MATH-36032 PSBC Project 2: Projectile Motion

We aim to solve two problems related to projectile motion. The projected object is a cannonball of mass 6kg, and is fired with an initial speed of 450ms^{-1}, at an angle θ from the ground, as shown in Figure 1.

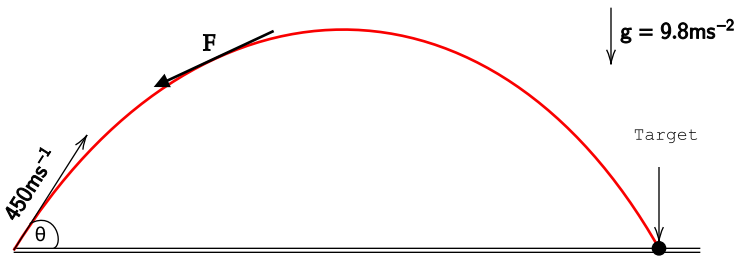

Figure 1: A cannonball fired from the origin to destroy a target on the same horizontal ground.

We assume only the following two forces act on the cannonball:

- The gravity: $\vec{g} = \begin{pmatrix} 0 \\ -9.8 \end{pmatrix}$ with gravitational constant assumed to be 9.8ms^{-2}.
- The friction force: $\vec{F} = K|\vec{v}|^2$ acting in the opposite direction to the velocity $\vec{v}$ and proportional to the speed squared $|\vec{v}|^2$, with constant $K = 0.00002\text{kgm}^{-1}$. Note that this can be written as

$$\vec{F} = -K|\vec{v}|\vec{v}.$$

We consider Newton's Second Law of Motion to formulate a differential equation system that will model this motion.

Newton's Second Law of Motion states that the sum of forces $\vec{F}$, acting on an object is equal to the mass, m, multiplied by the acceleration, $\vec{a}$, of the object. i.e.

$$\vec{F} = m\vec{a}.$$

Let $\vec{s}(t) = \begin{pmatrix} s_x(t) \\ s_y(t) \end{pmatrix}$ be the displacement of the cannonball from the origin at a given time t. Since $\vec{a} = \ddot{\vec{s}}$, we can write the following using Newton's second law of motion.

$$m\frac{d^2\vec{s}}{dt^2} = m\vec{g} - K|\vec{v}|\vec{v}. \tag{1}$$

Therefore, rearranging for acceleration we have a system of second order ordinary differential equations which we can solve for $\vec{s}(t)$.

$$\begin{bmatrix} \ddot{s}_x(t) \\ \ddot{s}_y(t) \end{bmatrix} = \begin{bmatrix} 0 \\ -9.8 \end{bmatrix} - \frac{K}{m}\sqrt{\dot{s}_x(t)^2 + \dot{s}_y(t)^2}\begin{bmatrix} \dot{s}_x(t) \\ \dot{s}_y(t) \end{bmatrix}, \tag{2}$$

with initial conditions expressed in terms of the initial speed and launch angle θ as

$$\vec{s}(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \text{and} \quad \dot{\vec{s}}(0) = \begin{pmatrix} 450\cos\theta \\ 450\sin\theta \end{pmatrix}. \tag{3}$$

Hence, we have a pair of initial value problems (IVPs) that can be solved using MATLAB.

1. Maximum horizontal distance

Problem

It is known that the maximum horizontal distance can be calculated in the absence of friction force explicitly, given by the launch angle $\theta = \pi/4$.

In this problem we want to find the maximum horizontal distance that can be achieved for the system above with the friction force, $\vec{F} = K|\vec{v}|^2$, and the associated launch angle θ.

1.1 Approach

Suppose we have a function `Distance(theta)` which returns the horizontal distance travelled by the cannonball for the given `theta`. Maximising this function for a given range of angles by using the `fminbnd` function will yield the maximum horizontal distance along with the associated angle θ.

