MXB103 Project Group 28: BUNGEE!

Table of Contents

1 Introduction

To promote Brisbane's "New World City" transformation, bungee jumping off the Story Bridge is the new suggested attration by Brisbane City Council. Further details of the proposal will be: Section 2: we would consider the proposal and the main question related to the project. Detail would be presenting on later. Section 3: we would discuss the equation representing the bungee jump model, the theory under the surface and each symbol definition. Section 4: We would analyse the model and implement the second order equation for an explanation. Parameters would be attached with comment. Then the model with Matlab coding format is displayed at the end of the section. Section 5: We would adjust the equation to enhance the precision of model predictions. Those predictions involved timing and bounces, maximum speed, acceleration and distance travelled experienced by the jumper. Finding the trigger time of the camera system, and also the research of water touch option. Section 6: We would conclude the whole research result and provided a suggestion for further plan.

2 The proposal

The proposal is a bungee jump from the Story Bridge. We have been tasked with creating a mathematical model of the bungee jumping process, solving it numerically. We are provided some questions to solve this project more efficiently.

There are six questions. First of all, the bunjee jump company consider standard bungee jump to consist of 10 "bounces" in around 60 seconds, at which the jump is considered complete. We are asked if our model results shows 10 "bounces" in around 60 seconds. For this problem, we use Modified Euler method for getting the time and distance values. After we got these values, we plot these values and determine if this is aligns with our assumed result. Second question is asking what is the maximum speed and when it occurs, to know about the "thrill level" of the bungee jump. To solve this problem, we use the same method of the first question to get the overall speed. From the results, we got the maximum speed and also the time when

the speed reaches maximum. For the next step, we plot the graph with speeds and time, and plot the point of the maximum speed on the graph. Another question is asking the maximum acceleration and when does it occur. To answer this problem, we calculate the acceleration by using the results for speed and time from Task One. Moreover, we plot the time and acceleration with the line of 2g to see if jumper experiences a force of 2g or not. For question 4, we are asked to get the distance traveled in 60 seconds. To solve this problem, we get the sum of calculating each speed times value of time. Question 5 is asking about the time when the camera should be triggerd to take the picure of the jumper. We used four approximations of when the jumper passed the deck position, and used a root finding method to get the accurate time when the jumper reaches the height of the deck where the camera is located. The final question is asking how close the jumper gets to the water. Moreover, we have to compute the required length and spring constant of the rope for jumper to touch water at the lowest point of their jump. To calculate how close jumper gets to the water with the current rope length and spring constant, we used the height of the jump point minus the maximum distance traveled by the jumper. We attempt many ways with changing either or both of the rope length and spring constant to provide true water touch experience for an 80kg jumper, while keeping as close as possible to 10 bounces in 60 seconds.

3 The model

The mathematical model equation for the bungee jump is:

$$\frac{dv}{dt} = g - C|v|v - \max(0, K(y - L))$$

where C = c/m, and K = k/m (note. 'k' is the spring constant of bungee rope, 'c' is drag coefficient, and 'm' is mass of jumper). This equation contains a few forces which are acting on the jumper's simultaneously during the jump. The forces are gravity(g), drag(-C|v|v), and tension(-max(0,k(y-L))). The maximum function will work only when jumper's position which is y is greater than unstretched rope which is represented by 'L' (e.g. when the rope is taut) and the minus sign works for ensuring that it acts upwards.

3.1 Assumptions and limitations

We assume all of parameters that the client already provide us to calculate the results and those values will not be changed during the bungee jumping. The mass of rope and the jumper's posture are not provided as factors or paramaters to be considered even those two values could affect the gravitational acceleration during the bungee.

3.2 Parameters

```
H = 74;             % Height of jump point (m)
D = 31;             % Deck height (m)
c = 0.9;            % Drag coefficient (kg/m)
m = 80;             % Mass of the jumper (kg)
L = 25;             % Length of bungee cord (m)
k = 90;             % Spring constant of bungee cord (N/m)
g = 9.8;            % Gravitational acceleration (m/s^2)
C = c/m;            % Scaled drag coefficient
K = k/m;            % Scaled spring constant
```

