Movimiento Browniano

Francisco Gerardo Meza Fierro

1 Introducción

Esta práctica consiste en examinar los efectos de la dimensión en el número de veces que una caminata con movimiento Browniano regresa al origen. Además, se analizará el tiempo de ejecución de una caminata en términos de su largo y dimensión, así como la comparación entre una implementación paralela y otra que no la aproveche.

2 Regresos al origen

La Figura 1 muestra que es evidente que mientras mayor sea la dimensión en donde cierta partícula se mueve de manera aleatoria, mayor será la probabilidad de que la partícula no regrese al origen durante la caminata.

Para determinar si el número de pasos (guardado en la variable **pasos**) o si el número de repeticiones del experimento (guardado en la variable **repetir**) causa un efecto significativo en la cantidad de veces que se regresa al origen, se hizo uso repetido de la función **sum(datos[1,])/repetir** alterando independientemente los valores en **pasos** y **repetir** en cada uso de la función, para comparar los promedios de veces en que se regresaba al origen en la primera dimensión.

Algunos de los resultados obtenidos fueron

repetir	pasos	promedio 1	promedio 2	promedio 3	promedio 4	promedio 5
100	200	10.85	9.93	9.77	10.17	10.34
10,000	200	10.3417	10.3475	10.3464	10.2261	10.1572
100	20,000	118.22	105.94	118.95	106.66	112.52

Es claro que incrementar la variable **pasos** es un factor significativo en el número de veces que la partícula regresa al origen.

3 Tiempo de ejecución

Para determinar el tiempo de ejecución de la caminata se hizo uso de **inicioT <- Sys.time()** y de **finT <- Sys.time()** al inicio y fin del código, respectivamente; donde la primer variable guarda la fecha y hora en que el programa empezó a correr y la segunda variable guarda la fecha y hora en que el programa terminó de ejecutarse. El tiempo total en que el programa se ejecutó se guardó en la variable **tiempo**.

Para esta sección, por comodidad se fijó la variable **repetir** en 1 y modificando tanto los **pasos** como la **dimensión**.

Estos fueron algunos de los tiempos (en segundos) obtenidos

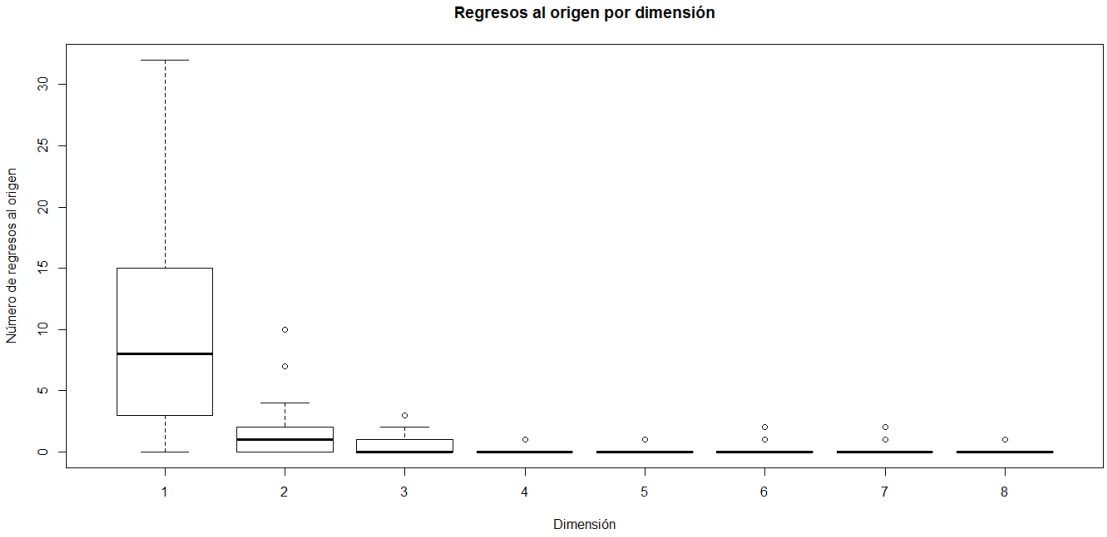

Figure 1: 100 repeticiones con 200 pasos

pasos	dimensión	tiempo 1	tiempo 2	tiempo 3	tiempo 4	tiempo 5
200	8	1.101715	1.149365	1.067676	0.9936671	0.9606109
2,000	8	1.326543	1.150309	1.209556	1.229474	1.222056
20,000	8	2.899392	2.845332	2.769233	2.763092	2.789775
200	80	2.238991	2.3301340	2.16574	2.516627	2.223787
200	800	16.41558	16.78842	15.61014	15.76165	15.43773

Puede asegurarse que mientras mayor sea la dimensión en la que la partícula se mueva, mayor será el tiempo de ejecución del programa.

4 Implementación paralela

Para comparar los tiempos de ejecución del programa entre una implementación paralela y otra que no lo aproveche, se modificará una vez más el código empleando la función **sapply** y eliminando los **clusters**.

Se corrieron ambos programas (el original y el modificado para esta sección) y se registraron los siguientes tiempos de ejecución

repetir	pasos	dimensión	tiempo en paralelo	tiempo sin paralelo
100	200	50	8.20787 seg.	12.285 seg.
200	400	65	35.51587 seg.	1.010146 mins.
500	1,000	80	1.457801 mins.	2.784404 mins.

Resulta evidente que el aprovechamiento del paralelismo es sumamente útil para ahorrar tiempo al momento de ejecutar programas que efectúan múltiples operaciones a la vez.