Computación, simulación y tratamiento estadístico de un Camino Aleatorio en una dimensión

Víctor Mira Ramírez[1], Vera Juan Moll[1]
* [1] *Departmento de Física Aplicada – Facultad de Ciencias, Universidad de Alicante (UA), Alicante, España.*

15 Noviembre de 2024

RESUMEN
En esta práctica se simula un camino aleatorio unidimensional para una partícula, la cual puede saltar a la izquierda o derecha con probabilidades definidas. Se analiza la distribución de la posición final tras N pasos y I iteraciones, generando el histograma normalizado, la media y la desviación estándar de dicha distribución. Se explora cómo varían estos resultados al modificar N e I, comparándolos con predicciones teóricas. Finalmente, se estudian los límites en los que la distribución se aproxima a las formas de Poisson y Gaussiana.

Key words: Camino aleatorio – Distribución de Poisson – Distribución Gaussiana – Mecánica Estadística – Histograma

1 SCRIPT UTILIZADO

Para la realización de la práctica realizamos un programa en Python que, como ya hemos comentado, representa un camino aleatorio en una dimensión con pasos a izquierda y derecha, los cuales no tendrán necesariamente la misma longitud, y las probabilidades de que estos tengan lugar. Este programa nos devolverá un histograma normalizado que describa la probabilidad de que la posición final de la partícula sea x.

Definimos una funición `random_walk()` que nos devolverá un *array* con las posiciones finales de cada *random walk*, así como el número de veces que saltamos a la derecha en cada iteración.

```
def random_walk(I, N, P_r, L, R) -> tuple:
    x_f = []
    dcha = []
    for i in range(I):
        print(f'Progreso: {int(i/I*100)}%',end='\r')
        posicion, contador = 0, 0
        for _ in range(N):
            if np.random.rand() < P_r:
                posicion += R
                contador += 1
            else:
                posicion -= L
        x_f.append(posicion)
        dcha.append(contador)
    return x_f, dcha
```

Comentar que en la práctica se sugirió usar la función `numpy.random.choice()` pero tras implementarlo () comprobamos que era notoriamente más lenta a hacerlo manualmente.

```
for _ in range(N):
   posicion+=np.random.choice([-L, R],p=[1-P_r, P_r])
```

Las funciones gaussianas y de poisson que se verán en las gráficas vienen dadas por la teoría y fueron incluidas en el script de esta forma:

```
def gaussiana(x, media, desviacion) -> float:
    num = exp(-(x-media)**2/(2*desviacion**2))
    den = sqrt(2*pi*desviacion**2)
    return  num / den

def poisson(x, N, P_r) -> float:
    num = exp(-N*P_r) * (N*P_r)**(x)
    den = scipy.special.factorial(x)
    return  num / den
```

2 ANÁLISIS DE LOS RESULTADOS

A continuación, ejecutamos el programa con valores de $Pr = 0{,}2$ (probabilidad de salto a la derecha), $L = 3$ (tamaño del paso izquierdo) y $R = 1$ (tamaño del paso derecho) para distintos valores de N (número de saltos) e I (número de iteraciones), obteniendo las imágenes de la siguiente página (Figuras 1, 2, 3).

Podemos observar como, a mayor número de iteraciones obtenemos una mejor aproximación a la Gaussiana y a Poisson. A medida que vamos rebajando el número de pasos e iteraciones, los resultados se van volviendo más imprecisos debido a la anchura de cada barra de la distribución, pero siguen siendo fieles a ambos tipos de distribuciones, con el pico del histograma coincidiente con el pico de las distribuciones Gaussiana y de Poisson.

Por otro lado, podemos observar en cada imagen los valores de la media y de la desviación estándar obtenidos en cada caso. De la media obtenemos el valor del centro de la distribución, aquel punto alrededor del cual se agrupan los datos. Con este valor podemos observar la tendencia central y comparar distintas distribuciones. En nuestro caso vemos que a medida que vamos disminuyendo el número de pasos e iteraciones las distribuciones van desplazándose hacia la derecha.

* Jose Antonio Pons Botella, Alberto Guilabert Martínez, Antonio Gómez Bañón, Universidad de Alicante (UA), Departamento de Física Aplicada, Spain

A su vez, vemos que las desviaciones también van decreciendo, es decir, a medida que N e I son más pequeños, la dispersión de los datos alrededor del valor medio es menor, debido principalmente a tener menor número de iteraciones y, por tanto, menos datos a representar.

