Práctica tema 3: Cálculo de la velocidad inicial para un *home run*

Jaime Bruno Gómez

Universidad Autónoma de Madrid. Computación Avanzada

Resumen

En este estudio se ha empleado el método de Euler para calcular la trayectoria de una pelota de béisbol y determinar la velocidad mínima necesaria para lograr un *home run*, considerando efectos como la resistencia del aire y el efecto Magnus. Se comparan resultados en condiciones ideales y realistas, utilizando modelos atmosféricos para evaluar la influencia de la altura en el alcance de la pelota. Los resultados confirman la validez del modelo numérico y muestran cómo la aerodinámica influye en la trayectoria, proporcionando una representación precisa del fenómeno físico.

1 Introducción y fundamento teórico

El estudio de la trayectoria de un proyectil es un problema fundamental en la física clásica. Si se consideran únicamente los efectos de la gravedad, el movimiento se describe mediante ecuaciones diferenciales de segundo orden cuya solución analítica es conocida. Sin embargo, en condiciones más realistas, es necesario incluir la resistencia del aire, lo que complica el sistema y requiere métodos numéricos para su resolución.

Uno de los métodos numéricos más sencillos y utilizados en la integración de ecuaciones diferenciales es el **método de Euler**. En este trabajo, se emplea dicho método para calcular la trayectoria de una pelota de béisbol en presencia y ausencia de resistencia del aire.

1.1. Método de Euler

El método de Euler es una técnica de integración numérica para resolver ecuaciones diferenciales ordinarias de la forma:

$$\frac{dy}{dt} = f(y, t) \tag{1.1}$$

Dado un valor inicial $y(t_0) = y_0$, el método de Euler aproxima el valor de y en un tiempo posterior mediante:

$$y_{n+1} = y_n + \Delta t f(y_n, t_n) \tag{1.2}$$

Donde Δt es el tamaño del paso temporal. Este método proporciona una buena aproximación para pequeños intervalos de tiempo y permite modelar sistemas dinámicos de manera iterativa.

1.2. Aplicación a la Trayectoria de un Proyectil

En esta práctica, se aplicará el método de Euler para hallar la trayectoria de una pelota de béisbol. Para un proyectil en movimiento bajo la acción exclusiva de la gravedad, las ecuaciones de movimiento son:

$$\frac{dx}{dt} = v_x, \quad \frac{dy}{dt} = v_y, \quad \frac{dz}{dt} = v_z \tag{1.3}$$

$$\frac{dv_x}{dt} = 0, \quad \frac{dv_y}{dt} = 0, \quad \frac{dv_z}{dt} = -g \tag{1.4}$$

Sin embargo, si se considera la resistencia del aire, esta se modela como una fuerza proporcional a la velocidad, con coeficiente de arrastre B_2. La ecuación de movimiento en presencia de resistencia se expresa como:

$$\frac{d\vec{v}}{dt} = -g\hat{k} - \frac{B_2|\vec{v}|\vec{v}}{m} \tag{1.5}$$

Donde $\vec{v} = (v_x, v_y, v_z)$. En el caso de una pelota de béisbol, el coeficiente B_2/m se puede aproximar como:

$$\frac{B_2}{m} = 0{,}0039 + \frac{0{,}0058}{1 + \exp\left(\frac{v - v_d}{\Delta}\right)} \tag{1.6}$$

donde $v_d = 35m/s$ y $\Delta = 5m/s$. Al aplicar el método de Euler, la discretización de las velocidades se queda entonces como:

$$\begin{array}{rcl} v_{x_{n+1}} & = & v_{x_n} - \Delta t \cdot B_2|\vec{v_n}|v_{x_n}, \\ v_{y_{n+1}} & = & v_{y_n} - \Delta t \cdot B_2|\vec{v_n}|v_{y_n}, \\ v_{z_{n+1}} & = & v_{z_n} - \Delta t \cdot (g + B_2|\vec{v_n}|v_{z_n}) \end{array} \tag{1.7}$$

Este conjunto de ecuaciones permite modelar la trayectoria realista de un proyectil en un medio con resistencia del aire, proporcionando una representación más precisa del fenómeno físico.

