SEAN HUGGINS PHYSICS 3926 PROJECT 1

Table of Contents

1. Introduction

The goal of this project is to determine an appropriate fence height at a pitch length of 400ft to be installed in all ALMLB stadiums, for when the RDH (Robotic Designated Hitter) is implemented in the league for the 2020/2021 season. To keep fans and players happy, the league reccomendeds that a minimum AB/HR ratio of 10 be maintaned. Our analysis will use iterative ODE solver numerical methods to solve the projectile motion problem, and we will determine an appropriate fence height to be installed by comparing AB/HR ratios (as determined by the solutions of the projectile motion problem), with various fence heights.

2. The baseball() Function

2.0 Description

The first necessary function, dubbed **baseball()** calculates the 2D projectile motion trajectory of a baseball as hit by the RDH (Robotic Designated Hitter), given an initial speed (m/s), angle in degrees from the horizontal axis, a tau time step (s), and a solution method.

Various constants were defined within this function, including the baseball's mass, diameter, cross-sectional area, the gravitational constant g, the density of the air, the drag coefficient Cd, and the length of the baseball pitch all in SI units.

Additionally, **baseball()** may take a variable number of extra arguments, with options to turn on and off plotting, air resistance, and to show or not show the ball's range, and its height when it passes the fence.

2.1 Procedure

To solve the projectile motion problem of the baseball, **baseball()** uses a stepwise ODE solver method. The three methods available for use are the Euler, Euler-Cromer, and Midpoint methods.

Initial conditions are set including a starting velocity based on the inputted launch angle and speed arguments, as well as a starting position at **(x,y) = (0, 1m)**.

These stepwise methods are iterative, so our function implements a while loop. This while loop terminates when the ball hits the ground. The loop algorithm is explained below:

- **Calculate the acceleration of the baseball using its equation of motion**
- **Calculate the ball's new velocity according to the method being used**
- **Calculate the ball's new position according to the method being used**
- **Repeat till the ball hits the ground**

This function finally outputs the range of the ball, and the height of the ball when it passes the length of the pitch (when it would hit, or otherwise fly over the fence), both calculated using pchip interpolation. These results will later allow us to determine whether or not the hit was a home run.

2.2 Testing baseball()

We first tested **baseball()** by recreating Figure 2.3 from p45 in the textbook, using three stepwise ODE solvers, the Euler, Euler-Cromer, and Midpoint methods.

Result from the **Euler** method:

```
figure(1);
baseball(50, 45, 0.1, 'euler', 'x-intercept', 1);
baseball(50, 45, 0.1, 'euler', 'air', 0, 'x-intercept', 1);
legend('Euler Method', 'x-intercepts', 'Theory (No air)');
```

