EEE4119F – Mechatronics II – Project Brief 2020

Save a city from an asteroid

Introduction

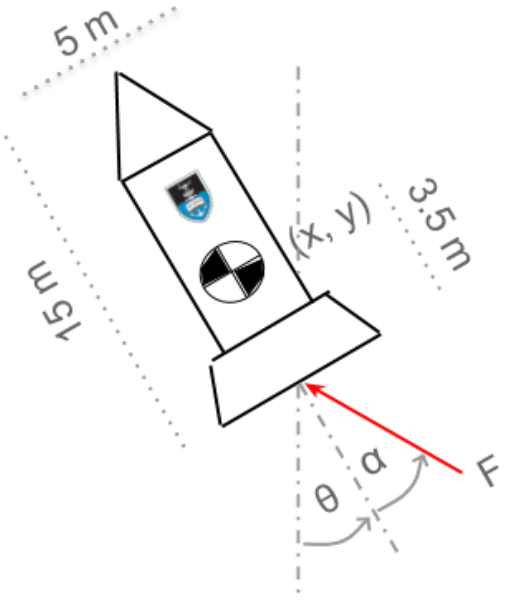
You are tasked with designing a controller for a rocket, such that it intercepts an asteroid hurtling towards a city on Earth. This is to be done in simulation using MATLAB and Simulink.

Rocket model

The rocket has a single thruster F which can be angled by α (alpha) rad. The output range is $0 \leq F \leq F_{max}$ and $-\pi/2 \leq \alpha \leq \pi/2$.

The rocket itself can be approximated as a box with width $L_1 = 5m$ and height $L_2 = 15m$. The distance from the bottom of the rocket to its center of mass is $3.5m$ (ie, $4m$ from the center of mass to the geometric center). Its mass is $m = 1000kg$

It has an angle of θ rad and position (x, y) with x and y being the center of mass of the rocket, as measured from the rocket's initial position: $(x = 0m, y = 0m)$ (it



starts slightly under ground). The rocket has been coated in Miracle Spray, so you can ignore any drag effects

Tip: you'll find that you can't simply find this model's transfer function. This means that you'll have to linearize the system around one or more operating points, or use some other nonlinear control approach. There are practice questions you can look at, or links on the forum on Vula

Asteroid model

The asteroid begins at an initial position $(x = x_i\ m,\ y = y_i\ m,\ \theta_{ast} = 0\ rad)$ where $x_i < 0,\ y_i > 0.$ It experiences forces due to gravity ($g = 9.81\ m/s^2$) as well as air resistance: $F_{drag} \propto c\sqrt{\dot{x}^2 + \dot{y}^2}$. The asteroid has a mass of $m = 10\ 000kg$ with an initial velocity such that it'll roughly make contact with a city at $(x = (2500\ \pm 500)\ m,\ y = 0\ m),$ unless you manage to stop it! Due to un-modelled atmospheric effects (turbulence, birds, etc) there is process noise in the equations of motion. All together, the equations are as follows:

$$\ddot{x} = F_{drag-x}/m + noise_x$$

$$\ddot{y} = F_{drag-y}/m - g + noise_y$$

$$\ddot{\theta}_{ast} = noise_{th}$$

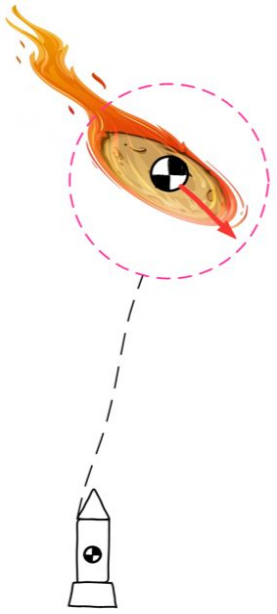
Estimating an *approximate* value for c will be your first task. Note that each student will receive a unique but constant c value, generated by their student number. (The idea behind this is that scientists estimate this factor while the rocket is being prepared, but you'll just run the simulation to gather data)

Tip: it may help to gather your simulation data using a smaller timestep. Filtering the (very noisey) data could also help. Bear in mind that the process noise (which is different from measurement noise!) will result in you receiving an inaccurate c value, which is okay! You can expect a value somewhere in the range of c = 100. Part of the idea is that, as the asteroid tumbles, it gets affected by the atmosphere in a varying way. The c value likely won't be absolutely repeatable

Also, the Simulink file sends simulation data to your Matlab workspace (you can look under the mask to see this) so you can start analyzing the data straight away. Think carefully about how you can use the equations above to find c

Requirements

In simulation, the rocket must detonate near the rocket. There are three levels to this task, each of which raises the difficulty of the problem:



Scenario 1

The rocket must detonate within 150 meters of the center of mass of the asteroid, before the center of mass of the asteroid falls to within 200 meters of the city, measured as height above ground.

Luckily, stellar work by the astronomy department means that the asteroid begins fairly far away, so a simpler strategy should work!

For example, can you predict when the asteroid will be directly above the rocket, and at what height?