1.3. Modelos de Atmósfera: Isotérmica y Adiabática

Para una representación más realista de la resistencia del aire, es importante considerar la variación de la densidad del aire con la altitud. Existen dos modelos comunes para describir esta variación:

- **Modelo Adiabático**: En este modelo, la temperatura varia con la altitud según una relación polinómica, lo que resulta en una ecuación mas precisa para la densidad:

$$\rho(h) = \rho_0 \left(1 - \frac{az}{T_0}\right)^{\alpha} \tag{1.8}$$

  donde $a = 6{,}5 \cdot 10^{-3} K/m$ es una constante, T_0 la temperatura a nivel del mar y el exponente $\alpha \approx 2{,}5$ para el aire.

- **Modelo Isotérmico**: En este modelo, se asume que la temperatura del aire es constante con la altitud, lo que lleva a una variación exponencial de la densidad según la ecuación:

$$\rho(h) = \rho_0 e^{-z/H} \tag{1.9}$$

  donde $H = 10^4 m$ es la altura de escala térmica y ρ_0 es la densidad a nivel del mar.

Para implementar esta variación de la densidad de la atmósfera con la altura, simplemente habrá que cambiar en las ecuaciones de movimiento el factor B_2 por $B_2\rho/\rho_0$

1.4. Efecto Magnus en la Pelota de Beisbol

El efecto Magnus es un fenómeno aerodinámico que ocurre cuando un objeto en rotación se desplaza a través de un fluido, lo que produce una fuerza lateral perpendicular a la dirección del movimiento.

Así, la fuerza de Magnus se implementaria en nuestras ecuaciones de la siguiente forma:

$$\frac{d\vec{v}}{dt} = \frac{S_0\vec{\omega} \times \vec{v}}{m} \tag{1.10}$$

Donde S_0 es el coeficiente de Magnus ($S_0/m = 4{,}1 \cdot 10^{-4}$ para pelota de béisbol), $\vec{\omega}$ es el vector de velocidad angular de la pelota y $\vec{v}$ es su velocidad lineal.

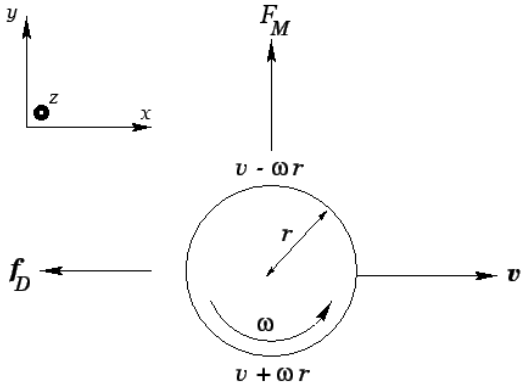

Figura 1: Representación de la fuerza de Magnus en una pelota

Como se observa en la figura 1, la fuerza de Magnus aparece perpendicular tanto a la dirección de desplazamiento como al eje de rotación, por lo que solo aparecerá en una de las tres ecuaciones de movimiento.

En nuestro caso, en el cual la pelota es una bola de beisbol, será el *pitcher* (lanzador) el que decida el eje de rotación de la pelota, y por tanto, la dirección de la fuerza de Magnus.

2 Caso experimental

El problema consiste en hallar la trayectoria de una pelota de béisbol, y calcular la velocidad mínima necesaria v_0 a la cual se debe lanzar para alcanzar el *home run*, considerándose *home run* una vez la pelota ha alcanzado los 168 metros. A lo largo del programa, se considera la dirección de desplazamiento en el eje x y la dirección en la cual actua la gravedad el eje z.

Tenemos 5 objetivos principales en esta práctica:

1. Calcular la velocidad inicial necesaria para hacer *home run*, en el caso ideal, para comprobar que nuestro programa funciona

