UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio Ciencias e Ingeniería

Implementación de métodos numéricos para resolver ecuaciones diferenciales estocásticas

Esteban Wirth Ordóñez

Matemáticas

Trabajo de fin de carrera presentado como requisito

para la obtención del título de

Matemático

Quito, 22 de diciembre de 2020

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ
COLEGIO CIENCIAS E INGENIERÍA

HOJA DE CALIFICACIÓN DE TRABAJO DE FIN DE CARRERA

Implementación de métodos numéricos para resolver ecuaciones diferenciales estocásticas

Esteban Wirth Ordóñez

Nombre del profesor, Título académico: Carlos Jiménez, Ph.D.

Quito, 22 de diciembre de 2020

DERECHOS DE AUTOR

Por medio del presente documento certifico que he leído la Política de Propiedad Intelectual de la Universidad San Francisco de Quito y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo de investigación quedan sujetos a lo dispuesto en la Política.

Así mismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo de investigación en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma del estudiante: .....

Nombres y Apellidos: Esteban Wirth Ordóñez

Código: 00131303

Cédula de Identidad : 1716755135

Lugar y fecha: Quito, 22 de diciembre de 2020

AGRADECIMIENTOS

Agradezco a mi tutor Carlos Jiménez por haberme guiado y apoyado para poder llevar a cabo este trabajo.

A John Skukalek le agradezco haberme enseñado qué en verdad es la matemática y la belleza y profundidad que tiene.

Finalmente quiero agradecer a la USFQ. A sus profesores y estudiantes, mis compañeros, que me han enseñado demasiado en estos años. Tratar de nombrarlos a todos o decir todo lo que me han aportado es verdaderamente imposible.

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

En este trabajo definimos el movimiento Browniano y la integral estocástica para poder establecer qué son las ecuaciones diferenciales estocásticas. Implementamos el método Euler-Maruyama y método de Milstein en python. Además damos definiciones de convergencia para poder establecer qué tan buenos son los métodos para la ecuación con la que se trabaja.

Palabras clave: Integral Estocástica, Método Euler-Maruyama, Cálculo de Itô, Métodos Numéricos, Análisis Numéricos.

ABSTRACT

In this paper we define Brownian motion and stochastic integral in order to establish what are the stochastic differential equations. We implemented the Euler-Maruyama and Milstein methods in python. Also we give the definitions of convergence in order to be able to tell how good our methods are.

Key words: Stochastic Integral, Euler-Maruyama Method, Itô Calculus, Numerical Methods, Numerical Analysis.

TABLA DE CONTENIDOS

LISTA DE FIGURAS

1. Introducción

En este trabajo contextualizaremos e implementaremos dos métodos numéricos para resolver las ecuaciones diferenciales estocástica. Además establecemos pautas para definir qué tanto aproximamos la respuesta real con nuestros algoritmos. Para contextualizar al cálculo estocástico presentamos y mostramos propiedades del movimiento browniano, la integral estocástica y la regla de la cadena estocástica. Los métodos numéricos son explicados e implementados a partir del trabajo de Desmond J. Higham, An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations (2001). Para definir qué tan bueno son nuestros algoritmos damos dos métricas para encontrar la diferencia entre la respuesta estimada y la real. Esto lo hemos hecho porque las ecuaciones diferenciales estocásticas juegan un papel fundamental para modelar varios fenómenos en el mundo. Por ejemplo, el movimiento de poblaciones, el precio de las acciones o la distribución de un virus pueden ser descritos a través de ecuaciones diferenciales estocásticas. Además encontrar respuestas analíticas a estas ecuaciones no es trivial por lo que no encontramos una solución a menos que usemos un método numérico. Para que estas soluciones tengan algún tipo de importancia debemos determinar su convergencia por lo cuál dedicamos una buena parte del paper para proveer las pautas para determinar la efectividad del algoritmo.

2. Movimiento Browniano

Definición 1: Movimiento Browniano. El Movimiento Browniano se define como un proceso estocástico definido sobre el intervalo $[0, \infty)$ cumpliendo las siguientes 3 características:

- W(0)=0 Con probabilidad 1

- Cada incremento es estacionario e independiente de los otros
- W(s)-W(t)=$\sqrt{(s-t)}$N(0,σ) tal que $s,t \in [0,\infty)$ y $t < s$
- t $\rightarrow$ W(t) es continuo

(Dobrow, 2016)

La primera característica dice que el movimiento Browniano debe iniciar en 0 todas las veces.

La segunda característica significa que cada incremento en el tiempo tiene la misma distribución de probabilidad y es independiente de cualquier cambio anterior.

La tercera característica dice que el cambio sobre $W(t)$ desde un tiempo a otro está dado por la distribución normal cuya varianza es $(s-t)\sigma^2$ y media 0. El movimiento Browniano estándard es cuando $\sigma = 1$. Desde este punto en adelante el movimiento Browniano será asumido estándard a menos que digamos lo contrario. Con estas características podemos observar unas propiedades interesantes de este proceso estocástico:

$$W(s) - W(t) = \sqrt{s-t}N(0,1) = \sqrt{(s-t)-0}N(0,1) = W(s-t) \tag{1}$$

Con la propiedad (1) podemos deducir cualidades interesantes de la continuidad y diferenciabilidad del movimiento Browniano. La siguiente explicacion no es del todo rigurosas pero da una idea de porque el movimiento Browniano es continuo lo cual está deacuerdo con su última propiedad.

$$\lim_{h\to 0} W(x+h) - W(x) = \lim_{h\to 0} W(h) = 0 \tag{2}$$

Con probabilidad 1.

En (2) observamos que cuando h es 0 el resultado es $W(0)$ que es 0 con probabilidad 1. Cuando analizamos a $W(h)$ observamos que tiene una distribución normal con esperanza 0 y varianza h. Es decir que mientras h tiende a 0 su esperanza y varianza tienden a 0 igualmente. Específicamente si tenemos cualquier $\varepsilon > 0$ y cualquier $0 < p < 1$ podemos definir un $\delta > 0$ donde $\mathbf{P}(W(x) - \varepsilon \leq W(x+\delta) \leq W(x) + \varepsilon) > p$. Esto significa que el movimiento Browniano es siempre continuo.

Ahora es natural preguntarnos si es que el movimiento Browniano es siempre continuo, si es que también es diferenciable. Hacemos la siguiente observación para dar una idea general de porque este no es el caso.

$$\lim_{h \to 0} \frac{W(x) - W(x+h)}{h} = \frac{W(h)}{h} \tag{3}$$

Vemos en (3) que la derivada escrita como un límite es $\frac{W(h)}{h}$ que tiene esperanza 0 pero varianza $\frac{1}{h}$. Por lo que mientras h tiende a 0 la varianza diverge (Ross, 2014). Por esto aunque el movimiento Browniano es siempre continuo nunca es diferenciable. Mientras que estas explicaciones ilustran que el movimiento Browniano es siempre continuo pero nunca diferenciable no son demostraciones rigurosas.

2.1. Cálculo Numérico Movimiento Browniano

Para poder calcular este proceso de forma computacional debemos trabajar con su discretización. Para hacer esto debemos definir el intervalo $[0, T]$ para no trabajar con un intervalo infinito. Ahora debemos dividir en varios segmentos de tamaño uniforme y "pequeño"para aproximar el Movimiento Browniano. Definimos a $\delta t = T/M$ donde M es la cantidad de intervalos en que queremos dividir a nuestro segmento original. Notemos que entre más pequeño es δt más cercano estará del Movimiento Browniano real. Para hacer esto debemos incrementar M cuanto podamos. Habiendo hecho esto aproximamos el Movimiento Browniano de la siguiente

forma:

$$W(t_i) - W(t_{i-1}) = \sqrt{\delta t} N(0,1) \tag{4}$$

Ahora que tenemos una aproximación discreta a nuestro proceso estocástico podemos modelarlo y graficarlo.

```
import math
import numpy as np
import matplotlib.pyplot as plt
M = 500
T = 1
deltat = T / M
W = [0] * M #declaración de vector con M espacios para llenar
W[0] = 0
i = 1
while i < M:
    W[i] = W[i - 1] + math.sqrt(deltat) * numpy.random.normal(0, 1)
    i = i + 1
tiempos = np.linspace(0, T, M)
plt.plot(tiempos, W)
```