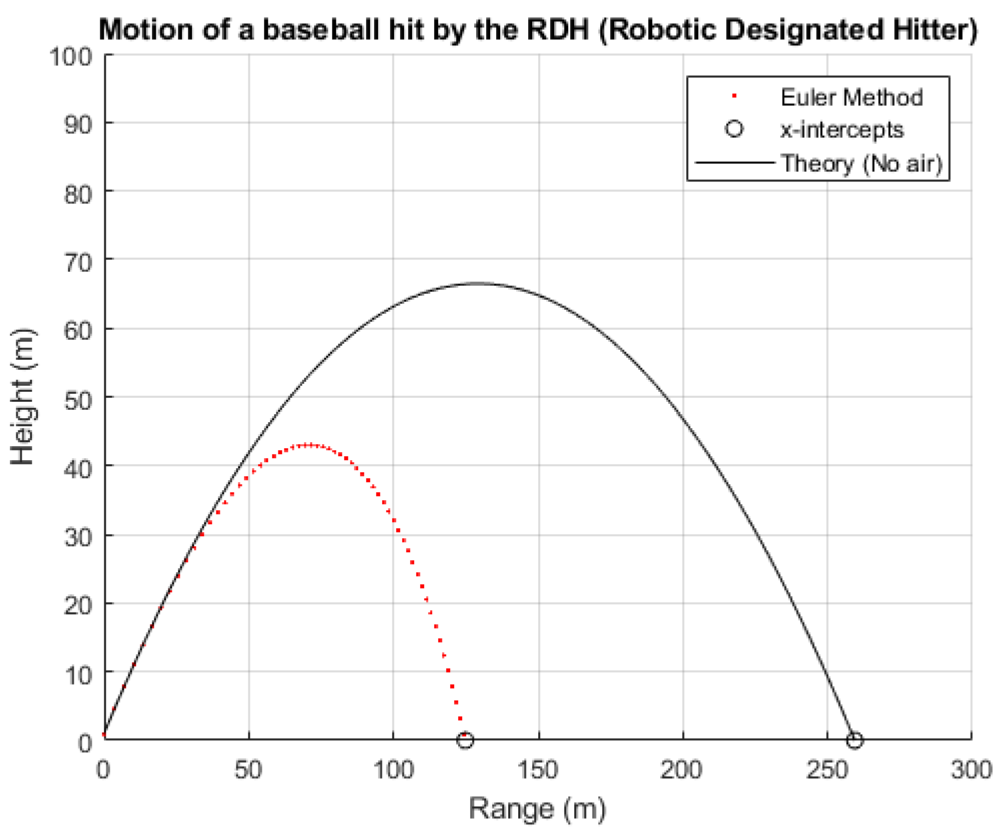

Figure 1: Euler method output from **baseball()** for an initial height of 1 m, initial speed of 50 m/s, angle of 45 degrees, and a time step of 0.1s

Result from the **Euler-Cromer** method:

```
figure(2);
baseball(50, 45, 0.1, 'euler-cromer', 'x-intercept', 1);
baseball(50, 45, 0.1, 'euler-cromer', 'air', 0, 'x-intercept', 1);
legend('Euler-Cromer Method', 'x-intercepts', 'Theory (No air)');
```

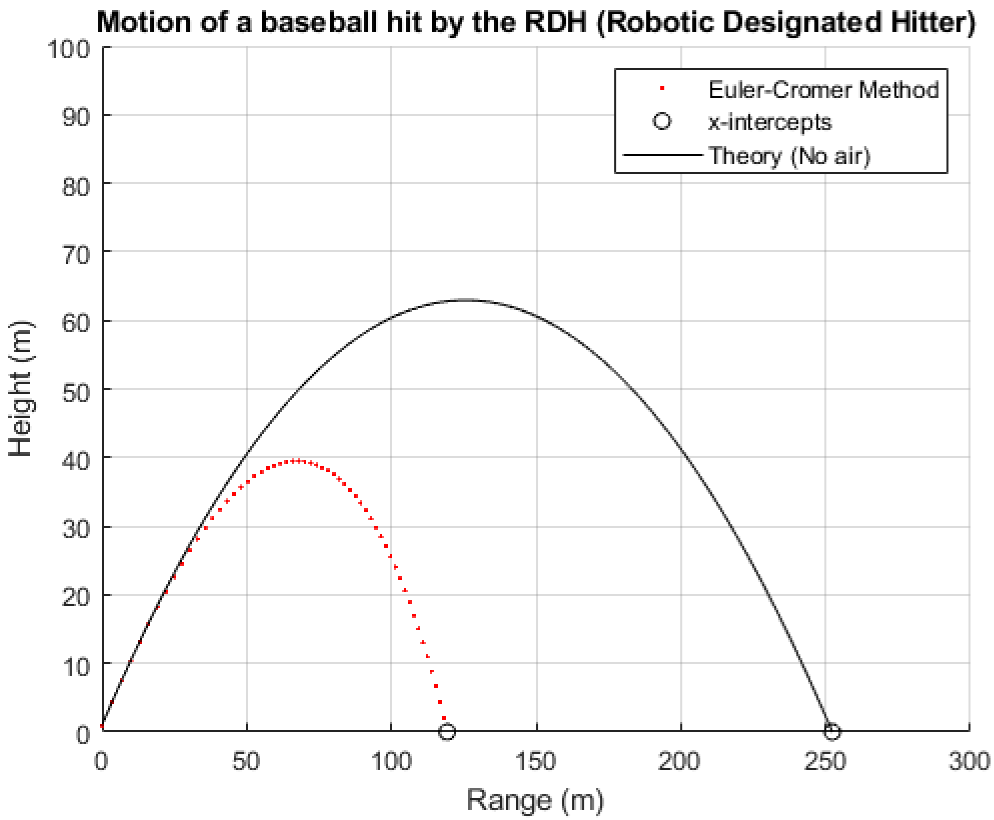

Figure 2: Euler-Cromer method output from **baseball()** for an initial height of 1 m, initial speed of 50 m/s, angle of 45 degrees, and a time step of 0.1s

Result from the **Midpoint** method:

```
figure(3);
baseball(50, 45, 0.1, 'midpoint', 'x-intercept', 1);
baseball(50, 45, 0.1, 'midpoint', 'air', 0, 'x-intercept', 1);
legend('Midpoint Method', 'x-intercepts', 'Theory (No air)');
```

