PSBC Project 2: Projectile motion

April 3, 2020

Introduction

Vectors in this report will be represented bold with a underline, for example $\underline{\boldsymbol{v}}$. Also we will be using radians to express angles.

Furthermore, any reference with prefix "App-", refers to a piece of code given in the appendix. An example of this is App-(1).

1 Question 1

For this question we are given that a projectile has been fired at $450\,\mathrm{m\,s^{-1}}$ at some angle θ. We want to find the maximum horizontal distance, with it's associated angled, of the projectile. The only forces that are applied to the particle are assumed to be gravity (with the gravitational constant $g = 9.8\,\mathrm{m\,s^{-2}}$), and the friction force $F = K|\underline{\boldsymbol{v}}|^2$ opposite to the direction of the velocity $\underline{\boldsymbol{v}}$ of the projectile, with constant $K = 0.00002\,\mathrm{kg\,m^{-1}}$. The mass of the projectile is given as $6\,\mathrm{kg}$.

1.1 Calculating the first order system

For this question, let $\underline{\boldsymbol{r}} = (x, y)$ be the position vector of the projectile. Now if we look at the vector of the friction force opposite to the direction of the velocity $\underline{\boldsymbol{v}}$, we would get:

$$\underline{\boldsymbol{F}} = K|\underline{\boldsymbol{v}}|^2(-\underline{\boldsymbol{e}}_r) = K|\underline{\boldsymbol{v}}|^2\left(-\frac{1}{|\underline{\boldsymbol{v}}|}\underline{\boldsymbol{v}}\right) = -K|\underline{\boldsymbol{v}}|\underline{\boldsymbol{v}}$$

where $\underline{\boldsymbol{e_r}}$ is the unit direction vector of the projectile. Using this, we now consider the forces acting on the particle in the x and y axis:

$$m\frac{d^2r_x}{dt^2} = \text{Friction} = -K|\underline{\boldsymbol{v}}|\frac{dr_x}{dt}$$
$$m\frac{d^2r_y}{dt^2} = \text{Weight} + \text{Friction} = -mg - K|\underline{\boldsymbol{v}}|\frac{dr_y}{dt}$$

where r_x and r_y are respective functions of t of their vector entries. Equivalently, we can write this as:

$$\frac{d^2r_x}{dt^2} = -\frac{K}{m}|\underline{\boldsymbol{v}}|\frac{dr_x}{dt}$$
$$\frac{d^2r_y}{dt^2} = -g - \frac{K}{m}|\underline{\boldsymbol{v}}|\frac{dr_y}{dt}$$

The initial conditions for this problem would be given by:

$$r_x(0) = 0, \qquad r_x'(0) = 450\cos\left(\frac{\pi}{4}\right),$$
$$r_y(0) = 0, \qquad r_y'(0) = 450\sin\left(\frac{\pi}{4}\right)$$

We now want to turn this problem into an equivalent first order system. Let $a = r_x'$ and $b = r_y'$. Then we can write the first order system of our problem as:

$$\begin{cases} r_x' = a \\ r_y' = b \\ a' = -\frac{K}{m}|\underline{\boldsymbol{v}}|a = -\frac{K}{m}a\sqrt{a^2+b^2} \\ b' = -g - \frac{K}{m}|\underline{\boldsymbol{v}}|b = -g - \frac{K}{m}b\sqrt{a^2+b^2} \end{cases}$$

with initial conditions:

$$(x_0, y_0, a_0, b_0) = \left(0, 0, 450\cos\left(\frac{\pi}{4}\right), 450\sin\left(\frac{\pi}{4}\right)\right)$$

1.2 Solving the first order system

To solve this first order system, create a function "projectileODE", which takes some angle θ and a event function, and returns a vector and matrix of the time and path respectively corresponding to the particle during flight. Note that the path matrix is constructed such that a row is in the form (r_x, r_y, a, b). The code for this function is given by App-(1).

As we have let the interval of time for the ODE be set between 0 and infinity, we use the event function to stop the ode45 when we have hit the "ground", that is when $y = 0$. The event function for this question we will use is given by App-(2).

The last function we will create, is simply a function to return the horizontal distance after the projectile has landed called "maxHori", for a given angle θ, using the created functions. This function is given by App-(3).