Scenario 2

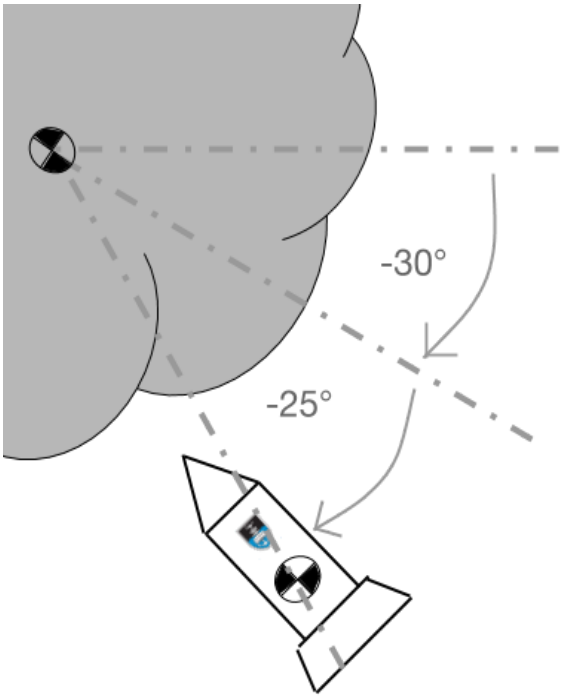
In this scenario, the asteroid begins in a state such that flying straight up is not an option.

One strategy would be to estimate its trajectory, and then find a point along that trajectory that you can reach at the same time that the asteroid is there.

Think carefully about what you need to do here! You may need to estimate a possible collision point between the rocket and asteroid, and/or estimate how long it'll take for the asteroid to get there. You'll certainly need to control the rocket, because it's fairly unstable. There are also other approaches to take, like Proportional Navigation (which you may need to modify slightly)

Scenario 3

Finally, in this OPTIONAL scenario, engineers have determined that the rocket will cause the most damage to the asteroid if it strikes from an appropriate angle. The rocket should strike the asteroid at an appropriate angle to the asteroid (defined below) or angle + 180 degrees. Specifically, the rocket should strike in the range $-30 \le \psi \le 30$ or $150 \le \psi \le 210$.



The angle is measured at the moment of detonation of the rocket, and is a function of the angle of the asteroid (θ_{ast}), and the position of the rocket relative to the position of the asteroid.

As an example, in the adjacent diagram the asteroid is at an angle of -30 degrees relative to the world frame. The rocket is approaching the asteroid at an angle of -25 degrees, relative to the angle of the asteroid. (Note that this is different from the angle of the rocket itself). This would be sufficient to destroy the asteroid.

Finally, as luck would have it, this scenario begins directly after the asteroid has bounced off the Earth's atmosphere. This means that the asteroid as a slight upwards velocity at $t = 0\ sec$ (which buys you more time), but it will still hit the city!

Allowable modification

After seeing the results of the survey and how overwhelmed students are from other courses, we have decided to give students the *option* of making the problem a bit easier by adjusting the initial position of the asteroid. In this scenario, the asteroid starts far higher and with a higher initial velocity, such that it's roughly at its terminal velocity. This results in it not accelerating as much, making for a more interceptable target

If you wish to opt into this, you may change the initialisation code by clicking on the arrow in the bottom left of the System Dynamics mask, then double clicking on the fcn block in the bottom left

which takes in ScenarioVar and rngSeed, and replacing the first two parts of the if-statement (lines 4 to 18) with:

```
if (Scenario == 1)
      x0 = -1000;
      y0 = 100000;
      th0 =   0;
      dx0 =   50;
      dy0 = -1100;
      dth0 = 0;

elseif (Scenario == 2)
      x0 = -2000 + rand_within_plus_minus(100);
      y0 = 100000 + rand_within_plus_minus(100);
      th0 = 0;
      dx0 = 70;
      dy0 = -1000;
      dth0 = 0.1;
```

You may also want to change YMAX in define_constants to `100000;`

Additional marks

If the first two scenarios are completed, additional marks will be given for,

- Maximizing the distance between the asteroid and city at the time of detonation
- Detonating the rocket in between the asteroid and the city, so that any remaining shrapnell is pushed away from the city (only if you didn't modify the initialization code)
- Completing (or showing progress towards completing) Scenario 3

You will be judged on the metric you do best at

Helper files

You will be given the following files, which may be useful when designing your controllers:

1. `mission_success.m` – a function that indicates whether the requirements have been met.
2. `make_animation.m` – a function that produces a simple animation of the rocket and asteroid. It shouldn't be taken too seriously!
3. `record_animation.m` - a function that writes the animation to disk, which might be useful for sharing videos. The quality is very bad, so you may want to modify the file to suit your needs
4. define_constants.m - a function that defines some variables for your workspace, so that the variables don't have to be defined in each function that use it
5. `AsteroidDynamics.p`, `RocketDynamics.p` - these define the dynamics of the asteroid and rocket respectively. The are used by the simulink model
6. `main.m` – a short demo of how to use these helper scripts.

You can look at their code and (with the exception of `mission_success`) modify them as needed. Let the TA know if you need an explanation of anything

NOTE: there is always the chance that there is a bug somewhere in either the model file, animation script or success evaluation script. Keep an eye out for Vula notifications, and if you see something is

obviously wrong (eg you found a way to "trick" the `mission_success` file in a way that obviously defeats the idea of the task) then let the TA know, and don't expect that you'll be allowed to make use of the bug.

Submission and ELO

A < 10 pg report summarizing your design and results is due by **23:59 on Sunday 21 June**. Your controller code should be submitted as well. While you can look under the mask of the model, you should not modify *any* code there - it will be automatically replaced by the original code, possibly breaking any changes you have made. If there is an error in the code, please contact the TA!

Note that the project is aimed at assessing the ECSA ELO 2 (see the course handout for more details).

Marking scheme

Marks will be awarded as follows:

Task	**Marks**
Model identification (c parameter, equations of motion)	30
Report on controller	20
Satisfy requirements in scenario 1	20
Satisfy requirements in scenario 2	20
Additional marks	10