4 The numerical method

$$y(i+1) = y(i) + h * v(i)$$

$$v(i+1) = v(i) + h(g - C|v(i)|v(i) - max(0, K(y(i) - L)))$$

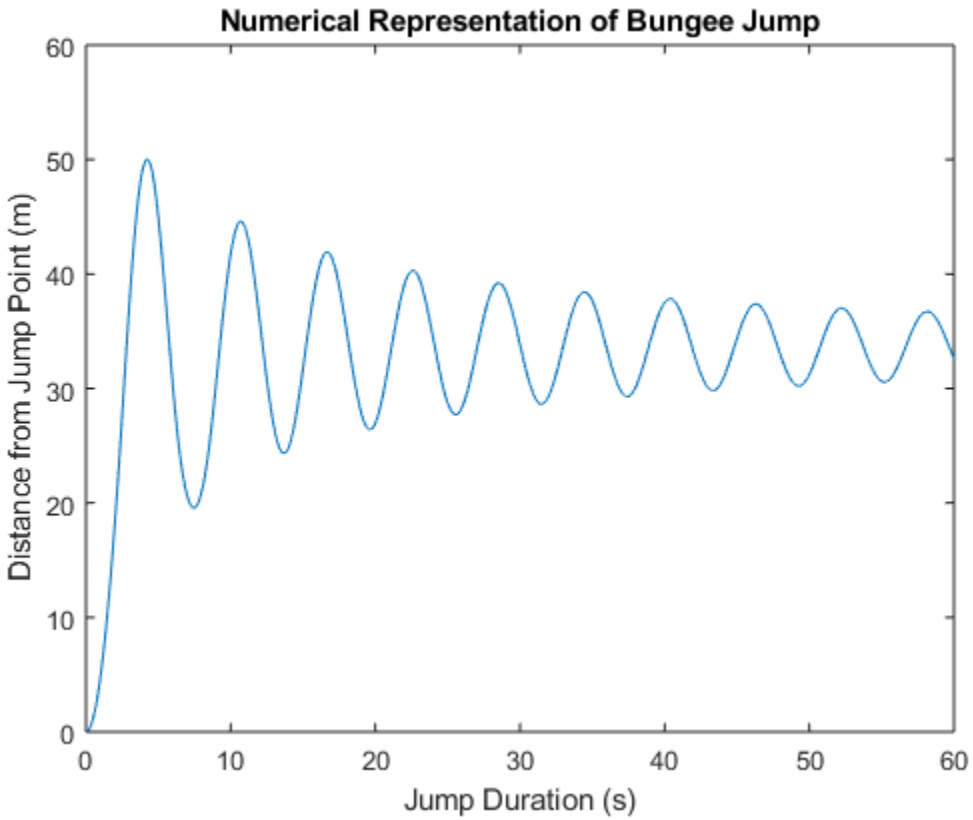
4.1 Parameters

```
T = 60;             % Final time in simulation (s)
n = 10000;          % Number of subintervals
```

4.2 Solution

The ordinary differential equations are solved using a modified Euler's method.

```
[t, y, v, h] = modified_euler_bungee(T, n, g, C, K, L);


figure
plot(t, y);
xlabel('Jump Duration (s)');
ylabel('Distance from Jump Point (m)');
title('Numerical Representation of Bungee Jump');
```