Finally, to find the maximum horizontal distance and the associated angle, we use the function "fminbnd", to minimise the horizontal distances between the associated angles $[0, \frac{\pi}{2}]$. As this function only minimises, then the function argument input for "fminbnd" will be the negative distance. The reason for this is that the maximum horizontal distance, is equivalent to the minimum negative maximum horizontal distance. Note that after we get the distance, we must take the negative of it, so that we get the distance as positive. Altogether, this is given by:

Main code

```
  [theta,distance] = fminbnd(@(theta)-maxHori(theta), 0, pi/2);
  theta
  distance = -distance
```

Which give the output of this as:

Console output for question 1

```
theta =
   0.778422268652416

distance =
      1.961776755647775e+04
```

So to get the maximum horizontal distance from the origin, we must fire at an angle of "theta", in which the horizontal distance travelled is about 19618 m when rounded.

2 Question 2

For this question, we are under similar constraints as in question 1 for the projectile, however we want to hit a target 15000 m horizontally away from the origin. Furthermore, 12000 m horizontally away from the origin are interceptors, of vertical length 1000 m, travelling vertically upwards at 100 $\mathrm{m\,s^{-1}}$. These interceptors are launched every 20 seconds from the ground. We want to find all launch angles and firing times (starting from $t = 0$), such that the cannonball at the origin can hit the target.

We can split this problem into two questions. What angles have a horizontal distance of 15000 m from the origin when fired. And what time frames does the projectile not get blocked by the interceptors, for each angle.

2.1 The angles associated with a horizontal distance of 15000 m

From question 1 we know that 15000 m is not the maximum horizontal distance from the origin. Furthermore, as we increase the angle θ from 0, towards the angle associated with the maximum horizontal distance from the origin, we get the horizontal distance from the origin increasing. Oppositely, as we increase the angle θ, past the angle associated with the maximum horizontal distance towards $\frac{\pi}{2}$, we get the horizontal distance from the origin decreasing. Therefore we can say that there can only exist two potential angles which will give us a horizontal distance from the origin as 15000 m. Hence, there must exist an angle in the boundary $(0, 0.778422268652416)$ and an angle in $(0.778422268652416, \frac{\pi}{2})$.

Now using the same functions given in questions 1, we can create a new function called "dist", which takes some angle θ and returns the "maxDist" function minus 15000. Then we can use the function "fzero", to look for the angles which make the function "dist" equal to zero, between the boundaries given before. The code for this is given by:

Main code for finding potential angles

```
% Find the exact angle associated with the max horizontal distance.
maxAngle = fminbnd(@(theta)-maxHori(theta), 0, pi/2);
% Function which just takes 15000 from the horizontal distance given by
% maxHori.
dist = @(theta) maxHori(theta) - 15000;
% Find the two angles, between the given boundaries.
angle1 = fzero(dist, [0,maxAngle])
angle2 = fzero(dist, [maxAngle,pi/2])
```

This gives the outputted angles as:

Output for potential angles

```
angle1 =
  0.425365590818037

angle2 =
  1.133637131955319
```

2.2 Finding the time frames for each angle

The first thing we want to calculate is the height and time of the projectile for each angle, when it has reached a horizontal distance of 12000 m from the origin. We do this by creating a function called “vert”, which takes an angle θ and a horizontal distance x, and returns the time and vertical distance of the projectile when it has launched at angle θ and reached a horizontal distance x from the origin. We do this by simply using the same function from question 1, “projectileODE”, where we change the event function passed in, such that it stops the ODE when it has reached a horizontal distance of x meters. The code for the function “vert” is given by App-(4), and the event function code is given by App-(5). The code called when using the vert function with our two calculated angles is given as:

Using vert with the two angles

```
  [time1, height1] = vert(angle1, 12000)
  [time2, height2] = vert(angle2, 12000)
```

This gives the output as:

Output of vert

```
time1 =
  29.890726332174467

height1 =
     1.117127012875581e+03

time2 =
  65.018575705556188

height2 =
     5.318965428650329e+03
```