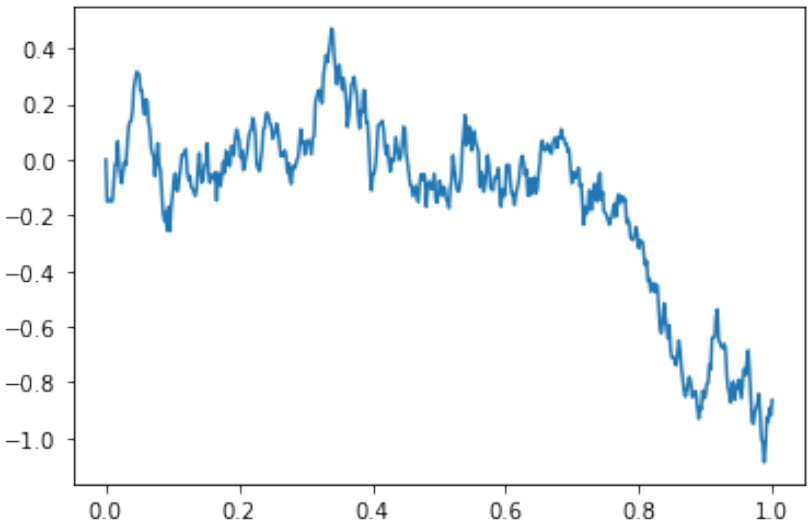

Figura 1: Gráfica de Movimiento Browniano 1

Acá tenemos que el movimiento empieza en 0 como dijimos que debía hacerlo, toma el estado anterior y lo cambia al nuevo estado usando una desviación estándard δt. Notemos que al usar solamente el anterior estado nuestro algoritmo cumple con la independencia entre estados.

Lastimosamente el modelo es bastante lento por lo que aproximar con una cantidad grande de subdivisiones del intervalo original se va volviendo más difícil. El siguiente modelo hace exactamente lo mismo pero genera todas las muestras de la distribución $N(0,1)$ y las guarda en un vector de tamaño M. Luego calculamos la posición en un momento i tomando la suma de todas las muestras de $N(0,1)$ anteriores.

```
import math
import numpy as np
import matplotlib.pyplot as plt
M = 50000
T = 5
deltat = T/M
s=np.random.normal(0,1,M)
```

```
W=math.sqrt(deltat)*np.cumsum(s,1)
tiempos=np.linspace(0,T,M)
print(W)
plt.plot(tiempos,W)
```

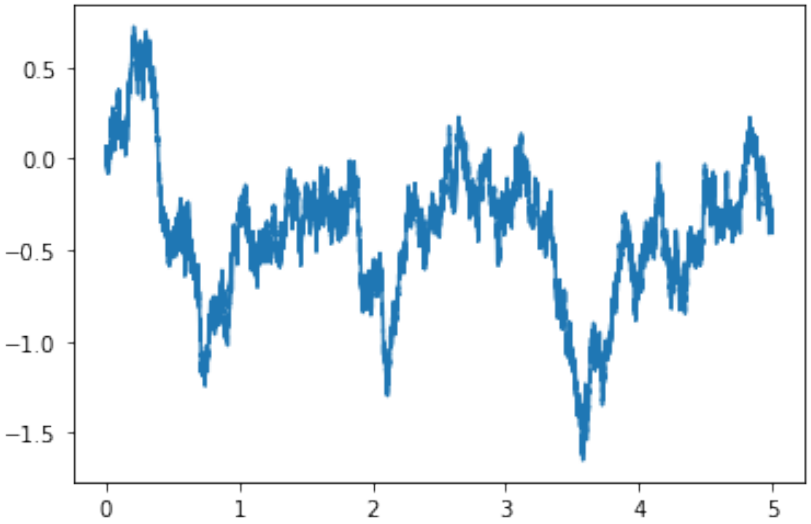

Figura 2: Gráfica de Movimiento Browniano 2

Finalmente este último modelo ayuda a mostrar porqué es importante tener un código que corra de forma rápida. Consdieremos la siguiente función aleatoria

$$\mu(W(t)) = e^{t+\frac{W(t)}{2}} \tag{5}$$

Modelaremos (5) de la siguiente forma.

```
import math
import numpy as np
import matplotlib.pyplot as plt
N = 5000
M = 1000
```

```
T = 1
deltat = T/M
s=math.sqrt(deltat)*np.random.normal(0,1,[N,M])
u=np.cumsum(s,1)
tiempos = np.linspace(0, T, M)
W = np.exp(tiempos+1/2*u)
Path1 = W[255]
Path2 = W[378]
Path3 = W[985]
AVGw = W.mean(0)
plt.plot(tiempos, AVGw) and plt.plot(tiempos,Path1)
and plt.plot(tiempos,Path2) and plt.plot(tiempos,Path3)
```

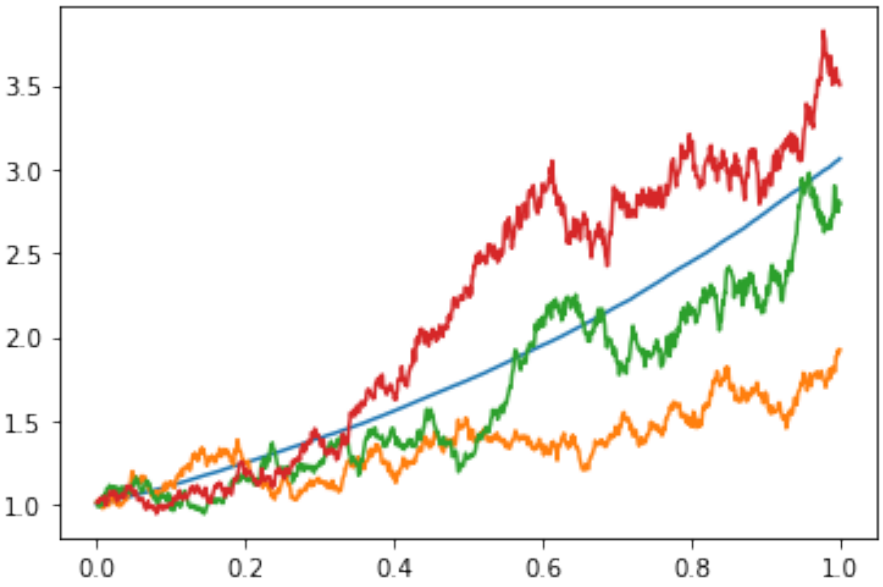

Figura 3: Gráfica de Movimiento Browniano 3

Lo que hemos hecho en este caso es modelar el movimiento de 5000 partículas que se mueven como es descrito en (5). Para hacer esto hemos segmentado el intervalo $[0, 1]$ en 1000 subdivisiones. En la gráfica mostramos el camino de 3 partículas y el camino promedio. Podemos observar que en las tres partículas no es del todo discernible que la función utilizada es

exponencial o cual es su comportamiento en general. Pero al observar el camino promedio de todas las partículas la función que recibimos es evidentemente exponencial. Interesantemente mientras que cada camino es nunca diferenciable observamos que en el camino promedio la función es aparentemente diferenciable en todo punto. Esto se puede formalizar pero escapa el enfoque de este trabajo. Para lograr modelar (5) de forma precisa debemos tomar muchas subdivisiones sobre el intervalo cerrado y para dar el camino promedio necesitamos modelar a muchas partículas. En general entre más subdivisiones y partículas la aproximación será mejor. Vemos que al incrementar estos dos valores la cantidad de iteraciones y cálculos que necesitamos son sumamente grandes. Por esto debemos tener un código que los haga de una forma veloz para dar respuestas precisas en tiempos razonables. Para lograr hacer esto evitamos ecuaciones que utilizan recurrencia y las sustituimos por operaciones sobre todo el arreglo lo cual minimiza la cantidad de tiempo de cómputo significativamente.

3. Integral Estocástica

Una definición rigurosa de una integral estocástica escapa el alcance de este trabajo pero daremos una forma de calcularla considerando sumas de Riemman para una integral Riemman-Stieltjes. La integral $\int_0^T f(t)d\alpha(t)$ puede ser aproximada con la siguiente sumatoria

$$\sum_{i=0}^{M-1} f(t_i^*)(\alpha(t_{i+1}) - \alpha(t_i)) \tag{6}$$

Usando un α que sea monotónicamente creciente Rudin (1976). De forma similar la integral estocástica $\int_0^T X(t)dW(t)$ puede ser aproximada con la sumatoria

$$\sum_{i=0}^{M-1} X(t_i)(W(t_{i+1}) - W(t_i)) \tag{7}$$

Utilizando el movimiento Browniano en vez de un α monotónicamente creciente y cambiando

la función determinística por una estocástica X. La integral estocástica está definida también para funciones determinísticas pero su versión más general es con una función estocástica.

