de deriva*, que describe la tasa instantánea de cambio de la esperanza condicionada de los incrementos, y $b(t,x)$ es el *coeficiente de difusión* del proceso, que refleja la tasa de cambio instantánea de la covarianza condicionada de los incrementos [17].

Las funciones $a(t,x)$ y $b(t,x)$ no se corresponden exactamente con los momentos infinitesimales sino que son los momentos truncados de los incrementos condicionados. En la definición de proceso de difusión aparecen sólo los dos primeros momentos truncados ya que, en general, los de orden superior son nulos [24].

La definición anterior resulta complicada, en determinadas condiciones, para comprobar si un determinado proceso es o no un proceso de difusión. Por ello, se utilizan las siguientes condiciones suficientes para que un proceso sea de difusión [30]:

- Tiene trayectorias continuas (casi seguramente).
- $\exists \delta > 0$ tal que $\forall x \in \mathbb{R}$, $lim_{h \downarrow 0} \ \frac{1}{h} \int |x-y|^{2+\delta} p(t,x,t+h,\{y\})dy = 0, \ \forall t \geq 0.$
- Existen $a(t,x)$, $b(t,x)$ tales que $\forall t \geq 0, \forall x \in \mathbb{R}$

$$lim_{h \downarrow 0} \ \frac{1}{h} \int (y-x) p(t,x,t+h,\{y\})dy = a(t,x)$$

$$lim_{h\downarrow 0}\ \frac{1}{h}\int (y-x)^2 p(t,x,t+h,\{y\})dy = b(t,x)$$

Con estas condiciones no se puede asegurar que si existen los momentos infinitesimales de orden superior a dos estos sean nulos, como ocurre con los momentos infinitesimales truncados. Pero ahora sí que coinciden los momentos infinitesimales y los momentos infinitesimales truncados hasta orden dos, por lo que podemos denominar a las funciones $a(t,x)$ y $b(t,x)$ momentos infinitesimales [24]. De esta forma, las funciones $a(t,x)$ y $b(t,x)$ se denominan media y varianza infinitesimal, respectivamente [24].

A continuación, consideramos ecuaciones diferenciales estocásticas de la forma [30]

$$dX_t = c(t,X_t)dt + \sigma(t,X_t)dB(t),\ t \geq 0, \tag{1.1}$$

donde $c(x)$ es una tasa superior dependiente del estado, $B(t)$ es un movimiento Browniano estándar y $\sigma(X_t)dB(t)$ introduce una fuente de volatilidad en la acumulación de tasas [30].

Las soluciones de estas ecuaciones diferenciales estocásticas que satisfacen las propiedades de Markov y tienen trayectorias continuas son los procesos de difusión, con coeficiente de deriva $c(t,x)$ y con coeficiente de difusión $\sigma(t,x)$ [30].

Para cualesquiera $x,y \in \mathbb{R}$ y $t \geq 0$, existe una constante $K_1 < \infty$ tal que las funciones de deriva y de difusión verifican la condición de Lipschitz

$$|c(t,x)-c(t,y)| + |\sigma(t,x)-\sigma(t,y)| < K_1|x-y|; \quad [19]$$

y existe una constante $K_2 < \infty$ de forma que verifican la siguiente restricción sobre el crecimiento

$$|c(t,x)| + |\sigma(t,x)| < K_2(1+|x|). \quad [19]$$

1.2. El proceso Browniano

La universalidad del movimiento Browniano, también llamado proceso Browniano o proceso de Wiener, dentro de la teoría de procesos estocásticos, es similar a la de la distribución normal dentro de la teoría de variables aleatorias. De hecho, el proceso movimiento Browniano se puede interpretar como una versión continua de un recorrido aleatorio. Aunque inicialmente este proceso surge como modelo para el movimiento de partículas suspendidas en un líquido o un gas, posteriormente ha sido utilizado en la representación de procesos aleatorios en diversos campos aplicados como las Finanzas, Ingeniería, Biología, Geofísica, Medio Ambiente, etc [11]. Además, juega un papel importante en Matemáticas puras, donde, por ejemplo, el proceso de Wiener dio lugar al estudio de las martingalas de tiempo continuo [38].

El proceso movimiento Browniano estándar se define como un proceso estocástico $\{B_t,\ t \geq 0\}$, satisfaciendo [18][32]:

- $B_0 = 0$.
- B_t es continuo casi seguramente.
- B_t es un proceso que posee incrementos independientes con $B_{s+t} - B_s \sim N(0,t)$, para cualquier $s \geq 0$, $t > 0$.

Una consecuencia inmediata es que $B_t \sim N(0,t)$ para $t \geq 0$.

De esta forma, todo movimiento Browniano es homogéneo, es una martingala [32] y, además [11] [18]:

- $E[B_t] = 0, \forall t \in \mathbb{R}^+$.
- $Var[B_t] = t$. [38][18]
- $Cov(B_t, B_s) = min\{t, s\}$.

El movimiento Browniano estándar es un proceso de difusión con momentos infinitesimales $a(t,x) = 0$ y $b(t,x) = 1$ [24].

Sean μ y $\sigma > 0$ constantes, entonces X_t es un proceso de difusión Browniano con deriva μ y coeficiente de difusión σ^2 si $\frac{X_t - \mu t}{\sigma}$ es un proceso de difusión Browniano estándar [18]. Es decir, si partimos proceso de difusión Browniano estándar B_t entonces X_t se construye como sigue

$$X_t = \mu t + \sigma B_t.$$

Un proceso estocástico $\{X_t,\ t \geq 0\}$ es un proceso movimiento Browniano si X_t es solución de la ecuación diferencial estocástica

$$dX_t = \mu(t)dt + \sigma(t)dW_t \tag{1.2}$$

donde W_t es un proceso Browniano estándar y $\mu(t)$ y $\sigma(t) > 0$ son variables deterministas que varían en el tiempo, siendo μ la deriva y σ^2 el coeficiente de difusión del proceso de difusión Browniano X_t, que construido de esta forma tiene trayectorias continuas e incrementos independientes [18].

1.3. El proceso Browniano geométrico

Un proceso estocástico $\{X_t, t \geq 0\}$ es un proceso Browniano geométrico si $\{log(X_t)\}$ es un proceso Browniano [2]. Esto implica que un proceso Browniano geométrico es la exponencial de un proceso Browniano. Así, todos los métodos de simulación aplicables al proceso Browniano también lo son al proceso Browniano geométrico mediante una transformación exponencial [18].

Un movimiento Browniano geométrico X_t es solución de la ecuación diferencial estocástica [30][11]

$$dX_t = \mu X_t dt + \sigma X_t dB_t \tag{1.3}$$

donde B_t es un proceso Browniano. Aplicando la fórmula de Itô a la ecuación (1.3), se tiene que su solución es el proceso Browniano geométrico clásico [18][2]

$$X_t = X_0 exp\left(\left[\mu - \frac{\sigma^2}{2}\right]t + \sigma B_t\right),$$

donde $a(x) = \mu x$ es el coeficiente de deriva y $b(x) = \sigma x$ es el coeficiente de difusión [2].

La esperanza y la varianza condicionadas de este proceso son [17]

$$E[X_t / X_s = x] = xe^{\mu(t-s)} \qquad Var(X_t / X_s = x) = x^2 e^{2\mu(t-s)}(e^{\sigma^2(t-s)} - 1).$$

Una propiedad importante del proceso Browniano geométrico es que los incrementos proporcionales $\frac{X_{i+1} - X_i}{X_i}$, con $i \geq 1$, son independientes, en lugar de serlo los incrementos

$X_{i+1} - X_i$ [18]. En matemáticas financieras, el proceso Browniano geométrico es el modelo más usado ya que la función exponencial toma únicamente valores positivos, a diferencia del modelo Browniano que puede tomar valores negativos, lo cual no es útil en finanzas, por ejemplo, en el modelado del precio de una acción o de los activos, con tasa de interés μ y volatilidad σ [18][17].

1.4. Proceso de Poisson

Mientras que el proceso Browniano es un proceso continuo en el espacio y en el tiempo, existen procesos que son continuos en el tiempo pero no en el espacio, admitiendo saltos. Los procesos más simples de este tipo son los procesos de recuento, y un caso especial de ellos es el proceso de Poisson, que es también un modelo universal [11].

El proceso de Poisson es un ejemplo clásico de Cadena de Markov en tiempo continuo. Su distribución de probabilidad y propiedades se caracterizan mediante un único parámetro λ, que representa la tasa de ocurrencia de los sucesos aleatorios que contabiliza, lo que lo convierte en un modelo idóneo para la inferencia, que se centra, por tanto, en la estimación puntual y por intervalos de confianza del parámetro λ, así como en el planteamiento de contrastes de hipótesis sobre dicho parámetro y funciones del mismo [11].

El proceso de Poisson se descubrió de forma independiente y repetida en varios entornos, incluidos experimentos sobre decaimiento radiactivo, llegadas de llamadas telefónicas y matemáticas de seguros [36][2]. Posteriormente, este proceso ha sido ampliamente utilizado como modelo para representar el número de fallos de un sistema en fiabilidad, el número de nacimientos o muertes en análisis de supervivencia, la distribución de excesos sobre umbrales en medicina, la ocurrencia de incendios forestales en medio ambiente, etc.

Matemáticamente, el proceso de Poisson $\{N(t),\ t \geq 0\}$ con parámetro λ (intensidad o tasa del proceso), se define como un proceso de recuento, es decir, $N(t) - N(0) = N(t)$ contabiliza el número de sucesos aleatorios ocurridos en un intervalo de longitud t, satisfaciendo las siguientes condiciones [11][18]:

- Posee incrementos independientes (el número de sucesos aleatorios ocurridos en dos intervalos temporales disjuntos se distribuye de forma independiente).
- Posee incrementos estacionarios (el número de sucesos aleatorios ocurridos en un intervalo temporal no depende de su localización sino de su longitud).
- Para cualquier $t \in \mathbb{R}_+$ y $h > 0$,

$$N(t+h) - N(t) \sim \mathcal{P}(\lambda h),$$

  donde se denota por $\mathcal{P}(\lambda h)$ la distribución de Poisson con parámetro λh.

De aquí se deduce que $N(t) \sim \mathcal{P}(\lambda t)$ y, por tanto, $E[N(t)] = Var(N(t)) = \lambda t$ [11].

Adicionalmente, se tiene que los tiempos aleatorios entre ocurrencias de sucesos se distribuyen según una exponencial con parámetro λ y que los tiempos de espera hasta que se produce la ocurrencia de n sucesos aleatorios se distribuyen según una Gamma con parámetros n, λ [11][18].

$$T_i \sim exp(\lambda) \quad S_n = \sum_{i=1}^{n} T_i \sim \Gamma(n, \lambda).$$

1.5. Proceso de Ornstein-Uhlenbeck

Si en la ecuación (1.1) presentada anteriormente consideramos

$$c(X_t) = p + rX_t, \quad \sigma(X_t) = \sigma,$$

la ecuación diferencial estocástica tiene la forma

$$dX_t = (p + rX_t)dt + \sigma dB(t), \quad t \geq 0$$

y su solución es el proceso de Ornstein-Uhlenbeck generalizado. Este modelo es una traslación del famoso modelo de Vasicek para la tasa de interés [2], para el que la parametrización usual es

$$dX_t = k(a - X_t)dt + \sigma dB(t), t \geq 0,$$

donde k es la velocidad de inversión media ($k > 0$) y a es un objetivo a largo plazo para X. El término $a - X_t$ refleja las fluctuaciones de los estados del proceso X_t respecto al objetivo a [17].

Este modelo se puede escribir mediante la segunda parametrización ($\sigma > 0, \mu, \theta \in R$) como [11]

$$dX_t = -\theta(X_t - \mu)dt + \sigma dB(t), t \geq 0, X_0 = x_0$$

cuya solución viene dada por [2]

$$X_t = \mu + (x_0 - \mu)e^{-\theta t} + \sigma \int_0^t e^{-\theta(t-u)} dB(t), t \geq 0.$$

En el caso en el que $\theta > 0$, el proceso es también ergódico con densidad gaussiana invariante $N(\mu, \sigma^2/2\theta^2)$ y función de covarianza [17]

$$Cov(X_t, X_s) = Var(X_0)e^{-\theta|t-s|}, t, s \in R.$$

El proceso de Ornstein-Uhlenbeck es un proceso estacionario de Gauss-Markov, es decir, un proceso gaussiano, un proceso de Markov y es temporalmente homogéneo.

Asumiendo $X_0 = x_0$ constante, la media del modelo es [17]

$$E[x_t] = x_0 e^{-\sigma t} + \mu(1 - e^{-\sigma t})$$

y la covarianza es [37]

$$Cov(x_t, x_s) = \frac{\sigma^2}{2\theta}(e^{-\sigma|t-s|} - e^{-\sigma(t+s)}).$$

Este modelo se utiliza en aplicaciones físicas y en matemáticas financieras. El proceso de Ornstein-Uhlenbeck es uno de los varios enfoques utilizados para modelar (con modificaciones) las tasas de interés, las tasas de cambio de divisas y los precios de las materias primas de forma estocástica. El parámetro μ representa el equilibrio o valor medio apoyado por fundamentos; σ el grado de volatilidad a su alrededor causado por los choques, y θ la tasa a la que estos choques se disipan y la variable se revierte hacia la media. Una aplicación muy útil del proceso es una estrategia comercial conocida como comercio de pares [6].

2 Simulación de procesos básicos y procesos de difusión con saltos

Para la simulación de los procesos de difusión estudiados en el Capítulo 1 de este Trabajo de Fin de Máster se utilizan aproximaciones que hacen uso del concepto de integral estocástica en el sentido de Itô.

Lema de Itô

Sea X_t un proceso de difusión cuya dinámica es

$$dX_t = f(t, X_t)dt + g(t, X_t)dB(t).$$

Si $Y_t = H(t, X_t)$ es función del proceso de difusión anterior, con H una función de clase $\mathcal{C}^2(\mathbb{R} \times \mathbb{R})$, entonces Y_t es un proceso de difusión con ecuación diferencial estocástica [29]

$$dY_t = \left(\frac{\partial H(t, X_t)}{\partial t} + f(t, X_t)\frac{\partial H(t, X_t)}{\partial X_t} + \frac{1}{2}g(t, X_t)\frac{\partial^2 H(t, X_t)}{\partial X_t^2} \right) dt + \left(g(t, X_t)\frac{\partial H(t, X_t)}{\partial X_t} \right) dB(t).$$

Integral estocástica en el sentido de Itô

Sea $g(t)$ una función de cuadrado integrable en $[0, T]$. Se define su integral estocástica en el sentido de Itô como

$$\int_0^T g(t)dB(t) = lim_{||\Delta_M|| \to 0} \sum_{i=2}^{M} g(t_{i-1})(B(t_i) - B(t_{i-1}))$$

donde $\Delta_M = \{0 = t_1, t_2, \cdots, t_M = T\}$ una partición del intervalo $[0, T]$ y $||\Delta_M|| = max_{2 \leq i \leq M}(t_i - t_{i-1})$ [21].

2.1. Simulación del proceso Browniano y del proceso Browniano Geométrico

2.1.1. Aproximación de Euler

En general, para ecuaciones diferenciales estocásticas de la forma (1.1)

$$dX_t = c(t, X_t)dt + \sigma(t, X_t)dB_t, \ X_{t_0} = X_0, \ t \in [0, T]$$

donde $B(t)$ es un movimiento Browniano estándar, se utilizan métodos numéricos basados en la aproximación de Euler para aproximar la solución de la ecuación. Para ello, se considera la discretización

$$0 = t_1 < t_2 < \cdots < t_M = T$$

y, para $i = 0, 1, \ldots, M - 1$, la aproximación de Euler, que notamos con la letra Y, ya que no es la solución exacta de la ecuación diferencial estocástica, es la siguiente:

$$Y_0 = X_0;$$
$$Y_{i+1} = Y_i + c(t_i, Y_i)(t_{i+1} - t_i) + \sigma(t_i, Y_i)(B_{i+1} - B_i),$$

donde $Y(t_i) = Y_i$, $B_{(t_i)} = B_i$ [28]. Si $\Delta t = \frac{1}{M}$, es decir, es paso de discretización es constante, entonces

$$Y(t) = Y_i + \frac{t - t_i}{t_{i+1} - t_i}(Y_{i+1} - Y_i) = Y_i + \frac{t - t_i}{M}(Y_{i+1} - Y_i), \quad t \in [t_i, t_{i+1}).$$

Bajo ciertas condiciones se puede probar que el esquema de Euler converge al verdadero proceso a medida que la discretización del tiempo se hace cada vez más fina [23].

Esta aproximación se puede implementar en MatLab. A continuación se muestra el código para simular una trayectoria de un proceso de difusión unidimensional solución de una ecuación diferencial estocástica de la forma (1.1) mediante la aproximación de Euler. Para ello se crea la función *Euler*, en un fichero llamado Euler.m, cuyos parámetros son $X(t_0) = X_0$; el intervalo de tiempo, cuyos extremos son *time1* y *time2*; y M, para la partición del intervalo temporal. Además, utiliza una función c que depende del tiempo y de X_t, que representa la función de deriva $C(t, X_T)$; y una función S que depende del tiempo y de X_t, que representa la difusión $\sigma(t, X_t)$. En la expresión de la aproximación se considera $B(t_{i+1}) - B(t_i) = Z\sqrt{\Delta t}$. Esta función devuelve t, la partición del intervalo temporal, y el vector y, que contiene la aproximación de Euler. Entre dos puntos de tiempo cualesquiera, la trayectoria se aproxima por interpolación lineal.

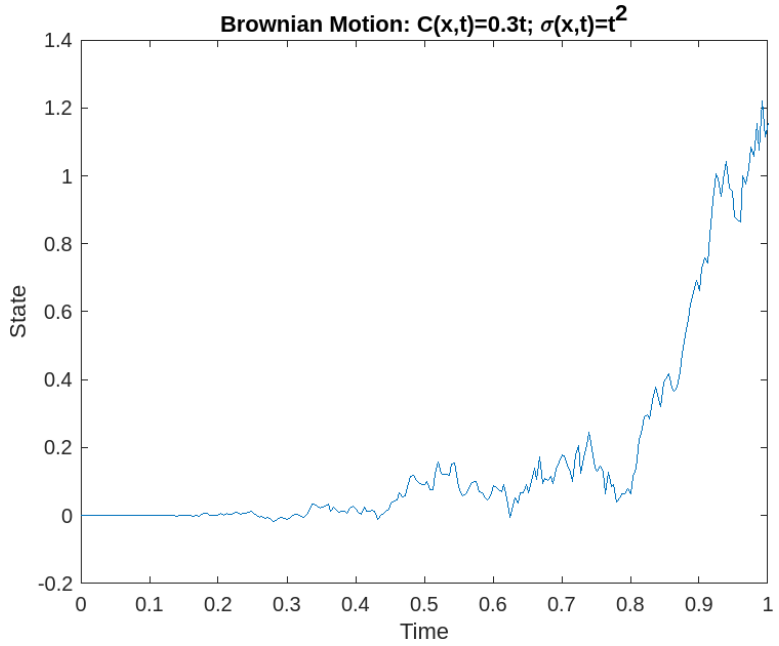
```
function [t,y]=Euler(xo,time1,time2,M)
        y=zeros(1,M+1);
        y(1)=xo;
        t=time1:(time2-time1)/M:time2;
        Z = normrnd(0,1,M);

        for i = 1:(length(t)-1)
                y(i+1)=y(i)+c(t(i),y(i))*(t(i+1)-t(i))+
                 S(t(i),y(i))*(sqrt((time2-time1)/M)*Z(i));
        end
end
```

Para simular una trayectoria del proceso Browniano utilizando esta función, se definen las funciones de deriva y de difusión como variables deterministas que varían en el tiempo, también en el fichero Euler.m.

```
function c=c(t,x)
        c=0.3*t;
end

function S=S(t,x)
        S=t^2;
end
```

y, a continuación, se evalúa la función *Euler* considerando el intervalo de tiempo $[0, 1]$, $M = 250$ y $X_0 = 0$.

```
[t,y]=Euler(0,0,1,250);
plot(t,y)
xlabel('Time')
ylabel('State')
title('Brownian Motion: C(x,t)=0.3t; \sigma(x,t)=t^2')
```

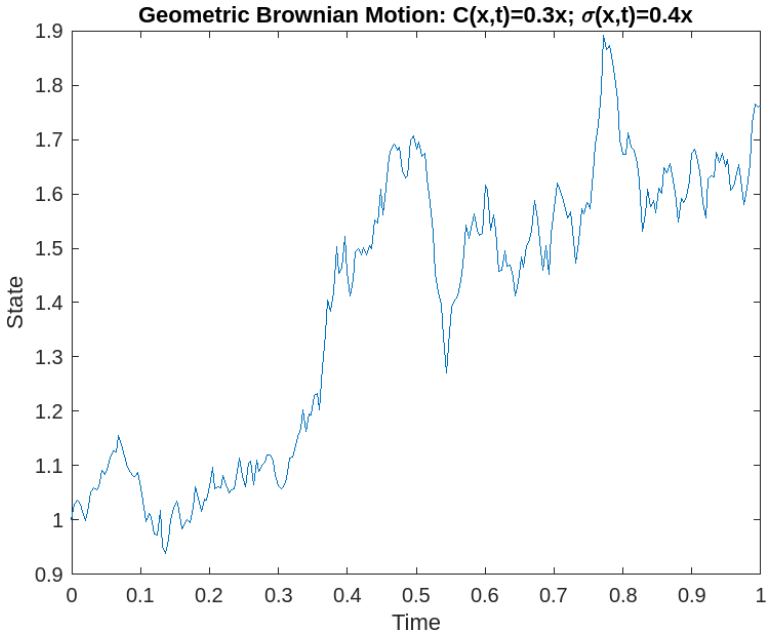


Para simular una trayectoria del proceso Browniano geométrico mediante la función *Euler* se definen las funciones de deriva y de difusión de la forma $C(t, X_t) = \mu X_t$ y $\sigma(t, X_t) = \sigma X_t$, sustituyendo las anteriores en el fichero Euler.m.

```
function c=c(t,x)
        c=0.3*x;
end

function S=S(t,x)
        S=0.4*x;
end
```

y, a continuación, se evalúa la función *Euler* considerando el intervalo de tiempo $[0, 1]$, $M = 250$ y $X_0 = 1$.

```
[t,y]=Euler(1,0,1,250);
plot(t,y)
xlabel('Time')
ylabel('State')
title('Geometric Brownian Motion: C(x,t)=0.3x; \sigma(x,t)=0.4x')
```



Por otro lado, la aproximación de Euler se encuentra implementada en MatLab a través de varias funciones que se estudian a continuación.

Con el fin de aproximar y simular las trayectorias de un proceso Browniano, se hace uso de la función *bm* disponible en MatLab, que genera un objeto bm (*Brownian motion*) que se deriva de la ecuación diferencial estocástica (1.2), donde ahora X_t, μ y dW_t son vectores de dimensión N y σ es una matriz de dimensión $N \times N$. El vector μ es la tasa de deriva, la matriz σ es la tasa de volatilidad instantánea y dW_t es un vector de componentes Brownianos correlacionados con deriva cero[30]. La función *bm* tiene como parámetros μ y σ. Estos pueden ser introducidos de dos formas: bien como un vector y una matriz, de forma que se asume una especificación paramétrica estática, que no varía en el tiempo; o bien como funciones de MatLab con input el tiempo t de observación real valuado e input opcional un vector estático X_t de dimensión N. El objeto bm generado es de la clase *Movimiento Browniano* y proporciona las siguientes propiedades:

- StartTime: el tiempo inicial de la primera observación, que por defecto es 0.
- StartState: valores iniciales de las variables de estado, que por defecto es 1.
- Correlation: la correlación entre variables aleatorias gaussianas dibujadas para generar el vector proceso Browniano.
- Simulation: el método de simulación, que por defecto es la simulación por aproximación de Euler.
- Drift: la función de deriva.
- Diffussion: la función de difusión.

Además, estas propiedades se guardan bajo los parámetros opcionales *Name* y *Value* de la función *bm*, que se utilizan de la siguiente forma: con *Name* se especifica el nombre de una

de las propiedades anteriores y con *Value*, el valor de dicha propiedad. Estos parámetros deben usarse de forma conjunta y puede utilizarse más de uno a la vez.

Por otro lado, para aproximar y simular las trayectorias de un proceso Browniano geométrico se dispone de la función *gbm* en MatLab, que en este caso genera un objeto gbm (*Geometric Brownian motion*) que se deriva de la ecuación diferencial

$$dX_t = \mu X_t dt + D(t, X_t)\sigma dB_t,$$

donde de nuevo X_t, dW_t son vectores de dimensión N y σ es una matriz de dimensión $N \times N$. Ahora, μ y $D(t, X_t)$ son matrices de dimensión $N \times N$. El vector μ es la tasa de deriva, la matriz σ es la tasa de volatilidad instantánea y dW_t es un vector de componentes Brownianas. La matriz $D(t, X_t)$ es diagonal y cada elemento de la diagonal principal es el elemento correspondiente del vector de estados X_t. La función *gbm* tiene los mismos parámetros que la función *bm*, incluidos los parámetros opcionales, y se introducen de la misma forma.