2. Calcular esta misma velocidad junto con su ángulo óptimo, para el caso con rozamiento

3. Ser capaces de dibujar la trayectoria de nuestra pelota, teniendo en cuenta rozamiento y efecto Magnus

4. Observar como varía el alcance de la pelota en función de la altura a la cual se encuentra nuestro campo

5. Analizar distintos tipos de lanzamientos del *pitcher*, debido al efecto Magnus

2.1. Código del programa

Comenzaremos implementando las librerías necesarias y definiendo constantes:

```
from math import cos,sin,sqrt,pi
import numpy as np
import matplotlib.pyplot as plt

#Definimos las constantes necesarias

g=9.8                       #Gravedad
x0, y0, z0= 0., 0., 0.      #Punto inicial
t0=0                        #Tiempo inicial
dt=0.01                     #Paso de tiempo
rho_0=1.225                 #Densidad del aire al nivel del mar
a=6.5*10**(-3)              #Constante de temperatura
T0=288.15                   #Temperatura al nivel del mar
alpha=2.5                   #Exponente de temperatura
vd, D = 35, 5               #Constantes para el coeficiente de arrastre
HR=168                      #Distancia necesaria para hacer home run
z02=3715                    #Altura a la que se encuentra el campo de beisbol 2
z03=12500                   #Altura a la que se encuentra el campo de beisbol 3
z_0=10000                   #Constante de altura en la atmósfera isotérmica
S0m=4.1*10**(-4)            #Coeficiente de Magnus
w=314.16                    #Velocidad angular de la bola de beisbol (3000RPM)
```

Figura 2: Definición de librerías y constantes

Es importante comentar que en el único caso donde tenemos soluciones analíticas simples es aquel en el que no tenemos rozamiento con el aire. Por ello, podemos usar este caso para ver si nuestro código funciona correctamente, antes de pasar al caso con rozamiento.

Para el caso numérico, lo calculamos mediante un bucle con listas y la función *append*, donde podemos observar las ecuaciones de movimiento discretizadas. Para el caso analítico, usamos las ecuaciones de un movimiento parabólico (ver figura 3).

Para compararlas, simplemene usaremos la función *print* (ver figura 4).

```python
#Paramos el bucle cuando la pelota haga HR:
while alcance<HR:
    #Variamos v0 para hallar la mínima
    v0+=1
    v0x=v0*(np.cos(np.radians(theta)))
    v0z=v0*(np.sin(np.radians(theta)))
    #Listas para la trayectoria de la bola de béisbol
    t,x,z,vx,vz=[t0],[x0],[z0],[v0x],[v0z]

    i=0
    #Paramos cuando la pelota llegue al suelo
    while z[i]>=0:
        i+=1
        t.append(t[i-1]+dt)
        x.append(x[i-1]+vx[i-1]*dt)
        z.append(z[i-1]+vz[i-1]*dt)

        vx.append(vx[i-1])
        vz.append(vz[i-1]-g*dt)

    alcance=(x[-1]+(x[-2]-x[-1])/2)

v0_num=v0

#Ahora, lo calculamos analíticamente para ver si nuestro programa funciona

v0_an=np.sqrt((168*g)/(np.sin(2*np.radians(theta))))
```

Figura 3: Cálculo numérico y analítico de la velocidad mínima para el caso ideal

```python
#Exponemos los resultados
print("En el caso ideal a 45º la velocidad inicial calculada es:")
print(f"{v0} m/s mediante métodos numéricos")
print(f"{v0_an:.{2}f} m/s mediante métodos analíticos\n")
```

Figura 4: Print para comparar si ambos resultados son correctos

Una vez visto que funciona como debería, pasamos a calcular el ángulo al cual la velocidad necesaria para hacer *home run* se hace mínima.

Este proceso requiere de varios bucles dentro de otros, por lo que requiere tiempo a la hora de calculase. Por esta razón, solo se ha calculado una vez guardándose sus valores, y en el código final aparece comentado. El código es igual al caso anterior, solo que se encuentra dentro del siguiente bucle para variar el ángulo de lanzamiento:

```python
rangetheta=np.arange(30,41,1)        # Distintos ángulos de lanzamiento
for theta in rangetheta:
```

Figura 5: Bucle con distintos ángulos a comparar para el lanzamiento

Y las ecuaciones de movimiento discretizadas con rozamiento y el modelo de atmósfera adiabática:

```python
while z[i]>=0:
    i+=1
    t.append(t[i-1]+dt)
    x.append(x[i-1]+vx[i-1]*dt)
    z.append(z[i-1]+vz[i-1]*dt)

# Nos encontramos en el modelo de atmósfera adiabática
    B2m=0.0039+0.0058/(1+np.exp((v[i-1]-vd)/D))
    rho= rho_0*(1-(a*z[i])/T0)**alpha
    B2m_=B2m*rho/rho_0

    vx.append(vx[i-1]-B2m_*v[i-1]*vx[i-1]*dt)
    vz.append(vz[i-1]+(-B2m_*v[i-1]*vz[i-1]-g)*dt)
    v.append(np.sqrt(vx[i]**2+vz[i]**2))
```

Figura 6: Ecuaciones de movimiento con rozamiento

Con la función *min* encontramos el menor valor de velocidad incial calculada, y lo mostramos en pantalla con *print*.