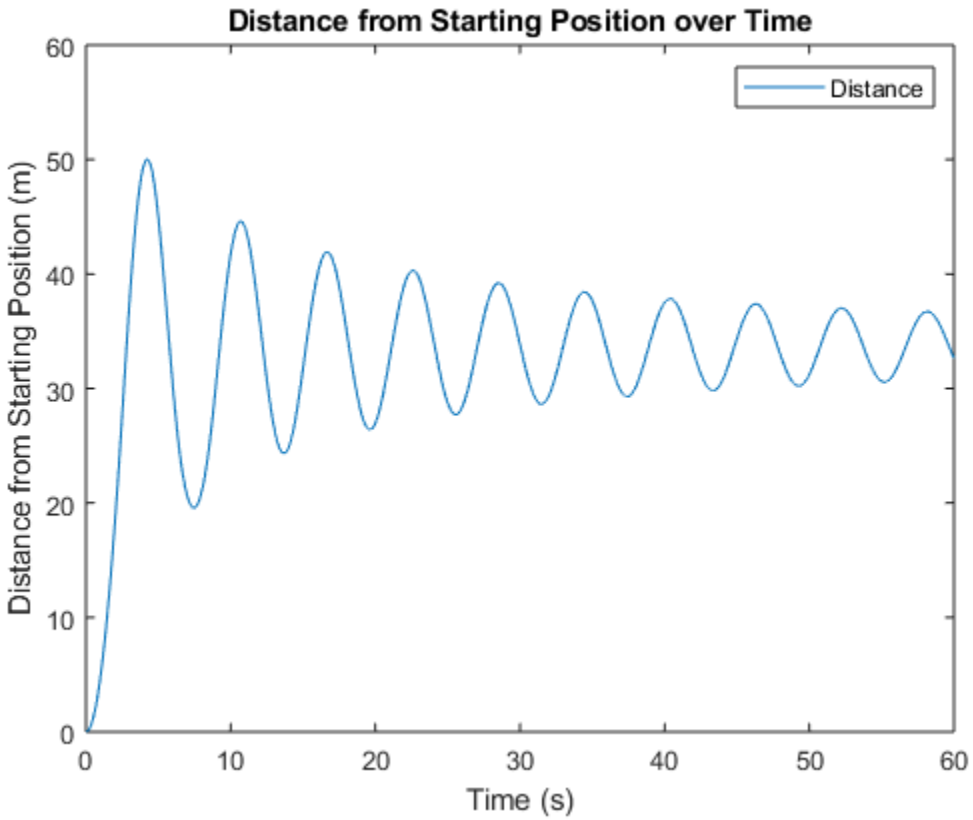
5 Analysis

In this section, the model predictions are analysed with respect to the key questions being asked about the proposal by the client.

5.1 Timing and bounces

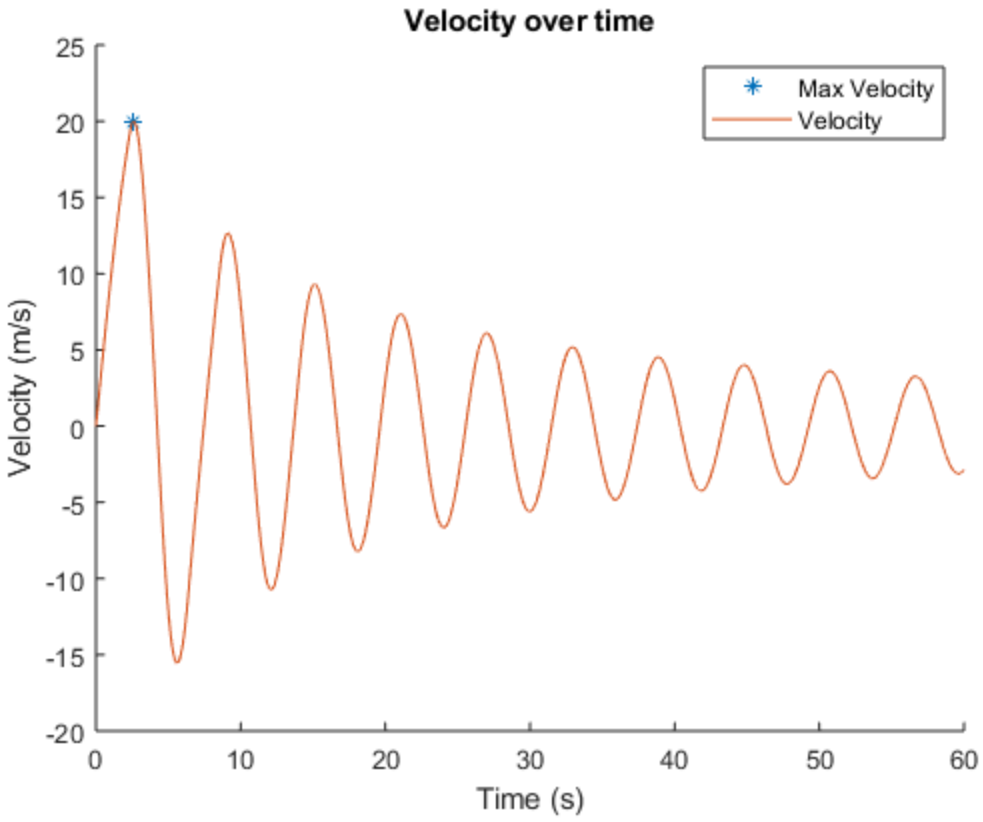
```
disp("Task 1");
disp('Does the jumper experience 10 "bounces" in the first 60 seconds
 of their jump?');

% Using the Modified Euler method to calculate.
[mod_t, mod_y, mod_v, ~] = modified_euler_bungee(60, 10000, 9.8,
 0.01125, 1.125, 25);

figure
plot(mod_t,mod_y); % Distance over time
xlabel("Time (s)");
ylabel("Distance from Starting Position (m)");
title("Distance from Starting Position over Time");
legend('Distance');
% Print the results to the command window
disp("The jumper experiences ten bounces during the jump");
```

```
Task 1
Does the jumper experience 10 "bounces" in the first 60 seconds of
 their jump?
The jumper experiences ten bounces during the jump
```



