Physics 3926 Project 1

Katie Brown

September 29, 2021

1 Introduction

The goal of this project was to simulate projectile motion of baseballs being hit by a Robotic Designated Hitter (RDH). The motion of these balls can be described by a system of simple ordinary differential equations (ODE's) relating their position and velocity:

$$\frac{d\vec{v}}{dt} = -g\hat{y} - \frac{C_d \rho A v}{2m}\vec{v} \tag{1}$$

$$\frac{d\vec{r}}{dt} = \vec{v} \tag{2}$$

This system of ODE's can be easily solved with several different numerical methods: Euler Method, Euler-Cromer Method and Midpoint Method. In each case, the first step of each iteration is to calculate the acceleration (based on the current velocity) using equation 1. Each method then calculates the new position and velocity vectors by adding increments of $\Delta\vec{v} = \tau\vec{a}$ and $\Delta\vec{r} = \tau\vec{v}$ to the existing position and velocity, respectively. The only difference between the three methods is that in calculating $\Delta\vec{r}$, Euler Method uses the existing $\vec{v}$, Euler-Cromer uses the newly computed $\vec{v}$ and Midpoint uses the average of both.

2 Program

2.1 Main Functions

The primary function in this program is called `simulation`, which calculates the trajectory of the baseball for a given launch speed and angle. When calling the function, it is necessary to specify the numerical method, the time-step, and whether to account for air resistance. The Boolean variable ABHRcal is also used to specify whether or not the function is being used to calculate the AB/HR ratio, which controls whether or not the simulation ends at the location of the fence.

```
def simulation(initialSpeed, theta0, tau, solveMethod, airRes, ABHRcal):
```

Listing 1: Definition of the simulation function

This function keeps track of the position, velocity, and acceleration of the ball - all as 2-element arrays corresponding to the x-y vectors. At each step in the simulation, the function calculates the acceleration (using the current velocity), then updates the position and velocity using the specified numerical method.

```
a[0] = - airResCoef * v[0] * np.linalg.norm(v)
a[1] = -g - airResCoef * v[1] * np.linalg.norm(v)

if solveMethod == 'Euler':
    r = r + v * tau
    v = v + a * tau

elif solveMethod == 'Euler-Cromer':
    v = v + a * tau
    r = r + v * tau

elif solveMethod == 'Midpoint':
    v_last = np.copy(v)
    v = v + a * tau
    r = r + tau * (v + v_last) / 2

trajectory.append(r)
```

Listing 2: One iteration of the simulation

The simulation ends (as the loop breaks) when the ball hits the ground (when the y-component of the position is equal to or less than 0) or the ball reaches the location of the fence (if ABHRcal = True). `simulation` returns a list of all the position vectors generated.

Two supplementary functions `getXPos` and `getYPos` were used to break up the list of position vectors generated by `simulation` into lists of all the x-positions and y-positions.

```
def getXPos(initialSpeed, theta0, tau, solveMethod, airRes, ABHRcal):
    xPositions = []
    trajectory = simulation(initialSpeed, theta0, tau, solveMethod, airRes, ABHRcal)
    for i in trajectory:
        xPositions.append(i[0])
    xPosArray = np.array(xPositions)
    return xPosArray
```

Listing 3: The `getXPos` function calls `simulation` and adds the first element of each sub-array to the `xPositions` list.

2.2 Part 1

In order to calculate the horizontal range of the ball, `getXPos` was called and `np.amax` was used to determine the largest x-position. Using Euler's Method with initial conditions of $v_0 = 50$ m/s, $\theta = 45°$ and $\tau = 0.1$ s, the horizontal range was calculated to be 282.84 m (with no air resistance) and 134.43 (with air resistance).

Using the same conditions (and no air resistance), I used `getXPos` and `getYPos` to generate the trajectories with Euler, Euler-Cromer and Midpoint methods, then plotted each using matplotlib (see Figure 1).

