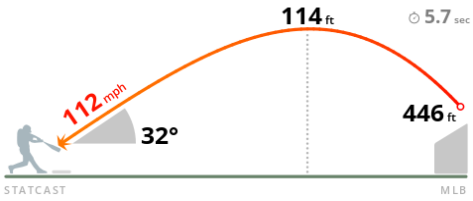
Project 2:
Hitting a home run, with air resistance

Overview: In this project, you use your knowledge of physics to determine the net force on a baseball after it is hit, including the nonlinear effects of air resistance (drag), then use numerical techniques to determine the time of flight, range, and maximum height.

**** Read the entire assignment before beginning Phase 1, so that you know what to expect. ****

Consider the trajectory of a baseball, hit at a "launch angle" of 32°, with an "exit velocity" of 112mph, as shown below. Using a combination of raw data and modeling software, its range was determined to be 446ft, the maximum height was 114ft, and the time of flight was 5.7s.



[source: mlb.statcast]

We would like to see how close we can get to these values, first without air resistance, then with air resistance, and both using numerical techniques. (Note that we will be neglecting "lift", which occurs when there is backspin on the baseball.)

Phase 1. Comparing the analytic solutions to the numeric, without drag (and setting up the structure you will need for adding drag in the next phase)

[20pts] If we know the second derivative of a function, then we can construct the function, as demonstrated in class. Specifically, if the time interval Δt is small enough, we can use the second derivative to find the first derivative and then we can use both to find the function. That is…

$$f'(t+\Delta t) = f'(t) + \Delta t \cdot f''(t)$$

then…

$$f(t+\Delta t) = f(t) + \Delta t \cdot f'(t) + \frac{1}{2}(\Delta t)^2 \cdot f''(t)$$

In this project, at every stage, we will know the net force on the baseball, so we can compute the components of its acceleration, a_x and a_y, where $a_x = dv_x/dt = d^2x/dt^2$ and $a_y = dv_y/dt = d^2y/dt^2$. This means we can use the components of the net force to find $x(t)$ and $y(t)$.

DESIGN SPECIFICATIONS

1. Compute $x(t)$ and $y(t)$ numerically, using the net force on the baseball, without air resistance, but treating the net force as non-constant (to make it easier to add drag in the next phase), and using the analytic time of flight to determine the upper limit of time t.
2. Compare the numeric solutions to the analytic solutions. (Design two checks, one for $x(t)$ and one for $y(t)$, that should return zero if each is the same as its analytic solution.)
3. Use one FOR loop to compute both $x(t)$ and $y(t)$.
4. Use "feet" as the unit of distance for $x(t)$ and $y(t)$.
5. Plot two trajectories, y vs. x, one analytic and one numeric. Include a grid, a meaningful title and legend, and proper axis labels. Font sizes should be appropriate for the size of the figure.

Phase 1. (continued)

For Spec #1, you will need to look up the mass of a baseball, so that you can compute the components of the net force, then divide by the mass to find the components of its acceleration. You should treat the net force and acceleration as NOT constant, i.e., compute their components within each loop of the numeric computation, as this will change in the next phase when we add air resistance. (This might seem strange to you, but it will make adding air resistance **<u>much</u>** easier to do in the next phase, as the drag force does not depend on the mass.)

You can start with the script from class. A calculation of the analytic time of flight is already there, which sets the time scale for everything. However, you should write the rest of the script yourself, as it should be VERY different from what was done in class.

For Spec #2, you should construct two functions, each of which should equal an array of zeros, then for each use a suitable expression to sum the values in a way that it is convincing when the answer is zero. (We have done this quite a few times now.) It is not necessary to plot these two "checking" functions, because we are focusing on y vs. x, not x vs. t or y vs. t. Instead, output one value for $x(t)$ and another value for $y(t)$ that capture the essence of how close these two numeric solutions are to their analytic counterparts.

