# Actividad 6 - Mesa de Pool

Exactas Programa

Verano 2023

## Mesa de Pool

El objetivo de esta actividad es modelar la dinámica de las bolas en una mesa de pool.
**Deben subir la resolución a:**`https://bit.ly/entregas-v2023` .

### Una sola bola

Por simplicidad, empezaremos **modelando una única bola**. Luego, lo haremos con muchas en simultáneo, pero sin interacción entre sí.

1. Implementar la función `dar_pasito(x,y,vx,vy,dt)`: recibe las coordenadas $x$ e $y$, la velocidad en $x$ e $y$ y el paso temporal $dt$ para avanzar; devuelve una tupla con las coordenadas de la partícula después de avanzar $dt$ tiempo, según las ecuaciones (1) y (2):

$$x(t + dt) = x(t) + v_x * dt \tag{1}$$

$$y(t + dt) = y(t) + v_y * dt \tag{2}$$

2. Implementar una función `simular_bola_ocho(x, y, vx,vy,dt, L, n_pasos)` que llama sucesivamente a la función `dar_pasito` `n_pasos` veces y que devuelva dos listas: `posiciones_x`, y `posiciones_y`. Note que el primer valor de estas listas debe ser el de las condiciones iniciales. Además, toma como parámetro a `L`, la mitad del largo de nuestra mesa de pool cuadrada (la mesa va de `-L` a `L`). Pruebe esta función con `n_pasos`=3 y los valores iniciales dados en el pie de la Figura 1.

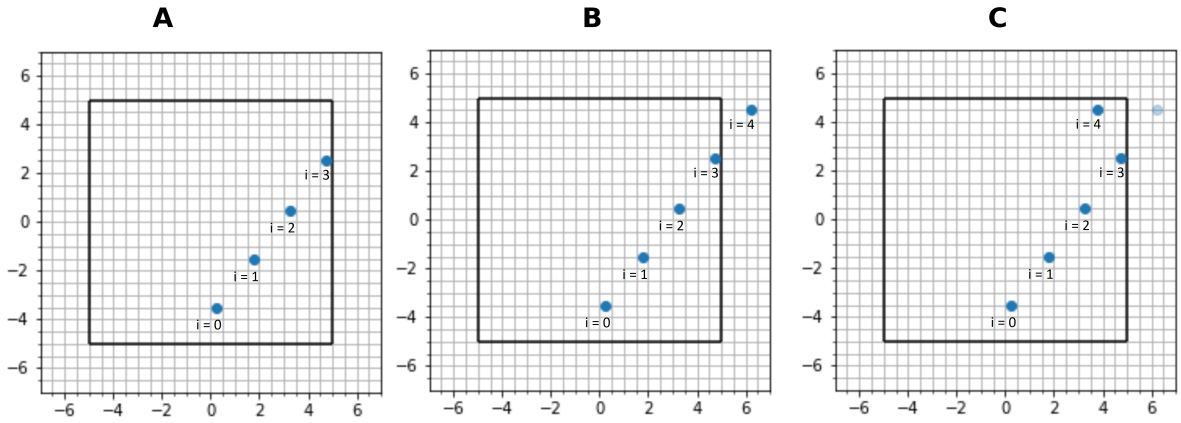


Figura 1: Trayectoria de una partícula (o la bola ocho) inicialmente ubicada en `x=0.25`, `y=-3.5`, con una velocidad `vx=1.5`, `vy=2.0` y un paso temporal de `dt=1.0`, en una mesa de pool de tamaño [-5,5] x [-5,5] (`L=5`).