A estos objetos bm y gbm se le pueden aplicar distintas funciones de MatLab como *simulate*, *interpolate* y *simByEuler*. Al objeto gbm también se le puede aplicar la función *simBySolution*, que es exclusiva para el proceso Browniano geométrico ya que aplica un enfoque de Euler al proceso logarítmico transformado utilizando la fórmula de Itô.

La función *simulate* simula las trayectorias de N variables de estado correlacionadas, impulsadas por fuentes de riesgo del movimiento Browniano, aproximando procesos estocásticos de tiempo continuo. Para ello, la función tiene como argumento un objeto que corresponda a un modelo de ecuación diferencial estocástica bm o gbm (aunque también acepta otros tipos de modelos, como se verá en los siguientes capítulos).

Además de este argumento obligatorio, tiene argumentos opcionales que permiten elegir el esquema de aproximación utilizado para simular las trayectorias. Esta función devuelve una matriz *Paths* tridimensional, de dimensiones $(N+1) \times N \times N$, que contiene las trayectorias simuladas, siendo cada fila de la matriz la transposición del vector X_t en el tiempo t; un vector columna *Times* de dimensión $N+1$ con los tiempos de observación asociados a las trayectorias simuladas; y una matriz tridimensional Z, de dimensiones $N \times N \times N$, con las variables aleatorias dependientes usadas para generar el vector de movimiento Browniano utilizado para simular las trayectorias de *Paths*.

A estas trayectorias simuladas se les puede aplicar la función *interpolate*, que realiza una interpolación Browniana, en una serie temporal específica, basada en un enfoque de muestreo de Euler constante por partes. La serie temporal se introduce como argumento de la función en forma de un vector de dimensión N; además se requiere un objeto que corresponda a un modelo de ecuación diferencial estocástica bm o gbm (aunque también acepta otros tipos de modelos) y la matriz *Paths* de las trayectorias simuladas. Esta función también tiene argumentos opcionales, que deben ser introducidos por pares formados por el nombre del argumento y su valor. Algunos de estos argumentos opcionales son 'Times', que contiene tiempos de observación asociados a la serie temporal; 'Refine', de tipo lógico, que indica si se refina la interpolación a medida que hay nueva información disponible (este argumento es falso por defecto); 'Processes', que contiene una secuencia de funciones de procesamiento ejecutadas en cada tiempo de interpolación; y 'DeltaTime', que indica la longitud de los períodos de simulación. Este último argumento opcional es útil, por ejemplo, para especificar condiciones de contorno, o trazar gráficos. Los output de *interpolate* son una matriz tridimensional XT de dimensiones $N \times N \times N$ con las variables de estado interpoladas y un vector columna T de dimensión N con los tiempos de interpolación asociados.

Ejemplo 1

La función *interpolate* es útil en los casos en los que se necesita conocer el vector de estado en tiempos de muestra intermedios que inicialmente no están disponibles [9]. Para aproximar estos estados intermedios se utiliza la técnica de muestreo conocida como puente Browniano, que captura la distribución conjunta correcta mediante el muestreo de una distribución gaussiana condicional [2] [11]. Se utiliza a continuación la interpolación sin refinamiento, definiendo en primer lugar el siguiente proceso Browniano correlacionado, mediante la función *bm*:

$$dX_{1t} = 0.3dt + 0.2dW_{1t} - 0.1dW_{2t}$$

$$dX_{2t} = 0.4dt + 0.1dW_{1t} - 0.2dW_{2t}$$

$$E[dW_{1t}, dW_{2t}] = \rho dt = 0.5dt$$

```
mu    = [0.3; 0.4];
sigma = [0.2 -0.1; 0.1 -0.2];
rho   = [1 0.5; 0.5 1]; % matriz de correlación
obj   = bm(mu,sigma,'Correlation',rho);
```

Mediante la función *simulate* se simulan las trayectorias simulando una sola prueba Monte Carlo de observaciones diarias durante 250 días (un año de días hábiles). La orden *rng default* provoca que se generen de forma repetida los mismos números aleatorios.

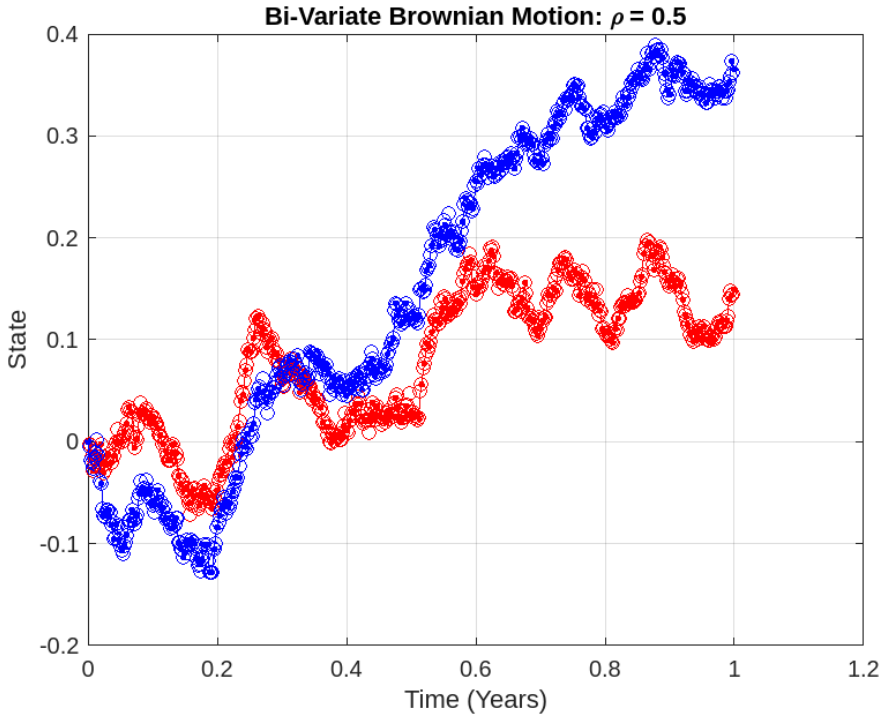
```
rng default
dt    = 1/250;  % 1 día hábil = 1/250
[X,T] = simulate(obj,250,'DeltaTime',dt);
```

Seguidamente, se usa la función *interpolate* para interpolar en la serie temporal simulada mediante un puente Browniano.

```
t = ((T(1) + dt/2):(dt/2):(T(end) - dt/2));
x = interpolate(obj,t,X,'Times',T);
```

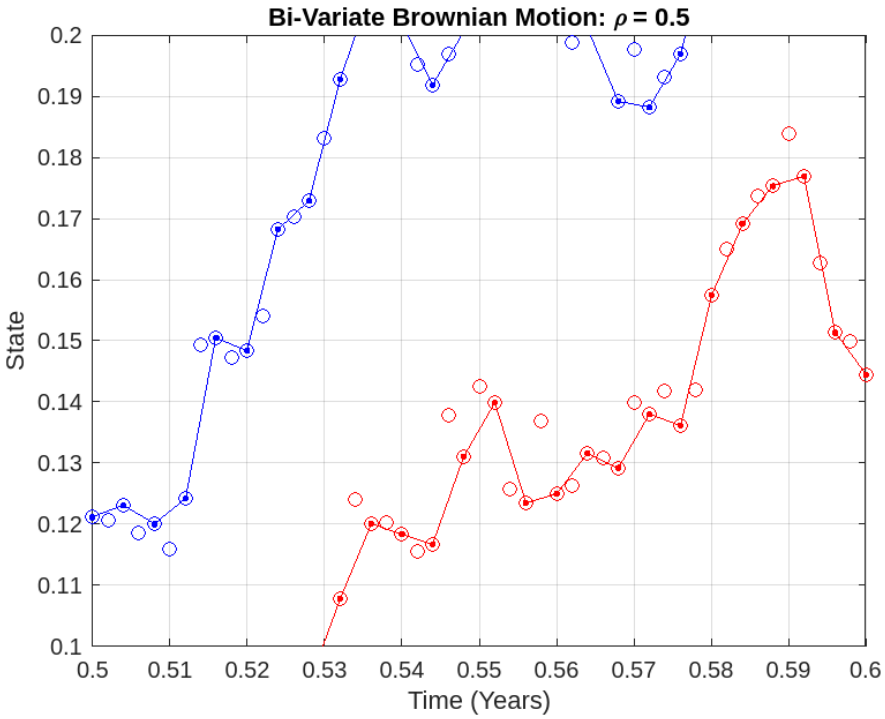
Por último, se grafican los valores simulados e interpolados, añadiendo títulos a la gráfica y a los ejes. Se muestran como puntos sólidos los estados simulados del modelo, mientras que las líneas que conectan estos puntos y los círculos abiertos son los estados intermedios que se obtienen al interpolar.

```
plot(T,X(:,1),'.-r',T,X(:,2),'.-b')
grid on;
hold on;
plot(t,x(:,1),'or',t,x(:,2),'ob')
hold off;
xlabel('Time (Years)')
ylabel('State')
title('Bi-Variate Brownian Motion: \rho = 0.5')
```



Observamos de cerca una parte de la gráfica:

```
axis([0.4999 0.6001 0.1 0.2])
```



Ejemplo 2

De forma adicional, se puede considerar un puente Browniano como una simulación Monte Carlo de una distribución gaussiana condicional. En este caso, vamos a ver el comportamiento en un solo intervalo temporal, por lo que se vuelve a generar el objeto bm anterior, se simulan sus trayectorias y se define el intervalo *times* intermedio mencionado, que se divide en subintervalos.

```
mu    = [0.3; 0.4];
sigma = [0.2 -0.1; 0.1 -0.2];
rho   = [1 0.5; 0.5 1];
obj   = bm(mu,sigma,'Correlation',rho);

rng default;
dt    = 1/250;
[X,T] = simulate(obj,250,'DeltaTime',dt);

n     = 125;
times = (T(n):(dt/10):T(n + 1));
```

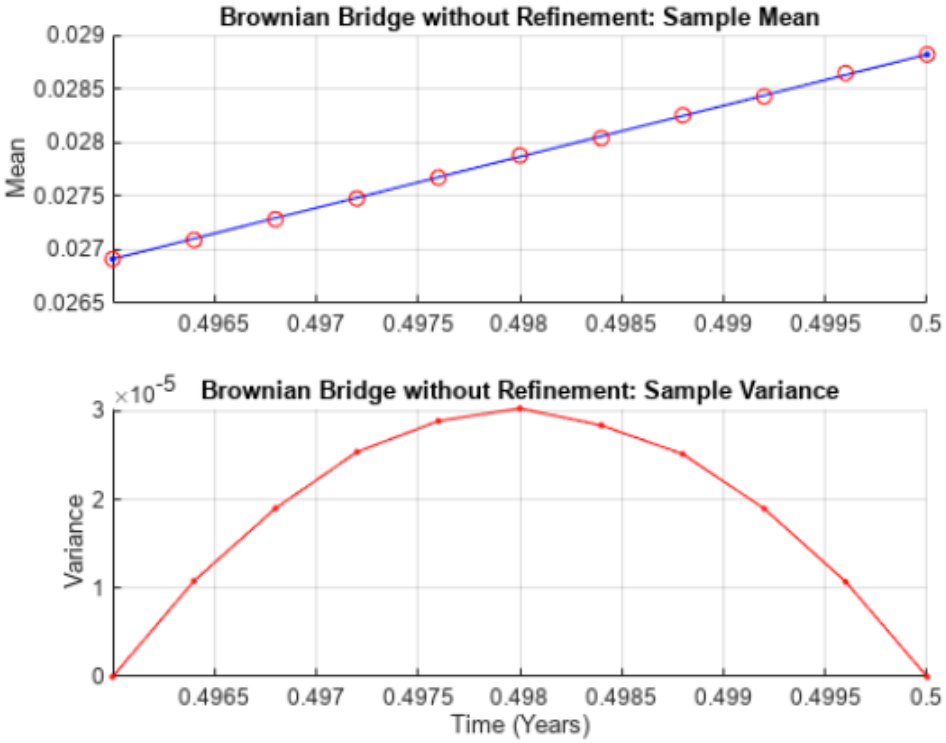
Seguidamente, en cada subintervalo se toman 25000 valores independientes de una distribución gaussiana, condicionados a los estados simulados a la izquierda y a la derecha mediante la función *interpolate*. En los vectores *average* y *variance* se almacenan la media y la varianza de las variables de estado interpoladas (sin refinamiento).

```
nTrials  = 25000;  % número de pruebas en cada instante t

average  = zeros(length(times),1);
variance = zeros(length(times),1);
for i = 1:length(times)
t = times(i);
x = interpolate(obj,t(ones(nTrials,1)),X,'Times',T);
average(i)  = mean(x(:,1));
variance(i) = var(x(:,1));
end
```

La interpolación estocástica realizada con la función *interpolate* está diseñada para interpolar en una serie temporal existente e ignorar nuevos estados interpolados a medida que se dispone de información adicional (interpolación sin refinamiento). Por último, se grafican la media y la varianza de cada variable de estado interpolada.

```
subplot(2,1,1);
hold on;
grid on;
plot([T(n) T(n + 1)],[X(n,1) X(n + 1,1)],'.-b')
plot(times, average, 'or')
hold off;
title('Brownian␣Bridge␣without␣Refinement:␣Sample␣Mean')
ylabel('Mean')
limits = axis;
axis([T(n) T(n + 1) limits(3:4)]);
```

```
subplot(2,1,2)
hold on;
grid on;
plot(T(n),0,'.-b',T(n + 1),0,'.-b')
plot(times, variance, '.-r')
hold('off');
title('Brownian Bridge without Refinement: Sample Variance')
xlabel('Time (Years)')
ylabel('Variance')
limits = axis;
axis([T(n) T(n + 1) limits(3:4)]);
```



Se puede observar que la media condicional forma una línea recta mientras que la varianza condicional es una función cuadrática que alcanza su máximo en el punto medio del intervalo. Si la interpolación hubiese sido con refinamiento, se hubiesen tenido en cuenta los nuevos datos para las siguientes interpolaciones y como consecuencia la varianza sería constante cero, no mostraría variabilidad.

Ejemplo 3

Por otra parte, para simular las trayectorias de N variables de estado correlacionadas se cuenta, además de con la función *simulate*, con la función *simByEuler*, que utiliza el enfoque de Euler para aproximar procesos estocásticos en tiempo continuo. Esta función tiene como argumentos obligatorios un proceso de difusión definido mediante una ecuación diferencial estocástica, en particular admite los objetos bm y gbm, y un número entero positivo 'NPeriods'

que representa el número de períodos de simulación. Además, hay pares de argumentos opciones (nombre del argumento y su valor correspondiente) que permiten, por ejemplo, realizar simulaciones cuasi-Monte Carlo o indicar el número de trayectorias a simular. Los outputs de esta función son una matriz *Paths* tridimensional, de dimensiones $(NPeriods + 1) \times N \times N$, con las trayectorias simuladas, donde cada fila de la matriz es la transposición del vector X_t en el tiempo t; un vector columna *Times* de dimensión $NPeriods + 1$ con los tiempos de observación asociados a las trayectorias simuladas, cada elemento de este vector está asociado con la fila correspondiente de *Paths*; y una matriz tridimensional Z, de dimensiones $(NPeriods\, N) \times N \times N$, con las variables aleatorias dependientes utilizadas para generar el vector de movimiento Browniano usado para simular las trayectorias.

A continuación, se utiliza la función *simByEuler* para realizar una simulación cuasi-Monte Carlo a partir del siguiente modelo Browniano geométrico, que se define mediante la función *gbm*:

$$dX_t = 0.25X_t dt + 0.3X_t dW_t$$

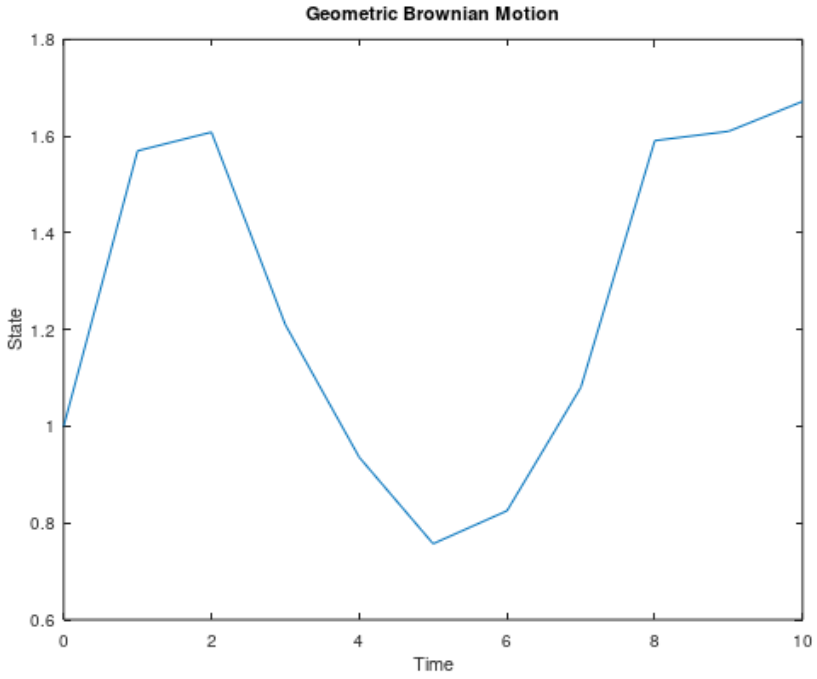
```
gbm_obj = gbm(0.25, 0.3)
```

```
gbm     =
   Class GBM: Generalized Geometric Brownian Motion
   -------------------------------------------------
     Dimensions: State = 1, Brownian = 1
   -------------------------------------------------
      StartTime: 0
     StartState: 1
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
         Return: 0.25
          Sigma: 0.3
```

```
[paths,time,z] =simByEuler(gbm_obj,10,'ntrials',4096,
        'montecarlomethod','quasi','quasisequence','sobol',
        'BrownianMotionMethod','brownian-bridge');
```

El argumento opcional 'ntrials' indica el número de trayectorias que se van a simular, el parámetro 'montecarlomethod' indica que se va a realizar una simulación Monte Carlo con valor 'quasi' señalando que la simulación es cuasi-Monte Carlo. El parámetro 'quasisequence' con valor 'sobol' realiza una secuencia cuasi-aleatoria de baja discrepancia para impulsar los procesos estocásticos, formando particiones uniformes cada vez más finas del intervalo unitario para luego reordenar las coordenadas en cada dimensión. Finalmente, el parámetro 'BrownianMotionMethod' con valor 'brownian-bridge' indica que en este caso el método de construcción del movimiento Browniano es un puente Browniano, calculando en primer lugar el último paso de la trayectoria del movimiento Browniano. Por último, se realiza la gráfica correspondiente a una de las trayectorias.

```
plot(time, paths(:,:,1)), xlabel('Time'), ylabel('State')
title('Geometric Brownian Motion')
```



Ejemplo 4

Exclusivamente para los objetos gbm se dispone además de la función *simBySolution* que proporciona una solución aproximada de la ecuación diferencial estocástica subyacente aplicando un enfoque de Euler al proceso logarítmico transformado utilizando la fórmula de Itô. Requiere un objeto gbm como argumento además del número de períodos de simulación 'NPeriods'. Esta función tiene los mismos pares de argumentos opcionales que la función *simByEuler*.

A continuación, se simulan mercados de valores mediante la simulación de un modelo Browniano Geométrico. Para ello se cargan los datos *Data_GlobalIdx2* disponibles en MatLab, a los que aplicamos la función *tick2ret* que convierte series de precios en series de retorno. Se definen los argumentos de la función *gbm* a partir de los datos, usando la función media *mean*, la función desviación típica *std* y la función *corrcoef*, que calcula los coeficientes de correlación. Se crea el vector de dimensión $nVariables \times 1$ cuyos elementos son todos 100 y se crea el modelo Browniano geométrico que representa el modelo de mercado, indicando la correlación y los valores iniciales de las variables de estado.

```
load Data_GlobalIdx2
prices  = [Dataset.TSX Dataset.CAC Dataset.DAX Dataset.NIK
Dataset.FTSE Dataset.SP];

returns =  tick2ret(prices);

nVariables  = size(returns,2);
expReturn   = mean(returns);
sigma       = std(returns);
correlation = corrcoef(returns);
```

```
X            = 100*(ones(nVariables,1));

GBM = gbm(diag(expReturn),diag(sigma), 'Correlation', correlation,
        'StartState', X);
```

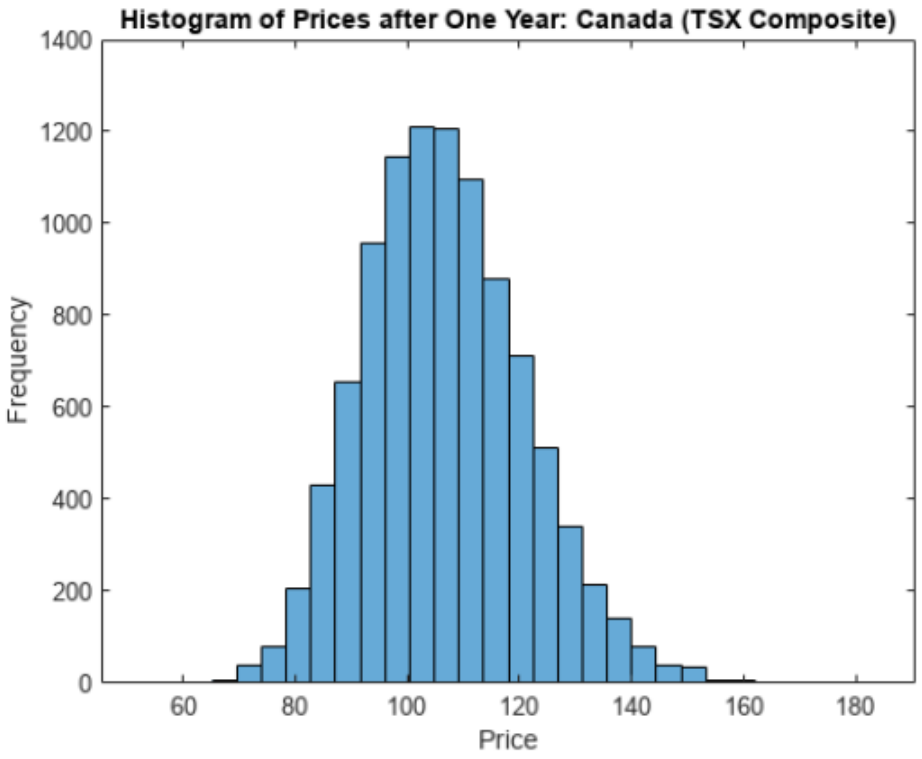
Se utiliza la función *simulate* utilizando el número de períodos de observación 'nPeriods' e indicando que el incremento temporal es de un día y que el número de trayectorias simuladas es 10000. Puesto que la salida 'Z' no se necesita para este ejemplo, no se solicita.

```
nPeriods = 249;
dt       =   1;      % incremento temporal=1 día
rng(142857,'twister')
[X,T] = simulate(GBM, nPeriods, 'DeltaTime', dt, 'nTrials', 10000);

whos X
```

```
  Name        Size                    Bytes  Class     Attributes

  X         250x6x10000           120000000  double
```

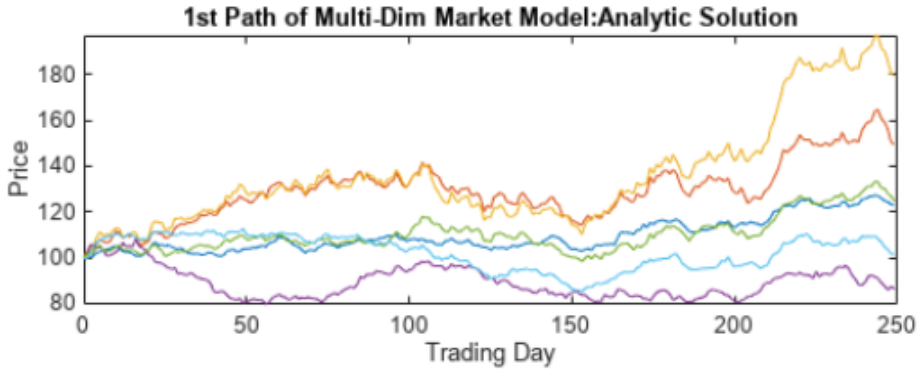
Con la función *histogram* se ve que los precios después de un año (250 días hábiles) tienen carácter lognormal.

```
histogram(squeeze(X(end,1,:)), 30), xlabel('Price')
ylabel('Frequency')
title('Histogram of Prices after One Year: Canada (TSX Composite)')
```



Se usa la función *simBySolution* para simular 10 trayectorias y se grafica la primera de ellas.

```
[X,T] = simBySolution(GBM, nPeriods,'DeltaTime', dt, 'nTrials', 10);

plot(T, X(:,:,1)), xlabel('Trading Day'),ylabel('Price')
title('1st Path of Multi-Dim Market Model: Analytic Solution')
```



Ejemplo 5

La función *simByEuler* evalúa directamente la ecuación diferencial estocástica que define el proceso de difusión, produciendo un error de discretización. Por el contrario, la función *simBySolution* proporciona una descripción más precisa del modelo subyacente. A continuación, se ilustra la diferencia con un ejemplo. Se crea un objeto gbm y se simulan trayectorias, cambiando el número de pasos temporales intermedios dentro de cada incremento de tiempo dt, usando tanto la función *simByEuler* como la función *simBySolution* bajo las mismas condiciones.