5.2 Maximum speed experienced by the jumper

```
---------------------------------

disp("Task 2");
% Maximum speed experienced by the jumper
disp("What is the maximum speed experienced by the jumper?");


figure;
hold on;
max_speed = max(mod_v);
indexOfFirstMax = find(mod_v == max_speed, 1, 'first');
maxY = mod_v(indexOfFirstMax);
maxT = mod_t(indexOfFirstMax);
plot(maxT,maxY,'*'); % Maximum velocity
plot(mod_t,mod_v);  % velocity over time
title("Velocity over time");
xlabel("Time (s)");
ylabel("Velocity (m/s)");
legend("Max Velocity","Velocity");

% Print the results to the command window
disp("The max speed of the jumper is " + max_speed + "m/s and it
 occurs after "+maxT+" seconds");

Task 2
What is the maximum speed experienced by the jumper?
The max speed of the jumper is 20.0143m/s and it occurs after 2.604
 seconds
```

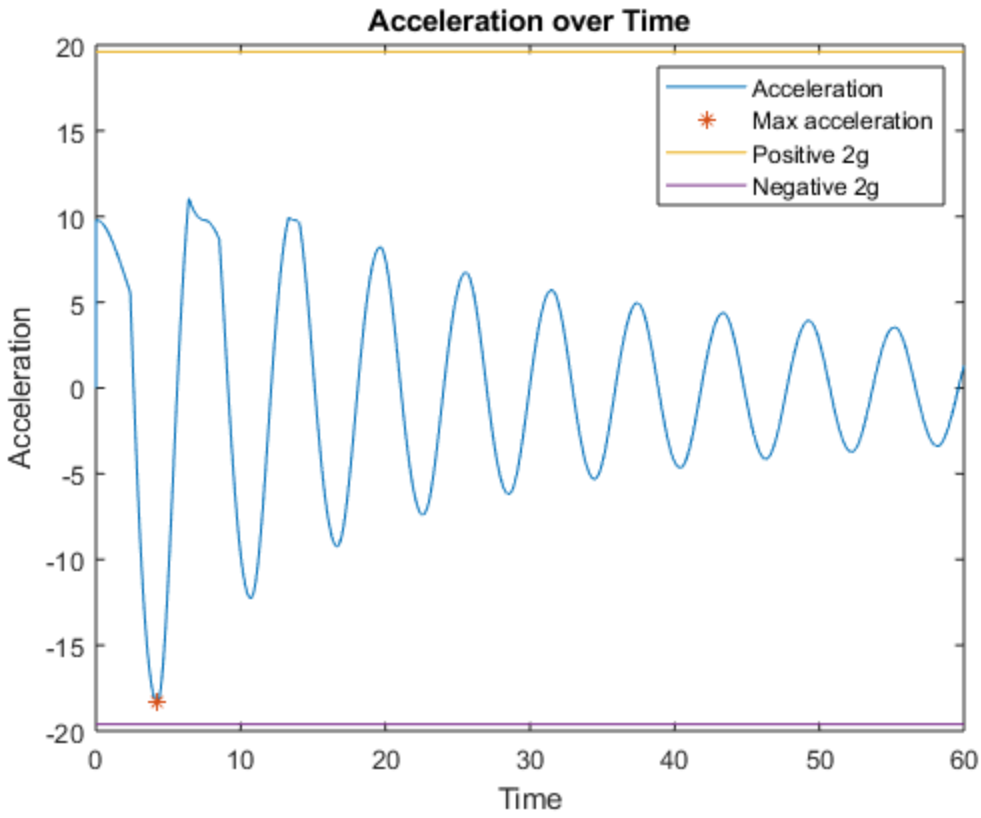


5.3 Maximum acceleration experienced by the jumper

```
---------------------------------

disp("Task 3");
% Acceleration experienced by the jumper during the jump
disp("What is the maximum acceleration experienced by the jumper
 during the jump?");


n = length(mod_t);
a = zeros(1,n); % create acceleration array
for i=2:n  % calculate accelertation at each step
    a(i) = (mod_v(i)-mod_v(i-1))/(mod_t(i)-mod_t(i-1));
end
a1 = max(a);  % maximum positive acceleration
a2 = min(a);  % maximum negative acceleration
if a1 > abs(a2)  % Determine the greater acceleration
    max_accel = a1;
else
    max_accel = a2;
end
indexOfFirstMax = find(a == max_accel, 1, 'first');
maxA = a(indexOfFirstMax);
```

```
maxT = mod_t(indexOfFirstMax);

% Two arrays to represent 2 times gravity (9.8m/s^2) on the graph
 (+/-)
pos_2g = ones(1,n) .* (9.8 * 2);
neg_2g = pos_2g .* -1;

figure;
plot(mod_t,a); % acceleration over time
hold on;
plot(maxT,maxA,'*'); % max acceleration
% plot positive and negative 2g on the graph for reference
plot(mod_t,pos_2g);
plot(mod_t, neg_2g);

title("Acceleration over Time");
xlabel("Time");
ylabel("Acceleration");
