Matlab Project Assignment – Falling like a Rock

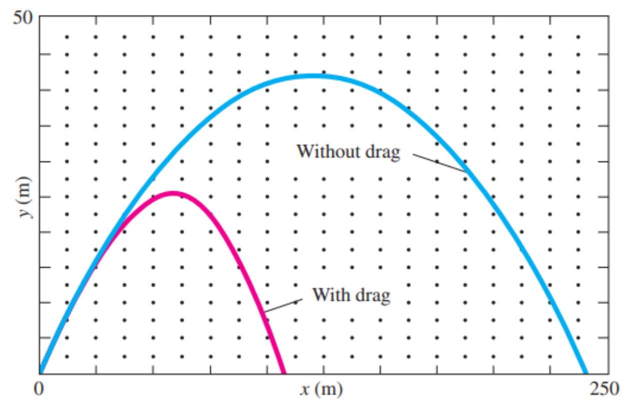
Dwayne "The Rock" Johnson starred in an entertaining movie called "Skyscraper" which had a movie poster that showed some questionable physics and briefly became a fun internet meme. Check out [this link] for some analysis and funny ideas about this movie poster from the internet.

The conclusion from the (tongue in cheek) internet analysis is that The Rock would have to run about as fast as Usain Bolt, who can sprint at a max speed of about 27 mph or 12 m/s. Though The Rock is extremely fit and built for strength, it is extremely unlikely that he could run as fast as Usain Bolt who is built for speed and the fastest sprinter in the world.

In the actual movie, the scene plays out slightly differently, and there does not appear to be similar physics-based dynamic analysis about the actual movie scene. Here is [a YouTube video showing the actual scene] from the movie. The objective of this Matlab assignment is to create a mathematical model of The Rock attempting this jump from the movie and determining whether it is feasible for The Rock to make the jump from a realistic physics-based perspective.

Part 1: Baseball Projectile Model with Aerodynamic Drag

The Rock's jump can be modeled with reasonable accuracy using projectile motion equations from physics, but we should add in the effect of air resistance to the projectile motion model to account for the drag force acting against The Rock during the jump because he presents a large area and would have to be moving pretty fast. [This reference from a Dynamics textbook] will be helpful to develop the Matlab model of projectile motion with drag.



We'll start with the algorithm and values from this Dynamics textbook reference and replicate their results for a baseball as a starting point to validate the model of projectile motion with air resistance. The algorithm uses the same basic projectile motion model from physics, but modifies the acceleration terms to account for the effects of the aerodynamic drag force. **You will need to write a standalone Matlab function called "projectile_motion_drag" that computes 2D projectile motion trajectories for given parameters passed into the function.** This function should take as inputs the initial conditions and mass and drag properties of the modeled problem. Outputs should be provided as a result of "projectile_motion_drag" and they should at least include the x and y coordinates of the trajectory over time. The equations for the algorithm and commented pseudocode are listed below:

% create projectile_motion_drag function in projectile_motion_drag.m file

% inputs should be: v (velocity magnitude), angle (jump angle), g (gravity), m (projectile mass)

% A (projected area), cd (drag coefficient), rho (air density)

% y (initial vertical position), x (initial horizontal position)

% outputs should be: x (horizontal position vector), y (vertical position vector)

function [x,y] = projectile_motion_drag(v,angle,g,m,A,cd,rho,y,x)

% establish initial values for a_x, v_x, x, a_y, v_y, y based on function inputs (assume a_x = a_y = 0)

% calculate drag constant D based on function inputs rho, cd, and A

$$D = \frac{1}{2}(rho * cd * A)$$

% establish an initial guess for the time t to start simulation, where end_t is around 100 seconds and Δt is a sufficiently small time step to effectively capture data (0.01 or 0.001 s) t = 0 : Δt : end_t;

% create a for or while loop to loop through time step values

% implement the following projectile motion equations in code for each timestep n:

$$v(n) = \sqrt{v_y^2(n) + v_x^2(n)} \quad \text{Eq(1)}$$

$$a_y(n+1) = -g - \left(\frac{D}{m}\right) * v(n) * v_y(n) \quad \text{Eq(2)}$$

$$v_y(n+1) = v_y(n) + a_y(n) * \Delta t \quad \text{Eq(3)}$$

$$y\ (n+1) = y(n) + v_y(n) * \Delta t + \frac{1}{2} a_y(n) * \Delta t^2 \quad \text{Eq(4)}$$

$$a_x(n+1) = -\left(\frac{D}{m}\right) * v(n) * v_x(n) \quad \text{Eq(5)}$$

$$v_x\ (n+1) = v_x\ (n) + a_x\ (n) * \Delta t \quad \text{Eq(6)}$$

$$x\ (n+1) = x(n) + v_x\ (n) * \Delta t + \frac{1}{2}\ a_x\ (n) * \Delta t\text{^}2 \quad \text{Eq(7)}$$