3. Graficar los valores obtenidos completando el siguiente código que utiliza la librería **matplotlib**:

```
import matplotlib.pyplot as plt
import numpy as np

#---
#Aca van las funciones
#---

fig, ax = plt.subplots()    #Creamos una figura y un eje de matplotlib
ax.plot( ... , ... , 'o')   #Grafica la pelotita (COMPLETAR)

ax.set_aspect("equal")  #Hace que la escala de los ejes sea la misma

#Graficamos los bordes de la mesa: vlines y hlines dibujan rectas verticales
# y horizontales respectivamente:
ax.vlines(-5, ymin=-5, ymax=5, color="black")   #Izquierda
ax.vlines(5, ymin=-5, ymax=5, color="black")    #Derecha
ax.hlines(-5, xmin=-5, xmax=5, color="black")   #Abajo
ax.hlines(5, xmin=-5, xmax=5, color="black")    #Arriba

#Seteamos ticks (las lineas en los numeros de la posicion)
#Largos cada dos metros:
ax.set_xticks(np.arange(-6, 7, 2))
ax.set_yticks(np.arange(-6, 7, 2))
#Cortos (por eso el minor=True) cada 0.5 metros:
ax.set_xticks(np.arange(-7, 7, 0.5), minor=True)
ax.set_yticks(np.arange(-7, 7, 0.5), minor=True)

ax.grid(which="both") #Dibujamos la grilla
ax.set_xlim([-7, 7]) #Seteamos limites del grafico en x
ax.set_ylim([-7, 7]) #idem en y
plt.show()
```

El resultado debería ser igual al de la primera imagen de la Figura (1).

4. Probar la función `simular_bola_ocho` de nuevo, con `n_pasos`=4 y volver a graficar. Notar que el último punto se fue del rango de la caja (Fig1B). Implementar la función `rebotar_der(x,vx,x_max)`, que toma una coordenada $x$, una velocidad $v_x$ y y el borde derecho de la caja $x\_max$. Devuelve la

coordenada y la velocidad corregidas a través de las ecuaciones (3) y (4) respectivamente

$$x' = x - 2 \times (x - x_{max}) \tag{3}$$

$$v_x = -v_x \tag{4}$$

si la coordenada $x$ excede el valor $x_$max; caso contrario, devuelve $x$ y $v_x$, sin modificación alguna.

5. Modificar la función `simular_bola_ocho` para que invoque a `rebotar_der`. Considerar de nuevo `n_pasos`=4 *pasitos*. Graficar la trayectoria y comparar con Figura 1C.

6. Implementar las funciones `rebotar_izq(x,vx,x_min)`, `rebotar_arriba(y,vy,y_max)` y `rebotar_abajo(y,vy,y_min)` que, análogamente a `rebotar_der`, toman como primer parámetro una coordenada, como segundo una velocidad, como tercero el borde correspondiente y corrigen los dos primeros si corresponde.

7. Modificar la función `simular_bola_ocho` para que invoque a las cuatro funciones de rebotes después de dar cada pasito. Reproducir la Figura 2, utilizando los parámeteros iniciales dados en el pie. Para eso, también borrar la `'o'` en la línea `ax.plot(... 'o')`.

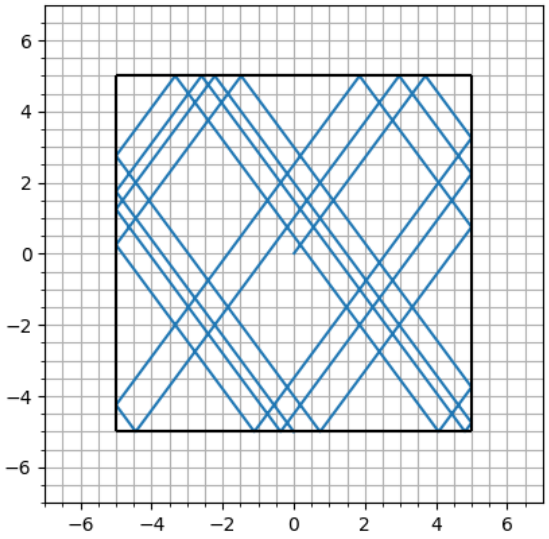


Figura 2: Gráfico de la trayectoria larga con los siguientes parámetros iniciales: `x=0`,`y=0`,`vx=1.0`,`vy=1.35`, `dt=0.01`, `L=5`, `n_pasos=10000`

8. Implementar una función `pelicula_bola_ocho` que tome los mismos parámetros que `simular_bola_ocho`, pero que, en lugar de almacenar las posiciones en x e y en dos listas, vaya imprimiendo las posiciones en un archivo (pueden incluir el nombre del archivo como parámetro adicional).