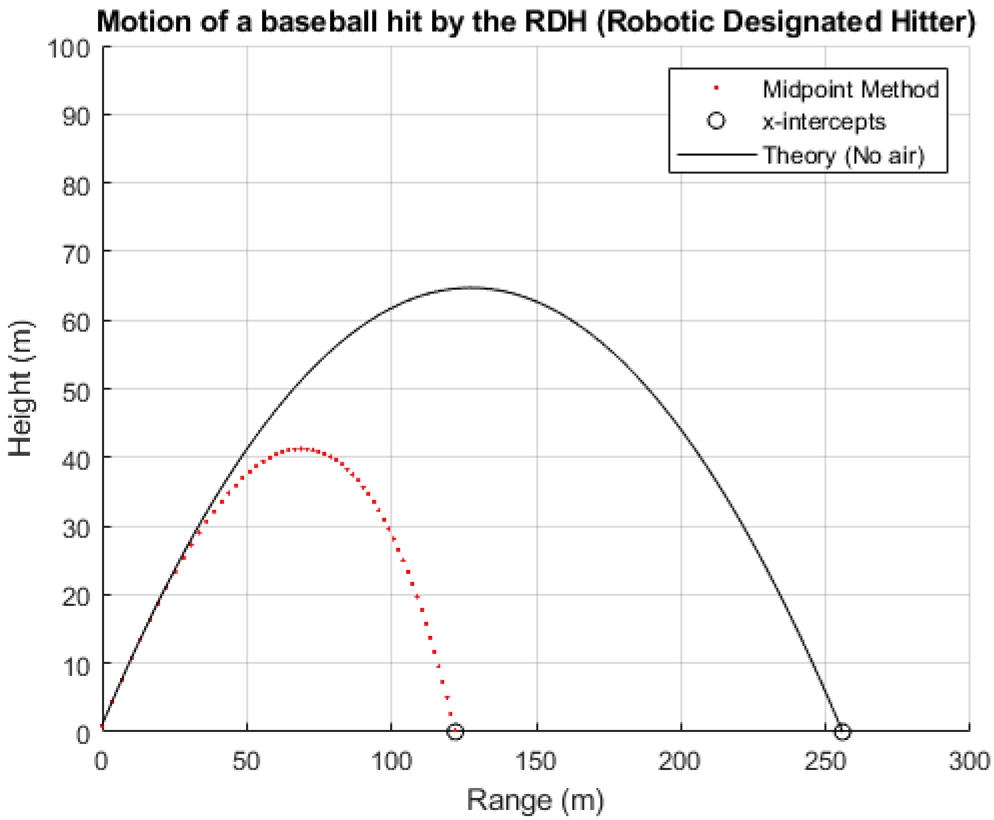

Figure 3: Midpoint method output from **baseball()** for an initial height of 1 m, initial speed of 50 m/s, angle of 45 degrees, and a time step of 0.1s

These figures agree nicely with the figure in the textbook which implemented the Euler method and used the same input parameters.

3. The ABtoHR() Function

3.0 Description

The second step in our analysis was to create a function **ABtoHR()** which determines the at bats to home runs ratio for our RDH. This function takes arguments of a certain number of at bats (the number of trajectories to test for) and a fence height.

It was determined that the RDH's hits could be modeled with a normal distrubtion of initial speeds and launch angles.

A for loop was used to determine: out of how many at bats with an initial launch angle and speed as defined by their normal distributions, was the trajectory a home run. Whether a not a trajectory was a home run was easily calculated using the x-intercept and fence-intercept values which were returned from **baseball()**'s Midpoint method.

ABtoHR() returns the at bats to home runs ratio AB/HR, or returns 0 if there were no home runs.

4. Discussion

4.0 Results

Our initial goal was to determine how a high a fence should be placed so that the RDH's AB/HR ratio stays above 10.

This was an easy task using the **ABtoHR()** function. The loop below calculates the AB/HR ratio for a varying fence height, using 10,000 atBats per fence height, a quite suitable number which will allow us to determine an appropriate fence height with suitable precision.

```
%Determine a number of at bats
atBats = 10000;
%Determine a maximum fence height
maxFenceHeight = 15;
%Retrieve AB/HR ratios for various fence heights
for fenceHeight = 0:maxFenceHeight*10
    i = fenceHeight+1;
    ratios(i) = ABtoHR(atBats, fenceHeight/10);
end
```

A quadtratic was also fitted to our data and its roots were used to determine the minimum fence height.

```
%Retrieve quadratic coefficients a, b, and c from our ratios vs.
%fenceheight data
quadCoeffs = coeffvalues(fit((0:0.1:maxFenceHeight)',
 ratios', 'poly2'));
%subtract c by 10 so that our zeros determine the intersection of AB/
HR =
%10 and the quadtratic fit
quadCoeffs(3) = quadCoeffs(3) - 10;
%Determine the roots. The positive root is our minimum fence height
r = roots(quadCoeffs);
minFenceHeight = r(r>0);
```