legend("Acceleration", "Max acceleration", "Positive 2g", "Negative
 2g");

disp("The maximum acceleration experienced by the jumper is
 "+max_accel+"m/s^2 during the jump, which is almost 2g");
```

Task 3
What is the maximum acceleration experienced by the jumper during the jump?
The maximum acceleration experienced by the jumper is -18.3366m/s^2 during the jump, which is almost 2g



5.4 Distance travelled by the jumper

```
---------------------------------

disp("Task 4");
% Calculate the total distance traveled by the jumper during their
 jump
disp("What is the total distance traveled by the jumper during their
 jump?");

distance_traveled = 0; % initial value
time_step = mod_t(2) - mod_t(1); % calculate the change in time
for i = 1:n  % Sum all the distances
    distance_traveled = distance_traveled + abs( mod_v(i) *
 time_step);
end

% Display the result to the command window
disp("The jumper traveled " + distance_traveled + "m during their
 jump");

Task 4
What is the total distance traveled by the jumper during their jump?
The jumper traveled 281.0496m during their jump
```

5.5 Automated camera system

```
---------------------------------

disp("Task 5");
% Calculating the time to set the trigger for the camera
disp("What time should the camera be set to trigger?");

% calculate the four values closest to H-D
H = 74;
D = 31;
deck = H-D;
index = find(mod_y > deck, 1, 'first'); % first value of y that is
 larger than D-H, y3
%
y1 = mod_y(index-2);
y2 = mod_y(index-1);
y3 = mod_y(index);
y4 = mod_y(index+1);

poly_points = [y1 y2 y3 y4];


t1 = mod_t(index-2);
t2 = mod_t(index-1);
t3 = mod_t(index);
t4 = mod_t(index+1);

poly_times = [t1 t2 t3 t4];


p = inter_ploy(poly_times, poly_points,poly_times);
[dis,time] = root_finding(y2,y3,t2,t3,8);
time;
disp("The camera should be triggered "+time+" seconds after the jumper
 to capture the perfect photo.");

Task 5
What time should the camera be set to trigger?
The camera should be triggered 3.333 seconds after the jumper to
 capture the perfect photo.
```