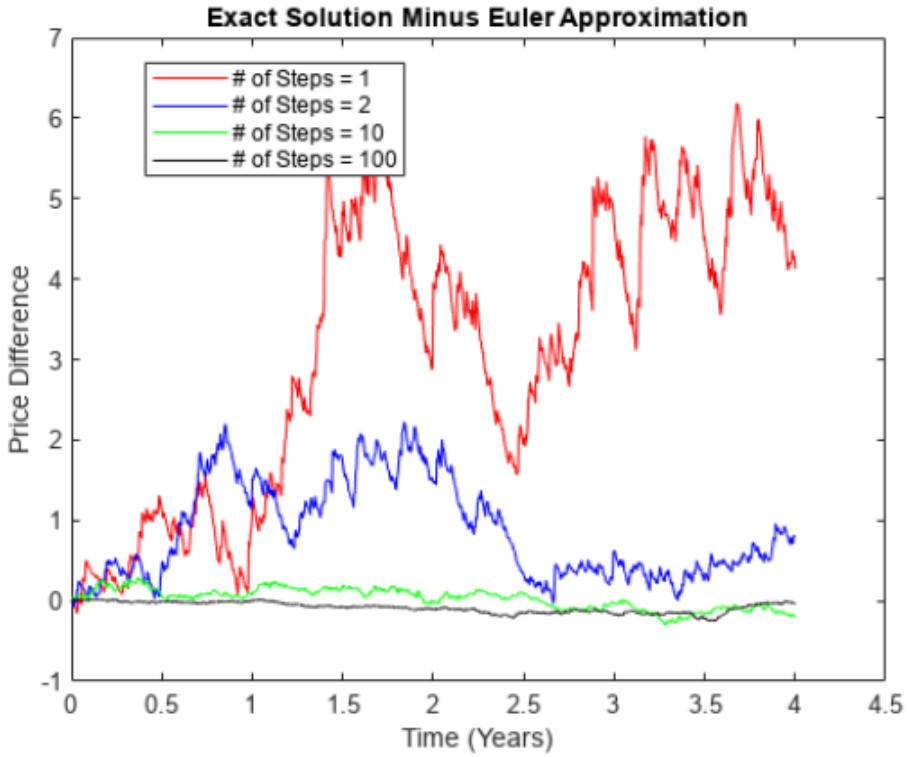
$$dX_t = 0.1X_t dt + 0.4X_t dW_t$$

```
nPeriods = 1000;
dt       = 1/250;
obj      = gbm(0.1,0.4,'StartState',100);
rng(575,'twister')
[X1,T1]    = simBySolution(obj,nPeriods,'DeltaTime',dt);
rng(575,'twister')
[Y1,T1]    = simByEuler(obj,nPeriods,'DeltaTime',dt);
rng(575,'twister')
[X2,T2]    = simBySolution(obj,nPeriods,'DeltaTime',dt,'nSteps',2);
rng(575,'twister')
[Y2,T2]    = simByEuler(obj,nPeriods,'DeltaTime',dt,'nSteps',2);
rng(575,'twister')
[X3,T3]    = simBySolution(obj,nPeriods, 'DeltaTime',dt,'nSteps',10);
rng(575,'twister')
[Y3,T3]    = simByEuler(obj,nPeriods,'DeltaTime',dt,'nSteps',10);
rng(575,'twister')
```

```
[X4,T4]     = simBySolution(obj,nPeriods,'DeltaTime',dt,'nSteps',100);
rng(575,'twister')
[Y4,T4]     = simByEuler(obj,nPeriods,'DeltaTime',dt,'nSteps',100);
```

En la siguiente gráfica se compara la diferencia entre la solución exacta (la generada con la función *simBySolution*) y la aproximada con la función *simByEuler*, es decir, se muestra el error de la aproximación. Se puede observar que el error disminuye a medida que aumenta el número de pasos temporales intermedios.

```
plot(T1,X1 - Y1,'red')
hold on;
plot(T2,X2 - Y2,'blue')
plot(T3,X3 - Y3,'green')
plot(T4,X4 - Y4,'black')
hold off
xlabel('Time (Years)')
ylabel('Price Difference')
title('Exact Solution Minus Euler Approximation')
legend({'# of Steps = 1'  '# of Steps = 2'  '# of Steps = 10'
        '# of Steps = 100'},'Location', 'Best')
hold off
```



Ejemplo 6

Seguidamente, se muestra un ejemplo en el que se usan dos técnicas distintas para generar un objeto gbm, utilizando los datos *Data_GlobalIdx2* de MatLab, para simular el siguiente modelo de mercado:

$$dX_t = \mu X_t dt + \sigma X_t dW_t$$

donde μ es una matriz diagonal.

```
load Data_GlobalIdx2
prices  = [Dataset.TSX Dataset.CAC Dataset.DAX Dataset.NIK
Dataset.FTSE Dataset.SP];
returns =  tick2ret(prices);
```

La primera técnica genera variables gaussianas correlacionadas para formar un proceso de movimiento Browniano con componentes dependientes, que después se ponderan mediante una volatilidad diagonal.

```
expReturn   = diag(mean(returns));
sigma       = diag(std(returns));   % volatilidad
correlation = corrcoef(returns);
GBM1        = gbm(expReturn,sigma,'Correlation',correlation);
```

La segunda técnica genera variables gaussianas independientes para formar un proceso de movimiento Browniano estándar, que luego se pondera con el factor de Cholesky inferior de la matriz de covarianza deseada, mediante la función *cholcov*.

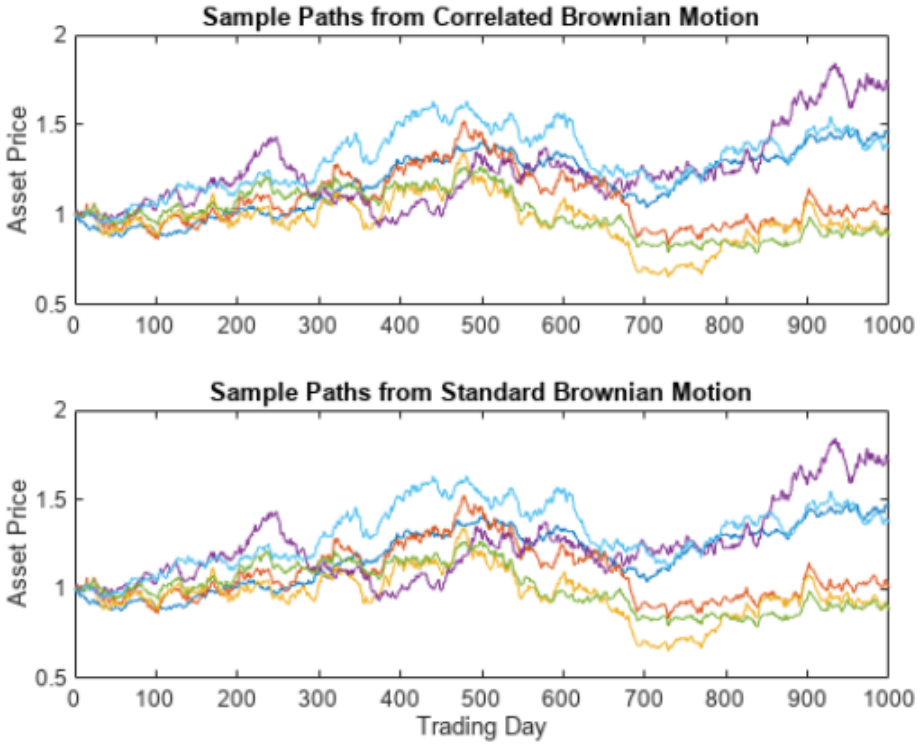
```
covariance = cov(returns);
sigma      = cholcov(covariance)';
GBM2       = gbm(expReturn,sigma);
```

Con la función *simByEuler* se simulan 1000 observaciones de cada técnica.

```
rng(22814,'twister')
[X1,T] = simByEuler(GBM1,1000); %movimiento Browniano correlacionado
rng(22814,'twister')
[X2,T] = simByEuler(GBM2,1000); %movimiento Browniano estándar
```

Gracias a la orden *rng(22814,'twister')*, que fija la semilla para generar números aleatorios, las trayectorias generadas con cada técnica son idénticas:

```
subplot(2,1,1)
plot(T, X1)
title('Sample␣Paths␣from␣Correlated␣Brownian␣Motion')
ylabel('Asset␣Price')
subplot(2,1,2)
plot(T, X2)
title('Sample␣Paths␣from␣Standard␣Brownian␣Motion')
xlabel('Trading␣Day')
ylabel('Asset␣Price')
```



Ejemplo 7

Se considera el siguiente movimiento Browniano geométrico, en el que la deriva varía en el tiempo:

$$dX_t = r(t)X_t dt + \sigma X_t dW_t$$

Para generar este modelo se utilizan los datos *Data_GlobalIdx2* disponibles en MatLab y se van a simular las trayectorias de observaciones diarias durante 250 días (un año de días hábiles).

```
load Data_GlobalIdx2
returns = tick2ret(Dataset.CAC);
```

En primer lugar se genera el modelo con deriva no dependiente del tiempo, calculada como la media muestral de los rendimientos (*yields*), para realizar comparaciones más adelante.

```
dt       = 1/250;
sigma    = std(returns)*sqrt(250);
yields   = Dataset.EB3M;
yields   = 360*log(1 + yields);

nPeriods = length(yields);  % observaciones simuladas
rng(5713,'twister')
obj = gbm(mean(yields),diag(sigma),'StartState',100)
```

```
obj =
   Class GBM: Generalized Geometric Brownian Motion
   -------------------------------------------------
     Dimensions: State = 1, Brownian = 1
   -------------------------------------------------
      StartTime: 0
     StartState: 100
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
         Return: 0.0278117
          Sigma: 0.231906
```

Se simulan las trayectorias con la función *simulate*.

```
[X1,T] = simulate(obj,nPeriods,'DeltaTime',dt);
```

En segundo lugar, se genera el modelo definiendo $r(t)$ como la serie temporal histórica de los rendimientos mediante la función *ts2func* de MatLab, teniendo así una tasa de rendimiento dinámica determinista. La función *ts2func* convierte una matriz de series temporales en funciones de tiempo y estado, y la sincroniza con un vector de tiempo opcional, realizando una interpolación constante por tramos. $r(0,100)$, por ejemplo, evalúa la función en $t = 0$, $X_t = 100$ y devuelve el primer rendimiento observado.

```
r = ts2func(yields,'Times',(0:nPeriods - 1)');
r(0,100)
```

```
ans = 0.0470
```

Una vez que se dispone de la función $r(t)$, se crea el objeto gbm y se simulan las trayectorias.

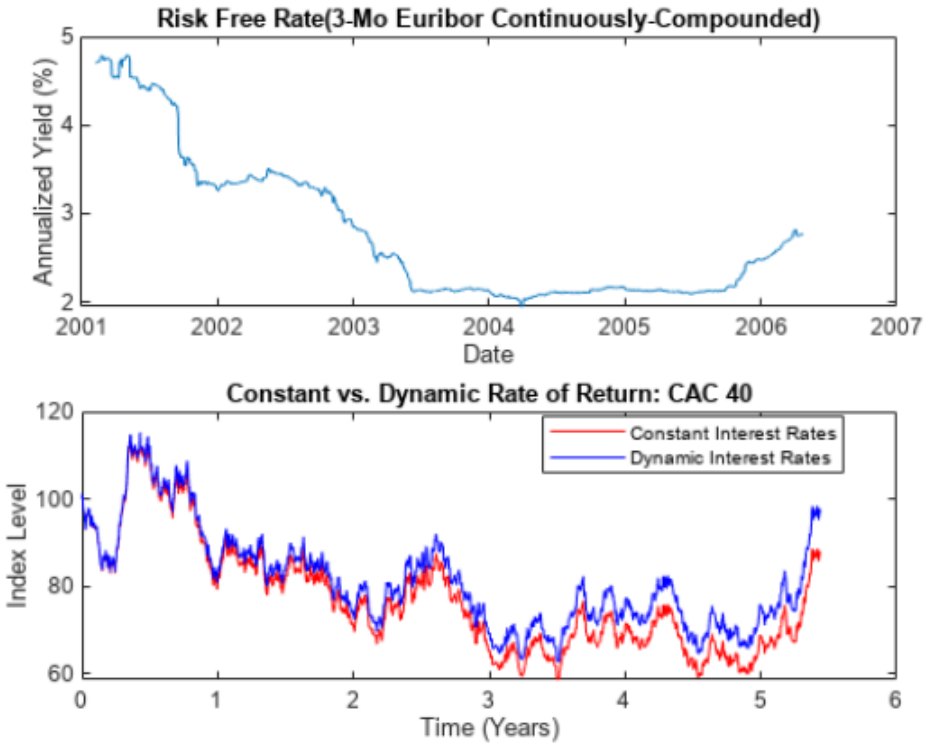
```
rng(5713,'twister')
obj = gbm(r, diag(sigma),'StartState',100)
```

```
obj =
   Class GBM: Generalized Geometric Brownian Motion
   -------------------------------------------------
     Dimensions: State = 1, Brownian = 1
   -------------------------------------------------
      StartTime: 0
     StartState: 100
    Correlation: 1
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
         Return: function ts2func/vector2Function
          Sigma: 0.231906
```

```
X2 = simulate(obj,nPeriods,'DeltaTime',dt);
```

Puesto que se ha usado la misma orden *rng(5713,'twister')*, se pueden comparar las dos simulaciones, que son similares pero no idénticas.

```
subplot(2,1,1)
plot(dates,100*yields)
datetick('x')
xlabel('Date')
ylabel('Annualized Yield (%)')
title('Risk Free Rate(3-Mo Euribor Continuously-Compounded)')
subplot(2,1,2)
plot(T,X1,'red',T,X2,'blue')
xlabel('Time (Years)')
ylabel('Index Level')
title('Constant vs. Dynamic Rate of Return: CAC 40')
legend({'Constant Interest Rates' 'Dynamic Interest Rates'},
        'Location', 'Best')
```



Ejemplo 8

En este último ejemplo se considera el siguiente movimiento Browniano geométrico:

$$dX_t = r(t)X_t dt + V(t, X_t)X_t dW_t$$

donde X_t representa el precio de las acciones, $r(t)$ es la tasa de rendimiento, que es una función determinista del tiempo, y $V(t, X_t)$ es la volatilidad, que es una función tanto del tiempo como del precio de las acciones.

Tanto el rendimiento como la volatilidad se definen en una cuadrícula discreta de forma que los valores intermedios se obtienen por interpolación lineal, con las funciones *interp1* e *interp2* de MatLab. Las columnas de la superficie de volatilidad corresponden al dinero relativo simple y las filas corresponden al tiempo hasta el vencimiento.

```
times=[0 0.25 0.5 1 2 3 4 5 6 7 8 9 10]; % en años
rates=[0.1 0.2 0.3 0.4 0.5 0.8 1.25 1.75 2.0 2.2 2.25 2.50 2.75]/100;

% superficie de volatilidad
surface=[28.1 25.3 20.6 16.3 11.2 6.2 4.9 4.9 4.9 4.9 4.9 4.9
22.7 19.8 15.4 12.6 9.6 6.7 5.2 5.2 5.2 5.2 5.2 5.2
21.7 17.6 13.7 11.5 9.4 7.3 5.7 5.4 5.4 5.4 5.4 5.4
19.8 16.4 12.9 11.1 9.3 7.6 6.2 5.6 5.6 5.6 5.6 5.6
18.6 15.6 12.5 10.8 9.3 7.8 6.6 5.9 5.9 5.9 5.9 5.9
17.4 13.8 11.7 10.8 9.9 9.1 8.5 7.9 7.4 7.3 7.3 7.3
17.1 13.7 12.0 11.2 10.6 10.0 9.5 9.1 8.8 8.6 8.4 8.0
17.5 13.9 12.5 11.9 11.4 10.9 10.5 10.2 9.9 9.6 9.4 9.0
18.3 14.9 13.7 13.2 12.8 12.4 12.0 11.7 11.4 11.2 11.0 10.8
19.2 19.6 14.2 13.9 13.4 13.0 13.2 12.5 12.1 11.9 11.8 11.4]/100;

tenor = [0 0.25 0.50 0.75 1 2 3 5 7 10]; % vencimiento, en años
moneyness = [0.25 0.5 0.75 0.8 0.9 1 1.10 1.25 1.50 2 3 5];

strike = 100;

mu= @(t) interp1(times,rates,t);
sigma= @(t,S) interp2(moneyness,tenor,surface,S/strike,tenor(end)-t);
```

Se genera el objeto gbm haciendo uso de las funciones mu y sigma.

```
price = 100;
mdl = gbm(mu,sigma,'StartState',price);
```

Para simular las trayectorias, se va a realizar una mejora en el rendimiento de la simulación Monte Carlo mediante computación paralela. Primero se generan las variables aleatorias gaussianas que impulsan la simulación (Z), lo que permite que las rutas simuladas resultantes coincidan con las de la simulación convencional y garantiza que todas las rutas simuladas sean independientes.

```
dt = 1/12;
NPeriods = 120;
NTrials = 100;

rng default
Z = randn(NPeriods,1,NTrials);

tStart = tic;
paths = simBySolution(mdl,NPeriods,'NTrials',NTrials,'DeltaTime',dt,
'Z',Z);
time1 = toc(tStart);
```

Se han simulado las trayectorias del movimiento Browniano geométrico sin paralelización utilizando la función *simBySolution*. El objeto *time1* contiene el tiempo invertido en realizar esta simulación, en segundos.

A continuación, se simulan las trayectorias usando de nuevo la función *simBySolution*, pero esta vez realizando la simulación en paralelo mediante un bucle parfor. El objeto *time2* contiene el tiempo invertido en realizar la simulación en paralelo, en segundos.

```
tStart = tic;
parPaths = zeros(NPeriods+1,1,NTrials);
parfor i = 1:NTrials
        parPaths(:,:,i) = simBySolution(mdl,NPeriods,'DeltaTime',dt,
                'Z',Z(:,:,i));
end
time2 = toc(tStart);
```

Se puede ver que la simulación en paralelo es considerablemente más rápida.

```
time1            % tiempo transcurrido de la simulación convencional
time2            % tiempo transcurrido de la simulación paralela
speedup=time1/time2 % factor de aceleración
```

```
time1 =
    6.1329
time2 =
    2.5918
speedup =
    2.3663
```

2.1.2. Esquema de Milstein

Otro método numérico utilizado para aproximar la solución numérica de la ecuación (1.1) es el Esquema de Milstein, que hace uso del lema de Itô y cuya convergencia es mayor que la de la aproximación de Euler, aunque no lo suficiente como para justificar alguna preferencia por este método numérico [2]. Considerando de nuevo la discretización

$$0 = t_0 < t_2 \cdots < t_M = T$$

y para $i = 0, 1, \ldots, M-1$, la solución aproximada, que notamos con la letra Y, ya que no es la solución exacta de la ecuación diferencial estocástica, se expresa en término de los estados del proceso Y, de los incrementos temporales y de los incrementos simples y cuadráticos del movimiento Browniano:

$$Y_0 = X_0 = X(t_0) = 0;$$

$$\begin{aligned} Y_{i+1} &= Y_i + c(t_i, Y_i)(t_{i+1} - t_i) + \sigma(t_i, Y_i)(B_{i+1} - B_i) + \\ &\quad + \frac{1}{2}\sigma(t_i, Y_i)\sigma_x(t_i, Y_i)\left\{(B_{i+1} - B_i)^2 - (t_{i+1} - t_i)\right\}. \end{aligned}$$

Vamos a implementar en MatLab el Esquema de Milstein para simular una trayectoria de un proceso de difusión unidimensional solución de una ecuación diferencial estocástica de la forma (1.1). Para ello se crea la función *milstein*, en un fichero llamado milstein.m, cuyos parámetros son los mismos que los de la función *Euler*: $X(t_0) = X_0$; el intervalo de tiempo, cuyos extremos son *time1* y *time2*; y M, para la partición del intervalo temporal, que en esta simulación es equidistante. Además, utiliza una función c que depende del tiempo y de X_t, que representa la función de deriva $C(t, X_T)$; una función S que depende del tiempo y de X_t, que representa la difusión $\sigma(t, X_t)$; y la función derivada de S con respecto a x, de nombre *SSx*. Esta función devuelve t, la partición del intervalo temporal, y el vector y, que contiene la aproximación realizada. Entre dos puntos de tiempo cualesquiera, la trayectoria se aproxima por interpolación lineal.

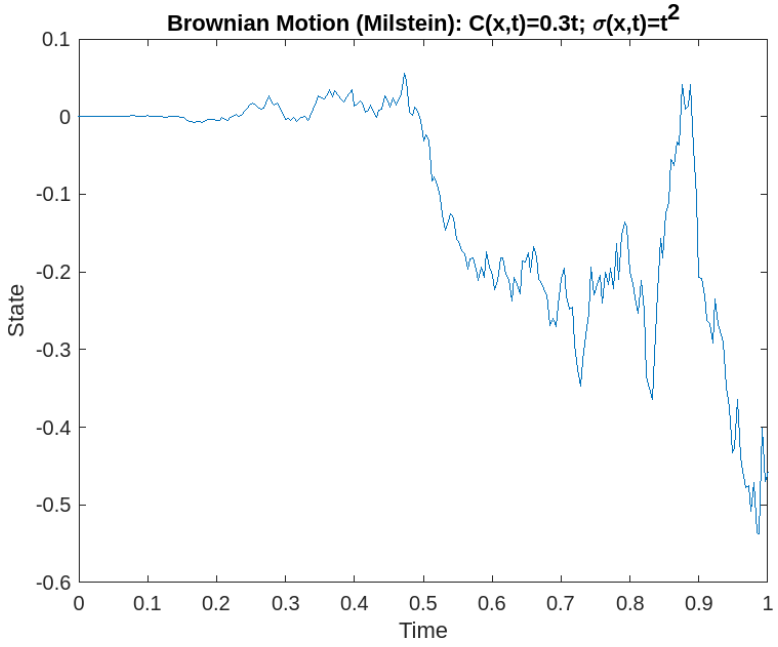
```
function [t,y]=milstein(xo,time1,time2,M)
        y=zeros(1,M+1);
        y(1)=xo;
        t=time1:(time2-time1)/M:time2;
        Z = normrnd(0,1,M);

        for i = 1:(length(t)-1)
                y(i+1)=y(i)+c(t(i),y(i))*(t(i+1)-t(i))+
                S(t(i),y(i))*(sqrt((time2-time1)/M)*Z(i))+
                0.5*S(t(i),y(i))*SSx(t(i),y(i))*((sqrt((time2-
                time1)/M)*Z(i))^2-(t(i+1)-t(i)));
        end
end
```

Para simular una trayectoria del proceso Browniano utilizando esta función, se definen, en el mismo fichero milstein.m, las funciones de deriva y de difusión como variables deterministas que varían en el tiempo. En el caso del proceso Browniano, la derivada con respecto de x de la función de difusión es cero.

```
function c=c(t,x)
        c=0.3*t;
end

function S=S(t,x)
        S=t^2;
end

function SSx=SSx(t,x)
        SSx=0;
end
```

y, a continuación, se evalúa la función *milstein* considerando el intervalo de tiempo $[0,1]$, $M = 250$ y $X_0 = 0$.

```
[t,y]=milstein(0,0,1,250);
plot(t,y)
xlabel('Time')
ylabel('State')
title('Brownian␣Motion␣(Milstein):␣C(x,t)=0.3t;␣\sigma(x,t)=t^2')
```

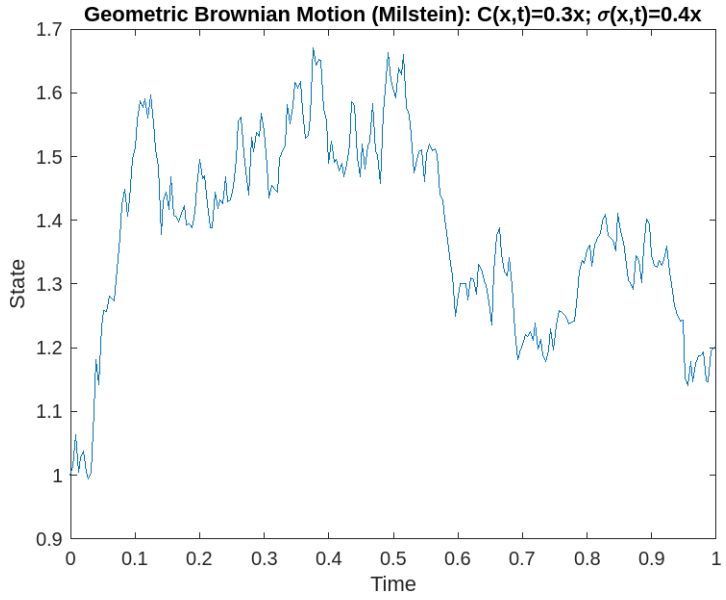


Se procede como anteriormente, en la implementación de los esquemas de aproximación para generar trayectorias del movimiento Browniano. Es decir, se declaran los parámetros de entrada de la función *milstein*, que incluyen la condición inicial, la función c, la función S, junto al resto de parámetros de entrada. De nuevo, se realiza una especificación lineal de las funciones S y c.

```
function c=c(t,x)
        c=0.3*x;
end

function S=S(t,x)
        S=0.4*x;
end

function SSx=SSx(t,x)
        SSx=0.4;
end
```

y, a continuación, se evalúa la función *milstein* considerando el intervalo de tiempo $[0,1]$, $M = 250$ y $X_0 = 1$.

```
[t,y]=milstein(1,0,1,250);
plot(t,y)
xlabel('Time')
ylabel('State')
title(['Geometric Brownian Motion (Milstein): C(x,t)=0.3x; ' ...
'\sigma(x,t)=0.4x'])
```



2.1.3. Movimiento Browniano: método basado en los incrementos

Otra forma de aproximar y simular las trayectorias de un proceso Browniano $\{B(t) : t \geq 0\}$es el método basado en los incrementos, que consiste en dividir el intervalo de tiempo $[0, T]$ en un cuadrícula de forma que

$$0 = t_1 < t_2 \cdots < t_{M+1} = T$$

considerando un paso de discretización constante $t_{i+1} - t_i = \Delta t$ para todo $i = 1, \ldots, M$. Así, se establece $B(0) = B(t_1) = B_0 = 0$ y se calculan los siguientes valores de B en los puntos de la cuadricula sumando una variable Normal de media cero y varianza 1 al valor en el punto anterior, siguiendo el algoritmo:

1. Se establece $i = 1$.
2. Se genera una (nueva) variable aleatoria $Z \sim N(0, 1)$.
3. Se avanza una unidad en i, es decir, se pasa a la siguiente iteración del bucle.
4. Se establece $B(t_{i+1}) - B(t_i) = Z\sqrt{\Delta t}$.
5. Si $i \leq M$ se itera el algoritmo.