A continuación, pasaremos a calcular y dibujar la trayectoria de la pelota de béisbol, lo que resulta más visual hacerlo en gráficos 3D, sobre todo cuando introduzcamos el efecto Magnus.

Usaremos la velocidad mínima y el ángulo optimo calculados, junto con otros ángulos para ver, efectivamente, que se trata del ángulo óptimo. El bucle para calcular la trayectoria es igual que el de la figura 5.

Para dibujar en 3D:

```python
fig=plt.figure(1)                          #Plot sin efecto Magnus
fig_=plt.figure(2)                         #Plot con efecto Magnus

nm=fig.add_subplot(111,projection='3d')
ym=fig_.add_subplot(111,projection='3d')
```

Figura 7: Preparar los ejes en 3D

En el caso con rozamiento pero sin efecto Magnus:

```python
#Dibujo la trayectoria sin Magnus a la altura del mar
nm.plot(x, y, z, label=f"y(t) - {theta[m]}º")
nm.legend()
nm.set_xlabel("Eje x (m)")
nm.set_ylabel("Eje y (m)")
nm.set_zlabel("Eje z (m)")
nm.set_title("Trayectoria de la pelota de beisbol sin efecto Magnus al nivel del mar")
```

Figura 8: Dibujar las trayectorias con rozamiento al nivel del mar

Si queremos introducir el efecto Magnus, deberemos modificar las ecuaciones de movimiento de la siguiente forma:

```
#Usando el método de Euler:
t1.append(t1[i-1]+dt)
x1.append(x1[i-1]+vx1[i-1]*dt)
y1.append(y1[i-1]+vy1[i-1]*dt)
z1.append(z1[i-1]+vz1[i-1]*dt)

#Nos encontramos en el caso de la atmósfera adiabática
B2m=0.0039+0.0058/(1+np.exp((v1[i-1]-vd)/D))
rho= rho_0*(1-(a*z1[i])/T0)**alpha
B2m_=B2m*rho/rho_0

vx1.append(vx1[i-1]-B2m_*v1[i-1]*vx1[i-1]*dt)
vy1.append(vy1[i-1]+(-B2m_*v1[i-1]*vy1[i-1]+S0m*w*vx1[i-1])*dt)
vz1.append(vz1[i-1]+(-B2m_*v1[i-1]*vz1[i-1]-g)*dt)
v1.append(np.sqrt(vx[i]**2+vy[i]**2+vz[i]**2))
```

Figura 9: Ecuaciones de movimiento teniendo en cuenta el efecto Magnus, con el eje de giro en dirección +z

y lo dibujaríamos igual que en la figura 8. Para ver si la pelota efectivamente ha hecho *home run*, creamos una variable llamada *alcance1* que será la media de los dos último puntos (ya que el último punto calculado puede estar "dentro"de la tierra, ver figura 10). Este valor lo comparamos con el valor de *home run* y hacemos un *print* en aquellos que sean mayores (ver figura 11).

```
distance = np.sqrt(x1[-1]**2 + y1[-1]**2)        #Último punto de la trayectoria
distance_1= np.sqrt(x1[-2]**2 + y1[-2]**2)       #Penultimo punto de la trayectoria

#Se calcula el alcance promediando los dos últimos puntos
alcance1.append((distance+distance_1)/2)
```

Figura 10: Calcular el alcance de la pelota

```
#Imprimimos los ángulos con los que se consigue home run
found=False
print("En el caso con rozamiento, sin fuerza de Magnus:")
for m in range(len(theta)):
    if alcance[m]>HR:
        print(f"Con un ángulo de {theta[m]}º se consigue home run\n")
        found=True
if not found:
    print("No se consigue home run para ningún ángulo")

print("En el caso de rozamiento y fuerza de Magnus en el eje y:")
for m in range(len(theta)):
    if alcance1[m]>HR:
        print(f"Con un ángulo de {theta[m]}º se consigue home run\n")
```

Figura 11: Hacer *print* en aquellos que sí hacen *home run*

Para ver el efecto de la altura (densidad atmosférica) en el alcance de la pelota, compararemos tres casos: un lanzamiento a nivel del mar, uno a 3715m y otro a 12500m (sin Magnus para simplificar). Para calcular la trayectoria, se usa el mismo bucle que en la imagen 6, cambiando la altura inicial, y la condición del *while* dependiendo del caso. Se guardará la variable *alcance* (previamente explicado) y se mostrará con *print*.