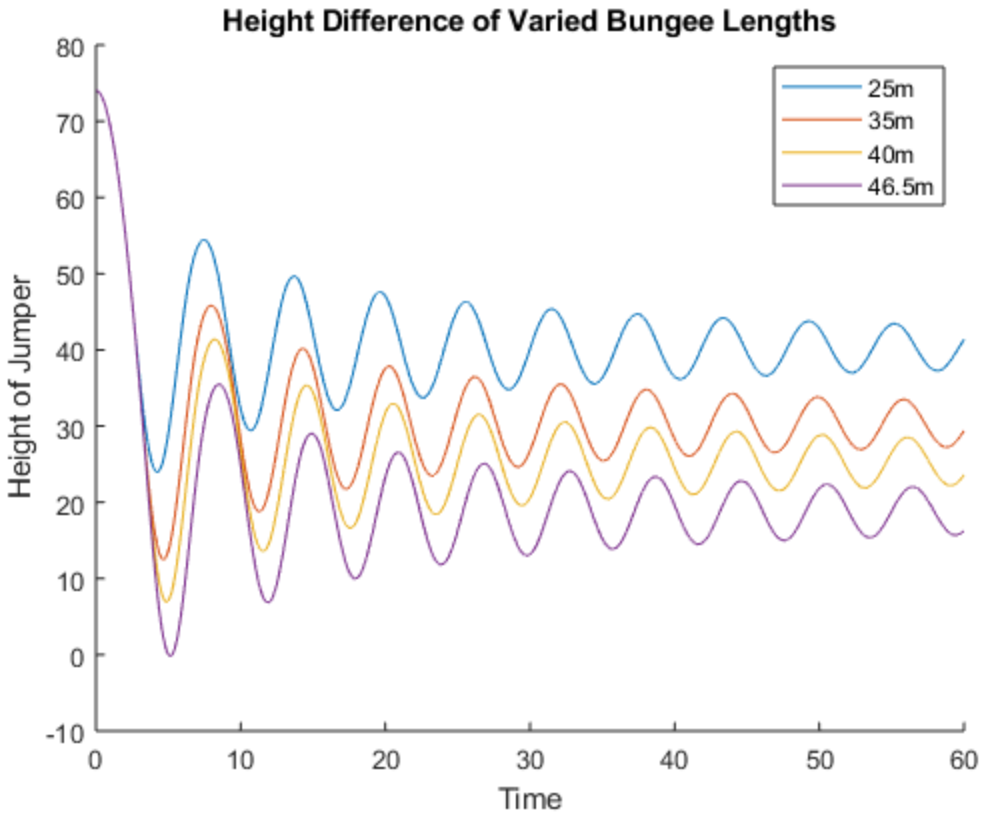
5.6 Water touch option

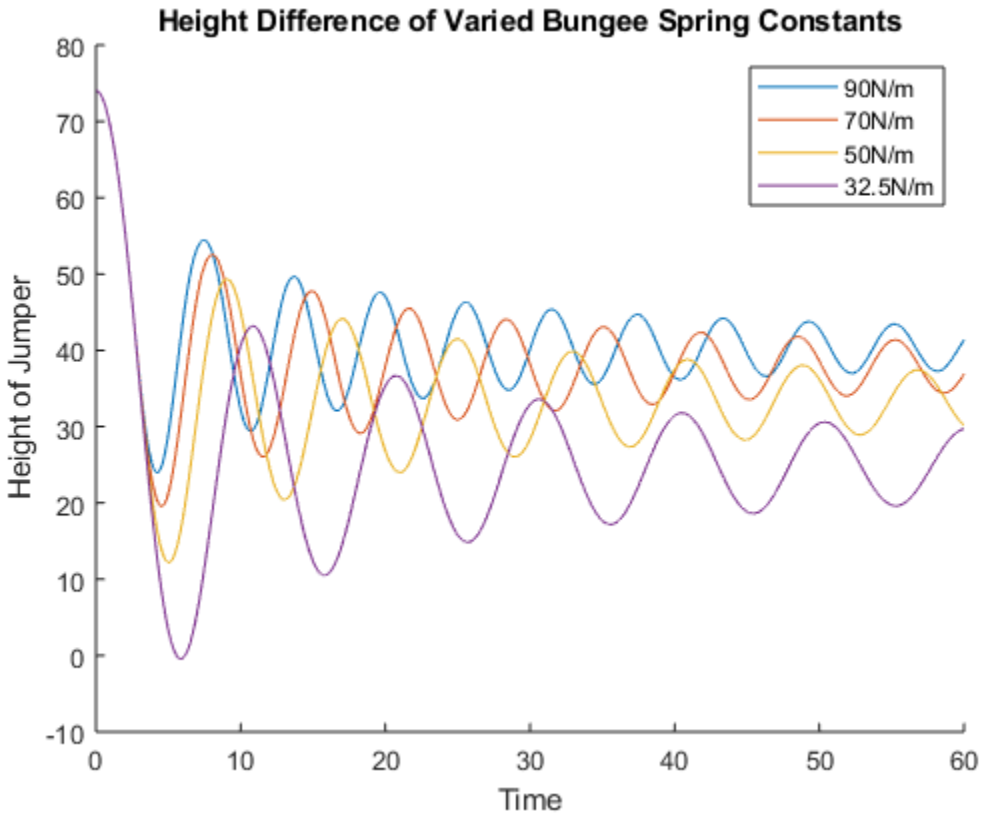
```
---------------------------------

disp("Task 6");
disp("How close does the jumper get to the water?");


[mod_t, mod_y, mod_v, mod_h] = modified_euler_bungee(60, 10000, 9.8,
 0.01125, 1.125, 25);
```