% if using a for loop, make sure to include a conditional if statement to break out of the loop (i.e. use a break statement if y(n+1) <= 0)

% if using a while loop, make sure to increment time step n and consider the conditional logic

% this will stop the simulation after the projectile hits the ground 0 or a target vertical position

% end loop

% end function

Next, create a separate .m file (in same working folder) to define the variable inputs to the function and call the projectile_motion_drag function 3 times to plot the following (see example plot below):

1. Baseball hit at sea level with cd = 0.5 and rho = 1.2 kg/m^3 (sea level air density)
2. Baseball hit at sea level with no drag (cd = 0) and rho = 1.2 kg/m^3 (sea level air density)
3. Baseball hit in Denver with cd = 0.5 and rho is 82% of sea level air density

The baseball parameters are given on the last page of the Dynamics textbook reference.

% define variables v (velocity magnitude), g (gravity), angle (launch angle in degrees)

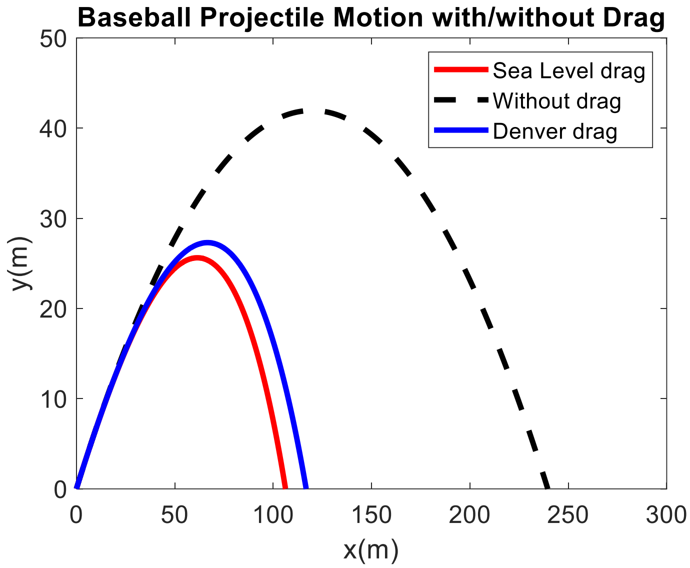
% m (projectile mass), A (projected drag area), cd (drag coefficient, rho (air density), and initial x and y

% call the projectile_motion_drag function with different cd and rho values as outlined above

% plot the results for the baseball at sea level with and without drag and in Denver (see plot below)