También hay que comentar que en el caso de 12500m se pasa a usar el modelo de atmósfera isotérmica, al considerar que a estas alturas la temperatura es prácticamente constante, como se ve a continuación:

```
#Tercer caso: sin magnus a 12500 m
i=0
while z3[i]>=12500:
    i+=1
    t3.append(t3[i-1]+dt)
    x3.append(x3[i-1]+vx3[i-1]*dt)
    y3.append(y3[i-1]+vy3[i-1]*dt)
    z3.append(z3[i-1]+vz3[i-1]*dt)

    #Densidad del aire a una altura z (modelo de atmósfera isotérmica)
    B2m=0.0039+0.0058/(1+np.exp((v3[i-1]-vd)/D))
    rho= rho_0*np.exp(-z3[i]/z_0)
    B2m_=B2m*rho/rho_0

    vx3.append(vx3[i-1]-B2m_*v3[i-1]*vx3[i-1]*dt)
    vy3.append(vy3[i-1]-B2m_*v3[i-1]*vy3[i-1]*dt)
    vz3.append(vz3[i-1]+(-B2m_*v3[i-1]*vz3[i-1]-g)*dt)
    v3.append(np.sqrt(vx3[i]**2+vy3[i]**2+vz3[i]**2))
```

Figura 12: Bucle en el caso de 12500m. Usamos el modelo de atmósfera isotérmica

A la hora de hacer el *plot* de las trayectorias, podemos hacer más claro el *home run* dibujando el suelo del campo de béisbol de la siguiente forma:

```
field_radius = 168                          # Radio del campo en metros
theta = np.linspace(-pi/4, pi/4, 100)       # Ángulo de apertura de 90º para el campo

# Pasamos polares a cartesianas
field_x = field_radius * np.cos(theta)
field_y = field_radius * np.sin(theta)
field_z = np.zeros_like(field_x)

# Añadir el punto inicial y final para cerrar el campo
field_x = np.concatenate(([0], field_x, [0]))
field_y = np.concatenate(([0], field_y, [0]))
field_z = np.concatenate(([0], field_z, [0]))

#Dibujamos el campo de beisbol
nm.plot(field_x, field_y, field_z, color='green')
ym.plot(field_x, field_y, field_z, color='green')

# Pintar el suelo del campo de verde
nm.plot_trisurf(field_x, field_y, field_z, color='green', alpha=0.5)
ym.plot_trisurf(field_x, field_y, field_z, color='green', alpha=0.5)
```

Figura 13: *Plot* del campo de béisbol

A continuación, pasaremos a la última parte de nuestro código, que consiste en dibujar distintos tipos de lanzamiento del *pitcher* dependiendo de en que dirección hace girar la pelota, gracias al efecto Magnus:

- **Overhand curveball**: La bola gira hacia adelante (eje de rotación +y), generando una fuerza vertical -z
- **Fastball**: La bola gira de lado (eje de rotación +z o -z), generando una fuerza horizontal
- **Sidearm curveball**: La bola gira hacia atrás (eje de rotación -y), generando una fuerza vertical +z

Introducimos las siguientes constantes iniciales, extraídas de casos reales de internet:

```
x0,y0,z0=0,0,1.5  #Posición de lanzamiento de la bola
distancia=20  # Distancia del pitcher al golpeador
theta0M= 0.5  # Ángulo de lanzamiento
v0M= 90  # Velocidad de lanzamiento
```