1.1.1 Observations

Observation I: To implement the function `Distance(theta)` we must solve the initial value problem described by (2) and (3) above. This can be done by using the `ode45` solver. However, `ode45` only solves first order ODEs which are in the form $\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x})$. Therefore, we must reduce the second order system into a first order system. We do this by using the following substitution:

$$\underbrace{\begin{bmatrix} \dot{s}_1(t) \\ \dot{s}_2(t) \\ \dot{s}_3(t) \\ \dot{s}_4(t) \end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} s_2(t) \\ -\dfrac{K}{m} s_2(t)\sqrt{s_2(t)^2 + s_4(t)^2} \\ s_4(t) \\ -9.8 - \dfrac{K}{m} s_4(t)\sqrt{s_2(t)^2 + s_4(t)^2} \end{bmatrix}}_{\boldsymbol{f}(t,\boldsymbol{x})}, \tag{1.1}$$

$$\text{where} \quad s_1(t) = s_x(t), \quad s_2(t) = \dot{s}_x(t), \quad s_3(t) = s_y(t), \quad s_4(t) = \dot{s}_y(t).$$

Therefore, the initial conditions become

$$s_1(0) = 0, \quad s_2(0) = 450\cos\theta, \quad s_3(0) = 0, \quad s_4(0) = 450\sin\theta. \tag{1.2}$$

Observation II: The `ode45` function requires a *span* (range of values for independent variable t) over which to find the solution. To obtain a solution for when the the cannonball reaches the ground, we require the time, t, it takes for the cannonball to reach the ground. However, we have no information for this exact time.

Instead, we consider the system when there is no resistance force $\vec{F}$ and the only force acting on the cannonball is gravity. Considering the "SUVAT" in the vertical component, $s_y = v_0 t\sin\theta - \frac{1}{2}gt^2$, we can find the time when the cannonball reaches the ground by setting $s_y = 0$. This gives

$$t = \frac{2v_0\sin\theta}{g}, \tag{1.3}$$

where v_0 is the initial speed and θ is the launch angle.

Since air resistance works against the motion of the cannonball, it is clear that any time of flight which includes the air resistance will be lower. Hence, (1.3) can be used as the upper bound for the span.

Observation III: As discussed in Observation II, the span is an upper bound for the values of t. Therefore, the solution will have values for t greater than the time taken for the cannonball to reach the ground. To avoid this, we can add an event function in the `options` argument of `ode45`. This event will simply stop solving the ODE once the cannonball touches the ground. i.e. $s_y = 0$.

Observation IV: `fminbnd(f,x0,x1)` finds the *minimum* value of a function `f` over the range `[x0,x1]`. However, we aim to find the maximum value of the function `Distance(theta)`. In order to achieve this we can we can use `fminbnd` on `-Distance(theta)`, i.e. the negative values for the distances, to find the maximum positive distance.

1.2 Implementation

First, we implement a function called `Projectile(theta)`, which solves initial value problem described in (1.1) and (1.2) for a given `theta` using `ode45`.

```
function [t, x] = Projectile(theta, events)
% Projectile solves the ODE system for a given firing angle.
m = 6;
K = 2e-5;
g = 9.8;
v_0 = 450;
span = [0, (2*v_0*sin(theta))/g];

opts = odeset('Refine', 32, 'Events', events);

% ODE system described in (1.1)
f = @(t, s) [s(2); (-K/m)*sqrt(s(2)^2+s(4)^2)*s(2); s(4); -g - (K/m)*sqrt(s(2)^2 + s(4)^2)*s(4)];

% Initial conditions described in (1.2)
ic = [0; v_0*cos(theta); 0; v_0*sin(theta)];

% Solve the ODE numerically
[t, x] = ode45(f, span, ic, opts);
end
```

Listing 1: Solves the IVP described in (1.1) and (1.2).

Furthermore, we define the events function discussed in Observation III as follows:

```
function [value, isterminal, direction] = GroundEvent(~, x)

% The event when vertical displacement = 0. Terminates the ODE.
value(1) = x(3);
isterminal(1) = 1;
direction(1) = -1;
```

Listing 2: `GroundEvent` function described in Observation III.

This will halt `ode45` from solving further, once the vertical displacement, $s_y = 0$. Now consider the `Distance(theta)` function which calls `Projectile(theta)` and returns the horizontal distance reached by the cannonball.

```
function dist = Distance(theta)
% Distance solves coupled ODE system for given launch angle and returns the
% distance

% Solve the ODE numerically
[~, x] = Projectile(theta, @GroundEvent);
dist = x(end,1);
end
```

Listing 3: Returns the distance travelled for a given `theta`.