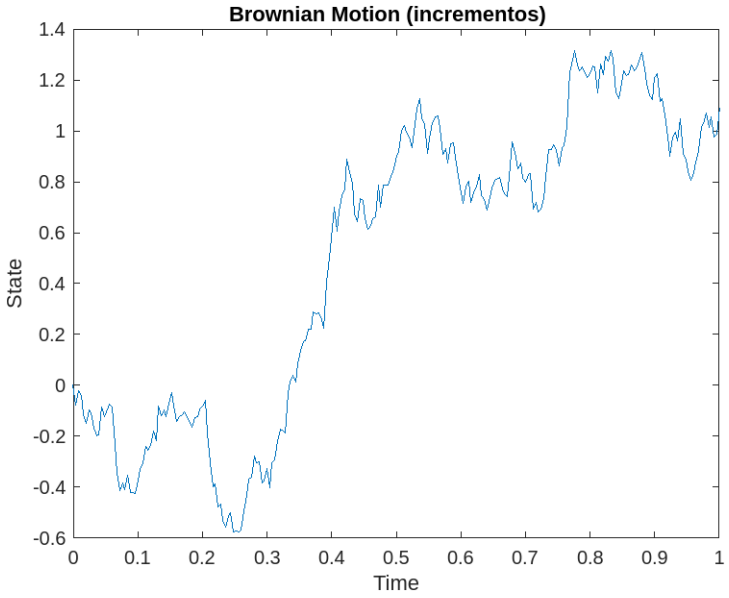
Finalmente, entre dos puntos cualesquiera t_i y t_{i+1} la trayectoria es aproximada por interpolación lineal.

Dicho algoritmo se implementa en la función *incrementos*, que sigue la misma filosofía que las funciones *Euler* y *milstein* anteriores.

```
function [t,b]=incrementos(xo,time1,time2,M)
        b=zeros(1,M+1);
        b(1)=xo;
        t=time1:(time2-time1)/M:time2;
```

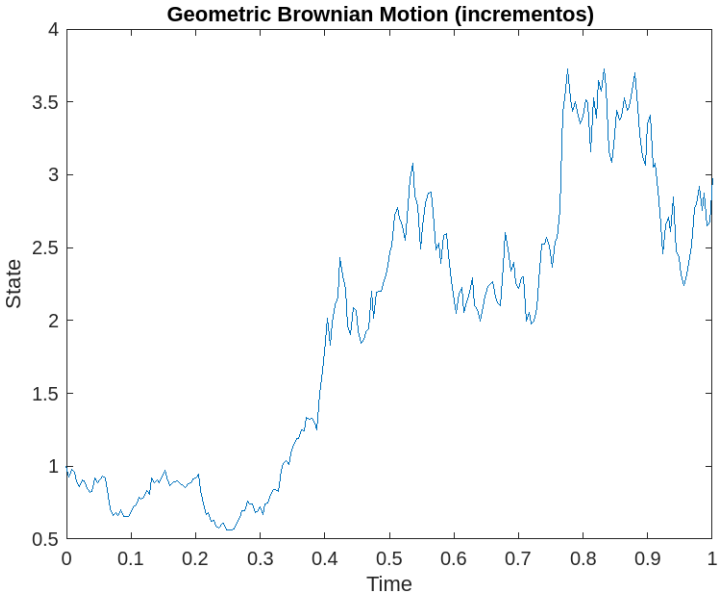
```
        Z = normrnd(0,1,M);

        for i = 1:(length(t)-1)
            b(i+1)=b(i)+sqrt((time2-time1)/M)*Z(i);
        end
end
```

```
[t,b]=incrementos(0,0,1,250);
plot(t,b)
xlabel('Time')
ylabel('State')
title('Brownian Motion (incrementos)')
```



Como se menciona en la sección 1.3, todos los métodos de simulación aplicables al proceso Browniano también lo son al proceso Browniano geométrico mediante una transformación exponencial. De esta forma, para generar una trayectoria del movimiento Browniano geométrico a partir de este ejemplo, basta con aplicar la función exponencial al objeto b generado.

```
plot(t,exp(b))
xlabel('Time')
ylabel('State')
title('Geometric Brownian Motion (incrementos)')
```



2.1.4. Movimiento Browniano: expansión Karhunen-Lóeve

Basándonos en el Teorema de Karhunen-Lóeve, se pueden aproximar las trayectorias de un movimiento Browniano representándolo como una combinación lineal infinita de funciones ortogonales de la forma

$$B(t) = \sum_{i=0}^{\infty} Z_i \phi_i(t)$$

con

$$\phi_i(t) = \frac{2\sqrt{2T}}{(2i+1)\pi} \sin \frac{(2i+1)\pi t}{2T}, \quad 0 \leq t \leq T,$$

donde Z_i, $i = 0, 1, \ldots$ son variables aleatorias independientes e idénticamente distribuidas según una $N(0,1)$, y $[0,T]$ es el intervalo de tiempo considerado.

Para la simulación de una aproximación finita de esta expansión se considera la partición

$$0 = t_1 < t_2 \cdots < t_{M-1} = T$$

considerando un paso de discretización constante $t_{i+1} - t_i = \Delta t$ para todo i.

En primer lugar se escribe la función $\phi_i(t)$ que depende de i y de t, y para su implementación se incluye el extremos del intervalo, T, como parámetro (*time*).

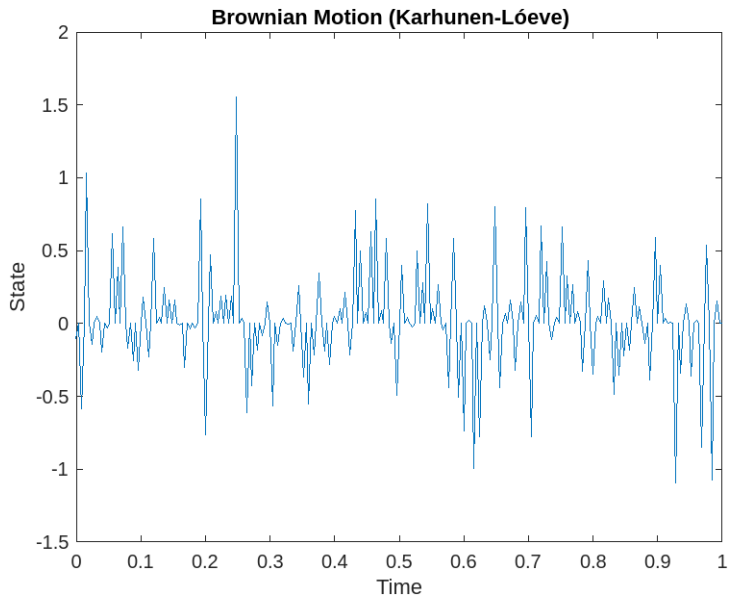
```
function phi=phi(t,i,time)
  phi=((2*sqrt(2*time))/((2*i+1)*pi))*sin(((2*i+1)*pi*t)/(2*time));
end
```

A continuación, se crea la función *KL* que realiza la aproximación finita (hasta 1000 en este caso) de la expansión de Karhunen-Lóeve.

```
function [t,b]=KL(time,M)
        b=zeros(1,M+1);
        t=0:time/M:time;
```

```
        for j = 1:(length(t)-1)
                Z = normrnd(0,1,1,1000);
                f = zeros(1,1000);

                for i = 1:1000
                        f(i)=phi(j,i-1,time);
                end
                b(j)=Z*f';
        end
end
```

```
time=1; %T
[t,b]=KL(time,250);
plot(t,b)
xlabel('Time')
ylabel('State')
title('Brownian Motion (Karhunen-Lóeve)')
```



2.2. Procesos de difusión. Simulación

Para lograr aproximar y simular las trayectorias de un proceso de difusión cualquiera, sin imponer restricciones a las funciones de deriva y de difusión, contamos con la función *sde* de MatLab, que genera un objeto sde (*Stochastic Differential Equation*) que se deriva de la ecuación diferencial estocástica (1.1),

$$dX_t = F(t, X_t)dt + G(t, X_t)dW(t)$$

donde ahora X_t y F son vectores de dimensión N, G es una matriz de dimensión $N \times M$ y dW_t es un vector de componentes Brownianos de dimensión M. El vector F es la función vectorial tasa de deriva evaluada y la matriz G es la evaluación de la función matricial de difusión.

La función *sde* tiene como parámetros F y G, que pueden introducirse de dos formas. La primera opción es introducirlos como funciones de MatLab con inputs el tiempo de observación t y un vector estático X_t de dimensión N. La segunda opción es introducir dichos parámetros como objetos *drift* y *diffusion* de MatLab, respectivamente, como veremos más adelante. En ambos casos, el objeto sde generado y proporciona las siguientes propiedades:

- StartTime: el tiempo inicial de la primera observación, que por defecto es 0.
- StartState: valores iniciales de las variables de estado, que por defecto es 1.
- Correlation: la correlación entre variables aleatorias gaussianas dibujadas para generar el vector proceso Browniano.
- Simulation: el método de simulación, que por defecto es la simulación por aproximación de Euler.
- Drift: la tasa de deriva.
- Diffussion: la función de difusión.

Además, estas propiedades pueden usarse como parámetros opcionales de la función *sde* especificando *Name*, el nombre de una de las propiedades, y *Value*, su valor correspondiente. Estos parámetros deben usarse de forma conjunta y puede utilizarse más de uno a la vez.

A ese objeto sde se le pueden aplicar las funciones *simulate*, *interpolate* y *simByEuler* vistas anteriormente, de la misma forma que se usaban con los objetos *bm* o *gbm*. La función *simBySolution*, sin embargo, no puede usarse con este modelo ya que, como se vio en el capítulo 1, es específica para el movimiento Browniano geométrico.

Ejemplo 1

A continuación se muestra un ejemplo del uso de las funciones *simulate* y *simByEuler* para aproximar y simular las trayectorias de un objeto sde. Para ello se cargan de nuevo los datos *Data_GlobalIdx2* disponibles en MatLab, a los que aplicamos la función *tick2ret* que convierte series de precios en series de retorno. Se definen los argumentos de la función *sde* a partir de los datos, usando la función media *mean*, la función desviación típica *std* y la función *corrcoef*, que calcula los coeficientes de correlación. Se crea el vector X de dimensión $nVariables \times 1$ cuyos elementos (entradas) son todos iguales a 100 y se crean las funciones F, tasa de deriva, y G, difusión, con argumentos el tiempo t y un vector de dimensión N: en el caso de la función F el vector es el producto del vector X por la media; y en el caso de la función G, el vector es el producto del vector X por la desviación típica. El objeto sde se crea con estas dos funciones F y G, indicando la correlación y los valores iniciales de las variables de estado.

Puesto que los parámetros obligatorios para el objeto sde son únicamente una función de tasa de deriva y una función de tasa de difusión, que se conocen a través de los valores $(t, X(t))$, es necesario introducir información adicional para determinar la dimensionalidad del modelo. Para ello, se especifica un vector de estado inicial, *StartState*.

```
load Data_GlobalIdx2
prices  = [Dataset.TSX Dataset.CAC Dataset.DAX Dataset.NIK
        Dataset.FTSE Dataset.SP];

returns =  tick2ret(prices);

nVariables  = size(returns,2); %N
expReturn   = mean(returns);
sigma       = std(returns);
correlation = corrcoef(returns);
t           = 0;
X           = 100*(ones(nVariables,1));

F = @(t,X) diag(expReturn)* X;
G = @(t,X) diag(X) * diag(sigma);

SDE = sde(F, G, 'Correlation', correlation, 'StartState', X);
```

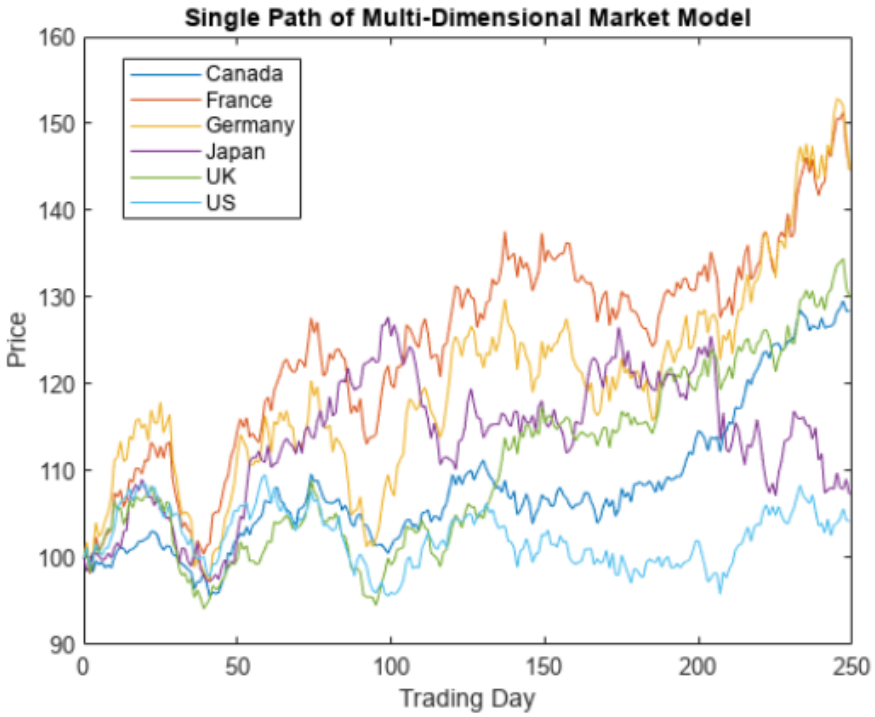
Se utilizan las funciones *simByEuler* y *simulate* con el número de períodos de observación 'nPeriods' e indicando que el incremento temporal es de un día, para generar una trayectoria. Con la orden ***whos*** *S* vemos que la dimensión de *S* es $(nPeriods + 1) \times nVariables$.

```
nPeriods = 249;
dt       =   1;      % incremento temporal=1 día
rng(142857,'twister')
[S,T] = simByEuler(SDE, nPeriods, 'DeltaTime', dt);

rng(142857,'twister')
[S,T] = simulate(SDE, nPeriods, 'DeltaTime', dt);

whos S
```

```
  Name        Size              Bytes  Class     Attributes

  S         250x6               12000  double
```

Con la función *plot* se grafica la trayectoria del modelo de mercado.

```
plot(T, S(:,:,1)), xlabel('Trading␣Day'), ylabel('Price')
title('Single␣Path␣of␣Multi-Dimensional␣Market␣Model')
legend({'Canada' 'France' 'Germany' 'Japan' 'UK' 'US'},
        'Location', 'Best')
```

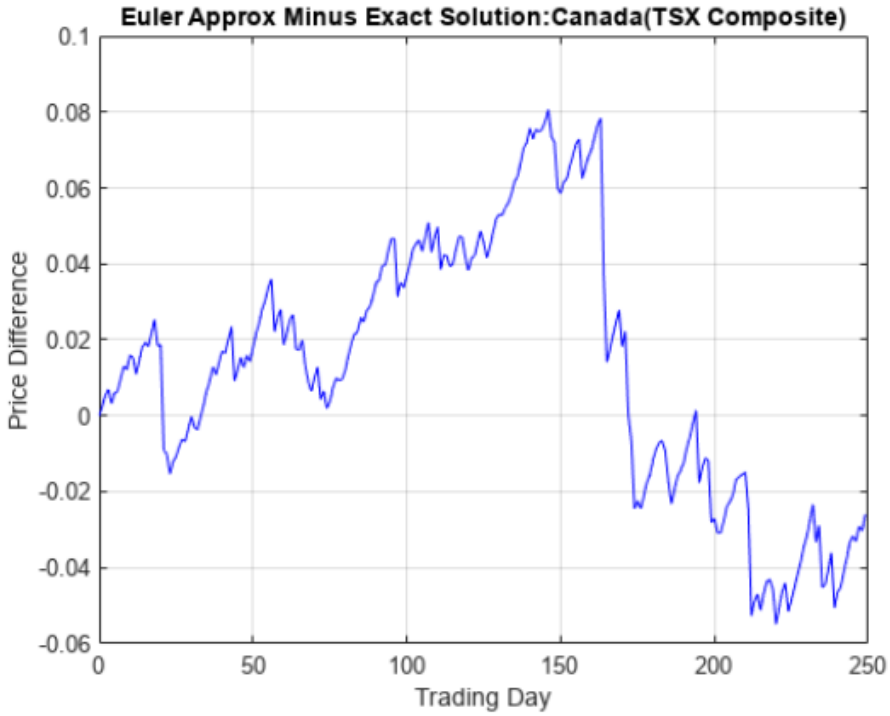


Ejemplo 2

Si consideramos el ejemplo anterior y el ejemplo 4 del capítulo anterior, vemos que se utilizan los mismo datos para estudiar el modelo con el objeto sde (cuyas trayectorias se almacenan en $[S, T]$) y el modelo Browniano geométrico con el objeto gbm (cuyas trayectorias se almacenan en el $[X, T]$), por lo que podemos graficar la diferencia entre las dos simulaciones, siendo la del objeto gbm, con la función *simBySolution* la solución exacta.

```
subplot(1,1,1)
plot(T, S(:,1,1) - X(:,1,1), 'blue'), grid('on')
xlabel('Trading Day'), ylabel('Price Difference')
title('Euler Approx Minus Exact Solution:Canada(TSX Composite)')
```

La siguiente gráfica muestra, por tanto, el error cometido al aproximar el modelo mediante un objeto sde, al no poder hacer uso de la *simBySolution*, que cuando los parámetros introducidos son constantes proporciona soluciones exactas.



2.2.1. Deriva y difusión. Simulación

La segunda forma de introducir las funciones F y G de la ecuación

$$dX_t = F(t, X_t)dt + G(t, X_t)dW(t)$$

anterior es utilizar las funciones *drift* y *difussion* de MatLab. Para poder hacer uso de estas dos funciones es necesario sustituir la función *sde* por la función *sdeddo* de MatLab, cuya sintaxis y finalidad son idénticas, la única diferencia es la forma de introducir los parámetros F y G.

La función *sdeddo* tiene como parámetros F, un objeto de clase drift, y G, un objeto de clase difussion. El objeto drift, creado con la función *drift*, especifica la tasa de deriva de la ecuación diferencial estocástica y el objeto difussion, creado con la función *difussion*, especifica la difusión de la ecuación.

La tasa de deriva introducida con la función *drift* tiene que ser de la forma

$$F(t, X_t) = A(t) + B(t)X_t$$

donde A y B son funciones del tiempo. Los parámetros de la función son A, un vector de dimensión N, y B, una matriz de dimensión $N \times N$.

La tasa de difusión introducida con la función *difussion* ha de ser una función matricial de la forma

$$G(t, X_t) = D(t, X_t^{\alpha(t)})V(t)$$

donde D es una función evaluada en el tiempo y en $X_t^{\alpha(t)}$; la matriz $X_t^{\alpha(t)}$ denota una matriz

diagonal de dimensión $N \times N$, cuya diagonal es el vector X_t elevado al vector $\alpha(t)$; α es una función vectorial del tiempo, que devuelve un vector de dimensión N; y V es una función matricial del tiempo, que devuelve una matriz de dimensión $N \times N$.

Los parámetros de la función *difussion* son D, que se puede introducir bien como un vector columna de dimensión N o bien como una función determinista del tiempo o como una función de t y de X_t que genere un vector columna de dimensión N; y V, que se puede introducir bien como una matriz $N \times N$, donde cada fila se corresponde a una variable de estado y cada columna corresponde a una fuente de aleatoriedad introducida por el Browniano, o bien se puede introducir como una función determinista del tiempo o como una función de t y de X_t que genere una matriz de dimensión $N \times N$.

A continuación, se muestra un ejemplo del uso de las funciones *sdeddo*, *drift* y *difussion*, que utiliza los datos *Data_GlobalIdx2* de MatLab.

```
load Data_GlobalIdx2
prices   = [Dataset.TSX Dataset.CAC Dataset.DAX Dataset.NIK
        Dataset.FTSE Dataset.SP];

returns =  tick2ret(prices);

nVariables  = size(returns, 2);
expReturn   = mean(returns);
sigma       = std(returns);
correlation = corrcoef(returns);

F = drift(zeros(nVariables,1), diag(expReturn))
```

```
F =
   Class DRIFT: Drift Rate Specification
   -------------------------------------
      Rate: drift rate function F(t,X(t))
         A: 6x1 double array
         B: 6x6 diagonal double array
```

```
G = diffusion(ones(nVariables,1), diag(sigma))
```

```
G =
   Class DIFFUSION: Diffusion Rate Specification
   ---------------------------------------------
       Rate: diffusion rate function G(t,X(t))
      Alpha: 6x1 double array
      Sigma: 6x6 diagonal double array
```

```
SDEDDO = sdeddo(F, G, 'Correlation', correlation, 'StartState', 100)
```

```
SDEDDO =
   Class SDEDDO: SDE from Drift and Diffusion Objects
   ---------------------------------------------------
     Dimensions: State = 6, Brownian = 6
   ---------------------------------------------------
      StartTime: 0
     StartState: 100 (6x1 double array)
    Correlation: 6x6 double array
          Drift: drift rate function F(t,X(t))
      Diffusion: diffusion rate function G(t,X(t))
     Simulation: simulation method/function simByEuler
              A: 6x1 double array
              B: 6x6 diagonal double array
          Alpha: 6x1 double array
          Sigma: 6x6 diagonal double array
```

2.3. Proceso de Poisson compuesto

Un proceso de Poisson compuesto es un proceso estocástico $\{A(t)\}$ tal que $A(t) = \sum_{i=1}^{N(t)} Y_i$, donde $\{N(t)\}$ es un proceso de Poisson y Y_i son variables aleatorias i.i.d., independientes de $\{N(t)\}$ [33].

2.3.1. Simulación

Para simular las trayectorias del proceso de Poisson compuesto en el intervalo $[0, T]$ y se sigue el siguiente algoritmo:

1. Se simula la variable aleatoria de Poisson, $N(T)$, con parámetro λt y se llama $N(T) = M$.

2. Se simulan M variables aleatorias independientes e idénticamente distribuidas $U_1, \ldots, U_M$ según una distribución uniforme $\mathcal{U}(0, T]$, para conseguir una discretización del intervalo en los siguientes pasos.