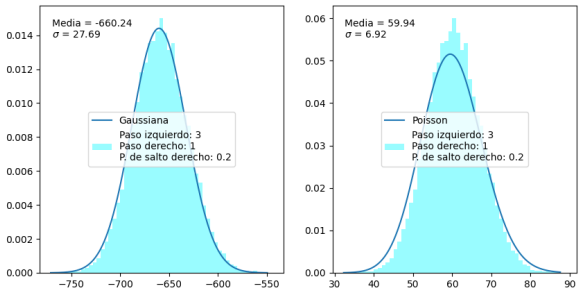

Figura 1: Random Walk para I=30000 iteraciones y N=300 saltos por iteración

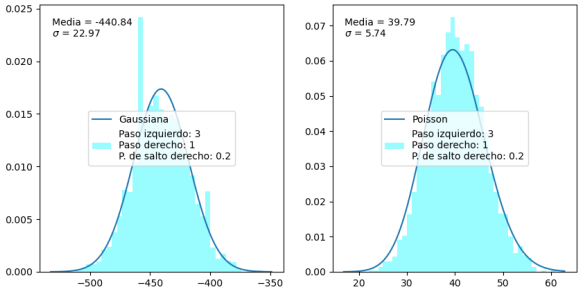

Figura 2: Random Walk para I=3000 iteraciones y N=200 saltos por iteración

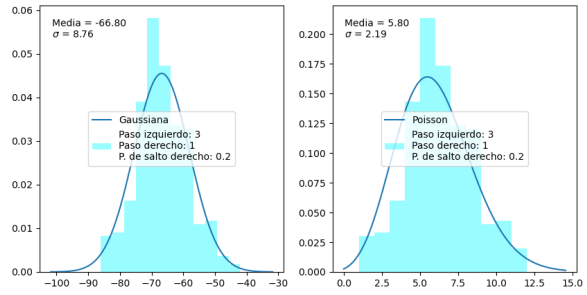

Figura 3: Random Walk para I=300 iteraciones y N=30 saltos por iteración

A lo largo de la práctica siempre nos aseguramos de que los histogramas estuvieran normalizados, y de ellos obtuvimos un valor éxperimentalpára la media y desviación típica de las distribuciones. Después las comparamos con las teóricas para asegurarnos de que se correspondían. Para ello utilizamos las siguientes fórmulas adaptadas para tamaños de saltos no necesariamente iguales (L y R).

La fórmula teórica obtenida para la media coincide siempre con el valor esperado independientemente de nuestras variables, pero nos dimos cuenta de que la desviación típica sólo era precisa cuando $L = R$, desviándose notoriamente, valga la redundancia, del valor experimental.

```
media_teorica      = N * (P_r * R - (1 - P_r) * L)
desviacion_teorica = sqrt(N*P_r*R**2+N*(1-P_r)*L**2)

media      = np.mean(walk)
desviacion = np.std(walk)
```

Comentar también que la distribuicón de *Poisson* aparece cuando la probabilidad de salto a la derecha (P_r) es relativamente baja, y que al aumentarla se desvía notoriamente como podemos ver en la figura 4.

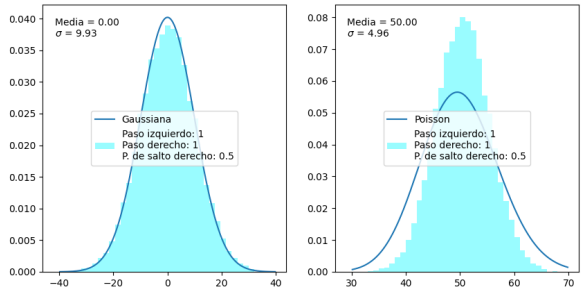

Figura 4: Random Walk para I=3000 iteraciones y N=100 saltos por iteración

En esta figura $L = R = 1$, obteniendo para la gaussiana:

```
Media: 0.0594
Desviación: 9.916871397102348

Media teórica: 0.0
Desviación teórica: 10.0
```

Pero sin embargo si imponemos $L = 3$ y $R = 1$ como venimos haciendo en el resto de la práctica, sin variar nada más, obtenemos:

```
Media: -99.9684
Desviación: 20.09395435050055

Media teórica: -100.0
Desviación teórica: 22.360679774997898
```

Lo cual nos sugiere más claramente esa necesidad de una baja probabilidad de salto a la derecha para que la distribución de *Poisson* aparezca.

LINKS

Código que genera este documento: randomwalk.tex
Script de `python`: randomwalk.py

BIBLIOGRAFÍA

Pons Botella, J. *Apuntes de Mecánica Estadística*, Universidad de Alicante, Departamento de Física Aplicada, 2024.