   Recordar la siguiente sintaxis en python para escribir una linea a un archivo :

```
archivo=open(nombre_archivo + ".txt","w")   #Abre o crea el archivo, segun corresponda
print('lo que quieras escribir',file=archivo)   #Escribe en el archivo
archivo.close() #Cuando terminamos de escribir, usamos esta linea para cerrar el archivo
```

   Recordar que el formato necesario para poder ver la pelicula con `pelicula.py` es el siguiente:

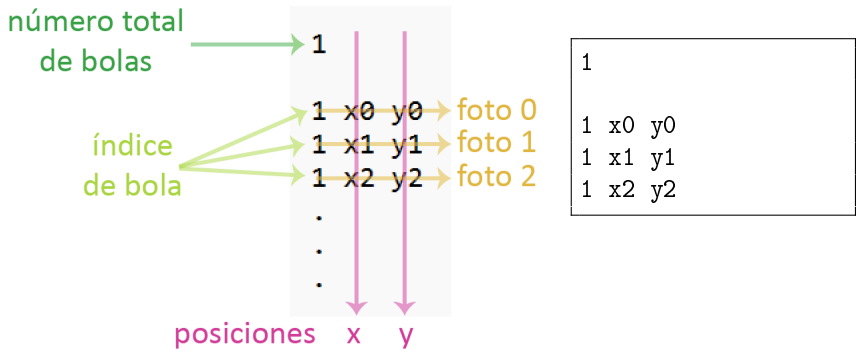


Figura 3: Formato de archivo para pelicula.py con una sola bola

9. Genere el archivo `trayectoria_larga.txt` (usando las mismas condiciones iniciales que en el punto 7) y visualizar la película utilizando el programa `pelicula.py`.

## Muchas bolas (sin interacción)

Pasemos ahora a **modelar muchas bolas** que chocan con las paredes de una mesa de pool (las bolas **no interactúan** entre sí).

Donde antes teníamos un número guardado en la variable `x` para la posición de la bola ocho en $x$, ahora tendremos una lista `x`, donde `x[i]` es la posición en $x$ de la *i-ésima* bola. Lo mismo vale para `y`, y las velocidades `vx` y `vy`, que también de acá en adelante son listas.

11. Crear una función `cond_ini(L,v_max,n_bolas)` que reciba el tamaño de la mesa $L$, la velocidad máxima (en valor absoluto) `v_max` y el número `n_bolas` de bolas; y que devuelva una tupla de cuatro listas: posiciones en x, posiciones en y, velocidades en x, y velocidades en y, para las `n_bolas` bolas.

    A cada una de las bolas le asignaremos *al azar* una posición en la mesa $[-L, L]\times[-L, L]$. Asignaremos también *al azar* velocidades iniciales entre $-V_{\max}$ y $V_{\max}$.

    Podemos usar el comando `(random.random()*2-1)*a` para generar un número *al azar, de manera pareja* entre $-a$ y $a$.

12. Crear una función `pelicula_muchas` que reciba los parámetros necesarios y realice la simulación de `n_bolas` bolas, y que guarde la trayectoria en un archivo .txt, como hicimos en la función `pelicula_bola_ocho`. El formato es similar al visto en la Figura 3, solo que ahora hay que modificar el número total de bolas y agregar las fotos de cada una. Por ejemplo, si tenemos 3 bolas con 2 fotos de cada una, quedaría así:

```
3

1 0 0
2 -4 -4
3 4.4 4
1 1 1
2 -3 -3
3 4 4
```

    Probar la función con los parámetros `L=5, v_max=2, n_bolas=10, dt=0.1` y `n_pasos=200` y visualizar con `pelicula.py`.