Finally, we have the function `FindMaxDistTheta` which uses `fminbnd` to find the maximum distance and the corresponding angle. This prints the maximum distance along with the launch angle in both degrees and radians.

```
function FindMaxDistTheta
% FindMaxDistTheta find the maximum horizontal distance & the launch angle.

[theta, dist] = fminbnd(@(theta) -Distance(theta), 0, pi/2);

fprintf('Max horizontal distance: %s metres\n', num2str(-dist));
fprintf('Launch angle: %.5f radians (5dp) | %.2f degrees (2dp)\n', theta, rad2deg(theta));
end
```

Listing 4: Finds the maximum distance and the associated angle.

1.3 Result

Below is the result of running `FindMaxDistTheta`:

```
>> FindMaxDistTheta

Max horizontal distance: 19617.7676 metres
Launch angle: 0.77842 radians (5dp) | 44.60 degrees (2dp)
```

Therefore, we can conclude that a maximum distance of 19617.77m (2dp) is achieved with a firing angle of 44.60° (2dp) or equivalently 0.77842 radians (5dp). The plot of the projectile motion is depicted below.

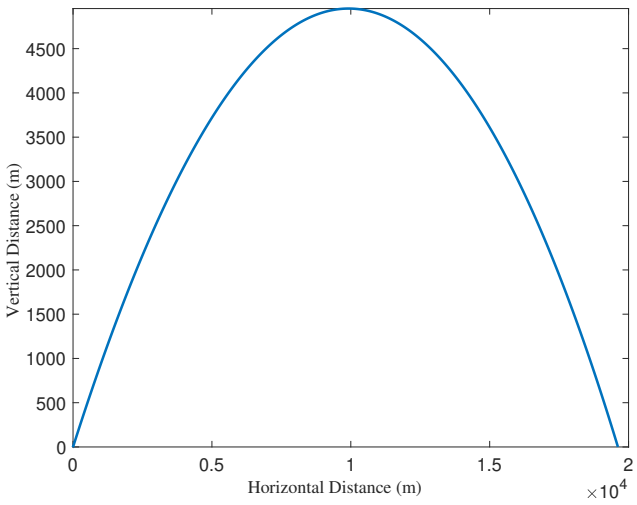

Figure 1.1: Projectile trajectory for maximum distance fired at 44.60°.

2. Firing at a target

Problem

We wish to fire cannonballs to destroy a target at 15000 metres from the origin. However, at 12000m there are interceptors which move vertically in an attempt to block the cannonballs. Each interceptor is a 1000m long, moves vertically upwards at a constant velocity 100ms^{-1}, and is launched every 20 seconds. Figure 2.1 shows the initial configuration at $t = 0$. An interceptor is released straight away from $(12000, 0)$ as t becomes greater than zero. We can conclude from the speed and the time interval that there will be a distance of 1000 metres between each interceptor because each one will travel 2000 metres before the next one is released (see Figure 2.1). For simplicity, we assume the size of the cannonballs and the interceptors is negligible.

The aim is to find all possible firing angles and their associated firing times for which we can hit the target successfully.

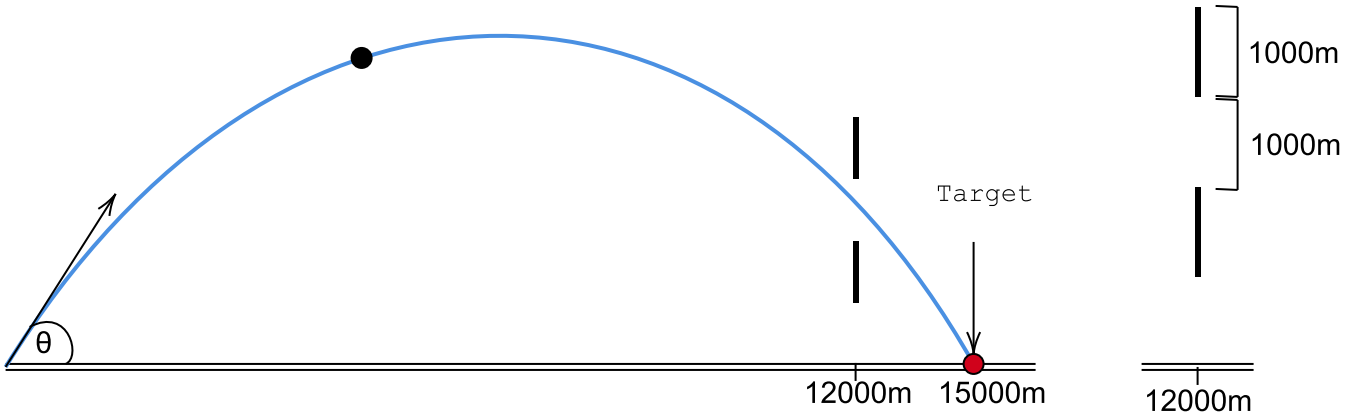

Figure 2.1: Setup: Fire cannons at the target avoiding the interceptors.