Una cosa interesante es que en la integral Riemman-Stieltjes el punto t_i^* que se escoge en el intervalo cerrado $[t_i, t_{i+1}]$ es completamente irrelevante ya que al hacer los intervalos más pequeños cada vez el valor al que se converge es el mismo. Para la integral estocástica esto no es cierto lo cual da pie a distintos tipos de integrales dependiendo de como escogemos el punto dentro de la partición.

Defínción 2: Integral Estocástica Itô

$$\sum_{i=0}^{M-1} X(t_i)(W(t_{i+1}) - W(t_i)) \tag{8}$$

donde $t_{i+1} - t_i = \delta t$ es constante. Mientras δt tiende a 0 la sumatoria da exactamente la integral estocástica Itô (Øskendal, 2000).

Analizaremos el caso donde $X(t) = W(t)$

$$\int_0^T W(t) dW(t) \tag{9}$$

Tomando la aproximación mediante sumatoria vemos que

$$\sum_{i=0}^{M-1} W(t_i)(W(t_{i+1}) - W(t_i)) = \frac{1}{2} \sum_{i=0}^{M-1} [W^2(t_{i+1}) - W^2(t_i) - (W(t_{i+1}) - W(t_i))^2] \tag{10}$$

$$\sum_{i=0}^{M-1} W(t_i)(W(t_{i+1}) - W(t_i)) = \frac{1}{2}(W^2(T) - W^2(0) - \sum_{i=0}^{M-1} (W(t_{i+1}) - W(t_i))^2 \tag{11}$$

Se puede demostrar que el último término de (11) tiene valor esperado T y varianza $O(\delta t)$ (Higham, 2001). Donde $O(\delta t)$ significa que si dividimos la varianza para δt y aproximamos δt a 0 el límite da como ersultado una constante. Es decir la varianza tiende a 0 mientras δt tiende a 0. Por lo que mientras δt tiende a 0 el valor de ese término tiende a T. De donde la integral estocástica Itô es

$$\int_0^T W(t)dW(t) = \frac{1}{2}(W^2(T) - T) \tag{12}$$

Mientras que no se ha dado una demostración de el valor T en esta respuesta al final de la sección se proporciona un código que calcula la sumatoria y su error comparándola con (12) y veremos que este se aproxima a 0 mientras δt se aproxima a 0 de donde sabemos que la expresión escrita es correcta.

Defínción 3: Integral Estocástica Stratonovich

$$\sum_{i=0}^{M-1} X(\frac{t_i + t_{i+1}}{2})(W(t_{i+1}) - W(t_i)) \tag{13}$$

donde $t_{i+1} - t_i = \delta t$ es constante. Mientras δt tiende a 0 la sumatoria da exactamente la integral estocástica Stratonovich (Øskendal, 2000).

Nuevamente analizaremos el caso $X(t) = W(t)$.

$$\sum_{i=0}^{M-1} W(\frac{t_i + t_{i+1}}{2})(W(t_{i+1}) - W(t_i)) \tag{14}$$

se puede demostrar que

$$W(\frac{t_i + t_{i+1}}{2}) = \frac{W(t_i) + W(t_{i+1}}{2} + \delta Z_i \tag{15}$$

donde cada δZ_i es independiente y está definida como $N(0, \frac{\delta t}{4})$ (Higham, 2001). De forma similar a la deducción en (10) y (11) obtenemos que la sumatoria se reduce a

$$\frac{1}{2}(W^2(T) - W^2(0) + \sum_{i=0}^{M-1} \delta Z_i (W(t_{i+1}) - W(t_i)) \tag{16}$$

en este caso el último término de (16) tiene valor esperado 0 y varianza $O(\delta t)$ por lo que mientras δt tiende a 0 la integral estocástica Stratonovich es

$$\int_0^T W(t) dW(t) = \frac{1}{2} W^2(T) \tag{17}$$

Nuevamente en el cálculo numérico de esta integral veremos que el error calculado comparando la aproximación de la sumatoria a (16) va a tender a 0 justificando esta expresión.
En las siguientes secciones utilizaremos la integral Itô para definir las ecuaciones diferenciales estocásticas.

3.1. Cálculo Numérico Integrales

Para realizar el cálculo numérico de estas integrales utilizamos la discretización del movimiento Browniano descrita anteriormente. El código utilizado es el siguiente

```
import math
import numpy as np
import matplotlib.pyplot as plt
M = 10000000
T = 1
deltat = T/M
```

```
s=np.random.normal(0,1,M)
dW=math.sqrt(deltat)*(s)
W= np.cumsum(dW)
W1=np.insert(W, 0,0)
dW1=np.insert(dW, M,0)
z=0.5*math.sqrt(deltat)*np.random.normal(0,1,M)
ito = np.dot(W1,dW1)
strat = 0.5*(W[M-1])*(W[M-1])-np.dot(dW,z)
itoerr =math.fabs(ito-0.5*(W[M-1]*W[M-1]-T))
straterr = math.fabs(np.dot(dW,z))
```

los valores calculados son los siguientes:

$$ito = 1{,}1542492426468398$$

$$strat = 1{,}6542852965529207$$

$$itoerr = 0{,}0001464280112919436$$

$$straterr = 0{,}00011037410521113682$$

Notemos que el error calculado para la integral de Stratonovich es únicamente el cálculo del último término tendiendo a 0 ya que este valor es el que proporciona el error.

4. Método Euler-Maruyama

Definición 4: Ecuación diferencial estocástica (EDE) autónoma. Una ecuación diferencial estocástica autónoma es una ecuación diferencial que da como solución una variable aleatoria sobre un intervalo cerrado de una variable x cuyas funciones no cambian o dependen

de x directamente. Usualmente la variable x es el tiempo y esta asumción refleja la realidad ya que asumimos que los factores que definen el sistema sobre el cuál esta variable aleatoria está siendo calculada se mantiene en el tiempo definido. Este tipo de ecuaciones tienen la siguiente forma:

$$X(t) = X(0) + \int_0^t f(X(s))ds + \int_0^t g(X(s))dW(s) \tag{18}$$

Donde las integrales escritas son integrales Itô descritas anteriormente (Øskendal, 2000).

No entraremos en mayor detalle de qué significa que la variable aleatoria resultante sea solución del sistema sino daremos un método numérico para modelar esta variable sobre el intervalo de tiempo definido. Para hacer esto subdividiremos las integrales en intervalos suficientemente pequeños y utilizaremos una regla de reucrsión para calcular el valor numérico del estado de la variable en ciertos momentos del intervalo. Notemos que la estructura de esta ecuación da un estado inicial y luego la forma en la que cambia a través de las integrales. Definiremos el siguiente intervalo cerrado $[\Delta t\, j, \Delta t(j+1)]$ con $j \in \{0, ...N-1\}$ donde $\Delta t N = T$. Podemos aproximar la anterior EDE como N ecuaciones que tienen el siguiente formato:

$$X(\Delta t(j+1)) = X(\Delta t\, j) + \int_{\Delta t\, j}^{\Delta t(j+1)} f(X(s))ds + \int_{\Delta t\, j}^{\Delta t(j+1)} g(X(s))dW(s) \tag{19}$$

La idea es que al hacer Δt suficientemente pequeña podemos aproximar las ecuaciones escritas anteriormente como

$$X_{j+1} = X_j + f(X_j)\Delta t + g(X_j)(W(\Delta t)) \tag{20}$$

Donde $X_j \approx X(\Delta t\, j)$. Con $j = 0, 1, 2, ..., N-1$

Ahora tenemos N ecuaciones con la estructura descrita anteriormente. Notemos que esto nos da una regla de recurrencia que nos permite modelar la variable porque tenemos como condición inicial dada por el problema el valor de $X(0) = X_0$. Por lo que podemos ya resolver esta EDE sin mayor problemas

4.1. Solución a EDE conocida

La ecuación que resolveremos es la ecuación de Black-Scholes. Esta es una ecuación diferencial parcial estocástica que se utiliza para modelar el valor de un activo financiero a través del tiempo considerando distintos factores como valor del activo y valor de la moneda en la cual está valorado el activo. De esa ecuación diferencial parcial es posible deducir la siguiente EDE:

$$X(t) = X(0) + \int_0^t \lambda X(s)ds + \int_0^t \mu X(s)dW(s) \tag{21}$$

Donde $X(0) = X_0$ con probabilidad 1 ; $f(X(s)) = \lambda X(s)$; $g(X(s)) = \mu X(s)$ λ y μ son constantes

.