What we should note about the interceptors, is that as each interceptor moves at exactly $100\,\mathrm{m\,s^{-1}}$ constantly, with a new interceptor being launched every 20 seconds, then a “pattern” emerges of the interceptors below the highest interceptor, which repeats every 2000 m from the ground to the highest interceptor. So, every 20 seconds the pattern returns to the pattern at $t = 0$. So we want to find the initial time frame for each angle, as after that time frame the ”pattern” of the intercepts will repeat at regular intervals. Using this fact then we are only interested in “time $\pmod{20}$”. Furthermore, as the “pattern” repeats every 2000 m, then we are only interested in “height $\pmod{2000}$”. Note, that we can only make these assumptions because the speed of the interceptors and launch time intervals stay constant for all time. Also note that at $t = 65$ the highest interceptor has a height greater than 9000. We find these values using the following code:

Modulus of height and time

```
  % The pattern of interceptors reset every 20 seconds, and every 2000 meters
  % so take mod of the times and heights.
  mtime1 = mod(time1,20)
  mtime2 = mod(time2,20)
  mheight1 = mod(height1,2000)
  mheight2 = mod(height2,2000)
```

Giving the output as:

Output of modulus

```
  mtime1 =
     9.890726332174467

  mtime2 =
     5.018575705556188
```

```

mheight1 =
     1.117127012875581e+03

mheight2 =
     1.318965428650329e+03
```

We want to find now, when the projectile is fired at $t = 0$ and arrives at $x = 12000\,\text{m}$, will it hit a interceptor? We can model the top and bottom of the "gap" between the two interceptors by:

$$g^{\text{Top}}(t) = 100t + 1000$$
$$g^{\text{Bottom}}(t) = 100t$$

Now using matlab and find the distance travelled by the gap after the modulus times, using the code:

Gap motion

```
% Now calculate the positions of the "open interval" after m time.
gap = @(t,offset) 100*t+offset; % Quick function to save time
gapT1 = gap(mtime1, 1000)
gapT2 = gap(mtime2, 1000)
gapB1 = gap(mtime1, 0)
gapB2 = gap(mtime2, 0)
```

Giving the following output:

Gap motion output

```
gapT1 =
     1.989072633217447e+03

gapT2 =
     1.501857570555619e+03

gapB1 =
     9.890726332174467e+02

gapB2 =
     5.018575705556188e+02
```

We can see that $\text{mheight1} \in [\text{gapB1}, \text{gapT1}]$ and $\text{mheight2} \in [\text{gapB2}, \text{gapT2}]$. So at $t = 0$, the projectile will make it past the interceptors for both angles. Now we want to calculate the delayed time we can shoot, that is for a constant τ then τ is the largest value such that the projectile can make it past without hitting the top of the interceptor below. We can find this by using the following equation:

$$Time = \frac{\text{Difference in distance}}{\text{Speed}}$$

This is implemented using the following code:

Remaining time

```
% Both projectials are in the gap, so calculate the delay time before
% interceptor reaches projectile.
dtime1 = (mheight1-gapB1)/100
dtime2 = (mheight2-gapB2)/100
```

With the following output:

Remaining time output

```
dtime1 =
   1.280543796581343

dtime2 =
   8.171078580947103
```

So the initial time interval for angles1 is [0, dtime1], and for angle2 is [0, dtime2]. Now, we use the fact that each interceptor is 1000 m travelling at 100 m s^{-1}. So for the bottom of the interceptor to travel past the height of the projectile, it will take:

$$\text{time} = \frac{\text{Length}}{\text{Speed}} = \frac{1000}{100} = 10\,\text{s}$$

Then as the gap is the same length and travels at the same speed as the interceptor, then it would also take 10 s for the end of the gap to reach the height of the projectile. Hence, we have a repeating pattern after the initial time interval for each angle. So altogether for angle1, the associated firing time intervals are given as:

$$[0, \text{dtime1}], [20n + 10 + \text{dtime1}, 20n + 20 + \text{dtime1}], \quad \text{for} \quad n = 0, 1, 2, \ldots$$

Similarly, for angle2, the associated firing time intervals are given as:

$$[0, \text{dtime2}], [20n + 10 + \text{dtime2}, 20n + 20 + \text{dtime2}], \quad \text{for} \quad n = 0, 1, 2, \ldots$$

The whole of the main code used for question 2 is given in App-(6).