3. Se ordenan las variables estadísticas anteriores para obtener las variables ordenadas (mediante la función *sort* de MatLab) $U_{(1)} \leq \ldots \leq U_{(M)}$, de forma que
   $U_{(1)} = \min(U_1, \ldots, U_M), \ldots, U_{(M)} = \max(U_1, \ldots, U_M)$.

4. Se establece $t_{(0)} = 0, t_{(1)} = U_{(1)}, \ldots, t_{(M)} = U_{(M)}$, obteniendo así la discretización del intervalo del tiempo, y la trayectoria del proceso de Poisson en $[0, T]$ es una función escalonada tal que $N(t) = k$ si $t_{(k-1)} < t \leq t_{(k)}$, $k = 1, \ldots, M + 1$.

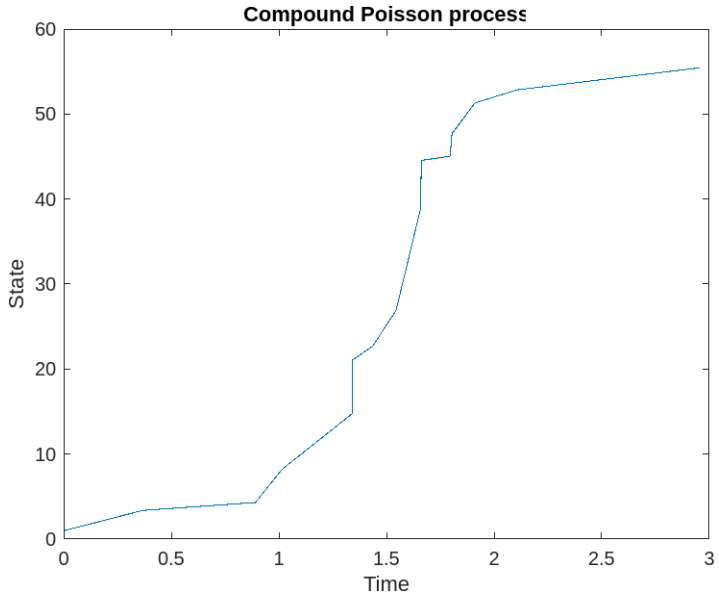
5. Se simulan $M + 1$ variables aleatorias independientes e idénticamente distribuidas Y_i con función de densidad de probabilidad $b(x)$ como exponencial, gamma, gamma invertida, Fisher, etc. En este caso se usa la distribución exponencial con parámetro $1/\lambda$.

6. Se obtiene la trayectoria de $A(t) = \sum_{i=1}^{N(t)} Y_i$, $0 \leq t \leq T$.

En el siguiente código se muestran los pasos anteriores en lenguaje MatLab. La función $N(t)$ creada en el código añade el instante t al vector que contiene la discretización del intervalo temporal (*time*), ordena el nuevo vector (*arrt*) y busca el instante t. De esta forma, se consigue la posición de t en *arrt*, que se corresponde con el subíndice k del paso 4.

```
function [time,S]=poissoncomp(lambda,TT)
        M=poissrnd(lambda*TT,1,1);              %paso 1
        time=[0  sort(unifrnd(0,TT,1,M))];      %paso 2, 3, 4
        function N=N(t)
                arrt=[time t];
                arrt=sort(arrt);
                N=find(arrt==t,1);
        end
        Y=exprnd(lambda,1,M+1);                 %paso 5
        function SS=SS(t)                       %paso 6
                SS=sum(Y(1:N(t)));
        end
        S=zeros(1,length(time));
        for i = 1:length(time)
                S(i)=SS(time(i));
        end
end
```

```
lambda=4;
TT=3;
[time,S]=poissoncomp(lambda,TT);
plot(time,S)
xlabel('Time')
ylabel('State')
title('Compound Poisson process')
```



2.4. Proceso de difusión con saltos

Un proceso de difusión con saltos viene dado por la siguiente ecuación diferencial estocástica:

$$dX_t = c(X_t)dt + \sigma(X_t)dB(t) - dS(t), t \geq 0, \tag{2.1}$$

$$S(t) = \sum_{i=1}^{N(t)} Z_i,$$

donde $c(x)$ es una tasa superior dependiente del estado, $B(t)$ es un movimiento Browniano estándar, $\sigma(X_t)dB(t)$ introduce una fuente de volatilidad en la acumulación de tasas, $N(t)$ es un proceso de Poisson independiente de intensidad λ, y $S(t)$ es un proceso de Poisson compuesto que modela las componentes Z_i acumuladas [33]. Denotamos por $b(x)$ la intensidad de las componentes i.i.d. (independientes e idénticamente distribuidas) Z_i.

Las soluciones de estas ecuaciones diferenciales estocásticas son los procesos de difusión a saltos[33]. El proceso de difusión a saltos X_t es Markoviano, con generador infinitesimal afín:

$$\Gamma h(x) := \Gamma_d h(x) + \Gamma_j h(x)$$

donde

$$\Gamma_d h(x) = c(x)\frac{\partial h(x)}{\partial x} + \frac{\sigma(x)^2}{2}\frac{\partial^2 h(x)}{\partial x^2}$$

y

$$\Gamma_j h(x) = \lambda \int_0^\infty [h(x-z) - h(x)]\, \nu(dz)$$

con $\nu(dz) = \lambda\, b(z)dz$ la medida de Lévy.

Se pueden estudiar "procesos de riesgo" dados por procesos de difusión a saltos, con saltos negativos, que son soluciones de la ecuación diferencial estocástica anterior, donde las componentes Z_i son las deudas o reclamaciones en el proceso de riesgo. Los procesos con saltos se han utilizado ampliamente para describir la evolución aleatoria de dinámica neuronal [27], dinámica de la humedad del suelo [15] o cifras financieras como precios de acciones, índices de mercado y tasas de interés [34].

2.4.1. Simulación

Haciendo uso de la fórmula de Itô, podemos simular las trayectorias de un proceso de difusión con saltos simulando la primera parte de la ecuación (2.1) siguiendo los métodos descritos anteriormente para la aproximación de Euler y el Esquema de Milstein, combinados con los pasos descritos para simular el proceso de Poisson compuesto. La aproximación de la solución de la ecuación, es decir, del proceso de difusión con saltos es la diferencia de estas dos simulaciones [33].

En primer lugar, se introducen las funciones de deriva y de difusión. En el caso de usar el Esquema de Milstein se introduce también la derivada con respecto a x de la función de difusión.

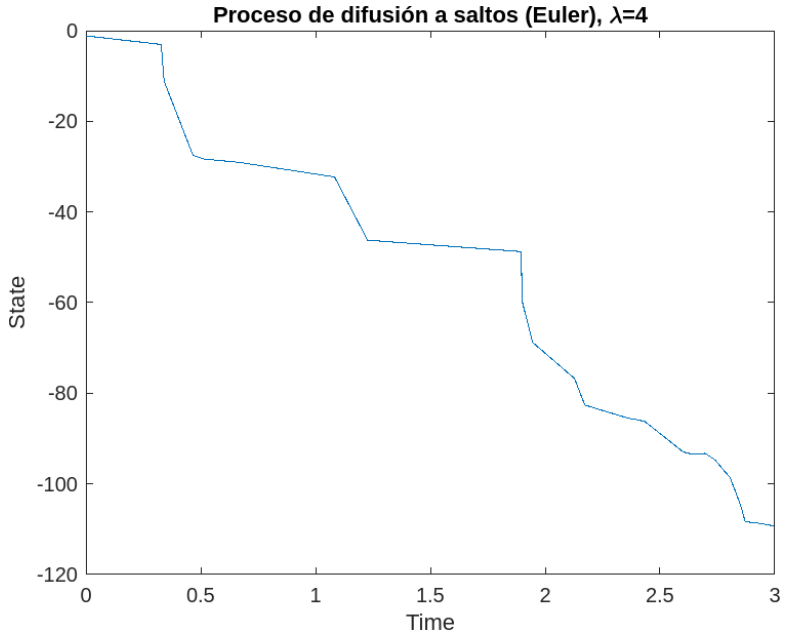
```
function c=c(t,x)    %deriva
        c=0.3*x;
end
```

```
function S=S(t,x)    %difusión
        S=0.4*x;
end

function SSx=SSx(t,x)
        SSx=0.4;
end
```

A continuación, se realizan la aproximación de Euler y la aproximación mediante el Esquema de Milstein, así como la simulación del proceso de Poisson compuesto, teniendo en cuenta que el intervalo temporal tiene que ser el mismo para todas las simulaciones. Se obtienen de esta forma dos aproximaciones distintas del proceso de difusión con saltos. Tenemos que tener en cuenta que se necesita la misma discretización del intervalo temporal, por lo que se cambia ligeramente las funciones *Euler* y *milstein* para utilizar el vector de tiempo generado con la función *poissoncomp*.

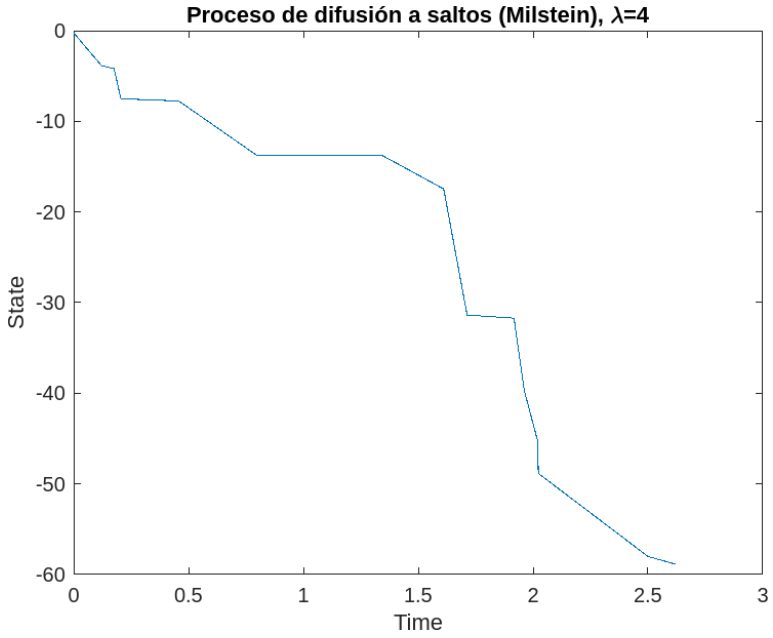
```
% Aproximación de Euler

function [t,y]=Euler(xo,TT,time)
        M=length(time)-1;
        y=zeros(1,M+1);
        y(1)=xo;
        t=time;
        Z = normrnd(0,1,M);

        for i = 1:(length(t)-1)
                y(i+1)=y(i)+c(t(i),y(i))*(t(i+1)-t(i))+
                S(t(i),y(i))*(sqrt(TT/M)*Z(i));
        end
end
```

```
lambda=4;
TT=3;
[time,yy]=poissoncomp(lambda,TT);

[t,y]=Euler(1,3,time);

plot(t,y-yy)
xlabel('Time')
ylabel('State')
title('Proceso de difusión a saltos (Euler), \lambda=4')
```




```
% Esquema de Milstein

function [t,y]=milstein(xo,TT,time)
        M=length(time)-1;
        y=zeros(1,M+1);
        y(1)=xo;
        t=time;
        Z = normrnd(0,1,M);

        for i = 1:(length(t)-1)
            y(i+1)=y(i)+c(t(i),y(i))*(t(i+1)-t(i))+
            S(t(i),y(i))*(sqrt(TT/M)*Z(i))+
            0.5*S(t(i),y(i))*SSx(t(i),y(i))*((sqrt(TT/M)*Z(i))^2
            -(t(i+1)-t(i)));
        end
end
```

```
lambda=4;
TT=3;
[time,yy]=poissoncomp(lambda,TT);

[t,y]=milstein(1,3,time);

plot(t,y-yy)
xlabel('Time')
ylabel('State')
title('Proceso de difusión a saltos (Milstein), \lambda=4')
```


Proceso de difusión a saltos (Milstein), λ=4
State
Time
0
-10
-20
-30
-40
-50
-60
0
0.5
1
1.5
2
2.5
3

3 Difusiones fraccionarias anómalas sobre la esfera

En este capítulo estudiamos las estructuras de correlación de los campos aleatorios dependientes del tiempo que surgen en el análisis de las soluciones de dos tipos de ecuaciones fraccionarias sobre la esfera unitaria $\mathbf{S}_1^2 \subset \mathbb{R}^3$, que muestran memoria larga (Teorema 1) y memoria corta (Teorema 2) [13].

Consideramos un movimiento Browniano con valores en la esfera y con parámetro el tiempo, esto es, con espacio de estados restringido, $B : (\Omega, \mathscr{A}, P) \times \mathbb{R}_0^+ \to \mathbf{S}_1^2$. Adicionalmente, consideramos un campo aleatorio real valuado $T : \mathbf{S}_1^2 \to \mathbb{R}$ isotrópico gaussiano para el cual se mantiene la expansión en términos de funciones esféricas harmónicas [25]. Vamos a estudiar los campos aleatorios dependientes del espacio-tiempo $X = T \circ B$ sobre la esfera $\mathbf{S}_1^2$ gobernados por diferentes ecuaciones diferenciales estocásticas, de forma que X es una transformación del espacio de estados de las componentes del movimiento Browniano y por tanto depende del parámetro temporal $t \in \mathbb{R}_0^+$ como $X_t(x) = T(x)$, siendo $x = B(t) \in \mathbf{S}_1^2$.

Vamos a estudiar, en primer lugar, los campos aleatorios que surgen como soluciones al problema de Cauchy

$$\begin{cases} \left(\gamma - \mathbb{D}_M + \frac{\partial^\beta}{\partial t^\beta}\right) X_t(x) = 0, \;\; x \in \mathbf{S}_1^2,\, t > 0,\, 0 < \beta < 1,\, \gamma > 0 \\ X_0(x) = T(x) \end{cases} \tag{3.1}$$

donde $\mathbb{D}_M$ es un operador diferencial adecuado, que se define a continuación, y $\frac{\partial^\beta}{\partial t^\beta}$ es la derivada fraccionaria de Dzerbayshan–Caputo. Al obtener la solución $X_t(x)$ de (3.1), se comprobará que su función de covarianza muestra un comportamiento de memoria larga.

Después, consideraremos la ecuación fraccionaria no homogénea

$$(\gamma - \mathbb{D}_M)^\beta X(x) = T(x), \;\; x \in \mathbf{S}_1^2,\, 0 < \beta < 1, \tag{3.2}$$

cuya extensión dependiente del tiempo es

$$\left(\gamma - \mathbb{D}_M - \varphi \frac{\partial}{\partial t}\right)^\beta X_t(x) = T_t(x), \;\; x \in \mathbf{S}_1^2,\, t > 0,\, 0 < \beta < 1,\; \gamma > 0,\; \varphi \geq 0. \tag{3.3}$$

Obtendremos como solución de (3.3) un campo aleatorio en la esfera cuya función de covarianza muestra un comportamiento de memoria corta.

El par $(B_t, T(x + B_t))$ describe un movimiento aleatorio en la esfera de radio unitario con una dinámica gobernada por las ecuaciones estocásticas fraccionarias (3.1) y (3.3). Este tipo de campos aleatorios se consideran en el análisis de la radiación cósmica de fondo de microondas (radiación CMB), en cuyo caso la estructura de correlación y el espectro de potencia angular resultan muy importantes [14][25]. Además, las difusiones sobre la esfera surgen en varios contextos: en el suavizado de superficie en computación gráfica [10] y en los patrones de migración global de los mamíferos marinos [7], por ejemplo. A nivel celular, la difusión es un modo importante de transporte de sustancias ya que, en general, las membranas biológicas son superficies curvas.

3.1. Campos aleatorios esféricos

En primer lugar, se introducen notaciones y algunas propiedades de los campos aleatorios isotrópicos en la esfera [25]. Consideramos el campo aleatorio gaussiano isotrópico integrable en media cuadrática

$$\left\{T(x);\, x \in \mathbf{S}_1^2\right\} \tag{3.4}$$

en la esfera $\mathbf{S}_1^2 = \left\{x \in \mathbb{R}^3 :\, |x| = 1\right\}$, para la cual

$$\mathbb{E}T(gx) = \mathbb{E}T(x) = 0$$

$$\mathbb{E}T^2(gx) = \mathbb{E}T^2(x) = c$$

$$\mathbb{E}[T(gx_1)T(gx_2)] = \mathbb{E}[T(x_1)T(x_2)] \text{ para } x_1,\, x_2 \in \mathbf{S}_1^2 \text{ arbitrarios}$$

para todo $g \in SO(3)$, donde $SO(3)$ es el grupo de rotaciones en $\mathbb{R}^3$ y c es una constante. Consideraremos la representación espectral

$$T(x) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} a_{l,m} \mathscr{Y}_{l,m}(x) = \sum_{l=0}^{\infty} T^l(x), \tag{3.5}$$

donde

$$a_{l,m} = \int_{\mathbf{S}^2} T(x) \mathscr{Y}_{l,m}^*(x) \lambda(dx), \;\; -l \leq m \leq l,\; l \geq 0 \tag{3.6}$$

son los coeficientes aleatorios de Fourier de T, $\mathscr{Y}_{l,m}(x) = (-1)^m \mathscr{Y}_{l,-m}^*(x)$ y $\mathscr{Y}_{l,m}^*(x)$ es el conjugado de $\mathscr{Y}_{l,m}(x)$. Las funciones esféricas harmónicas $\mathscr{Y}_{l,m}(\vartheta, \varphi)$ se definen como

$$\mathscr{Y}_{l,m}(\vartheta, \varphi) = \sqrt{\frac{2l+1}{4\pi} \frac{(l-m)!}{(l+m)!}} Q_{l,m}(cos\vartheta) e^{im\varphi}, \;\; 0 \leq \vartheta \leq \pi,\; 0 \leq \varphi \leq 2\pi$$

para todo $l \geq 0$, $|m| \leq l$. Para $m \geq 0$,

$$Q_{l,m}(z) = (-1)^m (1-z^2)^{\frac{m}{2}} \frac{d^m}{dz^m} Q_l(z) \;\; |z| < 1,\, |m| \leq l,$$

y

$$Q_{l,m}(z) = 0, \;\; m > l$$

son las funciones de Legendre asociadas y Q_l son los polinomios de Legendre con representación de Rodrigues, para $l \geq 0$,

$$Q_l(z) = \frac{1}{2^l l!} \frac{d^l}{dz^l} (z^2-1)^l, \;\; |z| < 1.$$

La convergencia en (3.5) se estudiará en el sentido de que

$$lim_{L \longrightarrow \infty} \mathbb{E} \left[\int_{\mathbf{S}_1^2} \left(T(x) - \sum_{l=0}^{L} \sum_{m=-l}^{+l} a_{l,m} \mathscr{Y}_{l,m}(x) \right)^2 \lambda(dx) \right] = 0 \tag{3.7}$$

donde $\lambda(dx)$ es la medida de Lebesgue en la esfera $\mathbf{S}_1^2$ y, para todo $x \in \mathbf{S}_1^2$ y $0 \leq \vartheta \leq \pi$,

$0 \leq \varphi \leq 2\pi$:

$$\lambda(dx) = \lambda(d\vartheta, d\varphi) = d\varphi d\vartheta sin\vartheta$$

y

$$x = (sin\vartheta cos\varphi,\ sin\vartheta sin\varphi,\ cos\vartheta).$$

Vamos a usar la siguiente notación

$$\sum_{l=0}^{\infty} \sum_{m=-l}^{+l} = \sum_{lm}$$

y escribimos $f(x)$ en lugar de $f(\vartheta, \varphi)$ cuando no haya confusión.

Los coeficientes aleatorios (3.6) son variables aleatorias complejas gaussianas de media cero tales que [4]

$$\mathbb{E}[a_{l,m} a^*_{l',m'}] = \delta_l^{l'} \delta_m^{m'} \mathbb{E}|a_{l,m}|^2, \tag{3.8}$$

donde δ_a^b es el símbolo de Kronecker y

$$\mathbb{E}|a_{l,m}|^2 = C_l,\ \ l \geq 0 \tag{3.9}$$

es el espectro de potencia angular del campo aleatorio T que, bajo el supuesto de gaussianidad, caracteriza completamente la estructura de dependencia de T.

Recordamos que las funciones esféricas harmónicas resuelven

$$\Delta_{\mathbb{S}_1^2} \mathscr{Y}_{l,m} = -\mu_l \mathscr{Y}_{l,m},\ \ l \geq 0,\ |m| \leq l \tag{3.10}$$

donde $\mu_l = l(l+1)$ y

$$\Delta_{\mathbb{S}_1^2} = \frac{1}{sin^2\vartheta} \frac{\partial^2}{\partial \varphi^2} + \frac{1}{sin\vartheta} \frac{\partial}{\partial \vartheta} \left(sin\vartheta \frac{\partial}{\partial \vartheta} \right)$$

es el operador esférico de Laplace o el operador de Laplace–Beltrami.

Teniendo en cuenta (3.8) y usando la fórmula de suma para funciones esféricas harmónicas

$$\sum_{m=-l}^{+l} \mathscr{Y}_{l,m}(y) \mathscr{Y}^*_{l,m}(x) = \frac{2l+1}{4\pi} Q_l\left(\langle x, y \rangle\right) \tag{3.11}$$

y el producto interno

$$\langle x, y \rangle = cos\, d(x,y) = cos\vartheta_x cos\vartheta_y + sin\vartheta_x sin\vartheta_y cos(\varphi_x - \varphi_y),$$

donde $d(x,y)$ es la distancia esférica entre los puntos x, y; la función de covarianza de $T(x)$ se escribe

$$\mathbb{E}[T(x)T(y)] = \sum_{lm} C_l \mathscr{Y}_{l,m}(x) \mathscr{Y}^*_{l,m}(y) = \sum_{l} C_l \frac{2l+1}{4\pi} Q_l\left(\langle x, y \rangle\right). \tag{3.12}$$

3.2. Subordinación y operadores fraccionarios

Sea $F(t)$, $t \geq 0$, un subordinador Lévy con función característica

$$\mathbb{E}e^{i\xi F(t)} = e^{-t\phi(\xi)} = e^{-t(ib\xi + \int_0^\infty (e^{i\xi y}-1)M(dy))}, \quad \xi \in \mathbb{R} \tag{3.13}$$

donde $b \geq 0$ es la deriva y $M(.)$ es la medida de Lévy en $\mathbb{R}_+ \setminus \{0\}$ satisfaciendo las condiciones:

$$\int_0^\infty (y \wedge 1)M(dy) < \infty \quad y \quad M(-\infty, 0) = 0.$$

La función característica del subordinador $F(t)$, $t \geq 0$, definida anteriormente, se puede escribir como

$$\mathbb{E}e^{-\xi F(t)} = e^{-t\Psi(\xi)} = e^{t\phi(i\xi)} = e^{-t(b\xi + \int_0^\infty (1-e^{-\xi y})M(dy))}, \quad \xi \geq 0 \tag{3.14}$$

donde $\Psi(\xi)$ es el exponente de Laplace de F. Si $F(t)$, $t \geq 0$, es el subordinador β-estable, entonces $\Psi(\xi) = \xi^\beta$, $\beta \in (0,1)$. De aquí en adelante, asumimos $b = 0$.

Usando (3.11), escribimos la densidad de transición de un movimiento Browniano en la esfera unitaria [40]

$$\begin{aligned} Pr\{x + B_t \in dy\}/dy &= Pr\{B_t \in dy | B_0 = x\}/dy \\ &= \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} e^{-t\mu_l} \mathscr{Y}_{l,m}(y)\mathscr{Y}^*_{l,m}(x) \\ &= \sum_l e^{-t\mu_l} \frac{2l+1}{4\pi} Q_l(\langle x, y\rangle) \end{aligned} \tag{3.15}$$

y escribimos

$$P_t f(x) = \mathbb{E}f(x + B_t) = \int_{\mathbf{S}_1^2} f(y) Pr\{x + B_t \in dy\} \tag{3.16}$$

donde $P_t f(x)$ es la solución del problema de valores iniciales

$$\begin{cases} \frac{\partial u}{\partial t} = \Delta_{\mathbf{S}_1^2} u, & x \in \mathbf{S}_1^2,\ t > 0 \\ u(x,0) = f(x) \end{cases} \tag{3.17}$$

para una función medible $f(x)$, $x \in \mathbf{S}_1^2$.

Sea f una función de cuadrado integrable en la esfera unitaria, es decir, $f \in L^2(\mathbf{S}_1^2)$. Definimos el siguiente operador

$$\mathbb{D}_M f(x) := \int_0^\infty (P_t f(x) - f(x)) M(dt) \tag{3.18}$$

donde, a partir de (3.15) y (3.16), tenemos que

$$P_t f(x) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} e^{-t\mu_l} \mathscr{Y}_{l,m}(x) f_{l,m} \tag{3.19}$$

y $f_{l,m}$ son los coeficientes de Fourier de f.