Figura 14: Constantes de lanzamiento del *pitcher*

y escribimos los bucles con las ecuaciones de movimiento según el lanzamiento. En el caso, por ejemplo, de la *overhand curveball*:

```
v0x=v0M*(np.cos(np.radians(theta0M)))
v0y=0.
v0z=v0M*(np.sin(np.radians(theta0M)))
t,x,y,z,vx,vy,vz,v=[t0],[x0],[y0],[z0],[v0x],[v0y],[v0z],[v0M]
i=0
while x[i]<distancia:
    i+=1

    #Usando el método de Euler:
    t.append(t[i-1]+dt)
    x.append(x[i-1]+vx[i-1]*dt)
    y.append(y[i-1]+vy[i-1]*dt)
    z.append(z[i-1]+vz[i-1]*dt)

    #Modelo de atmósfera adiabática
    B2m=0.0039+0.0058/(1+np.exp((v[i-1]-vd)/D))
    rho= rho_0*(1-(a*z[i])/T0)**alpha
    B2m_=B2m*rho/rho_0

    vx.append(vx[i-1]-B2m_*v[i-1]*vx[i-1]*dt)
    vy.append(vy[i-1]-B2m_*v[i-1]*vy[i-1]*dt)
    vz.append(vz[i-1]+(-B2m_*v[i-1]*vz[i-1]-g-S0m*w*vx1[i-1])*dt)
    v.append(np.sqrt(vx[i]**2+vy[i]**2+vz[i]**2))
```

Figura 15: Bucle del lanzamiento de *overhand curveball*

Finalmente ploteamos cada trayectoria como en la figura 8. A su vez, podemos introducir una "diana.en el plot que nos facilita ver la *strike zone*, es decir, el lugar donde la pelota tiene que acabar para que el lanzamiento del *pitcher* sea válido. Lo hacemos de la siguiente forma:

```
# Dimensiones aproximadas de la strike zone (en metros)
bajo_strike_zone = 1.0
alto_strike_zone = 1.6
ancho_strike_zone = 0.5

xs = [distancia,distancia,distancia,distancia]
ys = [ancho_strike_zone/2, -ancho_strike_zone/2, ancho_strike_zone/2, -ancho_strike_zone/2]
zs = [bajo_strike_zone, bajo_strike_zone, alto_strike_zone, alto_strike_zone]

# Convertir a una malla 2D (necesario para plot_surface)
X = np.array([xs[:2], xs[2:]])
Y = np.array([ys[:2], ys[2:]])
Z = np.array([zs[:2], zs[2:]])

# Dibuja la superficie de la diana y la pelota
Pplot.plot_surface(X,Y,Z, color='red',edgecolors='black',alpha=0.25)
Pplot.scatter(x0, y0, z0, color='white',edgecolors='black', s=100)
```

Figura 16: *Plot* de la *strike zone*

2.2. Resultados

A continuación, analizaremos tanto los resultados numéricos (expuestos en la terminal con *print*) como las figuras con *plot*.

Comenzamos obteniendo los resultados de las velocidades y ángulos óptimos:

```
En el caso ideal a 45º la velocidad inicial calculada es:
41.0 m/s mediante métodos numéricos
40.58 m/s mediante métodos analíticos

En el caso no ideal, teniendo en cuenta el rozamiento del aire:
El ángulo óptimo de lanzamiento en el caso no ideal es 33º, con una v0 necesaria de 62.05 m/s
```

Figura 17: Valor de velocidades mínimas de lanzamiento para hacer home run

Como vemos, nuestro código esta correcto, ya que en el caso ideal coincide la analítica con la numérica. A su vez, vemos que el lanzamiento en el caso ideal tiene un ángulo óptimo menor y una velocidad mayor, tal y como nos sugiere la literatura [1].

Observando ahora el *plot* de las trayectorias a nivel del mar, tanto con y sin efecto Magnus:

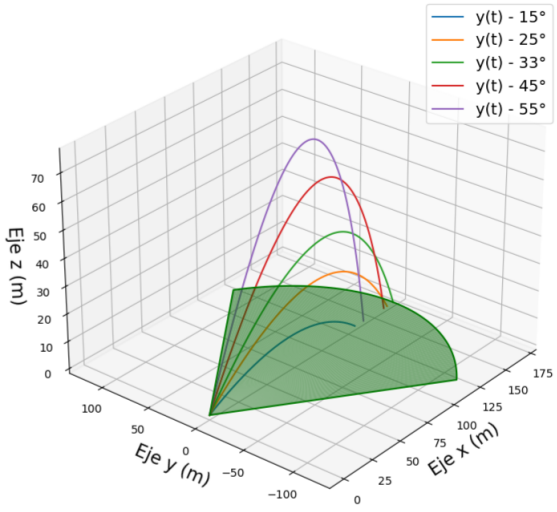

Figura 18: Trayectoria de la pelota sin efecto Magnus

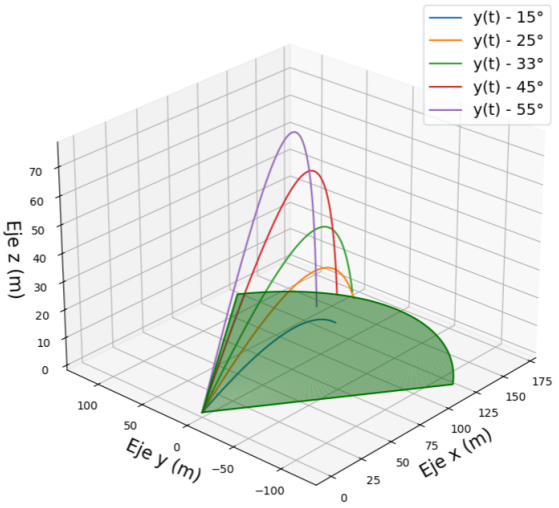

Figura 19: Trayectoria de la pelota con efecto Magnus

Podemos observar como, efectivamente, al introducir el efecto Magnus las trayectorias de la pelota se desvían horizontalmente. Es de esperar que, al introducir el efeceto Magnus, sea más sencillo hacer *home run*, como muestra la terminal:

```
En el caso con rozamiento, sin fuerza de Magnus:
Con un ángulo de 33° se consigue home run

En el caso de rozamiento y fuerza de Magnus en el eje y:
Con un ángulo de 33° se consigue home run
Con un ángulo de 45° se consigue home run
```

Figura 20: Ángulos a los cuales se consigue *home run*

Por otro lado, mostrando el cálculo del alcance para las distintas alturas:

```
Para una velocidad inicial de 62.05 m/s, y un ángulo de 33º:
El alcance sin Magnus a nivel del mar es de 168.00827263038644 m
El alcance con Magnus a nivel del mar es de 174.99561351571435 m
El alcance sin Magnus a 3715 m es de 189.07042817273376 m
El alcance sin Magnus a 12500 m es de 277.62348501387066 m
```

Figura 21: Alcance de la pelota para los distintos casos

También se obtiene un resultado razonable, ya que, a medida que aumentamos la altura, disminuye la densidad atmosférica, y por tanto el rozamiento. Esto hace que el alcance sea mayor. Por último, para el *plot* de los lanzamientos del *pitcher* obtenemos:

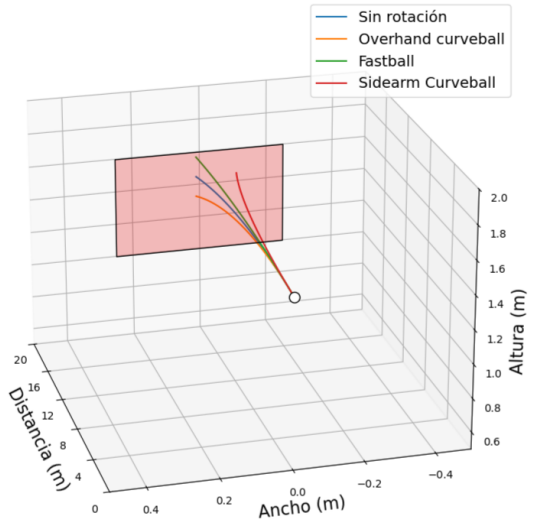

Figura 22: Distintos tipos de lanzamiento del *pitcher*

Observando la figura 22 vemos como la decisión del *pitcher* de hacia donde girar la pelota afecta a la trayectoria. El desplazamiento no es superior a los 10cm, pero apreciable para confundir al golpeador.

3 Conclusiones

El estudio realizado permite validar la implementación del método de Euler para la simulación numérica de el sistemas de una pelota de béisbol, confirmando la concordancia con soluciones analíticas en condiciones ideales. Se ha podido comprender la influencia de diversos factores en la trayectoria de una pelota de béisbol, como la resistencia del aire, la altura o el efecto Magnus.

La realización de este trabajo en el programa *Python* no solo ha reforzado mi conocimiento, si no que muestra la importancia del modelado computacional en la resolución de problemas físicos y su aplicabilidad en el ámbito deportivo.

Referencias

[1] Nicholas J Giordano. *Computational physics*. Pearson Education India, 2012.