% calculate the max vertical and horizontal distances in each case

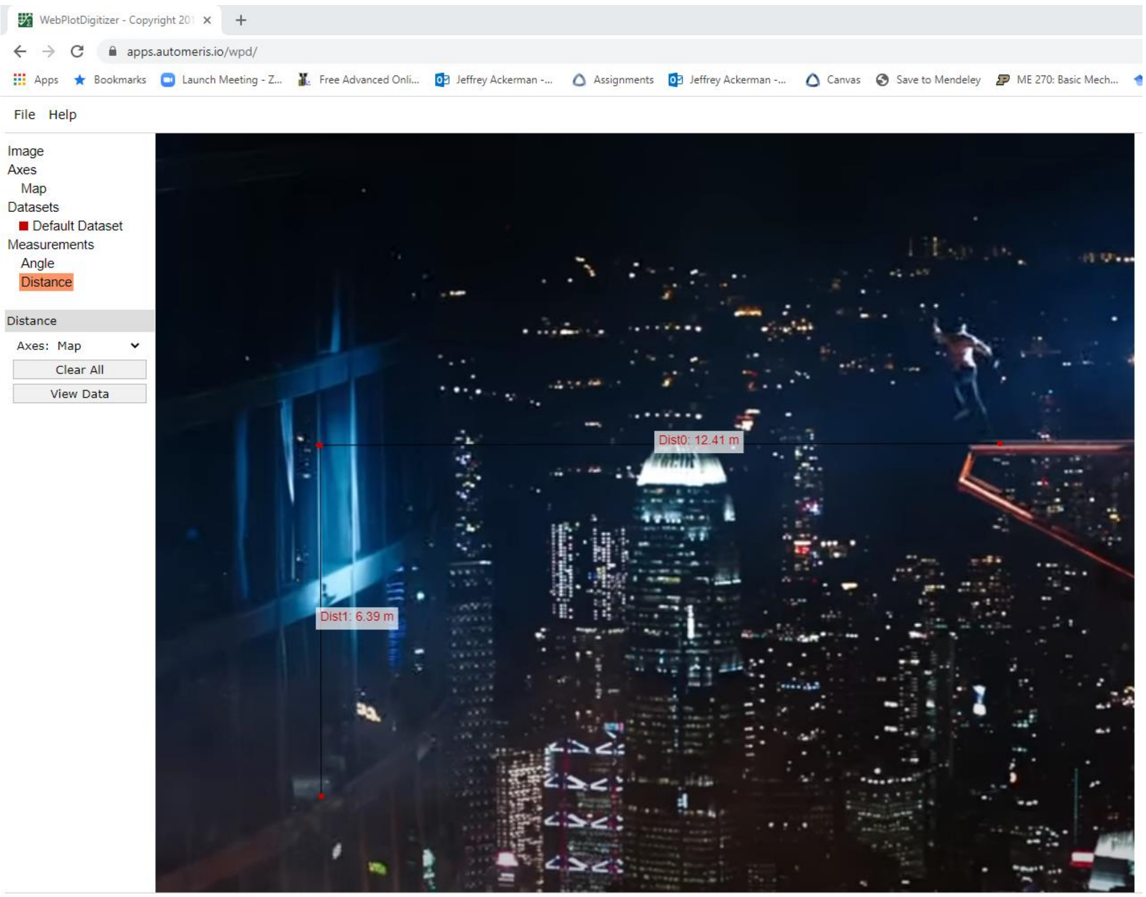
% print out the % difference in max vertical and horizontal distance relative to baseball at sea level



Part 2: Simulating The Rock's Skyscraper Jump

To simulate The Rock's jump, we'll need to make some reasonable estimates of the scenario to create a physics-based projectile motion analysis. The Rock weighs about 260 lb, or **118 kg**, with a height of about 6'5", or **1.95 m**. If we take a still screenshot of the jump scene from the movie, we can scale the distances in the image relative to The Rock's height to estimate the vertical and horizontal distance of the jump. We could process a screenshot of the movie still image in Matlab, but for simplicity we'll use an online tool called [Web Plot Digitizer] and scale the pixels of the image based on The Rock's height.

Based on the image analysis from the still movie image, The Rock needs to jump about **12.4 m** horizontally while falling about **6.4 m** vertically if he's going to make the jump. We could set the initial vertical position to 6.4 m with a vertical landing at 0 m, and the initial horizontal position at 0 m with a final horizontal position based on the projectile motion simulation. The angle of his jump off the crane is measured to be about **18 degrees** from horizontal.



To model the air resistance of The Rock's body, we can approximate him as a rectangle with a height of 1.95 m and a width of approximately 0.5 m to calculate the rectangle drag area. We'll approximate the drag coefficient as that of a person from this reference (**cd = 1.20**), which is roughly the same as that for a large rectangle. Use the same air density as for the sea level baseball problem (**1.2 kg/m^3**).

Dwayne "The Rock" Johnson is built like an NFL lineman. He actually played college football for Miami and entered the NFL draft, but was not drafted in the NFL. The average NFL player of a similar weight can do the 40 yard (36.6 m) dash in about 4.8s (average speed of 7.63 m/s). A person can run at a maximum speed in a short burst that's about 1.15x higher than their average speed in a 40 yd dash. If we assume that The Rock could sprint at this maximum speed just before the jump, how far would The Rock travel horizontally under these conditions? Would he make the jump? Also note that in the movie, The Rock is playing a character with a prosthetic leg and the Rock himself is 48 years old. Support your findings with plots and outputs for your report.