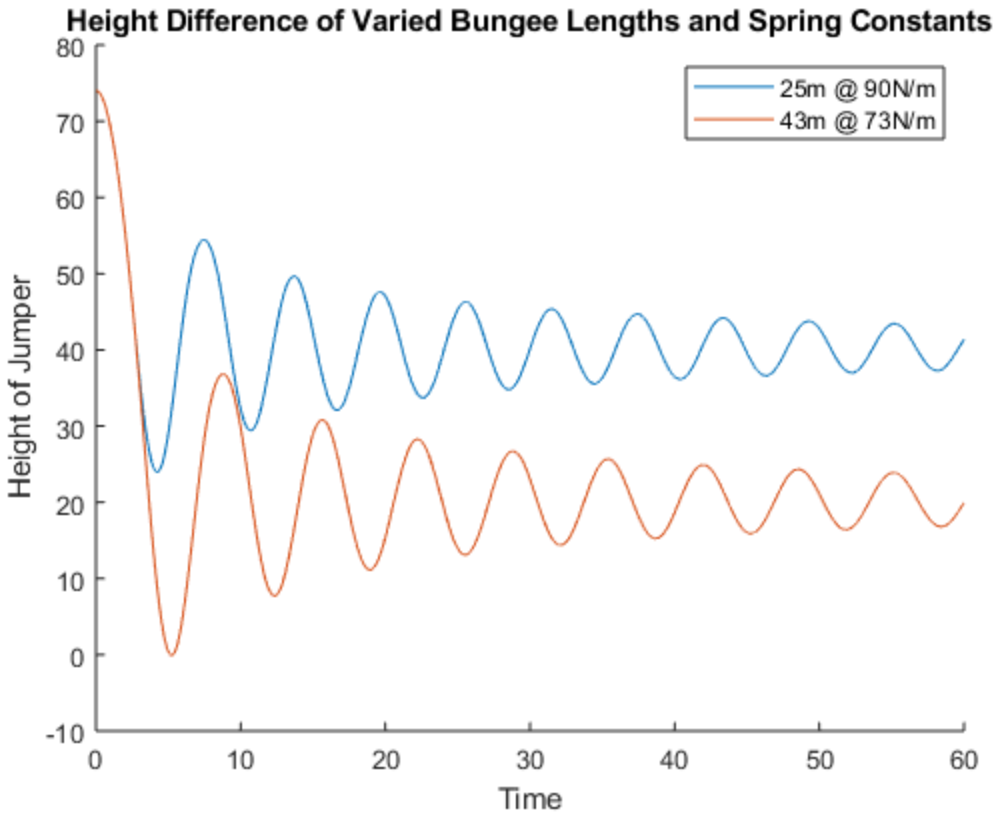
```
lowest_height = 74 - max(mod_y);
disp("The jumper gets "+lowest_height+" metres from the water on the
 first bounce.")



disp('Calculate the factors that would need to be changed to achive a
 "water touch" on the first bounce of the jump');


% Altering the length of the bungee cord
figure;
hold on;
bungee_lengths = [25 35 40 46.5]; % in metres

% Graph the bungee length versus the orginal length
for i=1:4
    [mod_t, mod_y, mod_v, mod_h] = modified_euler_bungee(60, 10000,
 9.8, 0.01125, 1.125, bungee_lengths(i));
    plot(mod_t,74-mod_y); % Distance over time
end
xlabel("Time");
ylabel("Height of Jumper");
title("Height Difference of Varied Bungee Lengths");
legend('25m', '35m', '40m', '46.5m')

% Calculate the max G force experienced by the jumper during the jump
g_force = max_acceleration(mod_t, mod_v);
disp("The jumper experiences "+g_force+"G's during the jump with a
 46.5m bungee cord, which is unsafe.");



% Altering the spring constant of the bungee cord
figure;
hold on;
spring_constants = [90/80 70/80 50/80 32.5/80];  % in N/m

% Graph the bungee spring constants verus the original
for i=1:4
    [mod_t, mod_y, mod_v, mod_h] = modified_euler_bungee(60, 10000,
 9.8, 0.01125, spring_constants(i), 25);
    plot(mod_t,74-mod_y); % Distance over time
end
% Calculate the G force experienced by the jumper
g_force = max_acceleration(mod_t, mod_v);
disp("The jumper experiences "+g_force+"G's during the jump with a
 32.5N/m bungee cord which is safe, but the jumper only experiences 6
 bounces in one minute");

xlabel("Time");
ylabel("Height of Jumper");
title("Height Difference of Varied Bungee Spring Constants");
legend('90N/m', '70N/m', '50N/m', '32.5N/m')
```

```
% Altering the length and spring constant of the bungee cord to
 achieve a
% safe G force and achive close to 10 bounces
figure;
hold on;
bungee_lengths = [25 43]; % in metres
spring_constants = [90/80 73/80]; % in N/m

% Graph the original ungee cord parameters against the new "water
 touch"
% cord parameters
for i=1:2
    [mod_t, mod_y, mod_v, mod_h] = modified_euler_bungee(60, 10000,
 9.8, 0.01125, spring_constants(i), bungee_lengths(i));
    plot(mod_t,74-mod_y); % Distance over time
end

% Calculate the G force experienced by the jumper
g_force = max_acceleration(mod_t, mod_v);
disp("The jumper experiences "+g_force+"G's during the jump with a
 43m cord at 73N/m which is safe and the jumper still experiences 9
 bounces");
xlabel("Time");
ylabel("Height of Jumper");
title("Height Difference of Varied Bungee Lengths and Spring
 Constants");
legend('25m @ 90N/m', '43m @ 73N/m')

% All bungee cord parameters were found by graphing arrays of values
 and
% investigating the values of length and spring to achieve the desired
% outcome.
```