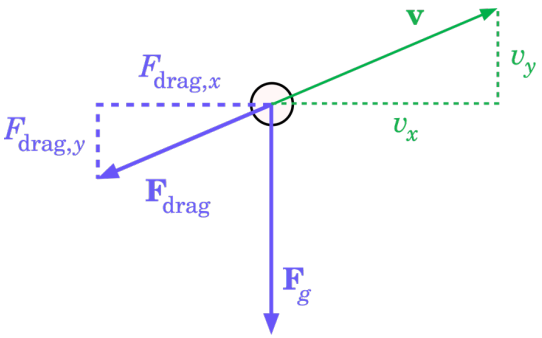
For Spec #3, constructing $x(t)$ and $y(t)$ are independent of each other, but that does not mean you need separate FOR loops to compute them. Both can be computed within the same loop.

For Spec #4, it might be easier to convert to meters early then convert back again to feet at the end.

UPLOAD: Script, Command Window, figure.

Phase 2. Adding air resistance (drag) and checking when drag is "turned off".

[20pts] We can show that the force of air resistance (drag) is proportional to the density of air (ρ_{air}), the cross-sectional area (A) of the baseball, and the speed squared (v^2) of the baseball. The direction of the drag force is directly opposite the velocity of the baseball, as shown. Mathematically…



$$\mathbf{F}_{\text{drag}} = -\frac{1}{2} C \cdot \rho_{\text{air}} \cdot A \cdot v \cdot \mathbf{v}$$

where C is a dimensionless constant (usually between 0.2 and 0.3 for a baseball), v is the speed of the baseball, and $\mathbf{v} = (v_x, v_y)$ is the velocity of the baseball. Therefore…

$$F_{\text{drag},x} = -\frac{1}{2} C \cdot \rho_{\text{air}} \cdot A \cdot v \cdot v_x \qquad F_{\text{drag},y} = -\frac{1}{2} C \cdot \rho_{\text{air}} \cdot A \cdot v \cdot v_y$$

We can now add this to our functioning script and see what happens.

The additional specifications are:

6. Update the script to include drag, as efficiently as possible. Make C a user input.
7. Add minor grid lines, and make all gridlines more visible, to make it easier to estimate values for maximum height and range, directly from the figure.
8. Check that the results are the same as before for $C = 0$. Submit only the Command Window.
9. Run with $C = 0.38$. Include a robust, meaningful legend. Submit only the figure (y vs. x).

Phase 2. (continued)

As in Project 1, rename your script before you start editing, so that you can keep the old one. ("Save As…" is useful for this.)

For Spec #6, look up or estimate values needed for the drag force, and compute the components of the net force at each time t. This is changing within each iteration of the FOR loop, so that is why we are calculating it within the loop rather than outside it. At any particular time t, starting at $t = 0$, we know the location (x, y) and velocity (v_x, v_y) of the baseball, which means we can compute the speed and net force on it $(F_{net,x}, F_{net,y})$, as well as its acceleration (a_x, a_y), which in turn means we can determine the location and velocity at $t + \Delta t$, and iterate until time runs out. Think about how to compute the drag force efficiently.

Don't be concerned that the trajectory of the ball continues after it has hit the ground. (In the next phase, we will develop the logic needed to determine when the ball "lands".) Note that both trajectories are on the same figure, i.e., the (analytic) solution without drag and the (numeric) solution with drag.

For Spec #7, use `grid minor` to add the minor grid lines. To make the grid lines more visible use `ax.GridAlpha` and `ax.MinorGridAlpha` (assuming you are using `ax = gca;` to make the fonts larger). Values of 0.4 for GridAlpha and 0.5 for MinorGridAlpha work well.

Make sure to update the title and legend to match this situation. Specifically, we are no longer comparing the analytic to the numeric solution. Now that we have shown in Phase 1 that they are equivalent, we don't need to mention whether a solution is numerical or not. Instead, we are focusing on comparing drag to no drag. Also, include the input value of C in the legend (and do this robustly).

UPLOAD: New script, Command Window, figure.

Phase 3. Exporting data and analyzing it in Excel

[10pts] Add code to your script to create an appropriate matrix of useful data, i.e., t, x, and y, then export the data to a text file (.txt or .csv) so that you can import it into an Excel workbook.