Sea $s > 0$. Introducimos el espacio de Sobolev [3]

$$H^s(\mathbf{S}_1^2) = \left\{ f \in L^2(\mathbf{S}_1^2) : \sum_{l=0}^{\infty} (2l+1)^{2s} f_l < \infty \right\} \tag{3.20}$$

donde

$$f_l = \sum_{|m|\leq l} |f_{l,m}|^2 = \sum_{|m|\leq l} \left| \int_{\mathbf{S}_1^2} f(x) \mathscr{Y}_{l,m}^*(x) \lambda(dx) \right|^2, \quad l = 0, 1, 2, \ldots$$

Proposición 1. Sea Ψ el símbolo de un subordinador introducido en (3.14). Sea $f \in H^s(\mathbf{S}_1^2)$ y $s > 4$. Entonces, tenemos la siguiente propiedad del operador $\mathbb{D}_M$:

$$\mathbb{D}_M f(x) = -\sum_{l=0}^{\infty} \sum_{m=-l}^{+l} f_{l,m} \mathscr{Y}_{l,m}(x) \Psi(\mu_l) \tag{3.21}$$

donde

$$f_{l,m} = \int_{\mathbb{S}_1^2} f(x) \mathscr{Y}_{l,m}^*(x) \lambda(dx)$$

son los coeficientes de Fourier de f.

Demostración. La serie (3.21) converge absoluta y uniformemente. En efecto, $f_l < l^{-2s}$ con $s > 4$ (siendo $f \in H^s(\mathbf{S}_1^2)$), $||Y_{l,m}||_\infty < l^{1/2}$ [39] y $\Psi(\mu_l) \leq l^2$ y por tanto, considerando que

$$\sum_m |f_{l,m}| \leq \left(\sum_m |f_{l,m}|^2 \right)^{\frac{1}{2}} (2l+1)^{\frac{1}{2}} = \sqrt{(2l+1) f_l} < C l^{-s+\frac{1}{2}}$$

para alguna constante positiva C, obtenemos la proposición. □

El operador (3.18) se puede reescribir como

$$\mathbb{D}_M f(x) = \int_{\mathbf{S}_1^2} (f(y) - f(x)) \hat{J}(x, y) \lambda(dy) \tag{3.22}$$

donde λ es la medida de Lebesgue en $\mathbf{S}_1^2$ y

$$\hat{J}(x, y) = \sum_{l=0}^{\infty} \frac{2l+1}{4\pi} Q_l \left(\langle y, x \rangle \right) \hat{\Psi}(\mu_l)$$

con $\hat{\Psi}(\mu) = \int_0^\infty e^{-s\mu} M(ds)$ si existe la integral. De hecho, podemos escribir

$$\begin{aligned}
\mathbb{D}_M f(x) &= \int_0^\infty (P_s f(x) - f(x)) M(ds) \\
&= \int_0^\infty \mathbb{E}[(f(x + B_s) - f(x))] M(ds) \\
&= \int_0^\infty \int_{\mathbf{S}_1^2} (f(y) - f(x)) Pr\{x + B_s \in dy\} M(ds) \\
&= \int_{\mathbf{S}_1^2} (f(y) - f(x)) \hat{J}(x, y) \lambda(dy)
\end{aligned}$$

donde

$$\begin{aligned}\hat{J}(x,y)\lambda(dy) &= \int_0^\infty Pr\{x + B_s \in dy\}M(ds) \\ &= \lambda(dy)\sum_l \frac{2l+1}{4\pi} Q_l\left(\langle y,x\rangle\right)\int_0^\infty e^{-s\mu_l}M(ds) \\ &= \lambda(dy)\sum_l \frac{2l+1}{4\pi} Q_l\left(\langle y,x\rangle\right)\hat{\Psi}(\mu_l).\end{aligned}$$

Además, de (3.19) y usando (3.11) en el último paso, el operador (3.18) se puede escribir de la siguiente manera

$$\begin{aligned}\mathbb{D}_M f(x) &= \int_0^\infty (P_s f(x) - P_0 f(x))M(ds) \\ &= \sum_{lm} f_{l,m}\mathscr{Y}_{l,m}(x)\int_0^\infty (e^{-s\mu_l} - 1)M(ds) \\ &= [3.14] = -\sum_{lm} f_{l,m}\mathscr{Y}_{l,m}(x)\Psi(\mu_l) \\ &= -\sum_{lm}\left(\int_{\mathbf{S}_1^2} f(y)\mathscr{Y}^*_{l,m}(y)\lambda(dy)\right)\mathscr{Y}_{l,m}(x)\Psi(\mu_l) \\ &= -\int_{\mathbf{S}_1^2} f(y)\left(\sum_{lm}\Psi(\mu_l)\mathscr{Y}_{l,m}(x)\mathscr{Y}^*_{l,m}(y)\right)\lambda(dy) \\ &= -\int_{\mathbf{S}_1^2} f(y)J(x,y)\lambda(dy)\end{aligned}$$

donde

$$J(x,y) = \sum_{l=0}^{\infty}\sum_{m=-l}^{+l}\Psi(\mu_l)\mathscr{Y}_{l,m}(x)\mathscr{Y}^*_{l,m}(y) = \sum_{l=0}^{\infty}\Psi(\mu_l)\frac{2l+1}{4\pi}Q_l\left(\langle x,y\rangle\right). \tag{3.23}$$

Atendiendo a estos resultados, se tiene que podemos dar dos formas alternativas del operador $\mathbb{D}_M$.

Ahora se define el siguiente semigrupo que será útil posteriormente.

Definición 1. $\mathbb{P}_t := exp(t\mathbb{D}_M)$ es el semigrupo asociado a (3.22) de símbolo $\hat{\mathbb{P}}_t = exp(-t\Psi)$ donde $-\Psi$ es el multiplicador de Fourier de $\mathbb{D}_M$.

3.3. Ecuaciones de evolución pseudofraccionarias sobre la esfera

Recordamos la derivada fraccionaria Dzerbayshan-Caputo (D-C)

$$\frac{\partial^\beta u}{\partial t^\beta}(x,t) = \frac{1}{\Gamma(1-\beta)}\int_0^t \frac{\partial u(x,s)}{\partial s}\frac{ds}{(t-s)^\beta} \tag{3.24}$$

para $0 < \beta < 1$, $x \in \mathbb{R}$, $t > 0$ [26]. La derivada fraccionaria D–C está relacionada con el inverso de un subordinador estable, sea $\mathfrak{L}_t^\beta$, en el sentido de que $u(x,t) = Pr\{\mathfrak{L}_t^\beta \in dx\}/dx$ resuelve la ecuación fraccionaria $\frac{\partial^\beta u}{\partial t^\beta}(x,t) = -\frac{\partial u}{\partial x}(x,t)$. La inversa $\mathfrak{L}_t^\beta$ de un subordinador

β-estable $\mathfrak{H}_t^\beta$ se puede definir por la siguiente relación

$$Pr\{\mathfrak{L}_t^\beta < x\} = Pr\{\mathfrak{H}_x^\beta > t\}$$

para $x, t > 0$ [26] y tenemos que $\mathbb{E}e^{-\lambda \mathfrak{L}_t^\beta} = E_\beta(-\lambda t^\beta)$, $\lambda \geq 0$, donde la función Mittag-Leffler de un parámetro se define como [26]

$$E_\beta(x) = \sum_{k=0}^{\infty} \frac{x^k}{\Gamma(\beta k + 1)}, \qquad x \in \mathbb{R},\ \beta > 0. \tag{3.25}$$

Además, para todo $\beta \in (0,1)$, se mantiene la estimación uniforme

$$\frac{1}{1+\Gamma(1-\beta)x} \leq E_\beta(-x) \leq \frac{1}{1+[\Gamma(1+\beta)]^{-1}x} \tag{3.26}$$

sobre todo $\mathbb{R}_+$ [1].

El campo aleatorio T introducido en (3.5) es gaussiano y por tanto sus coeficientes de Fourier $a_{l,m}$ son variables aleatorias gaussianas independientes de valor complejo y media cero. Denotamos por $F(\mathfrak{L}_t^\beta)$ el subordinador con símbolo Ψ cambiado en el tiempo por el inverso de un subordinador estable de orden $\beta \in (0,1)$.

A partir de ahora consideramos campos aleatorios de la forma

$$\sum_{l=0}^{\infty} \sum_{m=-l}^{+l} a_{l,m} \mathcal{T}_l(t) \mathscr{Y}_{l,m}(x) \tag{3.27}$$

con

$$\mathcal{T}_l(t) = \mathbb{E}[exp - \mu_l F(\mathscr{L}_t^\beta)], \qquad l \geq 0,\ t \geq 0, \tag{3.28}$$

donde la serie (3.27) converge en sentido $L^2(dP \times d\lambda)$ para todo $t \geq 0$, es decir,

$$lim_{L\to\infty} \mathbb{E}\left[\int_{\mathbf{S}_1^2} \left(X_t(x) - \sum_{l=0}^{L} \sum_{m=-l}^{+l} a_{l,m} \mathcal{T}_l(t) \mathscr{Y}_{l,m}(x)\right)^2 \lambda(dx)\right] = 0, \ \ \forall t. \tag{3.29}$$

Teorema 1. Consideremos $\gamma \geq 0$ y $\beta \in (0,1)$. La solución en $L^2(dP \times d\lambda)$ de la ecuación fraccionaria

$$\left(\gamma - \mathbb{D}_M + \frac{\partial^\beta}{\partial t^\beta}\right) X_t(x) = 0, \qquad x \in \mathbf{S}_1^2,\ t \geq 0 \tag{3.30}$$

con condición inicial $X_0(x) = T(x)$ es un campo aleatorio dependiente del tiempo en la esfera $\mathbf{S}_1^2$ escrito como

$$X_t(x) = \sum_{l=0}^{\infty} \sum_{m=-l}^{+l} a_{l,m} E_\beta(-t^\beta(\gamma + \Psi(\mu_l))) \mathscr{Y}_{l,m}(x) \tag{3.31}$$

donde

$$a_{l,m} = \int_{\mathbf{S}_1^2} X_0(x) \mathscr{Y}_{l,m}^*(x) \lambda(dx). \tag{3.32}$$

Demostración. Primero notamos que

$$-\frac{\partial}{\partial t}\mathbb{E}e^{-\xi(\gamma t+F(t))}\Big|_{t=0} = \xi\gamma + \Psi(\xi) \tag{3.33}$$

donde $\mathbb{E}e^{-\xi(\gamma t+F(t))}$ coincide con (3.14) para $\gamma = b$. De hecho, estamos tratando con el símbolo Ψ del subordinador F_t sin deriva. Además, la función Mittag-Leffler E_β es la función propia de la derivada fraccionaria D–C, esto es

$$\frac{\partial^\beta}{\partial t^\beta}E_\beta(-t^\beta\mu) = -\mu E_\beta(-t^\beta\mu), \quad \mu > 0. \tag{3.34}$$

Supongamos que (3.31) es cierta. Del hecho de que

$$\mathbb{D}_M\mathscr{Y}_{l,m}(x) = \int_0^\infty (P_s\mathscr{Y}_{l,m}(x) - \mathscr{Y}_{l,m}(x))M(ds)$$

donde $\mathscr{Y}_{l,m}(x) = (-1)^m\mathscr{Y}^*_{l,-m}(x)$ y

$$P_s\mathscr{Y}_{l,m}(x) = e^{-s\mu_l}\mathscr{Y}_{l,m}(x) \tag{3.35}$$

obtenemos que

$$\begin{aligned}\mathbb{D}_M\mathscr{Y}_{l,m}(x) &= \int_0^\infty (e^{-s\mu_l}\mathscr{Y}_{l,m}(x) - \mathscr{Y}_{l,m}(x))M(ds)\\ &= \int_0^\infty (e^{-s\mu_l} - 1)M(ds)\mathscr{Y}_{l,m}(x)\\ &= -\Psi(\mu_l)\mathscr{Y}_{l,m}(x).\end{aligned}$$

La fórmula (3.35) se puede obtener considerando que

$$\begin{aligned}P_s\mathscr{Y}_{l,m}(x) &= \mathbb{E}\mathscr{Y}_{l,m}(x + B_s)\\ &= \sum_{l'm'} e^{-s\mu_{l'}}\mathscr{Y}^*_{l',m'}(x)\int_{\mathbf{S}_1^2}\mathscr{Y}_{l,m}(y)\mathscr{Y}_{l',m'}(y)\lambda(dy)\\ &= \sum_{l'm'} e^{-s\mu_{l'}}\mathscr{Y}^*_{l',m'}(x)(-1)^{m'}\int_{\mathbf{S}_1^2}\mathscr{Y}_{l,m}(y)\mathscr{Y}^*_{l',-m'}(y)\lambda(dy)\\ &= \sum_{l'm'} e^{-s\mu_{l'}}\mathscr{Y}^*_{l',m'}(x)(-1)^{m'}\delta_l^{l'}\delta_m^{-m'} = e^{-s\mu_l}\mathscr{Y}_{l,m}(x).\end{aligned}$$

Así, conseguimos que

$$(\gamma - \mathbb{D}_M)X_t(x) = \sum_{l=0}^\infty\sum_{m=-l}^{+l} a_{l,m}(\gamma + \Psi(\mu_l))E_{\beta,1}(-t^\beta\gamma - t^\beta\Psi(\mu_l)))\mathscr{Y}_{l,m}(x)$$

y, de (3.34), llegamos a

$$\left(\frac{\partial^\beta}{\partial t^\beta} + \gamma - \mathbb{D}_M\right)X_t(x) = \sum_{l=0}^\infty\sum_{m=-l}^{+l} a_{l,m}\left(\frac{\partial^\beta}{\partial t^\beta} + \gamma + \Psi(\mu_l)\right)E_\beta(-t^\beta\gamma - t^\beta\Psi(\mu_l)))\mathscr{Y}_{l,m}(x) = 0$$

término a término, y por lo tanto la ecuación (3.30) se cumple. □

Observación 1. A partir de

$$T(x) = \sum_{lm} a_{l,m} \mathscr{Y}_{l,m}(x) \tag{3.36}$$

tenemos que

$$P_t T(x) = \mathbb{E}\left[T(x + B_t)|\mathfrak{F}_T\right] = \sum_{lm} e^{-t\mu_l} a_{l,m} \mathscr{Y}_{l,m}(x) = T_t(x). \tag{3.37}$$

Esto representa la solución a (3.30) con $\beta = 1$, $\gamma = 0$ y $\mathbb{D}_M = \Delta_{\mathbf{S}_1^2}$. De (**??**), para $\beta = 1$, $\Psi(\xi) = \xi$, esto es, para el subordinador elemental $F(t) = t$ (y $\mathfrak{L}_t^1 = t$), tenemos que

$$X_t(x) = \mathbb{E}\left[T(x + B(\gamma t + t))|\mathfrak{F}_T\right] = \sum_{lm} a_{l,m} e^{-t(\gamma+1)\mu_l} \mathscr{Y}_{l,m}(x).$$

Teorema 2. Consideremos $\gamma > 0$, $\varphi \geq 0$ y $\beta \in (0,1)$. Una solución en $L^2(dP \times d\lambda)$ a la ecuación fraccionaria

$$\left(\gamma - \mathbb{D}_M - \varphi \frac{\partial}{\partial t}\right)^{\beta} X_t(x) = T_t(x), \qquad x \in \mathbf{S}_1^2,\ t \geq 0 \tag{3.38}$$

donde $T_t(x)$ dado en (3.37) es un campo aleatorio dependiente del tiempo en la esfera $\mathbf{S}_1^2$ escrito como

$$X_t(x) = \sum_{lm} a_{l,m} e^{-t\mu_l} (\gamma + \varphi\mu_l + \Psi(\mu_l))^{-\beta} \mathscr{Y}_{l,m}(x), \tag{3.39}$$

donde $e^{-t\mu_l} a_{l,m}$ son los coeficientes aleatorios involucrados en la representación (3.37) del proceso de innovación $T_t(x)$ en términos de funciones esféricas harmónicas.

Demostración. Tenemos que

$$\begin{aligned} X_t(x) &= \left(\gamma - \mathbb{D}_M - \varphi \frac{\partial}{\partial t}\right)^{-\beta} T_t(x) \\ &= \int_0^\infty ds \frac{s^{\beta-1}}{\Gamma(\beta)} e^{s\varphi \frac{\partial}{\partial t} - s\gamma + s\mathbb{D}_M} T_t(x) \\ &= \int_0^\infty ds \frac{s^{\beta-1}}{\Gamma(\beta)} e^{s\varphi \frac{\partial}{\partial t} - s\gamma} \mathbb{P}_s T_t(x) \\ &= \int_0^\infty ds \frac{s^{\beta-1}}{\Gamma(\beta)} e^{-s\gamma} \mathbb{P}_s T_{t+\varphi s}(x) \end{aligned}$$

donde usamos la regla de traslación

$$e^{a\frac{\partial}{\partial z}} f(z) = f(z + a), \qquad a \in \mathbb{R},$$

la cual es válida para funciones continuas acotadas f en $(0, +\infty)$ [12]. Del hecho de que

$$\mathbb{P}_s \mathscr{Y}_{l,m}(x) = e^{-s\Psi(\mu_l)} \mathscr{Y}_{l,m}(x) \tag{3.40}$$

donde $\mathbb{P}_s = exp(s\mathbb{D}_M)$, tenemos que

$$\begin{aligned} X_t(x) &= \sum_{lm} a_{l,m} \left(\int_0^\infty ds \frac{s^{\beta-1}}{\Gamma(\beta)} e^{-s\gamma} e^{-(t+\varphi s)\mu_l} \mathbb{P}_s \mathscr{Y}_{l,m}(x) \right) \\ &= \sum_{lm} a_{l,m} \left(\int_0^\infty ds \frac{s^{\beta-1}}{\Gamma(\beta)} e^{-s\gamma} e^{-(t+\varphi s)\mu_l} e^{-s\Psi(\mu_l)} \right) \mathscr{Y}_{l,m}(x) \\ &= \sum_{lm} a_{l,m} e^{-t\mu_l} \left(\int_0^\infty ds \frac{s^{\beta-1}}{\Gamma(\beta)} e^{-s\gamma - s\varphi\mu_l - s\Psi(\mu_l)} \right) \mathscr{Y}_{l,m}(x) \\ &= \sum_{lm} a_{l,m} e^{-t\mu_l} \left(\gamma + \varphi\mu_l + \Psi(\mu_l)\right)^{-\beta} \mathscr{Y}_{l,m}(x), \end{aligned}$$

lo que concluye la demostración. □

Examinamos ahora el caso especial $\varphi = 0$.

Corolario 1. Sea $\beta \in (0, 1]$. La solución a

$$(\gamma - \mathbb{D}_M)^\beta X(x) = T(x) \tag{3.41}$$

se escribe como

$$X(x) = \sum_{lm} a_{l,m} \left(\gamma + \Psi(\mu_l)\right)^{-\beta} \mathscr{Y}_{l,m}(x). \tag{3.42}$$

Demostración. Para $\beta \in (0, 1)$ consideramos la siguiente relación relativa a la potencia fraccionaria de los operadores (potencial de Bessel). Para $f \in L^2(\mathbf{S}_1^2)$ tenemos que

$$\begin{aligned} (\gamma - \mathbb{D}_M)^\beta f(x) &= \frac{\beta}{\Gamma(1-\beta)} \int_0^\infty \frac{ds}{s^{\beta+1}} \left(1 - e^{-s\gamma + s\mathbb{D}_M}\right) f(x) \\ &= \frac{\beta}{\Gamma(1-\beta)} \int_0^\infty \frac{ds}{s^{\beta+1}} \left(f(x) - e^{-s\gamma} \mathbb{P}_s f(x)\right) \end{aligned}$$

donde $\mathbb{P}_s f$ es el semigrupo de transición asociado al operador $\mathbb{D}_M$ y $u(x,t) = \mathbb{P}_t f(x)$ resuelve el problema de Cauchy $(\frac{\partial}{\partial t} - \mathbb{D}_M)u(x,t) = 0$ con $u(x,0) = f(x)$. Por lo tanto, si asumimos que existe la siguiente representación espectral para la solución X como una función aleatoria en $\mathbf{S}_1^2$,

$$X(x) = \sum_{lm} \hat{a}_{l,m} \mathscr{Y}_{l,m}(x), \tag{3.43}$$

podemos escribir

$$\begin{aligned} (\gamma - \mathbb{D}_M)^\beta X(x) &= \frac{\beta}{\Gamma(1-\beta)} \sum_{lm} \hat{a}_{l,m} \int_0^\infty \frac{ds}{s^{\beta+1}} \left(\mathscr{Y}_{l,m}(x) - e^{-s\gamma} \mathbb{P}_s \mathscr{Y}_{l,m}(x)\right) \\ &= \frac{\beta}{\Gamma(1-\beta)} \sum_{lm} \hat{a}_{l,m} \int_0^\infty \frac{ds}{s^{\beta+1}} \left(1 - e^{-s\gamma} e^{-s\Psi(\mu_l)}\right) \mathscr{Y}_{l,m}(x) \\ &= \sum_{lm} \hat{a}_{l,m} \left(\gamma + \Psi(\mu_l)\right)^\beta \mathscr{Y}_{l,m}(x). \end{aligned}$$

La ecuación (3.41) se cumple sólo si

$$\hat{a}_{l,m} = a_{l,m} \left(\gamma + \Psi(\mu_l)\right)^{-\beta}.$$

Por otro lado, repitiendo los argumentos de la demostración del Teorema 2 tenemos que

$$\begin{aligned} X(x) &= \left(\gamma - \mathbb{D}_M\right)^{-\beta} T(x) \\ &= \int_0^\infty ds \frac{s^{\beta-1}}{\Gamma(\beta)} e^{-s\gamma + s\mathbb{D}_M} T(x) \\ &= \int_0^\infty ds \frac{s^{\beta-1}}{\Gamma(\beta)} e^{-s\gamma} \mathbb{P}_s T(x) \\ &= \sum_{lm} a_{l,m} \int_0^\infty ds \frac{s^{\beta-1}}{\Gamma(\beta)} e^{-s\gamma} e^{-s\Psi(\mu_l)} \mathscr{Y}_{l,m}(x) \\ &= \sum_{lm} a_{l,m} \left(\gamma + \Psi(\mu_l)\right)^{-\beta} \mathscr{Y}_{l,m}(x), \end{aligned}$$

llegando al resultado (3.42). □

Finalmente, estudiamos la covarianza de los campos aleatorios introducidos. Consideremos la representación

$$X_t(x) = \sum_{lm} a_{l,m} \mathcal{T}_l(t) \mathscr{Y}_{l,m}(x) = \sum_l \mathcal{T}_l(t) T^l(x) \tag{3.44}$$

introducido en (3.27) y recordamos que, para $x, y \in \mathbf{S}_1^2$,

$$\mathbb{E}[X_0(x) X_0(y)] = \sum_l \frac{2l+1}{4\pi} C_l Q_l\left(\langle x, y\rangle\right) = \mathbb{E}[T(x)T(y)]. \tag{3.45}$$

Además,

$$\mathbb{E}[T(x)T(y)] = \sum_l \mathbb{E}[T^l(x)T^l(y)]. \tag{3.46}$$

Esto se debe a que los coeficientes $a_{l,m}$ no están correlacionados sobre l.

Observación 2. Considerando (3.27), tenemos que

- para $X_t(x)$ como en el Teorema 1,

$$\mathcal{T}_l(t) = E_\beta\left(-t^\beta(\gamma + \Psi(\mu_l))\right) \geq \frac{1}{1 + \Gamma(1-\beta) t^\beta (\gamma + \Psi(\mu_l))}, \quad t \geq 0,\ l \geq 0, \quad [31] \tag{3.47}$$

- para $X_t(x)$ como en el Teorema 2,

$$\mathcal{T}_l(t) = e^{-t\mu_l} \left(\gamma + \varphi\mu_l + \Psi(\mu_l)\right)^{-\beta} \leq e^{-tl^2} \left(\gamma + \varphi l^2 + \Psi(l^2)\right)^{-\beta}, \quad t \geq 0,\ l \geq 0, \tag{3.48}$$

- para el caso estacionario $X(x)$ del Corolario 1,

$$\mathcal{T}_l(t) = \left(\gamma + \Psi(\mu_l)\right)^{-\beta} \leq \left(\gamma + \Psi(l^2)\right)^{-\beta}, \quad t \geq 0,\ l \geq 0. \tag{3.49}$$

Observación 3. Sea $B(\tau_t) = B \circ \tau_t$ un movimiento Browniano en la esfera unitaria, donde τ_t es un proceso aleatorio en el tiempo. Nos referimos a $B(\tau_t)$ como un cambio de coordenadas para el campo aleatorio T en la esfera. De los resultados anteriores, observamos que

$$X_t(x) = \mathbb{E}[T_0(x + B(\tau_t))|\mathfrak{F}_T] = \mathbb{E}[T_{\tau_t}(x)|\mathfrak{F}_T], \quad x \in \mathbf{S}_1^2,\ t > 0 \tag{3.50}$$

donde

$$T_t(x) = \sum_l e^{-t\mu_l} T^l(x), \quad x \in \mathbf{S}_1^2,\ t > 0. \tag{3.51}$$

Es más,

$$\mathcal{T}_l(t) = \mathbb{E}e^{-\mu_l \tau_t}. \tag{3.52}$$

Podemos enunciar el siguiente resultado, en el que se supone que los movimientos Brownianos esféricos $x + B_t$, $y + B_t$ subyacentes a $X_t(x) = T(x + B_t)$ y $X_t(y) = T(y + B_t)$ son independientes.