Appendix

Listing 1: projectileODE function

```
function [time, path] = projectileODE(theta, event)
%% Returns the time and path vector for an ODE under gravity and air resistance.

% Check for angle to be zero, in which case ignore as otherwise we get
% errors with ode45.
if(theta==0)
  time = [0];
  path = [0 0 0 0];
 return
end
% Set constants to be used, given by the question.
k= 0.00002;
m = 6;
g = 9.8;
v0 = 450;
% Set the options of the ODE. We bound the "event" function as an
% event to look for during the numerical integration (this allows us to use
% this function with different events depending on circumstance).
options = odeset('Events',@(t,z)event(t,z), 'refine',5,'AbsTol',1e-4,'MaxStep',1e-3);
% We set up out ode variable to be integrated according to our first order
% system. Note that z = (x y a b).
myode = @(t,z) [z(3); z(4); ...
     -(k/m)*sqrt(z(3)^2+z(4)^2)*z(3); ...
  -g-((k/m)*sqrt(z(3)^2+z(4)^2)*z(4))];
% Now solve the ODE. Note we set the time span between 0 and infinite,
% since we want the "event" to stop the ODE when it is triggered.
% Also we use symbolic and double transformations when theta=pi/2,
% we do not get the correct initial conditions, leading to a incorrect
% answer (Note that this method sacrifices performance for precision).
[time, path] = ode45(@(t,z)myode(t,z), [0 inf], [0 0 v0*double(cos(sym(theta))) v0*double(sin(sym(
    theta)))], options);
```

Listing 2: groundEvent function

```
function [value, isterminal, direction] = groundEvent(t, z)
%% An event function, to stop the ODE algorithm after the projectile hits the "ground", which is
    in this case when y=0.

% Trigger the event when "value" is zero.
value = z(2);

```

```
% Stop ODE algorithm when the event is triggered.
isterminal = 1;

% Used to only locate zeros when the event function is decrease
% (that is when the projectile is coming down to hit the ground).
direction = -1;
```

Listing 3: maxHori function

```
function distance = maxHori(theta)
%% Returns the horizontal distance travelled by a projectile under gravity and air resistance for
    some launch angle theta.

% Get the time vector and path matrix for the projectile.
[time, path] = projectileODE(theta, @groundEvent);
% Return the horizontal distance.
distance = path(end,1);
```

Listing 4: vert function

```
function [time,distance] = vert(theta, x)
%% Returns the time and vertical distance of a projectile under gravity and air resistance for
    some launch angle theta after it has travelled x meters.
% Get the time and path vectors for the projectile.
[time, path] = projectileODE(theta, @(t,z)horiEvent(t,z,x));
% Return the horizontal distance.
distance = path(end,2);
time = time(end);
```

Listing 5: horiEvent function

```
function [value , isterminal , direction] = horiEvent(t,z,x)
%% An event function to stop the ODE algorithm after the projectile reaches a horizontal distance
    x.

% Trigger the event when "value" is zero. In this case when the difference
% between distance x and the current horizontal distance has reached 0.
value = x-z(1);

% Stop ODE algorithm when the event is triggered.
isterminal = 1;

% Used to only locate zeros when the event function is decrease
% (that is when the projectile is coming down to hit the ground).
direction = -1;
```

Listing 6: Question 2 main code

```
% Find the exact angle associated with the max horizontal distance.
maxAngle = fminbnd(@(theta)-maxHori(theta), 0, pi/2);
% Function which just takes 15000 from the horizontal distance given by
% maxHori.
dist = @(theta) maxHori(theta) - 15000;
% Find the two angles, between the given boundaries.
angle1 = fzero(dist, [0,maxAngle])
angle2 = fzero(dist, [maxAngle,pi/2])

[time1, height1] = vert(angle1, 12000)
[time2, height2] = vert(angle2, 12000)

% The pattern of interceptors reset every 20 seconds, and every 2000 meters
% so take mod of the times and heights.
mtime1 = mod(time1,20)
mtime2 = mod(time2,20)
mheight1 = mod(height1,2000)
mheight2 = mod(height2,2000)
% Now calculate the positions of the "open interval" after m time.
gap = @(t,offset) 100*t+offset; % Quick function to save time
gapT1 = gap(mtime1, 1000)
gapT2 = gap(mtime2, 1000)
gapB1 = gap(mtime1, 0)
gapB2 = gap(mtime2, 0)
% Both projectials are in the gap, so calculate the delay time before
% interceptor reaches projectile.
```

```
dtime1 = (mheight1-gapB1)/100
dtime2 = (mheight2-gapB2)/100
```