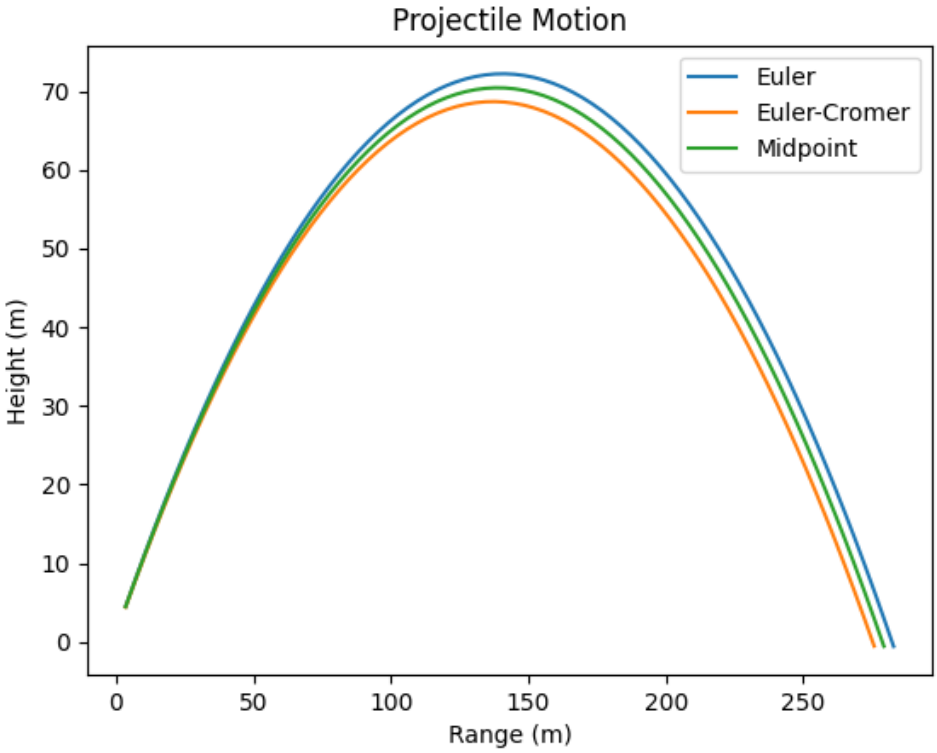

Figure 1: Trajectories calculated with various numerical methods for $v_0 = 50$ m/s, $\theta = 45°$ and $\tau = 0.1$ s.

2.3 Part 2

In order to complete parts 2 and 3, I defined the function `ABHRfun` to calculate the ratio of at-bats to home-runs. For a specified number of iterations, this function calculates the trajectory of the ball (employing `getXPos` and `getYPos`) using randomly generated initial speeds and angles (with $\mu_v = 44.7$ m/s, $\sigma_v = 6.7$ m/s, $\mu_\theta = 45°$ and $\sigma_\theta = 10°$). For each iteration, `ABHRfun` counts a home-run if the ball reaches the location of the fence (at 122 m) and has a vertical position higher than the height of the fence at this point. To eliminate the condition of clearing the fence (as is the

case in Part 2), simply take a fence height of 0 m. Finally, `ABHRfun` calculates the ratio of total iterations to iterations resulting in a home run and returns this value (the AB/HR ratio).

Using the Euler-Cromer method and accounting for air resistance, the computed AB/HR ratio was 4.76.

2.4 Part 3

Firstly, in order to observe how the AB/HR ratio changes according to fence height, I used `ABHRfun` to calculate the AB/HR ratio at fence heights ranging from 0 m to 40 m. This large range of heights was chosen to effectively illustrate how this ratio evolved (it had little variation from 0 to 15 m). The computed values are displayed in figure 2.

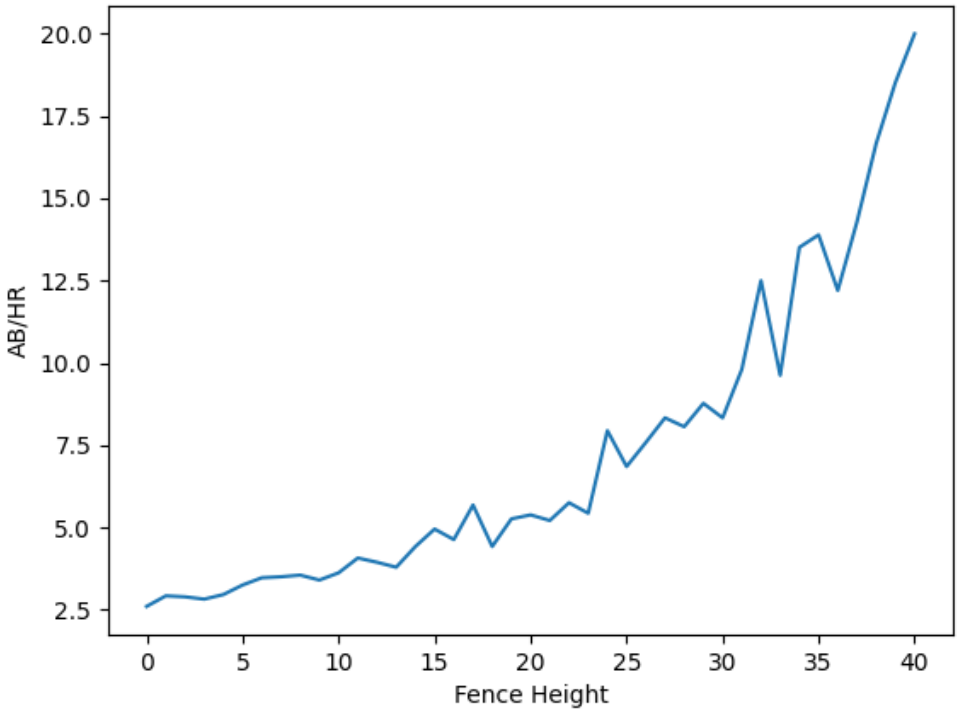

Figure 2: Ratio of at-bats to home-runs for a varying fence height.

Finally, `ABHRfun` was called for fence heights increasing in increments of 0.5 m until the AB/HR ratio reached 10. It was calculated that a 30.5 m tall fence would result in a AB/HR ratio of 10.

3 Conclusion

The purpose of this analysis was to determine the feasibility of replacing human hitters with the RDH. This program successfully modelled the trajectory of the baseballs under various conditions and computed the AB/HR ratio depending on the height of the fence at 400 ft. Using the `ABHRfun` function, it is simple to calculate that at Boston's Fenway Park (with a fence height of 11.3 m), the robot would have an AB/HR ratio of around 4.17. This is much lower than any human hitter, which is likely to cause backlash from fans. If the MLB wants to implement the use of the RDH, they should decrease the AB/HR ratio by reducing the mean initial speed or using a launch angle that produces a smaller range than $45°$. Alternatively, they could increase the standard deviation of v_0 or θ in order to produce more variation (and a less consistent AB/HR ratio).