Esta ecuación tiene la siguiente solución conocida

$$X(t) = X(0)exp((\lambda - \frac{1}{2}\mu^2)t + \mu W(t)) \tag{22}$$

(Øskendal, 2000)

Modelaremos el valor real de $X(t)$ con (22) y la discretización del movimiento Browniano usando δt. Para resolver nuestra EDE con el método numérico utilizaremos Δt. Necesitamos que exista un k en los enteros que cumpla $\delta tk = \Delta t$. Hacemos esto porque el modelo de nuestra solución real se encuentra sin necesidad de reucrsividad. En otras palabras necesitamos únicamente realizar operaciones sobre vectores lo cual no toma tanto tiempo de cómputo. Además usar un δt pequeño aquí nos beneficia enormemente porque el comportamiento del movimiento Browniano, y por tanto de la variable aleatoria, es muy cercano a la realidad. Por otro lado el método numérico necesita de la recursividad por lo que tiene un costo de cómputo mucho más alto. Por este motivo usar un valor Δt muy bajo toma mucho tiempo. Usando el siguiente código modelamos el comportamiento de la solución real y la aproximación numérica para ver qué tan bueno es nuestro modelo.

```
import math
import numpy as np
import matplotlib.pyplot as plt
M = 100000
T = 1
deltat = T/M
dW=math.sqrt(deltat)*np.random.normal(0,1,M)
W=np.cumsum(dW)
dt = np.full(M,deltat)
t = np.cumsum(dt)
Xzero = 1
```

```
a = 2

b = 1

Xtrue = Xzero*(np.exp((a-(b**2)/2)*t+b*W))

k=1000

m = M/k

n=int(m)

Deltat=deltat*k

X = np.zeros(n)

Xtemp = Xzero

for p in range(n):

    Winc = sum(dW[k*p:k*(p+1)])

    Xtemp = Xtemp + Deltat*a*Xtemp + b*Xtemp*Winc

    X[p] = Xtemp

tiempos = np.linspace(0, T, n)

plt.plot(t,Xtrue) and plt.plot(tiempos,X)
```

Graficando nuestra solución obtenemos lo siguiente.

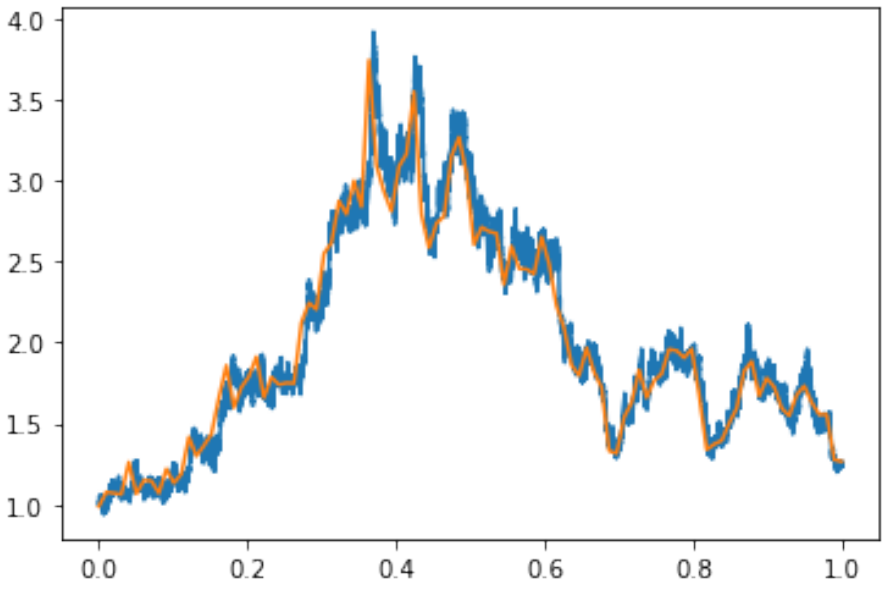

Figura 4: Solución Ecuación Black-Scholes

En azul está graficado el comportamiento de la solución conocida aproximando el movimiento browniano con 100000 (cien mil) subdivisiones sobre el intervalo $[0, 1]$. En naranja está graficada el resultado con el método numérico Euler-Maruyama usando únicamente 100 subdivisiones sobre el intervalo $[0, 1]$. Vemos que el método numérico con muy pocas subdivisiones logra dar el comportamiento general de la variable aleatoria. Los siguientes gráficos son usando 500 y 1000 subdivisiones respectivamente:

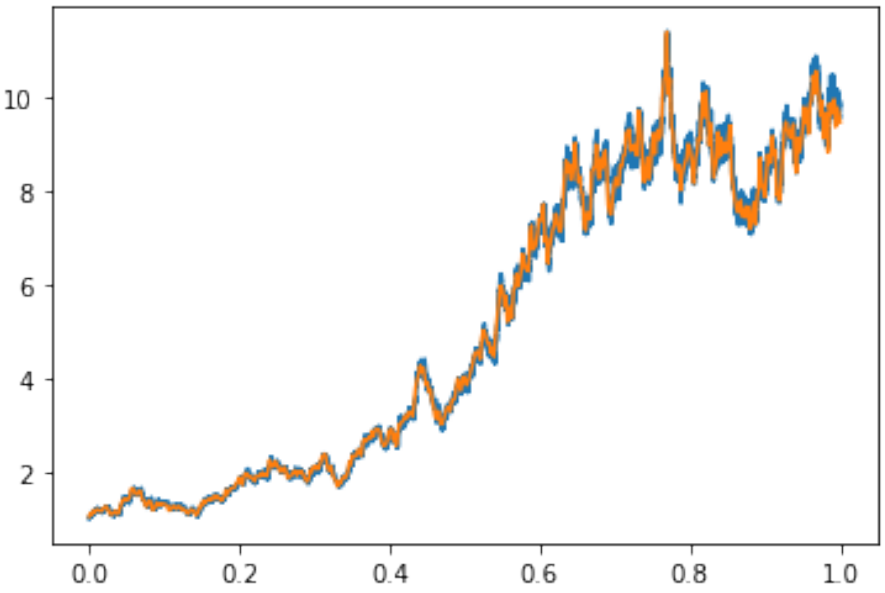

Figura 5: Solución Ecuación Black-Scholes 2

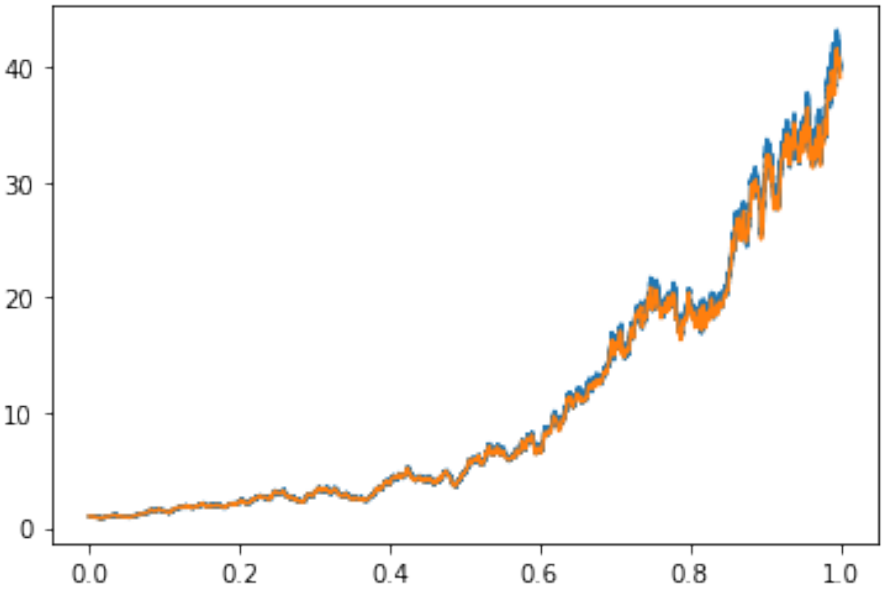

Figura 6: Solución Ecuación Black-Scholes 3

Vemos que la aproximación se vuelve cada vez mejor.

5. Convergencia del algoritmo

Para esta sección observaremos como el algoritmo se aproxima a la solución analítica. Por esto distinguiremos entre el valor obtenido por el algoritmo y la solución conocida de la siguiente forma: X_n será el resultado obtenido al realizar n pasos del algoritmo, $X(\sigma)$ será el valor obtenido por la función exacta de la solución donde $\sigma = n(\Delta t)$ donde Δt es el tamaño de paso que usamos en el algoritmo y n es el mismo que definimos anteriormente. Habiendo aclarado esto presentaremos el primer tipo de convergencia.