2.1 Approach

Suppose we have a function `FiringAngles(target)`, that finds the possible firing angles for a given `target`. This can be used to obtain all the possible firing angles that will successfully hit our target. Note, that the firing angles are independent of the firing times in order to reach the target. The firing times will only be used to avoid the interceptors.

Now suppose, we have a function `TimeDelay(angles)`, which for a firing angle returns one time delay such that the cannonball passes the interceptor just below the interceptor. i.e. the bottom of the interceptor is at h_θ, where h_θ is the height of the cannonball at distance 12000 metres from the origin, launched at angle θ.

Combining the two functions above will allow us find the solution to the problem.

2.1.1 Observations

Observation I: In order to implement `FiringAngles(target)`, we can make use of the `fzero` function and find the angles for which `Distance(theta) - target` is equal to zero. Note that there are two firing angles that can hit a given target. To use `fzero(f,x0)`, we are required to pass in `x0` which is an initial guess (which can be a range of values). To find the appropriate initial values for each angle, we plot the following graph.

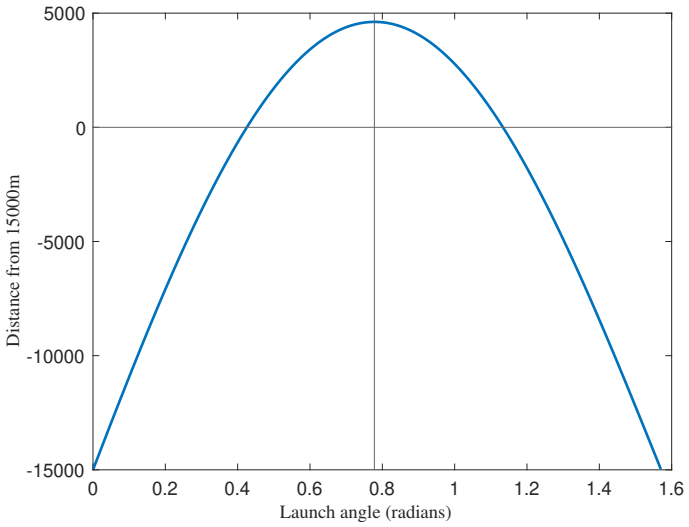

Figure 2.2: Plot of `Distance(theta)-15000`

It is clear that there are two zeros for the `theta` near 0.4 and 1.2. These are on either side of the angle $= 0.77842$ (corresponding to maximum horizontal distance in part 1). Hence, we can use the initial range $(0, 0.77842)$ and $(0.77842, \pi/2)$ for the two launch angles.

Observation II: The interceptors move at a constant velocity of 100ms^{-1}. So, in 20 seconds an interceptor travels 2000 metres. Observing the setup in Figure 2.1, we can see that the interceptors will be in the "same" position every 20 seconds. There is also an empty space of 1000 metres between each interceptor, which lasts for 10 seconds.

We want to find one time delay t_d (delay after $t = 0$), such that the cannonball is just below the interceptor (as mentioned earlier in the approach). The height of the point corresponding to the bottom of the first interceptor for a given time t, can be expressed as

$$100t + 1000.$$

Now, let h_θ be the height of the cannonball at 12000 metres from the origin and let t_θ be the time it takes for the cannonball, fired at angle θ, to reach this point. In order to have this height correspond to the bottom of the interceptor, we have,

$$h_\theta \equiv 100t + 1000 \mod 2000.$$

Note that we have modulo 2000, as the position of the interceptors is the same every 2000 metres travelled. This can be rearranged to

$$t \equiv \frac{h_\theta}{100} - 10 \mod 20.$$

Since t is the desired time for the cannonball to reach a distance of 12000 metres from the origin, we can take into account the time t_θ, it takes for the cannonball to reach this distance, and calculate the time delay t_d as follows,