10. In Excel, estimate the time of flight, maximum height, and range for $C = 0.38$.

For the time of flight, you need to recognize when the ball has "hit the ground". One efficient way to do this is to compare each value of y with the next value. For exactly one value of y, its sign is different from the sign of the next value, i.e., as y changes from + to –, it must pass through $y = 0$. Therefore, create a new column of values $y(t)/y(t+\Delta t)$. The one time that this ratio is negative is a good estimate of the time of flight. You can use the `MIN` function to determine the only negative value in this list of ratios, then you can use the `MATCH` function to find its row number. Finally, you can use `INDIRECT(ADDRESS(rowNumber, columnNumber))` to access this value of the time.

Think about how to do this efficiently, e.g., by having your calculations and labels organized in one column on the side rather than on top, so that you can use whole-column indexing, for example, `A:A` to search all of column A within the `MIN` and `MATCH` functions.

For the maximum height, you can use the `MAX` function in Excel.

For the range, you should access the value of x when the value of y changes from + to –.

Include your MATLAB figure showing the trajectory using the same data you are analyzing. Update the figure from Phase 2 to refer to this phase. (A figure from MATLAB is better than a figure from Excel, because the MATLAB figure has minor gridlines.)

UPLOAD: New script, MATLAB figure, first page of formulas, first page of values.

Phase 4. Getting MATLAB to do the same as Excel, and comparing results

[10pts] You can also estimate the time of flight, maximum height, and range in MATLAB. To determine the time of flight and range, within each iteration of the FOR loop, you should check each value of *y* against the previous value, to see if the ball has hit the ground. Then, grab the time and distance information, and hold on to it until the iterations are complete. The MAX function can be used to find the maximum height (outside the FOR loop).

11. In MATLAB, estimate the time of flight, maximum height, and range for $C = 0.38$.
12. Compare the results using MATLAB to the previous results using Excel from Phase 3.
13. Compute the "final" speed of the baseball, in mph, and the energy lost due to air resistance, in joules.
14. Choose longer variable names to make the output as meaningful as possible.

For Specs #11 and #12, the results should be exactly the same as before. Add a comment in your script about the results, i.e., either include the "values" page from Phase 3 or type the results into your script for comparison. (No calculations of percent difference are needed.)

For Spec #13, keep in mind that the "final" speed is the speed of the baseball when it lands, NOT the "last" speed after the FOR loop is finished. Therefore, grab the speed in the FOR loop at the same time that you grab the time and distance information. You should keep the speed in m/s, so that it's easier to compute the "final" kinetic energy of the baseball. To find the energy lost due to air resistance, use the change in kinetic energy, as the potential energy has not changed.

For Spec #14, you will not be using any of these variables in many calculations, so longer names are appropriate, with unit abbreviations, such as range_ft and energyLost_J. Be sure to keep other variable names short, e.g., the final speed in m/s should be simply vF, so that it's easier to use in expressions.

UPLOAD: New script, Command Window.

Phase 5. Exploring the results

[20pts] Once everything is working, let's explore the effects of drag, as compared to the results from mlb.statcast.

(a) For what value of *C* is the range within 0.1% of 446ft (as shown in the original diagram)?

(b) What are the maximum height and time of flight for this value of *C*?

(c) How well or poorly do these values compare to the corresponding values in the original diagram? Be quantitative.

(d) What is the "final" speed, in mph? How much energy is "lost", in J?

For part (a), compute the percent error in the range, then run your script and enter values of *C* until you are within 0.1% of the desired range, then clear the Command Window and run it one more time with that value of *C*. For part (b), make sure the maximum height and time of flight are output to the Command Window. For part (c), add efficient, robust code to compute the percent errors between your approximate values and the known values in the diagram, then write appropriate comments in your script. Note that the sign of the percent error is useful, so don't use the ABS function, with positive meaning the approximate values are too large. For part (d), these were calculated and output in Phase 4, so you should not need to do anything special.

UPLOAD: New script, Command Window.

Summary

Here is a table showing what you should include in your solution for each part.

Phase	Script?	Command Window?	ONE page of formulas?	ONE page of values?	MATLAB figure?
1	yes	yes	no	no	1
2	yes	yes	no	no	1
3	yes	no	yes	yes	1
4	yes	yes	no	no	0
5	yes	yes	no	no	0