Definición 5: Convergencia fuerte. El algoritmo converge fuertemente si existe una constante C y γ tal que

$$E_{\Delta t} := \mathbb{E}|X_n - X(\sigma)| \leq C(\Delta t)^{\gamma} \tag{23}$$

(Highman, 2020) Es decir el error de la convergencia fuerte con un cierto Δt está definido como la esperanza de la diferencia entre el algoritmo y la solución exacta. Si esta constante C existe tenemos que el algoritmo tiene una convergencia fuerte de γ. En este trabajo nos enfocaremos en el error en el último punto del intervalo en el cual estamos trabajando. En la experimentación numérica daremos valores encontrados para C y γ. El objetivo será dar un resultado que corrobore que para el método Euler-Maruyama $\gamma = \frac{1}{2}$ cuando las funciones f y g cumplen ciertos requerimientos. Esto puede ser demostrado pero escapa el enfoque de este trabajo. Con $\gamma = \frac{1}{2}$ y la ecuación (23) podemos ver que

$$E_{\Delta t} \leq C(\Delta t)^{\frac{1}{2}} \tag{24}$$

A partir de este resultado y la inequidad de Markov que dice que si $\mathbb{E}(X)$ es finito entonces podemos decir lo siguiente

$$\mathbb{P}(X \geq a) \leq \frac{\mathbb{E}|X|}{a} \tag{25}$$

Con las ecuaciones (24) y (25) obtenemos

$$\mathbb{P}(|X_n - X(\sigma)| \geq a) \leq \frac{C(\Delta t)^{\frac{1}{2}}}{a} \tag{26}$$

De donde, vemos que si tenemos una convergencia fuerte de $\frac{1}{2}$ podemos hacer que nuestro error en cualquier punto fijado en el intervalo sea tan pequeño como queremos con una probabilidad cercana a 1 si reducimos Δt lo suficiente.

Defínción 6: Convergencia débil. Un algoritmo converge débilmente a γ si existe una constante C y γ tal que

$$|\mathbb{E}(X_n) - \mathbb{E}(X(\sigma))| \leq C(\Delta t)^{\gamma} \tag{27}$$

(Highman, 2020)

Cuando tenemos funciones f y g que satisfagan ciertas condiciones el método Euler-Maruyama converge debilmente a 1. Este resultado igual será verificado en la subsección Experimentación Numérica.

5.1. Experimentación Numérica

Primero haremos un experimento numérico usando 300 caminos sobre el movimiento Browniano en el intervalo $[0,1]$. Con esto simulamos 300 veces nuestra solución de la variable

aleatoria con la ecuación exacta y $\delta t = 0{,}0001$. Finalmente resolvimos la EDE usando el método Euler-Maruyama variando el tamaño de Δt y comparábamos el error del valor final entre las simulaciones y la solución real. A partir de esto logramos calcular el valor de γ y C.

```
import math
import numpy as np
import matplotlib.pyplot as plt
M = 10000
y = 300
T = 1
C = [1000, 100, 50, 25, 20, 10, 5, 1]
c = 8
deltat = T/M
dt = np.full([y,M],deltat)
t = np.cumsum(dt,1)
dW=math.sqrt(deltat)*np.random.normal(0,1,[y,M])
W=np.cumsum(dW,1)
Xzero = 1
a = 2
b = 1
Xtrue = Xzero*(np.exp((a-(b**2)/2)*t+b*W))
Xfinal = np.zeros([y,c])
Error = np.zeros([y,c])
DT= np.zeros(c)
for i in range(c):
    k=C[i]
    n=int(M/k)
    Deltat=deltat*k
    DT[i] = Deltat
```

```
    for j in range(y):
        Xtemp = Xzero
        for p in range(n):
            Winc = sum(dW[j,k*p:k*(p+1)],0)
            Xtemp = Xtemp + Deltat*a*Xtemp + b*Xtemp*Winc
            Xfinal[j,i]=Xtemp
for i in range(c):
    for j in range(y):
        Error[j,i] = math.fabs(Xfinal[j,i]-Xtrue[j,M-1])
E=Error.mean(0)
logE = np.log(E)
logDT = np.log(DT)
plt.plot(logDT,logE)
```

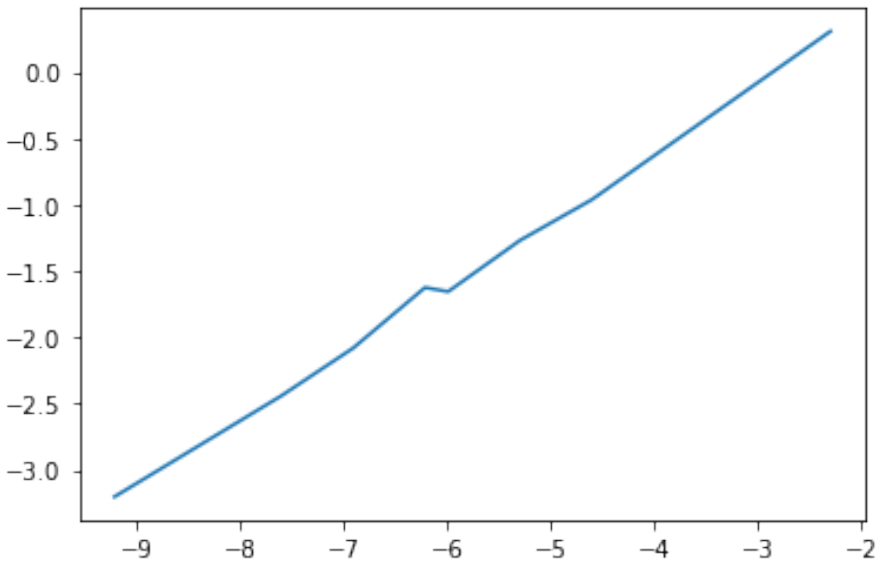

Figura 7: Convergencia Fuerte Método Euler-Maruyama

Al realizar una regresión lineal obtenemos que la ecuación de la línea es $y = 0{,}5062x + 1{,}4298$ con $R^2 = 0{,}9976$. Esto da un ajuste bastante bueno a la realción que se presenta entre la esperanza del error y el Δt utilizado. Además vemos que $\gamma = 0{,}5062$ lo cuál da un buen indicio

que en efecto nuestro algoritmo converge fuertemente a $\frac{1}{2}$. Además calculamos que $C = 4{,}1779$.

```
import math
import numpy as np
import matplotlib.pyplot as plt
M = 10000
y = 300
T = 1
C = [1000, 100, 50, 25, 20, 10, 5, 1]
c = 8
deltat = T/M
dt = np.full([y,M],deltat)
t = np.cumsum(dt,1)
dW=math.sqrt(deltat)*np.random.normal(0,1,[y,M])
W=np.cumsum(dW,1)
Xzero = 1
a = 2
b = 1
Xtrue = Xzero*(np.exp((a-(b**2)/2)*t+b*W))
Xfinal = np.zeros([y,c])
Error = np.zeros([y,c])
DT= np.zeros(c)
for i in range(c):
    k=C[i]
    n=int(M/k)
    Deltat=deltat*k
    DT[i] = Deltat
    for j in range(y):
        Xtemp = Xzero
```

```
        for p in range(n):
            Winc = sum(dW[j,k*p:k*(p+1)],0)
            Xtemp = Xtemp + Deltat*a*Xtemp + b*Xtemp*Winc
            Xfinal[j,i]=Xtemp
for i in range(c):
    for j in range(y):
        Error[j,i] = math.fabs(Xfinal[j,i]-Xtrue[j,M-1])
E=Error.mean(0)
logE = np.log(E)
logDT = np.log(DT)
plt.plot(logDT,logE)
```

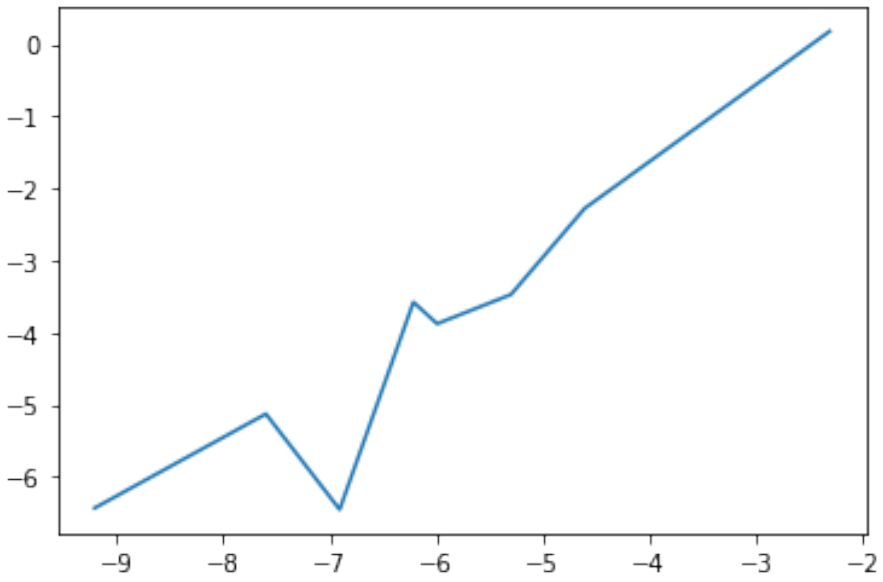

Figura 8: Convergencia Débil Método Euler-Maruyama

Donde vemos que $\gamma = 1{,}0048$ con $R^2 = 0{,}8865$ lo cual da un buen ajust a nuestro valor esperado. La falta de ajuste se podría disminuir corriendo una mayor cantidad de caminos para que la esperanza sobre cada camino se aproxime más a la verdadera y nuestro error lo refleje.