$$t_\theta + t_d \equiv \frac{h_\theta}{100} - 10 \mod 20$$

which gives,

$$t_d \equiv \frac{h_\theta}{100} - t_\theta - 10 \mod 20. \tag{2.1}$$

Finally, we can use this time delay t_d, to obtain all the possible firing times. It is clear that it will take the next interceptor 10 seconds to reach h_θ. Therefore, there is a 10 second interval $(t_d, t_d + 10)$, where the cannonball will successfully avoid hitting the interceptors. Moreover, we know that the position of the interceptors is time periodic, i.e. they are in the same position every 20 seconds. Hence, we can conclude that the time intervals where the cannonball will avoid the interceptors and hit the target successfully, are all the positive intervals $(t_d + 20n, t_d + 10 + 20n)$, where $n \in \mathbb{Z}$.

2.2 Implementation

First, we have the function `FindLaunchAngles(target)` which we will use to find the two possible launch angles that reach the target at 15000 metres.

```
function [ang1, ang2] = FindLaunchAngles(target)
% FindLaunchAngles Calculate the two launch angles to hit given target

distances = @(theta) Distance(theta) - target;
ang1 = fzero(distances, [eps, 0.77842]);  % use eps instead of 0 for fzero
ang2 = fzero(distances, [0.77842, pi/2]);
end
```

Listing 5: Returns the two launch angles that hit the `target`.

The angles retrieved from this can be used as inputs for `TimeDelay(angles)` to find the firing times that will avoid the interceptors. However, first we implement the function `GetHeightAndDurationAt12K(theta)`, which returns the height h_θ, and the duration t_θ, discussed in Observation II for a given angle θ.

```
function [height, duration] = GetHeightAndDurationAt12K(theta)
% GetHeightAndDurationAt12K Find the height of cannonball at horizontal
% distance=12000 for given firing angle and the time it takes to reach this height.

% Solve the ODE numerically till the horizontal distance = 12000
[t, x] = Projectile(theta, @InterceptorEvent);

duration = t(end);
height = x(end, 3);
end
```

Listing 6: Returns the height and flight time at 12000m for a given `theta`.

Notice, in Listing 6 we use `@InterceptorEvent` as the events function for the function `Projectile`, defined in Listing 1. This is because, we are only concerned for a solution until the cannonball reaches a distance of 12000 metres. Hence, we can stop solving further, once this distance is reached.

```
function [value, isterminal, direction] = InterceptorEvent(~, x)

% Terminate the ode when horizontal distance = 12000m
value(1) = x(1) - 12000;
isterminal(1) = 1;
direction(1) = 1;
```

Listing 7: `InterceptorEvent` function.

Finally, we can employ the above functions and use (2.1) to implement `TimeDelay(angles)` which takes a list of angles as the input.

```
function TimeDelay(angles)
% TimeDelay Find the time delay for firing angles that reaches x=12000 just
% below the interceptor.

for i = 1:length(angles)
    % Get the height and duration of particle when it reaches interceptors
    [height, duration] = GetHeightAndDurationAt12K(angles(i));

    % Find one time delay
    time_delay = mod((height/100) - duration - 10, 20); % as in (2.1)
    fprintf('Time delay for angle %f = %f\n', angles(i), time_delay);
end
end
```

Listing 8: Outputs one time delay t_d each, for the given `angles`.

2.3 Result

Let us first get the launch angles that hit the target at 15000 metres by calling `FindLaunchAngles(15000)`.

```
>> [angle1, angle2] = FindLaunchAngles(15000)
angle1 = 0.425365590820470
angle2 = 1.133637131950522
```

Firing Angles (in radians) to hit target at 15000m.

Now, we can use each angle and find the firing times for each angle by calling `TimeDelay` as follows:

```
>> TimeDelay([angle1, angle2])
Time delay for angle 0.425366 = 11.280544
Time delay for angle 1.133637 = 18.171079
```

Time delays for both firing angles.

From this, we can conclude that the two launch angles that can hit the target at 15000 metres successfully are $\theta_1 = 0.425366$ and $\theta_2 = 1.133637$ (in radians). Furthermore, the respective time intervals that successfully avoid the interceptors are

$$t_{\theta_1} = [0, 1.280544) \cup \{(11.280544 + 20n, 21.280544 + 20n) \mid n \in \mathbb{N}\}$$

and

$$t_{\theta_2} = [0, 8.171079) \cup \{(18.171079 + 20n, 28.171079 + 20n) \mid n \in \mathbb{N}\}.$$

in seconds.