4.1 Analysis

The following figure was produced to show the appropriate fence height in order for the RDH to maintain an AB/HR above 10. Ratios determined from our results in 4.0 are plotted against varying fence heights at 400ft, and the horizontal line shows a ratio of 10. The black circle is the intersection of our polynomial fit with the AB/HR = 10 line, and it shows our optimum fence height.

```
figure(4);
hold on
scatter(0:0.1:maxFenceHeight, ratios, 'r.');
plot([0 maxFenceHeight],[10 10], 'k--' );
plot(minFenceHeight, 10, 'ko');
txt = strcat({'\uparrow'},{' '}, {'Minimum Fence Height:'},{' '},
 {num2str(minFenceHeight)});
text(minFenceHeight, 9,txt{1});
ylabel('At Bats to Home Runs Ratio AB/HR');
xlabel('Fence Height (m)');
```

```
title('AB/HR ratios by the RDH vs. various fence heights at 400ft');
```

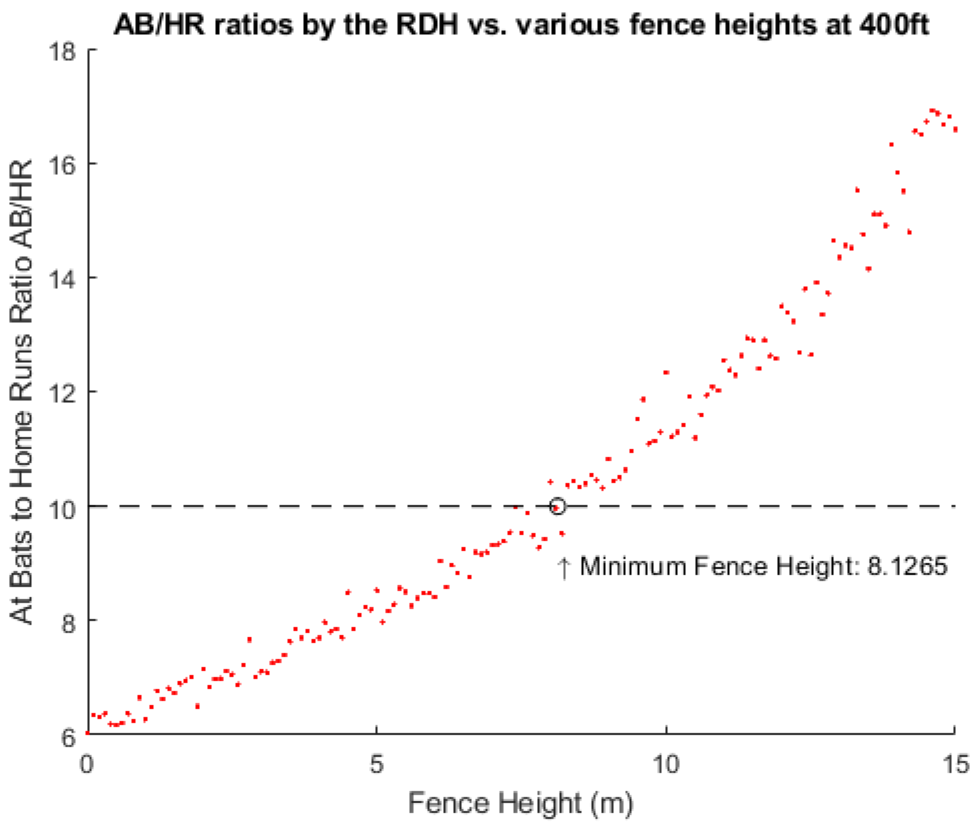

Figure 4: AB/HR ratios plotted against varying fence heights

This figure shows the fence height at which the AB/HR ratio begins to stay above 10. This is the fence height at 400ft which will be reccomended to the league.

5. Conclusion

In conculsion, using the **baseball()** function we were able to determine approximate trajectories of hits by the RDH using iterative ODE solver methods. The function **ABtoHR()** was then used to calculate the ratio of at bats to home runs using the **baseball()** function to determine if a trajectory was a home run for a normal distribution of initial speeds and launch angles. In our final analysis, several AB/HR ratios were calculated for different fence heights, and it was determined that the league should consider installing fences at a minimum height of:

```
disp(strcat(num2str(minFenceHeight),'m'));
```

8.1265m

in every stadium (standard 400ft pitch length) if the RDH is to be implemented for the 2020/2021 season. Any lower of a fence will allow the RDH to maintain an AB/HR ratio below 10, which would place our robotic friend beyond the ranks of the great Babe Ruth.

Published with MATLAB® R2018b