6. Método de Milstein

En la anterior sección observamos que el método Euler-Maruyama converge fuertemente a $\frac{1}{2}$. Por la ecuación (26) vemos que es sumamanete importante incrementar el valor de γ para tener un error pequeño con probabilidad cercana a 1. Por esto damos otro método para resolver una EDE que tenga una convergencia de 1 con los mismos requerimientos de f y g que tenía el método Euler-Maruyama. Este es el método de Milstein y está dado por la siguiente ecuación para resolver (18).

$$X_{j+1} = X_j + f(X_j)\Delta t + g(X_j)(W(\Delta t)) + \frac{1}{2}g(X_j)g'(X_j)(W(\Delta t)^2 - \Delta t) \tag{28}$$

Este método se obtiene de la expansión Itô-Taylor (Higham, 2001) que no será explicada en este trabajo. Saltaremos directamente a realizar un experimento numérico para mostrar que esta solución tiene una convergencia fuerte de 1.

6.1. Experimentación Numérica

Realizamos el experimento de la misma forma que fue descrita en la sección de convergencia del método de Euler-Maruyama.

```
import math
import numpy as np
import matplotlib.pyplot as plt
M = 10000
y = 300
T = 1
```

```
C = [1000, 100, 50, 25, 20, 10, 5, 1]
c = 8
deltat = T/M
dt = np.full([y,M],deltat)
t = np.cumsum(dt,1)
dW=math.sqrt(deltat)*np.random.normal(0,1,[y,M])
W=np.cumsum(dW,1)
Xzero = 1
a = 2
b = 1
Xtrue = Xzero*(np.exp((a-(b**2)/2)*t+b*W))
Xfinal = np.zeros([y,c])
Error = np.zeros([y,c])
DT= np.zeros(c)
for i in range(c):
    k=C[i]
    n=int(M/k)
    Deltat=deltat*k
    DT[i] = Deltat
    for j in range(y):
        Xtemp = Xzero
        for p in range(n):
            Winc = sum(dW[j,k*p:k*(p+1)],0)
            Xtemp = Xtemp + Deltat*a*Xtemp + b*Xtemp*Winc + 1/2*b*Xtemp*b*(Winc*Winc-
            Xfinal[j,i]=Xtemp
for i in range(c):
    for j in range(y):
        Error[j,i] = math.fabs(Xfinal[j,i]-Xtrue[j,M-1])
E=Error.mean(0)
```

```
logE = np.log(E)
logDT = np.log(DT)
plt.plot(logDT,logE)
```

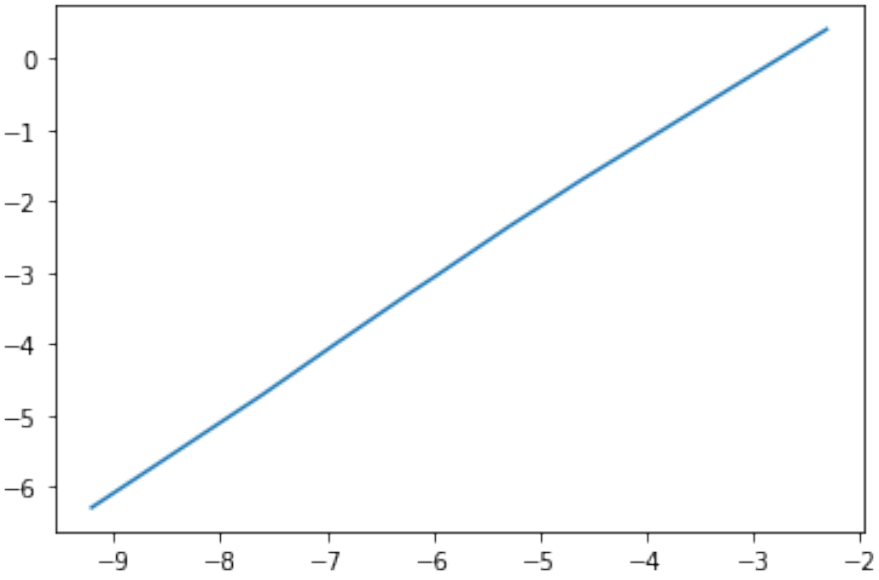

Figura 9: Convergencia Fuerte Método Milstein

Al realizar una regresión lineal vemos que la ecuación de la línea es $y = 0{,}9746x + 2{,}738$ con $R^2 = 0{,}9993$. Esto nos da una idea bastante clara que $\gamma = 1$ lo cual apunta a que este método es considereablemente más eficiente.

7. Estabilidad Lineal

En esta sección seguiremos utilizando la notación descrita anteriormente. Ya que tenemos una forma de analizar la convergencia del algoritmo dentro del intervalo daremos requisitos para determinar la convergencia mientras extendemos nuestro intervalo hacia el infinito. Primero describiremos dos tipos de convergencia a largo plazo para la solución exacta de la EDE.

Definición 7: Estabilidad de Esperanza Cuadrática. Decimos que la solución de una EDE tiene estabilidad de esperanza cuadrática cuando

$$\lim_{t\to\infty} \mathbb{E}([X(t)]^2) = C \in \mathbb{R} \tag{29}$$

Es decir el límite converge mientras t va al infinito.

Definición 8: Estabilidad Asintótica. Decimos que una EDE tiene estabilidad asintótica si con probabilidad 1

$$\mathbb{P}(\lim_{t\to\infty} |X(t)| = C) = 1 \tag{30}$$

Es decir que la solución a la EDE converge a una constante C con probabilidad 1. (Higham, 2001)

Al saber esto podemos hacer la pregunta ¿Con qué Δt nuestro algoritmo va a converger de esta misma forma? Es decir que Δt es necesario para que las siguientes expresiones se cumplan.

$$\lim_{n\to\infty} \mathbb{E}([X_n]^2) = C \tag{31}$$

$$\mathbb{P}(\lim_{n\to\infty} |X_n| = C) = 1 \tag{32}$$

Vemos que estas expresiones no pueden ser analizadas de forma general sino que debemos trabajar conociendo la forma de nuestra EDE para poder responder de forma apropiada estas preguntas. Por este motivo daremos las condiciones necesarias para que la Ecuación de Black-Scholes (21) converja de las formas que describimos anteriormente. En este caso permitiremos que λ y μ sean valores complejos para dar información más general sobre las convergencias de

la EDE y del algoritmo.

$$\lim_{t\to\infty} \mathbb{E}([X(t)]^2) = 0 \iff Re\{\lambda\} + \frac{1}{2}|\mu|^2 < 0 \tag{33}$$

$$\mathbb{P}(\lim_{t\to\infty} |X(t)| = 0) = 1 \iff Re\{\lambda - \frac{1}{2}\mu^2\} < 0 \tag{34}$$

(Higham, 2001) La estabilidad asintótica pide exactamente que la parte real de la ecuación exponencial que resuelve esta EDE sea negativa para que esta converja a 0. Interesantemente la condición de la estabilidad de media cuadrática causa que automáticamente se cumpla los requerimientos necesarios para la estabilidad asintótica, es decir es una condición más fuerte.

Asumiendo que tenemos λ y μ que cumplen con el requerimiento de estabilidad de media cuadrática, y por lo tanto asintótica también, daremos las condiciones sobre Δt para que el algoritmo también cumpla con ambas estabilidades.

$$\lim_{n\to\infty} \mathbb{E}([X_n]^2) = 0 \iff |1 + \Delta t\lambda|^2 + \Delta t|\mu|^2 < 1 \tag{35}$$

$$\mathbb{P}(\lim_{n\to\infty} |X_n| = 0) = 1 \iff \mathbb{E}\log|1 + \Delta t\lambda + \sqrt{\Delta t}\mu N(0,1)| < 0 \tag{36}$$

(Higham, 2001) La condición en (35) se encuentra con propiedades de la esperanza y la regla de recurrencia. Por otro lado (36) se puede encontrar usando la ley fuerte de grandes números y la ley del logaritmo iterado. Vemos que (36) puede ser reescrita como

$$\mathbb{P}(\lim_{n\to\infty} |X_n| = 0) = 1 \iff \mathbb{E}\log|1 + \Delta t\lambda + W(\Delta t\mu^2)| < 0 \tag{37}$$

(Higham, 2001) No se darán deducciones formales de esto pero se trabjará con un ejemplo en la siguiente subsección que mostrará que estas condiciones son correctas.