Task 6
How close does the jumper get to the water?
The jumper gets 23.9898 metres from the water on the first bounce.
Calculate the factors that would need to be changed to achive a "water touch" on the first bounce of the jump
The jumper experiences 2.1755G's during the jump with a 46.5m bungee cord, which is unsafe.
The jumper experiences 1.0492G's during the jump with a 32.5N/m bungee cord which is safe, but the jumper only experiences 6 bounces in one minute
The jumper experiences 1.8919G's during the jump with a 43m cord at 73N/m which is safe and the jumper still experiences 9 bounces


Height Difference of Varied Bungee Lengths
25m
35m
40m
46.5m
Height of Jumper
Time
80
70
60
50
40
30
20
10
0
-10
0
10
20
30
40
50
60


Height Difference of Varied Bungee Spring Constants
90N/m
70N/m
50N/m
32.5N/m
Height of Jumper
Time
80
70
60
50
40
30
20
10
0
-10
0
10
20
30
40
50
60



6 Conclusion

To sum up, this bungee project for supporting the feasibility of the proposal is realistic. As modified numerical method which is based on Euler's method, the standard bungee jump will consist of 10 bounces during 60 seconds and the maximum speed of the jumper will be around 20m/s after approximately 2.6 seconds. The jumper's experience for maximum acceleration during the bungee is around -18.34m/s ^ 2 which is almost reached to 2g and the jumper will travel a total distance of around 281m while bungee jumping.

For the camera, it should be triggered after around 3.3 seconds from the moment the bungee jump starts for capturing a great photo of the jumper.

Depending on the bouncing, the distance of jumper will be changed immediately. On the first bounce, the jumper will get around 24 metres from the water. If the client wants to have the 'water touch' option, they need to change some of factors for bungee. However, if the bungee lengths are changed to 46.5m, the jumper experiences over 2g, which is unsafe. However, if the other spring constants of the bungee rope are changed along with the length, it remains safe and still achieves just under bounces 10 in 60 seconds. The proposal of this bungee is essential work for ensuring the safety of the jumper and providing and exciting experience.

Published with MATLAB® R2019a