Teorema 3. Para $x, y \in \mathbf{S}_1^2$, para todo $g \in SO(3)$, tenemos que

$$\mathbb{E}[X_t(gx)X_s(gy)] = \sum_l \frac{2l+1}{4\pi} C_l \mathcal{T}_l(t)\mathcal{T}_l(s) Q_l\left(\langle x, y\rangle\right), \quad t, s \geq 0. \tag{3.53}$$

Demostración. Primero observamos que

$$\mathbb{E}[a_{l,m}a_{l',m'}] = (-1)^m \delta_l^{l'} \delta_{-m}^{m'} C_l \tag{3.54}$$

de la propiedad $\mathscr{Y}_{l,m}(x) = (-1)^m \mathscr{Y}^*_{l,-m}(x)$ de las funciones esféricas harmónicas. De la representación (3.44) podemos escribir

$$\begin{aligned}
\mathbb{E}[X_t(x)X_s(y)] &= \sum_{lm}\sum_{l'm'} \mathbb{E}[a_{l,m}a_{l',m'}]\mathcal{T}_l(t)\mathcal{T}_{l'}(s)\mathscr{Y}_{l,m}(x)\mathscr{Y}_{l',m'}(y) \\
&= \sum_{lm} C_l \mathcal{T}_l(t)\mathcal{T}_l(s)\mathscr{Y}_{l,m}(x)\mathscr{Y}^*_{l,m}(y) \\
&= \sum_l \frac{2l+1}{4\pi} C_l \mathcal{T}_l(t)\mathcal{T}_l(s) Q_l\left(\langle x, y\rangle\right)
\end{aligned}$$

donde $\mathcal{T}_l(t)$ viene dada como en (3.52) y usando la fórmula de la suma se llega a $Q_l\left(\langle x, y\rangle\right)$. □

Observación 4. Inmediatamente podemos ver que la varianza se convierte en

$$\mathbb{E}[X_t(gx)]^2 = \sum_l \frac{2l+1}{4\pi} C_l |\mathcal{T}_l(t)|^2, \quad \forall g \in SO(3). \tag{3.55}$$

Recordamos que C_l es el espectro de potencia angular de T y, por lo general, se supone que es $C_l \sim l^{-\gamma}$ con $\gamma > 2$ para l grande para asegurar sumabilidad (o $C_l \sim L(l)/l^\theta$, $\theta > 0$, donde $L(.)$ es una función que varía lentamente cuando $l \to \infty$). Por la Observación 2, tenemos el comportamiento de alta frecuencia también para $\mathcal{T}_l(t)$ tanto en la variable $t > 0$ como en la frecuencia $l > 0$.

Decimos que el proceso de media cero $X_t(x)$ muestra un comportamiento de memoria larga si

$$\sum_{h=1}^{\infty} \mathbb{E}[X_{t+h}(x)X_t(y)] = \infty, \qquad x, y \in \mathbf{S}_1^2, \tag{3.56}$$

lo cual ocurre si la dependencia de las variables aleatorias de los procesos $X_t(.)$ decrece lentamente en el tiempo. Por el contrario, decimos que X muestra un comportamiento de memoria corta si la serie (3.56) converge.

Observación 5. Escribimos

$$\mathcal{K}_t(x, y) = \sum_{h\geq 1} \mathbb{E}[X_{t+h}(x)X_t(y)], \qquad t \geq 0$$

para $x, y \in \mathbf{S}_1^2$, y

$$g_l(t) = \frac{1}{1 + \Gamma(1-\beta)t^{\beta}(\gamma + \Psi(\mu_l))}.$$

A partir de aquí, tenemos que:

- para $X_t(x)$ como en el Teorema 1, el campo aleatorio muestra un comportamiento de memoria larga

$$\begin{aligned}
\mathcal{K}_t(x, y) &= \sum_{h\geq 1}\sum_{l\geq 0} \frac{2l+1}{2\pi} C_l E_\beta(-t^\beta(\gamma + \Psi(\mu_l)))E_\beta(-(t+h)^\beta(\gamma + \Psi(\mu_l)))Q_l\left(\langle x, y\rangle\right) \\
&\geq \sum_{h\geq 1}\sum_{l\geq 0} \frac{2l+1}{2\pi} C_l g_l(t)g_l(t+h)Q_l\left(\langle x, y\rangle\right) \\
&\geq \sum_{l\geq 0} \frac{2l+1}{2\pi} C_l g_l(t)Q_l\left(\langle x, y\rangle\right) \sum_{h\geq 1} g_l(t+h) \\
&= \infty,
\end{aligned}$$

- para $X_t(x)$ como en el Teorema 2, el campo aleatorio tiene un comportamiento de memoria corta

$$\begin{aligned}
\mathcal{K}_t(x, y) &\leq \sum_{h\geq 1}\sum_{l\geq 0} \frac{2l+1}{4\pi} C_l e^{-2tl^2 - hl^2}(\gamma + \varphi l^2 + \Psi(l^2))^{-2\beta} Q_l\left(\langle x, y\rangle\right) \\
&\leq \sum_{l\geq 0} \frac{2l+1}{4\pi} \frac{C_l}{e^{l^2}-1} e^{-2tl^2}(\gamma + \varphi l^2 + \Psi(l^2))^{-2\beta} Q_l\left(\langle x, y\rangle\right) \\
&< \infty.
\end{aligned}$$

Como ilustración de este capítulo, se genera en MatLab el proceso solución de la ecuación fraccionaria (3.30) del Teorema 1, es decir, el campo dependiente del tiempo (3.31). Para ello, comenzamos con el código correspondiente al argumento de la función Mittag-Leffler, que en este caso es $-t^\beta(\gamma + \Psi(\mu_l))$: declaramos $\beta = 0.9$ y $\gamma = 1$ constantes, como función Ψ se toma la función identidad y se genera μ_l con $l = 1, \ldots, M = 10$. Se considera el intervalo temporal $[0, 100]$ con $T = 100$ nodos temporales y se genera el argumento, que depende del tiempo y de l. Este argumento se multiplica por $2l + 1$, aplicando la fórmula **??**11.1) que relaciona las funciones esféricas armónicas con las funciones de Legendre.

```
clear all
beta=0.9;
gamman=1;

M=10;
c=2;
for l=1:M
        mu(l)=(c/l^(3/2));
end

T=100;  % número de nodos temporales
x=linspace(0,T,T);

for t=1:T
        for l=1:M
                argumento(t,l)=-x(t)^(beta)*(gamman+mu(l)*(2*l+1));
        end
end
```

A continuación, se genera el término $E_\beta(-t^\beta(\gamma+\Psi(\mu_l)))$ del campo (3.31), haciendo uso de la función Mittag-Leffler (3.25), que truncamos en $k = 50$. Así, se obtiene la matriz $IFT2B$ que se usa más adelante con los polinomios de Legendre.

```
K=50;
for k=1:K
        seriet=(argumento).^(k);
        termino(:,:,k)=seriet/gamma(beta*k+1);
end
SS=sum(termino,3);
IFT2B=SS;
```

El siguiente fragmento de código sirve para declarar las coordenadas en la esfera $KL1$ del punto uniforme que vamos a considerar.

```
H=10;% REGULAR  SPHERICAL  GRID

phi=linspace(0,2*pi,H);
theta=linspace(0,pi,H);
phi=phi';
theta=theta';
[s1,s2]=size(phi);

for i=1:s1
        for j=1:s1
                x1(i,j)=cos(phi(j))*sin(theta(i));
                y1(i,j)=sin(phi(j))*sin(theta(i));
                z1(i,j)=cos(theta(i));
        end
end
```

```
x1=x1(:);
y1=y1(:);
z1=z1(:);
x2=x1;
y2=y1;
z2=z1;

v1=[x1,y1,z1];
v2=[x2,y2,z2];
[s11,s22]=size(v1);

KL1=[x(1:s11,1),y(1:s11,1),z(1:s11,1)];
```

El siguiente paso es generar una uniforme en la esfera. Primero, inicializamos el generador de números aleatorios para poder repetir los resultados de este ejemplo. Después, calculamos un ángulo de elevación para cada punto de la esfera; estos valores *elevation* están en el intervalo abierto $(-\pi/2, \pi/2)$, pero no están uniformemente distribuidos. Creamos también un ángulo acimut para cada punto de la esfera; estos valores *azimuth* sí se distribuyen uniformemente en el intervalo abierto $(0, 2\pi)$. Seguidamente, asignamos un radio a cada punto de la esfera; estos valores *radii* están en el intervalo abierto $(0, 1)$, pero no están uniformemente distribuidos. Finalmente, pasamos a coordenadas cartesianas mediante la función *sph2cart*.

```
rng(0,'twister')  %inicializa el generador de números aleatorios

rvals = 2*rand(1000,1)-1;
elevation = asin(rvals);

azimuth = 2*pi*rand(1000,1);

radii=ones(1000,1);

[x,y,z] = sph2cart(azimuth,elevation,radii);
```

Utilizamos $KL1$, generado anteriormente, para los polinomios de Legendre que se calculan con la función *LegendreP*, que genera el k-ésimo (con k variando desde 1 hasta $M = 10$) polinomio de Legendre del producto escalar de los puntos esféricos $v1$ y $KL1$. Los bucles en i y en j hacen que nos movamos en la primera dimensión de las coordenadas esféricas de $v1$ y $KL1$.

```
Tr=M;

for i=1:s11
        for j=1:s11
                for k=1:Tr
                        II(i,j)=dot(v1(i,:),KL1(j,:));
                        LL(i,j,k)=legendreP(k,II(i,j));
                end
        end
end
```

Seguidamente, se realiza la transformada de Fourier inversa de los coeficientes de Fourier, mediante la anidación de varios bucles *for*. Con la función *dot* de MatLab se realiza el producto escalar de la matriz correspondiente a la función Mittag-Leffler (*IFT2B*) y de los polinomios de Legendre, obteniendo una aproximación del valor de la log-intensidad en el tiempo t y la localización espacial esférica i, con respecto a diferentes posiciones esféricas uniformemente distribuidas, manteniendo el acimut de la uniforme y cambiando en polar (cuando variamos i mantenemos j).

```
[T,S2]=size(IFT2B(10:90,:));
u=1;

for t=10:10+T-1
        for i=1:s11
                for j=1:s11
                        for k=1:Tr
                                CCv1(k,u)=IFT2B(t,k);
                                CCv2(k,i,j)=LL(i,j,k);
                        end
                        XXX(i,j,u)=dot(CCv1(:,u),CCv2(:,i,j));
                end
        end
        u=u+1;
end

v=1;
for t=10:10+T-1
        for j=1:s11
                for hh=1:s1
                        for nn=1:s1
                           XXX2(hh,nn,j,v)=XXX((hh-1)*s1+nn,j,v);

                        end
                end
        end
        v=v+1;
end
```

Finalmente, representamos los valores sobre la esfera en diferentes tiempos de la log-intensidad Gaussiana.

```
tval=fix(linspace(0,T,T));
[X1,Y1]=size(x1);
s1=(X1)^(1/2);
for hh=1:s1
        for nn=1:s1
                xx11(hh,nn)=x1((hh-1)*s1+nn);
                yy11(hh,nn)=y1((hh-1)*s1+nn);
                zz11(hh,nn)=z1((hh-1)*s1+nn);
        end
end
```

Llegados a este punto, podemos representar gráficamente las realizaciones del campo solución X sobre la esfera, para los diferentes tiempos muestreados. Se muestran a continuación tres opciones de gráficas en la esfera.

```
% Gráfica opción 1

[xx1,xx2,xx3,xx4]=size(XXX2);
zzb=1;
for t=1:10:xx4
        for j=1
                SHMAP= XXX2(:,:,j,t);
                subplot(3,3,zzb)
                surf(xx11,yy11,zz11,SHMAP);
                mis_colores=jet(256);
                colormap(mis_colores)
                colorbar('vert')
                shading flat
        zzb=zzb+1;
        end
end
```

```
% Gráfica opción 2

tval=fix(linspace(0,T,T));
zzb=1;
for t=1:10:xx4
        for j=1
            SHMAP= XXX2(:,:,j,t);
            subplot(3,3,zzb)
            surf(xx11,yy11,zz11,SHMAP,'FaceColor','interp',...
            'EdgeColor','none','FaceLighting','gouraud')
            demcmap(SHMAP)
            colorbar
            title(['$t = $',num2str(tval(t))],'Interpreter','latex')
            zzb=zzb+1;
        end
end
```

```
% Gráfica opción 3

zzz=1;
for  t=1:10:xx4
        for j=1
            SHMAP= XXX2(:,:,j,t);
            subplot(3,3,zzz)
            hs = surf(xx11,yy11,zz11,SHMAP,'FaceColor','interp',...
            'EdgeColor','none','FaceLighting','gouraud')
            cmap = hsv2rgb([5/12 1 0.4; 0.25 0.2 1; 5/72 1 0.4]);
            cmapland = cmap;
```

```
            colorbar
            title(['$t␣=␣$␣',num2str(tval(t))],'Interpreter','latex')
            zzz=zzz+1;
        end
end
```

Se muestra la gráfica obtenida con el último código de programación, ya que refleja de forma clara la persistencia en el tiempo en los patrones esféricos introducida por la memoria larga.

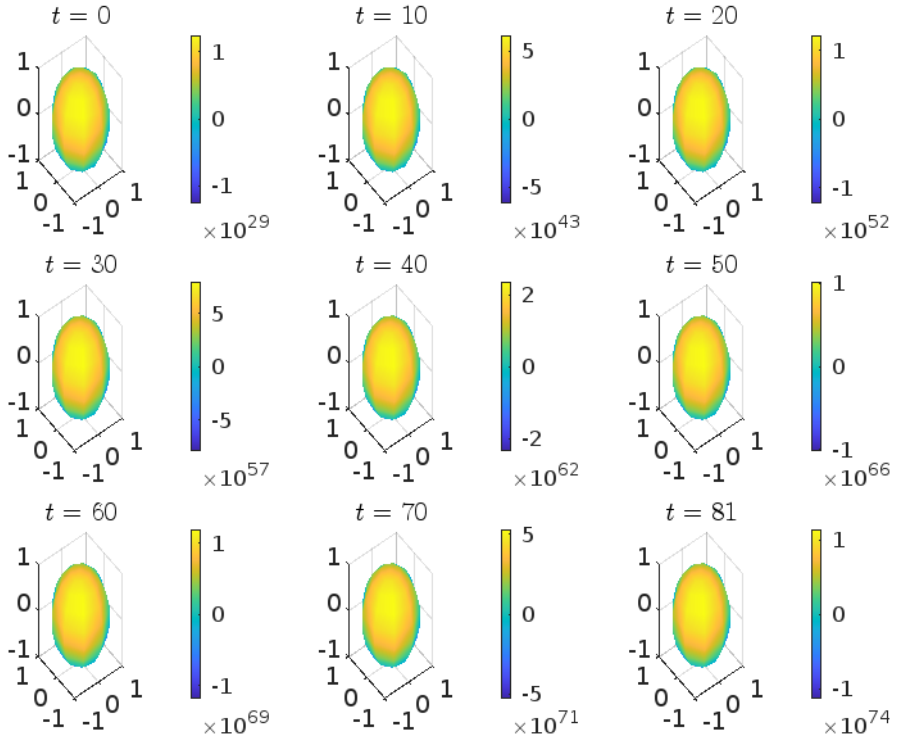


Figura 3.1: Realizaciones de X sobre la esfera para los diferentes tiempos muestreados.

Adicionalmente, se pueden utilizar las cotas (3.26) para generar cotas inferiores y superiores del proceso solución. Para ello, basta con sustituir la simulación de la función Mittag-Leffler por la de dichas cotas.

En primer lugar, utilizamos la cota inferior. El código es igual salvo en la expresión del campo dependiente del tiempo (3.31), en el que en lugar de utilizar la definición de la función Mittag-Leffler (3.25), se utiliza la cota inferior (3.26). Así, el argumento se mantiene igual, aunque cambiamos la generación de μ_l con $l = 1, \ldots, M$.

```
clear all
beta=0.9;
gamman=1;
M=10;
```

```
for l=1:M
        mu(l)=((l+1)/l)^(11/10);
end

T=100;  % número de nodos temporales
x=linspace(0,T,T);

for t=1:T
        for l=1:M
                argumento(t,l)=-x(t)^(beta)*(gamman+mu(l)*(2*l+1));
        end
end

for t=1:T
        for l=1:M
                cinf(t,l)=mu(l)/(1+gamma(1-beta)*argumento(t,l));
        end
end

IFT2B=cinf;
```

Una vez obtenida la matriz IFT2B, el resto del código es idéntico, obteniendo por ejemplo la siguiente gráfica:

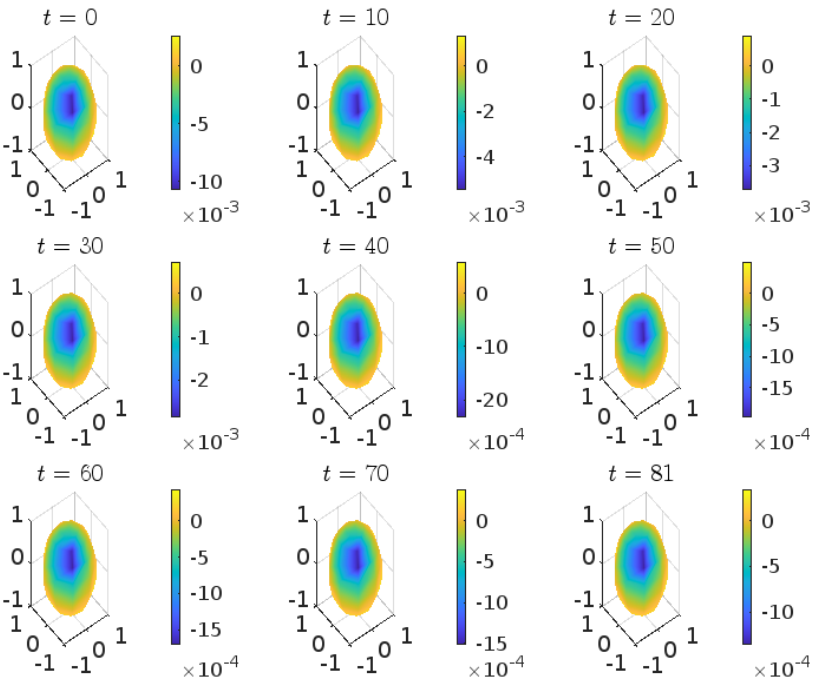


Figura 3.2: Cota inferior para las realizaciones de X sobre la esfera para los diferentes tiempos muestreados.

Se procede de igual forma con la cota superior:

```
clear all
beta=0.9;
gamman=1;
M=10;
T=100;  % número de nodos temporales
x=linspace(0,T,T);
for l=1:M
        mu(l)=((l+1)/l)^(11/10);
end
for t=1:T
        for l=1:M
                argumento(t,l)=-x(t)^(beta)*(gamman+mu(l)*(2*l+1));
        end
end
for t=1:T
    for l=1:M
        csup(t,l)=mu(l)/(1+((gamma(1+beta))^(-1))*argumento(t,l));
    end
end
IFT2B=csup;
```

Una vez obtenida la matriz IFT2B, el resto del código es idéntico, obteniendo la siguiente gráfica:

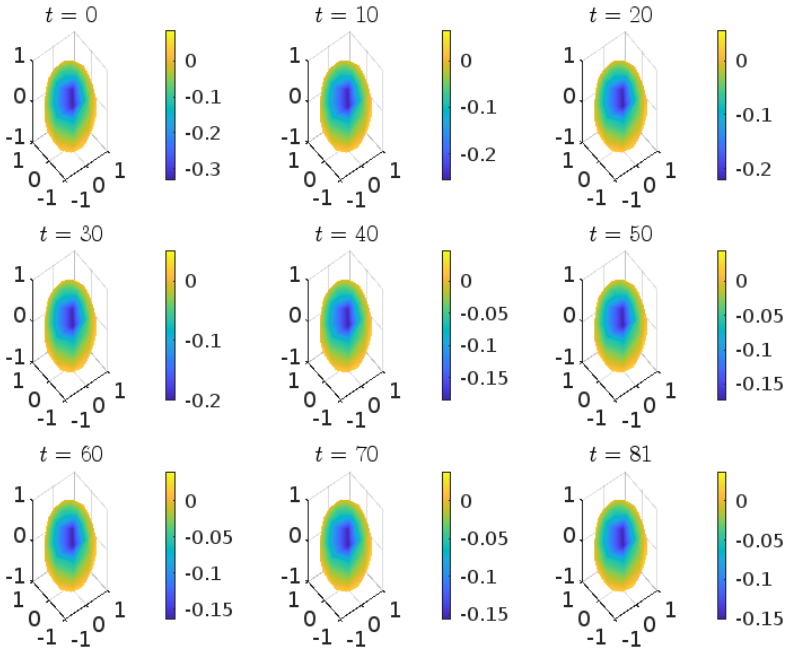


Figura 3.3: Cota superior para las realizaciones de X sobre la esfera para los diferentes tiempos muestreados.

3.4. Generación del modelo extendido

Consideramos a continuación la serie $L^2(\mathbb{S}_2, dv)$–valuada, que obedece la siguiente ecuación funcional, para $t \in \mathbb{T}$ y $\mathbf{x} \in \mathbb{S}_2 = \{\mathbf{x} \in \mathbb{R}^3;\ \|\mathbf{x}\| = 1\}$,

$$\begin{aligned}(\mathcal{I}_{L^2(\mathbb{M}_d, dv)} - B)^{\mathcal{A}_\theta/2}\widetilde{X}_t(\mathbf{x}) &= (\mathbf{\Phi}_3(B)\widetilde{X}_t)(\mathbf{x}) + \varepsilon_t(\mathbf{x}) \\ &= (\Phi_1\widetilde{X}_{t-1})(\mathbf{x}) + (\Phi_2\widetilde{X}_{t-2})(\mathbf{x}) + (\Phi_3\widetilde{X}_{t-3})(\mathbf{x}) + \varepsilon_t(\mathbf{x}),\end{aligned} \tag{3.57}$$

donde $\mathcal{A}_\theta$ denota un modelo paramétrico del operador de memoria. El espectro puntual (autovalores) $\{\alpha_n(\theta),\ n \in \mathbb{N}_0\}$ del operador de memoria $\mathcal{A}_\theta$ define los índices de diferenciación fraccionaria β en la ecuación (3.30). Por esta razón, el proceso que obedece (3.57) proporciona una versión muestreada del proceso solución de una versión multifraccionaria de la ecuación (3.30). El operador $\mathcal{A}_\theta$ admite una representación integral en términos de un núcleo $a(\mathbf{x}, \mathbf{y}, \theta)$ de la siguiente forma:

$$a(\mathbf{x}, \mathbf{y}, \theta) = \sum_{n \in \mathbb{N}_0} \alpha_n(\theta) P_n(\langle \mathbf{x}, \mathbf{y} \rangle), \quad \mathbf{x}, \mathbf{y} \in \mathbb{S}_2,$$

donde $\{P_n,\ n \in \mathbb{N}_0\}$ denota la base de polinomios de Legendre, cuya relación con la base de funciones esféricas harmónicas se especifica posteriormente, en la ecuación (3.59).