7.1. Ejemplo Numérico

Realizaremos un pequeño ejemplo sobre la ecuación de Black-Scholes que ilustra los valores de Δt con los cuáles la ecuación tiene estabilidad asintótica y de media cuadrática. Los parámetros escogidos fueron $\gamma = -3$ y $\mu = \sqrt{3}$. Es importante tener en cuenta que las gráficas a continuación están planteadas con una escala logarítmica sobre el eje y. A continuación mostramos el código y gráficas para la estabilidad asintótica.

```
import math
import numpy as np
import matplotlib.pyplot as plt
T = 500
C = [500, 1000, 1500, 2000, 2500]
c = len(C)
Xzero = 1
a = -3
b = math.sqrt(3)
X = np.zeros([C[c-1],c])
DT = np.zeros(c)
for i in range(c):
    DT[i]=T/C[i]
    Xtemp = Xzero
    for p in range(C[i]):
        Xtemp = Xtemp + DT[i]*a*Xtemp + b*Xtemp*math.sqrt(DT[i])*np.random.normal(0,1)
```

```
            X[p,i]=math.fabs(Xtemp)
Path1=X[:C[0],0]
Path2=X[:C[1],1]
Path3=X[:C[2],2]
Path4=X[:C[3],3]
Path5=X[:C[4],4]
T1=np.linspace(0,T,C[0])
T2=np.linspace(0,T,C[1])
T3=np.linspace(0,T,C[2])
T4=np.linspace(0,T,C[3])
T5=np.linspace(0,T,C[4])
plt.yscale("log")
plt.plot(T1, Path1)
plt.plot(T2, Path2)
plt.plot(T3, Path3)
plt.plot(T4, Path4)
plt.plot(T5, Path5)
```

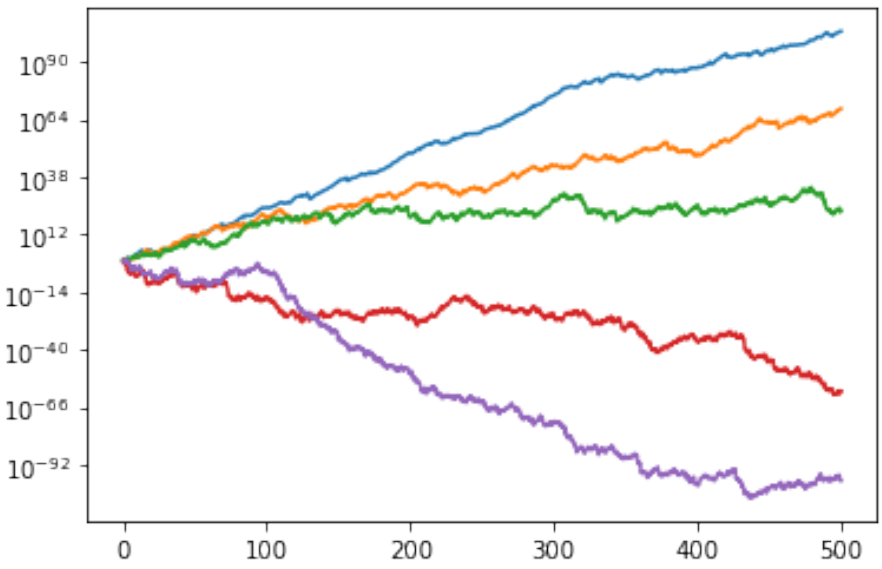

Figura 10: Estabilidad Asintótica

Acá vemos claramente que con los $\Delta t = \{1, \frac{1}{2}, \frac{1}{3}\}$ el algoritmo diverge mientras que con $\Delta t = \{\frac{1}{4}, \frac{1}{5}\}$ converge. Notemos que la diferencia entre los valores finales y pronóstico de nuestro algoritmo cambia dramáticamente con nuestro valor Δt. Cabe recalcar que la proporción del Δt más pequeño y más grande es de 5 mientras que entre los resultados finales es de 10^180.

A continuación analizaremos la estabilidad de media cuadrática sobre la misma ecuación de Black-Scholes. Para esto modelamos 5000 caminos sobre cada uno de los Δt mencionados anteriormente y tomamos el promedio sobre el valor del algoritmo en cada punto sobre el tiempo elevado al cuadrado.

```
import math
import numpy as np
import matplotlib.pyplot as plt
T = 50
C = [50, 100, 150, 200, 250]
c = len(C)
y = 5000
Xzero = 1
a = -3
b = math.sqrt(3)
Xp = np.zeros([C[c-1],c,y])
DT = np.zeros(c)
for i in range(c):
    DT[i]=T/C[i]
    for j in range(y):
        Xtemp = Xzero
        for p in range(C[i]):
            Xtemp = Xtemp + DT[i]*a*Xtemp + b*Xtemp*math.sqrt(DT[i])*np.random.normal
```

```
            Xp[p,i,j]=Xtemp*Xtemp
X = Xp.mean(2)
Path1=X[:C[0],0]
Path2=X[:C[1],1]
Path3=X[:C[2],2]
Path4=X[:C[3],3]
Path5=X[:C[4],4]
T1=np.linspace(0,T,C[0])
T2=np.linspace(0,T,C[1])
T3=np.linspace(0,T,C[2])
T4=np.linspace(0,T,C[3])
T5=np.linspace(0,T,C[4])
plt.yscale("log")
plt.plot(T1, Path1)
plt.plot(T2, Path2)
plt.plot(T3, Path3)
plt.plot(T4, Path4)
plt.plot(T5, Path5)
```

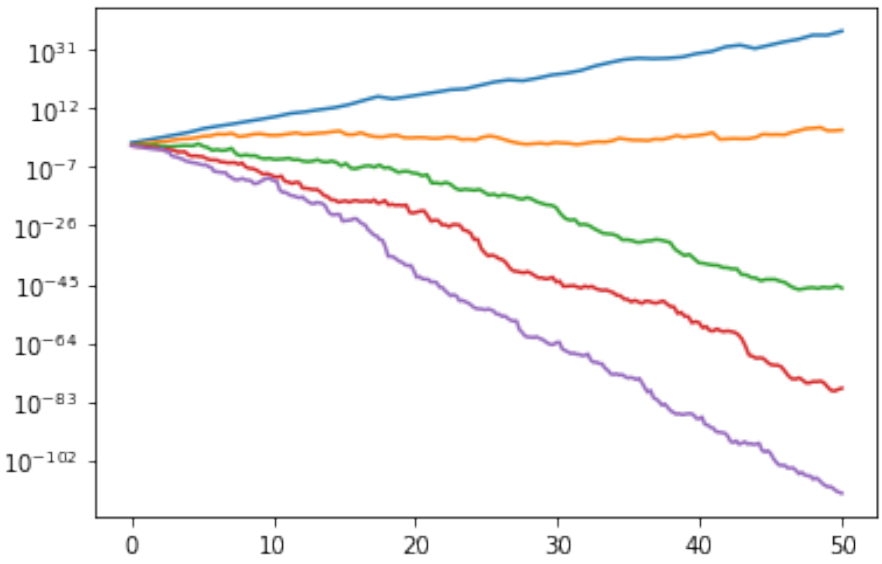

Figura 11: Estabilidad de Media Cuadrática

En este caso observamos convergencia sobre los 3 tamaños de cambio más pequeños. Esto es de esperarse con los valores de $\frac{1}{4}$ y $\frac{1}{5}$ eran esperados pero sobre $\frac{1}{3}$ esto no está del todo de acuerdo con la teoría. Al ingresar los valores de γ, β y Δt no cumplen con el requisito establecido en la desigualdad ya que la expresión es 1 y no menor a 1. Asumimos que por la cercanía a la desigualdad debemos utilizar un T más alto y una mayor cantidad de caminos para obtener un promedio más confiable.

