Deven Hurt, Raul Jordan, Ha Le, Maille Radford, PS12a Final Project

Table of Contents

Part 1a

```
close all
clear all
clc
% First we define all the constants necessary for our calculations.
% Here C = Coefficient of Drag of a ski jumper
% L = height of a person divided by two due to crouching
% R = average width of shoulders is 18 inches, so we divide it by two.
% A = Cross sectional area
% p = density
C = 1.1;
L = 1.8./2;
R = 0.23;
A = 2*R*L;
p = 1.2;

% mu = coefficient of static friction for ski jumper against the snow
% m = average mass of ski jumper in kg
% g = gravitational acceleration m/s^2
% h = height of ski jumper
% dt = differential in time
% N = number of iterations
mu = 0.05;
g = 9.8;
m = 70;
h = 1.8;
dt = 0.01;
N = 100;

% Defines the olympic standard x distance of a ski jump in meters
x_distance = 101.6;

% Defines the two different thetas we will be looping over to discover the
% best value. The first theta is the slope of the mountain as the skiier
% goes down, while the second theta is the slope as he/she comes back up
theta = linspace(30,40, 100);
theta2 = linspace(10,20,100);
```

```
% Initialize the best indeces for the values of theta and theta2,
% respectively as well as our best value for x
bestx = 0;
bestn = 1;
bestm = 1;

% Now we loop over both linspaces for theta
for n = 1:length(theta)
    for j = 1:length(theta2)

    % We calculate the height trigonometrically
    h = x_distance*sind(theta(n));

    % We obtain the first velocity calculation in our model, which will
    % take into account drag and the normal force of the mountain on the
    % skiier as explained in our model.
    v1 = sqrt((2*m*g*h - 2*mu*m*g*cosd(theta(n))*(h/sind(theta(n)))) ...
        /(m+p*A*C*(h/sind(theta(n)))));
```

Part 1b

```
    % L is the integral from 0 to any constant we want that we'll solve
    % for of 20/sqrt(1+4x^2)... The constant is 20/tan(theta) and we assign
    % the second height in our model to be 20 meters arbitrarily

    h2 = 20;

    % We define a function handle to perform our integration
    fun = @(q) h2./sqrt(1+4.*q.^2);

    % We obtain the length of the parabola by integrating and incorporate
    % this in our second velocity calculation
    L_par = integral(fun, 0, h2/tan(theta(n)));

    % We incorporate this length in our other velocity calculation which we
    % find using conservation of energy as explained in our model
    v2 = sqrt((2*m*g*h2 - 2*mu*m*g*cosd(theta(n))*...
        abs(L_par))./(m+p*A*C*L_par)) + v1;
```

Part 1 c

```
    % We now find the this velocity calculation as explained in our model
    % as the skiier takes off from the mountain again using conservation of
    % momentum and the second height we abritrarily defined for the
    % mountain a few lines above. We define this as a variable named "num"
    % for numerator to make our code look cleaner.
    num = m*v2^2 - 2*m*g*h2 - C*p*A*v2^2*(h2/sind(theta2(m))) ...
        - 2*mu*m*g*cosd(theta(n))*(h2./sind(theta(n)));

    % Now we actually compute this thirs velocity as the square root of the
    % numerator defined above divided by the mass of the skiier from
    % conservation of energy.
    v3 = sqrt(abs(num)/m);
```

```
% Using the BSA equation, except divided by 2, we obtain the cross
% sectional area
A = (sqrt(m*h/3600))/2;

% Now we find drag from the equation
D = (p*C*A)/2;


% define initial conditions
x0 = 0;
y0 = 5;
vx0 = v3*cosd(theta2(j));
vy0 = v3*sind(theta2(j));


% define data arrays for evolving our system
x = zeros(1,N);
x(n) = x0;
y = zeros(1,N);
y(n) = y0;
vx = zeros(N);
vx(1) = vx0;
vy = zeros(N);
vy(1) = vy0;

i = 1;
j = 1;

% Using evolver, we now step N times over these arrays and set the next
% element using simple kinematics
while i < N
    ax = -(D/m)*v3*vx(i);
    vx(i+1) = vx(i) + ax*dt;
    x(i+1) = x(i) + vx(i)*dt + 0.5*ax*dt^2;
    ay = -g - ((D/m)*v3*vy(i));
    vy(i+1) = vy(i) + ay*dt;
    y(i+1) = y(i) + vy(i)*dt + 0.5*ay*dt^2;

    % We now use this to calculate our best indeces for x, theta and
    % theta 2 by optimizing the system
    if y(i+1) < 0
         i = N;
    else
         i = i + 1;
    end

    if (x(i) > bestx)
        bestx = x(i);
        bestn = n;
    end

end
```

```
    end
end

% From our system evolver, we use the best indeces we calculated before and
% now obtain our best values which match what we were expecting before.
best_theta = theta(bestn);
best_theta2 = theta2(bestn);

fprintf('The best value for theta is%7.3f and for theta2 is%7.3f\n',...
    best_theta,best_theta2);

% Now we obtain the value of d matching the bestx we found from our
% looping. From our model, we see that d can be expressed as
% cosd(theta) = bestx/d so d = bestx / cosd(theta)
best_d = bestx / cosd(best_theta);
fprintf('The value of d is%7.3f\n', best_d);

figure;
grid on;
plot(x,y);
title('Trajectory After Take-Off');
xlabel('x position');
ylabel('y position');

figure;
hold on;
plot(theta,vx);
plot(theta2,vx);
legend('theta 1','theta 2');
title('Horizontal Velocity vs. Theta Values');
xlabel('Theta in Degrees');
ylabel('Velocity in m/s');
```

```
The best value for theta is 31.414 and for theta2 is 11.414
The value of d is 68.964
```

Published with MATLAB® R2014b