La ecuación (3.57) se interpreta en media cuadrática en el espacio $L^2(\mathbb{S}_2, dv)$. Los operadores Φ_i, $i = 1, 2, 3$, que aparecen en el término de autorregresión $\mathbf{\Phi}_3(z) = \sum_{i=1}^{3} \Phi_i z^i$, son operadores integrales cuyos núcleos $\boldsymbol{\phi}_i$, $i = 1, 2, 3$, son isotrópicos o invariantes frente a rotaciones. Se tiene entonces que todos ellos admiten la siguiente representación:

$$\boldsymbol{\phi}_i(\mathbf{x}, \mathbf{y}) = \sum_{n=0}^{\infty} \phi_n^{(i)} \frac{2n+1}{4\pi} P_n(\langle \mathbf{x}, \mathbf{y} \rangle), \quad \mathbf{x}, \mathbf{y} \in \mathbb{S}_2,\ i = 1, 2, 3 \tag{3.58}$$

en la norma del espacio $L^2(\mathbb{S}_2, dv)$. Estas representaciones se pueden expresar de forma equivalente en términos de los elementos de la base de esféricas harmónicas $\{Y_{m,l},\ m = -l, \ldots, l,\ l \geq 0\}$ aplicando la siguiente identidad:

$$\sum_{m=-l}^{l} Y_{m,l}(\mathbf{x}) Y_{m,l}(\mathbf{y}) = \frac{(2l+1)}{4\pi} P_l(\langle \mathbf{x}, \mathbf{y} \rangle),\ \mathbf{x}, \mathbf{y} \in \mathbb{S}_2,\ l \geq 0. \tag{3.59}$$

En las generaciones del modelo (3.57), se han considerado los parámetros $\phi_n^{(1)} = (0.6(n+1)/n)^{-3/2}(2n+1)^{-1}$, $\phi_n^{(2)} = (0.2(n+1/n))^{-3/2}(2n+1)^{-1}$ y $\phi_n^{(3)} = (-0.1(n+1)/n)^{-3/2}(2n+1)^{-1}$, $n = 1, \ldots, 10$. La figura 3.4 contiene las generaciones de los coeficientes aleatorios temporales $\{V_n,\ n \in \mathbb{N}_0\}$ respecto a la base de polinomios de Legendre $\{P_n,\ n \in \mathbb{N}_0\}$ del proceso X generado, que obedece la ecuación (3.57), para las frecuencias discretas de Legendre $n = 1, 4, 7, 10, 13, 16, 19, 22, 25, 28$. También se muestran las realizaciones de X sobre la esfera para los diferentes tiempos muestreados en las figuras **??** y 3.7, así como se muestran los coeficientes aleatorios, respecto a la base de polinomios de Legendre $\{P_n,\ n \in \mathbb{N}_0\}$, del proceso de innovación $L^2(\mathbb{S}_2, dv)$–valuado ε_t, que aparece en la ecuación (3.57), en las frecuencias discretas de Legendre $n = 1, 4, 7, 10, 13, 16, 19, 22, 25, 28$, en la figura 3.5.

A continuación, se muestra el código de programación en sintaxis MatLab con el que se

genera este modelo.

En primer lugar, se declara el parámetro M que contiene el orden de truncamiento, es decir, las frecuencias de Legendre analizadas, en este caso 30; y el parámetro T, que es el número de nodos temporales. Seguidamente, se utiliza la función *arima* para crear un modelo de media móvil integrada autorregresiva univariada, en la que se especifican los parámetros $\phi_n^{(1)} = (0.6(n+1)/n)^{-3/2}(2n+1)^{-1}$, $\phi_n^{(2)} = (0.2(n+1/n))^{-3/2}(2n+1)^{-1}$ y $\phi_n^{(3)} = (-0.1(n+1)/n)^{-3/2}(2n+1)^{-1}$. Con la función *simulate* se simula una trayectoria de este modelo. Así, mediante un bucle, se obtiene la matriz Y, que contiene en cada columna la trayectoria de respuesta simulada para cada orden de truncamiento, y la matriz IT, donde cada columna es la trayectoria de innovación simulada para cada orden de truncamiento.

```
clear all
M=30; % orden de truncamiento
T=100; %Número de nodos temporales

for k=1:M
        Mdl =arima('Constant',0.05,'AR',...
        {0.6*((k+1)/k)^(-3/2),0.2*(k+1/k)^(-3/2),...
        -0.1*(k+1/k)^(-3/2)},'Variance',(0.01*(k+1/k))^(-3/2));
        [Y, E] = simulate(Mdl,T,'NumPaths',1);

        R(:,k)=Y;
        IT(:,k)=E;
end
```

Mediante el siguiente bucle se grafica la trayectoria de respuesta correspondiente a los órdenes de truncamiento $MM = 1,4,7,10,13,16,19,22,25,28$, que se corresponden con los coeficientes aleatorios temporales respecto a la base de polinomios de Legendre del proceso X generado. Así, se obtiene la gráfica 3.4.

```
figure
MM=[1 4 7 10 13 16 19 22 25 28];

zz1=1;
for i=MM
        SSCF1= R(:,i);
        subplot(2,5,zz1);
        plot(SSCF1)
        title(['Scale','-',num2str(i)])
        zz1=zz1+1;
end
```

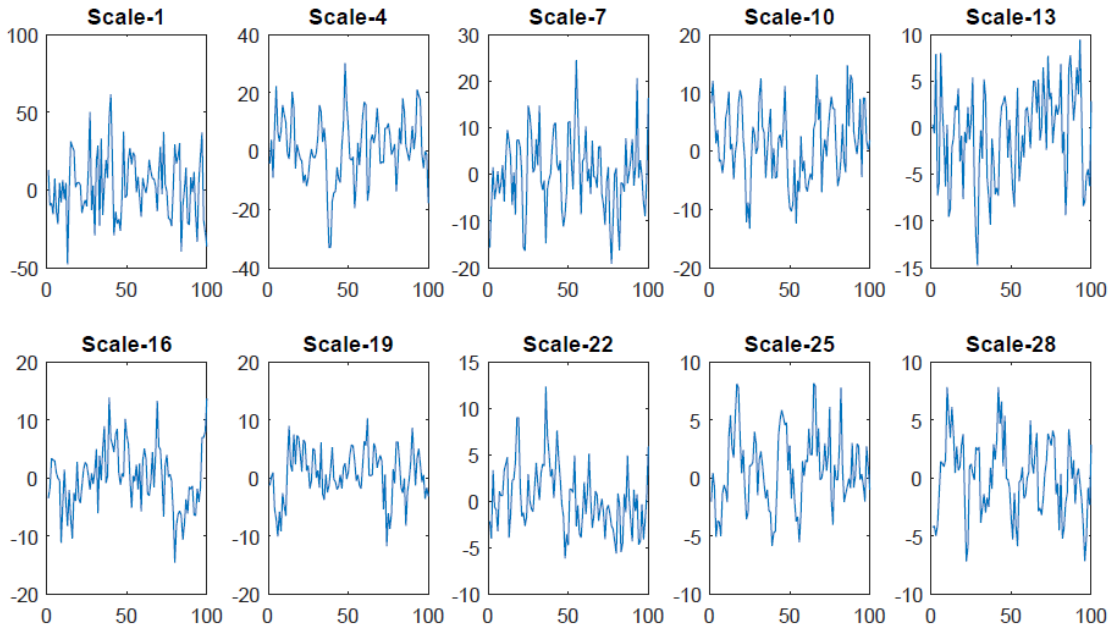


Figura 3.4: Coeficientes aleatorios temporales respecto a la base de polinomios de Legendre del proceso X generado, frecuencias discretas $n = 1, 4, 7, 10, 13, 16, 19, 22, 25, 28$.

Las gráficas de las trayectorias de innovación se generan de forma similar, obteniendo la gráfica 3.5.

```
zz1=1;
for i=MM
        SSCF2= IT(:,i);
        subplot(2,5,zz1);
        plot(SSCF2)
        title(['Scale','-',num2str(i)])
        zz1=zz1+1;
end
```

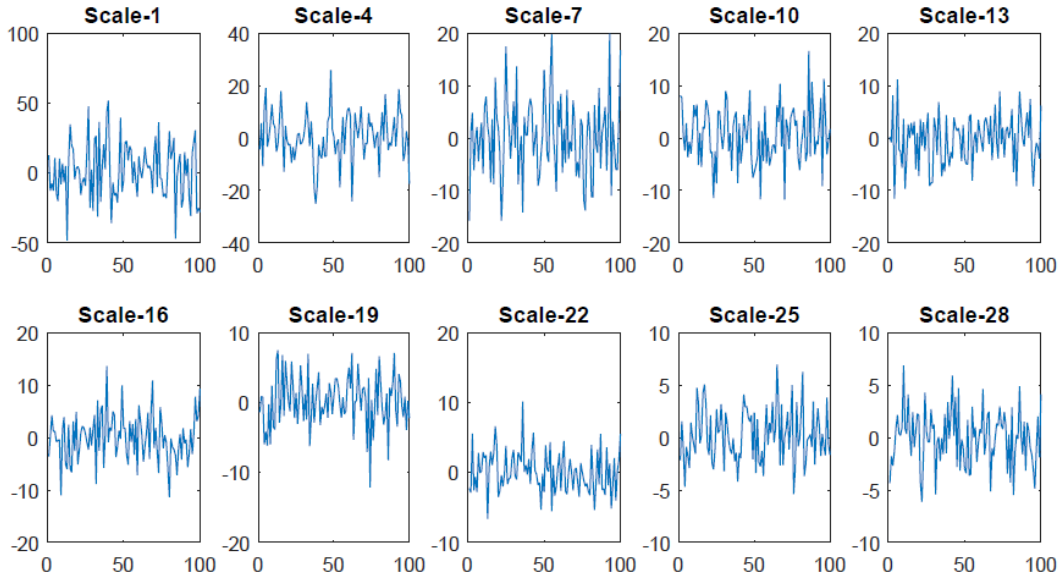


Figura 3.5: Coeficientes aleatorios temporales respecto a la base de polinomios de Legendre del proceso de innovación $L^2(\mathbb{S}_2, dv)$–valuado ε_t, frecuencias discretas $n = 1, 4, 7, 10, 13, 16, 19, 22, 25, 28$.

Seguidamente, se calculan y grafican los coeficientes beta de la ecuación esférica, obteniendo la gráfica 3.6.

```
betat= 1./((-9)*exp(polyval([1,1],linspace(-pi,pi,M))));
betatvf= betat'+ones(M,1);
x=linspace(0,M,M);
y=betatvf;
subtitle('COEFICIENTE BETA DE LA ECUACIÓN ESFÉRICA')
plot(x,y,'--gs',...
'LineWidth',2,...
'MarkerSize',10,...
'MarkerEdgeColor','b',...
'MarkerFaceColor',[0.5,0.5,0.5])
xlabel('Jacobi discrete frequencies')
```

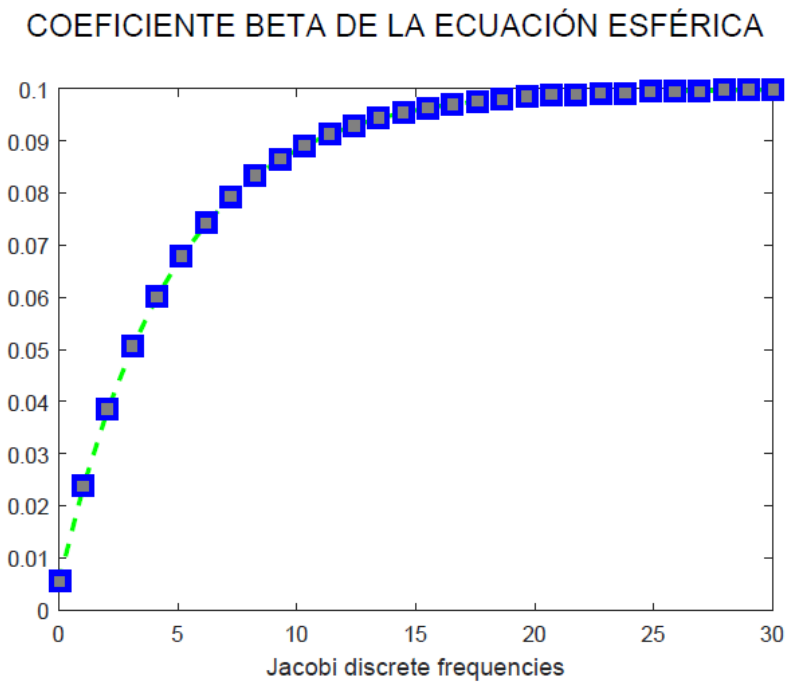


Figura 3.6: Coeficiente beta de la ecuación esférica.

Posteriormente, pasamos a la generación de la raíz cuadrada del operador de densidad. A cada trayectoria de respuesta simulada, guardada en la matriz R, se le calcula la transformación rápida de Fourier mediante la función *fft* de MatLab, obteniendo una nueva matriz *ptxx*, cuyas dimensiones se guardan en los objetos $S1$ y $S2$.

```
for k=1:M
        x=R(:,k);
        ptxx=abs(fft(x)); %transformación rapida de Fourier
        [S1,S2]=size(ptxx);
        f=linspace(-pi,pi,S1);
        f=f';
        LRDF=(4*(sin(f/2)).^(2)).^(- betatvf(k)/2);
        FC(:,k)=ptxx.*LRDF;
        w2=window(@blackmanharris,S1);
        for i=1:T    %T=100
                FC1B(i,k)=FC(i,k)*w2(i);
        end
        p1B=fit(f,FC1B(:,k),'cubicinterp');
        for i=1:T
                hh1B(i)=p1B(f(i));
        end
        SDS2B(:,k)=hh1B;
end
IFT2B=abs(ifft(SDS2B));
```

Así, se obtiene la matriz $IFT2B$ que se usa más adelante con los polinomios de Legendre. A partir de aquí, el código es idéntico al de la sección anterior a partir de la línea:

```
H=10;% REGULAR SPHERICAL GRID
```

A continuación, se muestra una de las gráficas obtenidas:

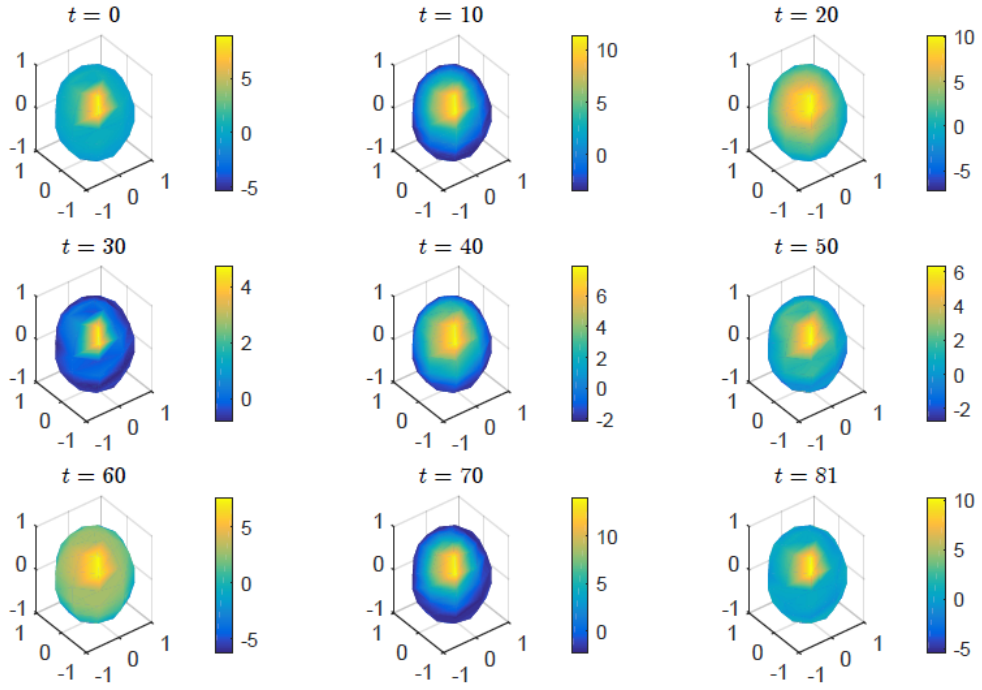


Figura 3.7: Realizaciones de X sobre la esfera para los diferentes tiempos muestreados.

4 Líneas abiertas y futuras. Aplicaciones en Astrofísica. Aplicaciones en cambio climático

Tradicionalmente, los campos aleatorios gaussianos isotrópicos se consideran el modelo estadístico subyacente de los datos del fondo cósmico de microondas (CMB). El enfoque multifraccional del movimiento Browniano se utiliza para investigar los datos CMB de la misión Planck, que consisten en mediciones de radiación CMB en ángulos estrechos de la esfera del cielo. Los resultados obtenidos sugieren que los exponentes de Hölder estimados para diferentes regiones del CMB cambian de un lugar a otro, por lo que los datos de CMB son multifraccionales [8].

Las aplicaciones sobre el modelado de los datos del CMB han generado un renovado interés desde un punto de vista teórico en la investigación de campos aleatorios en estructuras algebraicas. Por ejemplo, una de las características de esta radiación, la temperatura, está bien modelada como una sola realización de un campo aleatorio en la esfera $\mathbf{S}_1^2$ [5]. Ampliando la idea original de P. Lévy para el movimiento Browniano esférico, cada campo aleatorio de espín isotrópico gaussiano complejo se puede representar construyendo campos aleatorios sobre estructuras algebraicas: campos aleatorios en espacios homogéneos de un grupo compacto y en los paquetes de líneas de espín de la esfera $\mathbf{S}_1^2$ [5].

Además, se puede realizar un cálculo directo de la representación espectral de campos aleatorios esféricos homogéneos ponderados por espín, con espín entero arbitrario, generalizando resultados conocidos de la cosmología para la polarización del fondo de microondas cósmico spin-2 y utilizando una representación instructiva de funciones esféricas ponderadas por espín sobre el grupo Spin(3). En este grupo, el comportamiento de transformación de los campos ponderados por espín se puede tratar de forma más natural que sobre la esfera y se simplifican los cálculos para campos esféricos homogéneos [35].

El estudio de CMB es la principal herramienta para sondear modelos del Big Bang y determinar las principales constantes cosmológicas, por lo que ha sido objeto de un enorme interés en los últimos 20 años, dando lugar a dos grandes misiones satelitales, el WMAP (Sonda de anisotropía de microondas Wilkinson) de la NASA y el de la ESA (Agencia Espacial Europea). Actualmente, la polarización CMB atrae un interés similar, que será objeto del futuro satélite de la Misión LiteBird y de varios experimentos de observación. Por ejemplo, se espera que los datos de polarización puedan probar la existencia de ondas gravitacionales primordiales, proporcionando así la prueba definitiva del llamado escenario inflacionario en la dinámica del Big Bang. La función de espín también surge en otras observaciones cosmológicas muy importantes, sobre todo en los llamados datos de lentes gravitacionales débiles, el objeto del satélite de la Misión Euclid de la ESA [22].

Bibliografía

[1] Anh, V. V. & Leonenko, N. N. & Ruiz-Medina, M. D. (2016). Space-Time Fracional Stochastic Equations on Regular Bounded Open Domains. *Fractional Calculus & Applied Analysis.*, 19 (5), 1161–1199. https://doi.org/10.1515/fca-2016-0061

[2] Armussen, S. & Glynn, P. W. (2014, junio). *Stochastic Simulation. Algorithms and Analysis*. Springer.

[3] Aubin, T. (1998). *Some Nonlinear Problems in Riemannian Geometry*. Springer, Berlin.

[4] Baldi, P. & Marinucci, D. (2007). Some characterization of the spherical harmonics coefficients for isotropic random fields. *Statistics and Probability Letters*, 77, 490–496. https://doi.org/10.1016/j.spl.2006.08.016

[5] Baldi, P. y Rossi, M. (2014). Representation of Gaussian isotropic spin random fields. *Stochastic Processes and their Applications*, 124 (5), 1910-1941. https://doi.org/10.1016/j.spa.2014.01.007

[6] Barndorff-Nielsen, O. & Shephard, N. (2001). *Non-Gaussian Ornstein-Uhlenbeck based models and some of their uses in financial econometrics*. J.R. Statist. Soc.

[7] Brillinger, D.R. & Stewart, B.S. (1998). Elephant-seal movements: Modelling migrations. *Canad. J. Statist*, 26, 431–443.

[8] Broadbridge, P. & Nanayakkara, R. & Olenko, A. (2021, 18 abril). *On Multi-fractionality of Spherical Random Fields with Cosmological Applications.* Arxiv Physics, Cornell University. https://arxiv.org/abs/2104.13945

[9] *Brownian interpolation of stochastic differential equations (SDEs) for SDE, BM, GBM, CEV, CIR, HWV, Heston, SDEDDO, SDELD, or SDEMRD models - MATLAB interpolate - MathWorks España*. (s. f.). https://es.mathworks.com/help/finance/sde.interpolate.html

[10] Bulow, T. (2004). Spherical diffusion for 3D surface smoothing. *IEEE Trans. Pattern Anal. Mach. Intell*, 26 (12), 1650–1654. https://doi.org/10.1109/TPAMI.2004.129

[11] Capasso, V. & Bakstein, D. (2012, 27 julio). *An Introduction to Continuous-Time Stochastic Processes: Theory, Models, and Applications to Finance, Biology, and Medicine*. (2nd 2012 ed.). Birkhauser.

[12] D'Ovidio, M. (2015). Wright functions governed by fractional directional derivatives and fractional advection diffusion equations. *Methods and Applications of Analysis*, 22(1), 1–36. https://doi.org/10.4310/maa.2015.v22.n1.a1

[13] D'Ovidio, M., Leonenko, N. & Orsingher, E. (2016). Fractional spherical random fields. *Statistics and Probability Letters*, 116, 146–156. https://doi.org/10.1016/j.spl.2016.04.011

[14] D'Ovidio, M. & Nane, E. (2014). Time dependent random fields on spherical non-homogeneous surfaces. *Stochastic Process. Appl*, 124, 2098–2131. https://doi.org/10.1016/j.spa.2014.02.001

[15] Daly, E. & Porporato, A. (2006). Probabilistic dynamics of some jump-diffusion systems. *Physical Review E*, 73(2).

[16] Dynkin, E.B. (1965). *Markov Processes. Volume 1.* Springer-Verlag.

[17] Fuchs, C. (2013). *Inference for Diffusion Processes: With Applications in Life Sciences.* Springer Publishing.

[18] Glasserman, P. (2010). *Monte Carlo Methods in Financial Engineering.* Springer Publishing.

[19] Iacus, S.M.. (2008). *Simulation and inference for stochastic differential equations.* Springer-Verlag.

[20] Kijima, M. (1997). *Markov Processes for Stochastic Modeling.* Springer.

[21] Kuo, H.H. (2006). *Introduction to Stochastic Integration.* Springer.

[22] Lerario, A. & Marinucci, D. & Rossi, M. & Stecconi, M. (2022, 14 julio). *Geometry and Topology of Spin Random Fields.* Arxiv Mathematics, Cornell University. http://arxiv.org/abs/2207.08413

[23] Lord, R. & Koekkoek, R. & van Dijk, D. (2008, 6 febrero) *A comparison of biased simulation schemes for stochastic volatility models* https://papers.tinbergen.nl/06046.pdf

[24] Macià, F. & Oleaga, G. (2007). *Procesos estocásticos y ecuaciones diferenciales: una introducción.* https://dcain.etsin.upm.es/~fabricio/Docencia/ProcesosEstocasticos.pdf

[25] Marinucci, D. & Peccati, G. (2011). *Random Fields on the Sphere: Representation, Limit Theorems and Cosmological Applications.* Cambridge University Press.

[26] Meerschaert, M.M. & Sikorskii, A. (2012). *Stochastic Models for Fractional Calculus.* De Gruyter, Berlin

[27] Mousavi, S. M., Reihani, S. N. S., Anvari, G., Anvari, M., Alinezhad, H. G. & Tabar, M. R. R. (2016). Stochastic analysis of time series for the spatial positions of particles trapped in optical tweezers. *Scientific Reports*, 6(1).

[28] Navas, J. (s.f.). *Métodos Numéricos.* Ujaen.es. http://matema.ujaen.es/jnavas/web_modelos/pdf_mmb08_09/metodos%20numericos.pdf

[29] Olmos Ortega, M. E. (s.f.) *Procesos estocásticos, integral estocástica y ecuaciones diferenciales estocásticas: Lema de Itô.* https://www.uv.es/olmos/mod_din_estocastico.pdf

[30] Pavliotis, G. A. (2014, 19 noviembre). *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations: 60.* (2014 ed.). Springer.

[31] Simon, T. (2014). Comparing Fréchet and positive stable laws. *Electron. J. Probab.*, 19, 1–25. https://doi.org/10.1214/ejp.v19-3058

[32] Sondermann, D (2006, 27 julio). *Introduction to Stochastic Calculus for Finance: A New Didactic Approach: 579*. (2006. Corr. 3rd Printing 2007 ed.). Springer.

[33] Tabar, M. R. R. (2020). *Analysis and Data-Based Reconstruction of Complex Nonlinear Dynamical Systems: Using the Methods of Stochastic Processes.* Springer Publishing.

[34] Tankov, P. & Cont, R. (2004). *Financial Modelling with Jump Processes.* Chapman & Hall/CRC.

[35] Tessore, N. (2019, 24 abril). *The Spectral Representation of Homogeneous Spin-Weighted Random Fields on the Sphere.* Arxiv Physics, Cornell University. http://arxiv.org/abs/1904.09973

[36] tok.wiki. (s. f.). *Poisson point process*. https://hmong.es/wiki/Poisson_process

[37] tok.wiki. (s. f.). *Proceso de Ornstein-Uhlenbeck. Definición y Representación de la ecuación de Fokker-Planck*. https://hmong.es/wiki/Ornstein-Uhlenbeck_process

[38] tok.wiki. (s. f.). *Proceso Wiener. Caracterizaciones del proceso de Wiener*. https://hmong.es/wiki/Wiener_process

[39] Varshalovich, D.A., Moskalev, A.N. & Khersonskii, V.K. (2008). *Quantum Theory of Angular Momentum.*. World Scientific Publishing Co. Pte. Ltd., Singapore.

[40] Yosida, K. (1949). Brownian motion on the surface of the 3-Sphere. *Ann. Math. Stat.*, 20 (2), 292–296. https://doi.org/10.1214/aoms/1177730038