8. Regla de la Cadena Estocástica

Para esta sección escribiremos la ecuación (19) en términos de diferenciales y no una integral sobre un intervalo arbitrariamente pequeño. Esto es notación ya que el significado de ambas es el mismo.

$$dX(t) = f(X(t))dt + g(X(t))dW(t) \tag{38}$$

Ahora mostraremos otra diferencia sustancial entre el cálculo determinístico y el cálculo estocástico al mostrar que la regla de la cadena es distinto. Si tenemos dos funciones reales continuamente diferenciables f y g sabemos que la regla de la cadena establece lo siguiente

$$\frac{d}{dx}(f(g(x))) = f'(g(x))g'(x) \tag{39}$$

Si tratamos de extrapolar esto al cálculo estocástico usando un $X(t)$ que resuelve la ecuación (29) y una función V que sea continuamente diferenciable pensaríamos que podemos definir el diferencial de V de la siguiente forma:

$$d(V(X(t))) = V'(X(t))dX(t) \tag{40}$$

donde $V'(X(t))$ es la derivada de $V(X(t))$ con respecto a $X(t)$ y $dX(t)$ está definido como en (38). Al observar las propiedades del cálculo Itô podemos ver que esto no se cumple aunque los motivos de esto escapan el alcance de este trabajo. Entregaremos la definición para el diferencial de $V(X(t))$ y luego realizaremos un cálculo numérico para mostrar que esta definición se cumple.

$$d(V(X(t))) = V'(X(t))dX(t) + \frac{1}{2}[g(X(t))]^2 \frac{d^2V(X(t))}{dX^2}dt \tag{41}$$

(Jiménez & Romera, 2008)

8.1. Experimentación numérica

Para mostrar que la regla de la cadena propuesta es correcta trabajaremos con la siguiente EDE

$$dX(t) = (\alpha - X(t))dt + \beta\sqrt{X(t)}dW(t) \tag{42}$$

Es decir $f(X(t)) = \alpha - X(t)$ y $g(X(t)) = \beta\sqrt{X(t)}$. Esta EDE tiene aplicaciones en matemáticas financieras para analizar la volatilidad, taza de interés y precio de ciertos activos financieros. Interesantemente si $X(0) > 0$ con probabilidad 1 esta condición se mantiene en el resto del tiempo.

Usamos la función $V(X(t)) = \sqrt{X(t)} = M(t)$. Entonces obtenemos que

$$dM(t) = V'(X(t))dX(t) + \frac{1}{2}[g(X(t))]^2 V''(X(t))dt \tag{43}$$

$$dM(t) = \frac{1}{2\sqrt{X(t)}}[(\alpha - X(t))dt + \beta\sqrt{X(t)}dW(t)] + \frac{1}{2}\beta^2 X(t)\frac{-1}{4[X(t)]^{\frac{3}{2}}}dt \tag{44}$$

$$dM(t) = \frac{\alpha}{2\sqrt{X(t)}}dt - \frac{\sqrt{X(t)}}{2}dt + \frac{1}{2}\beta dW(t) + \frac{-\beta^2}{8\sqrt{X(t)}}dt \tag{45}$$

$$dM(t) = \frac{4\alpha - \beta^2}{8M(t)}dt - \frac{M(t)}{2}dt + \frac{\beta}{2}dW(t) \tag{46}$$

Lo que haremos a continuación es utilizar Euler-Maruyama sobre ambos diferenciales encontrados. Al aplicarlo en $dX(t)$ encontraremos un valor numérico para $X(t)$ y luego aplicaremos una raíz sobre todos los valores obteniendo $M(t)$. Aplicaremos después Euler-Maruyama a $dM(t)$ obteniendo $M(t)$ a partir de la regla de la cadena. Si estos valores coinciden veremos que la regla de la cadena se sotiene. Utilizaremos los siguientes valores para realizar este experimento $\alpha = 2$, $\beta = 1$, $X(0) = 1$ con probabilidad 1. A pesar de esto vemos que en efecto la estabilidad cuadrática es una condición más fuerte que la estabilidad asintótica.

```
import math
import numpy as np
import matplotlib.pyplot as plt
M = 10000
T = 1
deltat = T/M
dW=math.sqrt(deltat)*np.random.normal(0,1,M)
```

```
W=np.cumsum(dW)
dt = np.full(M,deltat)
t = np.cumsum(dt)
Xzero = 3
Mzero = math.sqrt(Xzero)
a = 2
b = 1
X = np.zeros(M)
Mt = np.zeros(M)
Mtemp = Mzero
Xtemp = Xzero
for p in range(M):
    Xtemp = Xtemp + (a-Xtemp)*deltat + b*math.sqrt(Xtemp)*dW[p]
    X[p] = Xtemp
    Mtemp = Mtemp+((4*a-b*b)/(8*Mtemp)-1/2*Mtemp)*deltat+1/2*b*dW[p]
    Mt[p] = Mtemp
V=np.sqrt(X)
Dif= np.abs(Mt-V)
print(min(Dif))
print(max(Dif))
print(sum(Dif)/len(Dif))
plt.plot(t,V)
plt.plot(t,Mt)
```

$$1{,}7002795105192092e-07$$

$$0{,}0009653204141064631$$

$$0{,}0002869177468225202$$

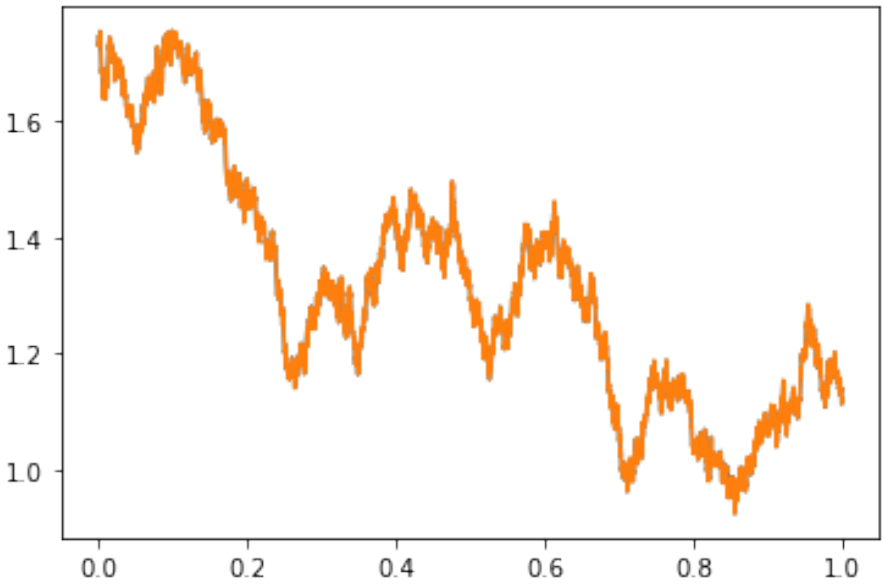

Figura 12: Regla de la Cadena Estocástica

Vemos a partir de la gráfica y los errores impresos que la regla de la cadena estocástica es correcta.

9. Conclusión y Recomendaciones

Durante todo este paper hemos logrado dar una introducción bastante completa de las ecuaciones diferenciales estocásticas y sus métodos de resolución numérica. Aprovechamos para mencionar que para mejorar este paper debemos indagar más sobre el cálculo estocástico y otorgar demostraciones más rigurosas de los resultados obtenidos. Además la investigación de teoremas particulares sobre las condiciones necesarias para que las convergencias se cumplan sería una adición sumamente positiva. Por otro lado se debe trabajar para generalizar lo descrito en este paper para el cálculo de stratonovich. Además mientras que tenemos un método para definir la convergencia de nuestros algoritmos se debe indagar más sobre la estimación del error máximo sobre nuestro modelo que es meramente mencionado en este trabajo aunque no se realizaron experimentos específicos sobre este. Finalmente se debe reestructurar la escritura

de los códigos para optimizar el uso de memoria y la velocidad del algoritmo para que nuestros resultados puedan ser considerablemente mejores. Vemos que, por ejemplo, en la convergencia débil del algoritmo Euler-Maruyama por la falta de memoria el resultado que se quería ejemplificar no fue tan contundente porque se necesitaban más corridas para estimar de mejor forma los valores esperados.

10. Bibliografía

Dobrow, R. P. (2016). Introduction to stochastic processes in r.

Higham, D. J. (2001). An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM Review*, *43*(3), 525–546.

Highman, D. (2020). Euler maruyama.

Jiménez, R. & Romera, R. (2008). Procesos estocásticos.

Ross, S. M. (2014). *Introduction to Probability Models*. Elsevier.

Rudin, W. (1976). *Principles of Mathematical Analysis*. McGraw-Hill.

Øskendal, B. (2000). *Stochastic Differential Equations an Introduction*. Springer-Verlag.