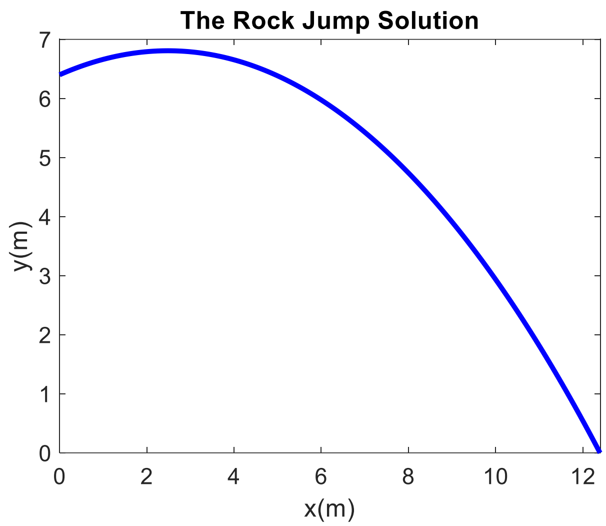
Part 3: Solving for the Required Initial Velocity to Make the Jump

We can design the code to solve for the exact initial velocity to achieve the jump based on a specified target. To do this, we'll use the fsolve function, which is a very useful optimization function which can repeatedly plug in input parameter values into a complex function until an output reaches 0 and the problem is solved. You will learn more about zero finding mathematical algorithms like fsolve in MATH

307 (Scientific Computing). For now, we just need to know how to setup fsolve to solve the problem and determine the initial velocity that would complete the jump in this situation. Note that you will need to have the “Optimization Toolbox” installed to use the fsolve function, or you can use the fzero function if you aren’t able to install the “Optimization Toolbox.”

This is the basic format of how to modify your existing code to use the fsolve function:

```
% define all required variables, including the target x_final value

% use fsolve on the find_solution function to solve for the initial velocity v_sol where x(end) = x_final

[v_sol] = fsolve(@(v)  find_solution(v, angle, g, m, A, cd, rho, y, x, x_final), v);

% set new initial v (velocity magnitude) to v_sol solution where x(end) = x_final

v = v_sol;

% call projectile_motion_drag function using v = v_sol to get the x and y position vectors

[x,y] = projectile_motion_drag(v, angle, g, m, A, cd, rho, y, x)

% plot and output results


% The function below should be in a separate file called find_solution.m (in same working folder)

% There shouldn’t be a need to change this provided find_solution.m function

% [output_driven_to_0] = solver_function_name(inputs)

function [x_sol] = find_solution(v,angle,g,m,A,cd,rho,y,x, x_final)

   % Call the projectile_motion_drag function to obtain results

   [x1,y1] = projectile_motion_drag(v,angle,g,m,A,cd,rho,y,x);

   % fsolve will input different v values until the final x value minus the target x value is very close to 0

   x_sol=x1(end)-x_final;

end
```



Note: The fsolve function is a powerful way to iteratively solve complex systems. If you are unable to get either the fsolve or fzero functions to work, as a backup option you could loop through different initial velocities to similarly solve the problem, but it's best to use either fsolve or fzero.

Assignment Deliverables:

Create a well-formatted white paper document (see template on Canvas) with clear plots (make sure the plots are professionally formatted with clear text at least as large as the body text) to present and discuss your results.

1. Part 1 – Baseball projectile motion with results with clear screenshot (**20 pts**).
    a. Discuss the results in a paragraph. What was the % difference in max vertical and horizontal distances relative to the baseball at sea level? What should you consider when designing for aerodynamic drag? (**5 pts**).
2. Part 2 – The Rock projectile motion initial estimate plot results with clear screenshot (**20 pts**).
    a. Discuss the results in a paragraph. Based on your analysis, is it plausible that The Rock could have practically or theoretically made the jump shown in the movie? (**5 pts**).
3. Part 3 – The Rock projectile motion problem screenshot solved with the correct velocity (**10 pts**).
    a. Discuss the results in 1-2 paragraphs. Is it physically possible that The Rock or a typical trained athlete could have made this jump from the movie based on professional baseball, NFL, or track and field athletes' sprinting and jumping abilities? Explain your thoughts and cite at least two external references using MLA format. (**5 pts**)
4. Submit your final **fully-commented** (including descriptive header) .m code files (**30 pts**).
    a. Write a custom help function for your projectile_motion_drag.m function which appears when you type "help projectile_motion_drag" in the Matlab command window (**5 pts**).
5. Combine all files into a single .zip file and submit via Canvas:
    a. Well-formatted white paper document following the template on Canvas (.pdf file)
    b. projectile_motion_drag_baseball.m
    c. projectile_motion_drag_the_rock.m
    d. projectile_motion_drag.m (function)
    e. find_solution.m (function)