Things to consider in your code

1. You may assume that the target is in the first quadrant and is within striking range. Credit will be lost if you can't hit all possible targets in the first quadrant.

2. We will need your answer for θ in radians to four decimal places, and five significant figures. That is, $\theta =$ x.xxxx. Clearly, for the basic targets in the first quadrant, $0 \le \theta \le \pi/2$. Note that you are actually actually free to determine for yourself what it means to "hit the target." The real requirement is that if you add or subtract 0.0001 radians, the trajectory would change from an undershot to an overshot.

3. You may hard–code the parameters $g = 9.81$ m/s^2, $C_D/m = 0.002$ m^{-1}, and $v_0 = 1500$ m/s into your program. If you have already worked out the equations to account for the attractors, you will have noticed that α/m appears in the set of coupled ODE's that you need to integrate. To make things easier for you, assume that your call to `Sensors.m` returns a matrix, each row of which contains the values of $(\alpha_i/m, x_i, y_i)$ for each attractor. You can find a sample `Sensors.m` file at the end of this document.

4. This will be a timed competition (using `tic toc`) and extra credit will be awarded to the ten fastest groups. We will call your `Target.m` file with a script similar to the one listed at the end of this document, `FireTest.m`. Of course, the target coordinates will vary from those given in the sample script.

5. Just for the record, the correct system of equations for n attractors should be:

$$\begin{aligned}
\dot{x} &= v\cos\theta \\
\dot{y} &= v\sin\theta \\
\dot{v} &= -\frac{C_D}{m}v^2 - g\sin\theta + \sum_{i=1}^{n}\frac{\alpha_i/m}{d_{a,i}^3}\Big[(x-x_{a,i})\cos\theta + (y-y_{a,i})\sin\theta\Big] \\
\dot{\theta} &= -\frac{g}{v}\cos\theta + \sum_{i=1}^{n}\text{your proprietary stuff accounting for each attractor}
\end{aligned}$$

   where $d_{a,i} = |(x-x_{a,i})\,\mathbf{i} + (y-y_{a,i})\,\mathbf{j}|$ is the distance from the projectile to the i^{th} attractor, and n is the number of attractors, each of which has strength α_i, and is located at $(x_{a,i}, y_{a,i})$.

Things to consider in your report

- You must work in groups of two or three.
- Your document should read like a real report with all of the usual sections. You may write it in a style, and at a level, that your fellow engineering students would understand. Also, you should consider writing it so that you could pick it up in five years, read it, and be able to tell exactly what you did and why.
- How did your `Target.m` file determine your initial guesses for θ_0^* in the bisection method? Why?
- What stopping conditions for the integration does your program use, and why?
- Explain your accuracy settings for MatLab's `ode45` integrator.
- What other factors does your program incorporate to be fast and state-of-the-art?
- If you were writing your code for speed, what did you do to make it run faster? Why?

Submitting your report

- You only need to submit one hard–copy of the report for each group. Be sure to include the names of all group members on the report. Also, include the names of all team members in each and every Matlab script.
- You only need to submit one electronic file for each group. Place all of your Matlab code into one folder labeled `Code` and all of your LaTeX files into another folder labeled `Report`. Place both of these folders into another folder that uses your last name as the name of the folder. Zip this folder and make sure the name of the zip file matches your last name.
- More detail will follow this weekend.

Sample code files

- The main program that we will use to test your code will look like the following, but with different target locations. Please note the two sample solutions given are for no attractors. (Actually, they have attractors but $\alpha/m = 0$ for each attractor.)

```
clc
clear all
close all
format long
%---------------------------------
target = [75, 500];
tic
for I = 1:4
    Target(target);
end
Target(target)    % Print results of last run
toc

% ans = 1.4223 with no attractors    <---- sample solution
%---------------------------------
target = [1000, 800];
for I = 1:4
    Target(target);
end
Target(target)    % Print results of last run
toc

% ans = 0.7035 with no attractors    <---- sample solution
%---------------------------------
target = [1400, 200];
for I = 1:4
    Target(target);
end
Target(target)    % Print results of last run
toc
%---------------------------------
format short
```

- Here is a sample `Sensors.m` file. Note that the number of attractors and the attractor α/m, x and y values will change when we run your code.

```
function AttractorData = Sensors

%--------------------------------
% Determine values for attractor strength and location
% This returns an array of [alpha/m, x, y]
%--------------------------------
% Attractors off (zero strength)

% AttractorData = [ 0, 400, 375;      %<--- 2 attractors
%                   0, 800, 600];

%--------------------------------
% Attractors on

AttractorData = [ -1e+06, 400, 375;      %<--- 2 attractors
                  +1e+06, 800, 600];

%--------------------------------
```