EEE4119F: Save a City from an Asteroid Project Final Report

Prepared by:

Justin Cross

CRSKAT005

Prepared for:

EEE4119F

Department of Electrical Engineering

University of Cape Town

20/05/2025

Plagiarism Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.

3. This report is my own work.

4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

[illegible]

20/05/2025

Contents

Introduction

The aim of the project was to utilise control techniques to control the movement of a rocket in order to intercept an asteroid hurdling towards a city and prevent its impact to save humanity. The project problems deal with different scenarios (described below) where objectives needed to be met in order to successfully destroy the asteroid before making contact with the city. The control process involved rocket and asteroid modelling, rocket-to-asteroid navigation, rocket-stability control and interception.

Scenarios (Rocket objectives)

Scenario 1:

The rocket must detonate within 150 meters of the centre of mass of the asteroid, before the centre of mass of the asteroid falls to within 200 meters of the city.

Scenario 2:

Same objective as scenario 1, except in this scenario, the asteroid begins in a state in which flying the rocket straight up is not an option.

Scenario 3:

The rocket must strike/detonate on/near the asteroid in a 60-degree arc in front the asteroid or behind the asteroid to inflict maximum damage and destroy the asteroid. In other words, the rocket can strike the front of the asteroid between -30 and 30 degrees, or behind the asteroid between 150 and 210 degrees.

System modelling (Unchanged as GA2 satisfied)

Rocket modelling

Rocket parameters:

An assessment of the rockets parameters was made prior to the modelling of the rocket, and these parameters and specifications are discussed below:

The rocket has a mass of 1000kg with its shape being approximated as a box with width L1 = 5m and height L2 = 15m. It has a shifted centre of mass which occurs 3.5m away from the base of the rocket and along the axis of symmetry on the x axis in its body frame.

The moment of inertia of the rocket is as follows:

$$\boldsymbol{I = \frac{1}{12}(m)(L1^2 + L2^2) + m(\frac{L2}{2} - 3.5)^2 \ldots eqn(1)}$$

The rocket contains a single thruster F which can be angled by α rad. The range at which the rocket can angle its thruster is $-\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$ and it has an output force range of $0 \leq F \leq F_{max}$.

The rocket has a body angle of θ and a position (x,y) which has been used to describe the rockets positions, velocities and accelerations in the modelling process below.

Rocket model:

The rocket model is described using an equation of motion founded by applying Lagrangian mechanics and generalised force techniques to accurately represent the rocket's motion.

1. **Generalised coordinates:**
   The rockets chosen generalised coordinates are:
   $q = [x;\ y;\ th]$ → represents the position of the rocket in 2D
   $dq = [dx;\ dy;\ dth]$ → represents the linear and angular velocities
   $ddq = [ddx;\ ddy;\ ddth]$ → represents the linear and angular accelerations
2. **Kinetic and Potential energies of rocket:**
   The kinetic energy was calculated as follows:

$$\boldsymbol{T = 0.5(m)(v^T)(v) + 0.5(I)(\omega)^2 \ldots eqn(2)}$$

v → represents the linear velocity in the inertial frame computed by taking the jacobian of the position of the rocket in the inertial frame.
Ɯ → represents the angular velocity in the inertial frame.

The potential energy was calculated as follows:

$$\boldsymbol{V = mgy_{rocket} \ldots eqn(3)}$$

Where y_{rocket} represents the vertical position of the rocket in the inertial frame.

3. **Mass, Gravity and Coriolis matrices:**
    - The Mass matrix was calculated by finding the Hessian of the total kinetic energy.
    - The Coriolis matrix was calculated by finding derivative of the mass matrix.
    - The Gravity matrix was calculated by finding the Jacobian of the total potential energy.
4. **Generalised forces:**
   The rocket's thrust force needed to be modelled as a generalised force by modelling the force in the inertial frame and applying the following equation to achieve the desired generalised forces.

$$\boldsymbol{Q = \sum F\left(\frac{\partial r_F}{\partial q}\right) \ldots eqn(4)}$$

   Where ∂r_F is the location of the thrust force in the inertial frame.
5. **Equation of motion:**
   The final equation of motion to successfully model the rocket's dynamics was acquired using the manipulator equation as follows:

$$\boldsymbol{M * ddq + C + G^T = Q \ldots eqn(5)}$$

   Where:
   M = Mass matrix; C = Coriolis matrix; G = Gravity matrix; Q = Generalised force

Asteroid modelling

The asteroid model begins at an initial position of x = x_i [m], y = y_i [m], θ_{ast}= 0 [rad], and the forces affecting its movement are gravity as well as air resistance where

$$\boldsymbol{F_{drag} \propto \sqrt{dx^2 + dy^2} \ldots eqn(6)}$$

Due to the presence of unmodelled atmospheric effects (e.g. turbulence, birds, etc) the equations of motion for the asteroid model contain noise parameters as shown the follow EOMs:

$$\boldsymbol{ddx = \frac{F_{drag}}{m} + noise_x \ldots eqn(7)}$$

$$\boldsymbol{ddy = \frac{F_{drag}}{m} - g + noise_y \ldots eqn(8)}$$

$$\boldsymbol{dd\theta = noise_\theta \ldots eqn(9)}$$

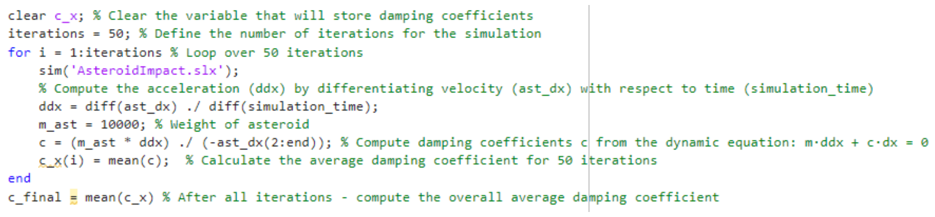
By using the equation of motion for the x component of acceleration for the asteroid, the drag coefficient c was able to be calculated.

$$\boldsymbol{F_{drag\,x} = -c(dx) \ldots eqn(10)}$$

By applying Newtons 2nd law:

$$\boldsymbol{c = \frac{-m(ddx)}{dx} \ldots eqn(11)}$$

Through the use of simulated data, the drag coefficient was effectively calculated. This was done by applying the equation for c determined above and using multiple iterations (50 iterations) of simulated data to create a large dataset making the estimation more reliable. The mean of all the estimated c values was taken to achieve one final, effective drag coefficient value for the asteroid. A code snippet is provided for further insight into the estimation strategy:



```
clear c_x; % Clear the variable that will store damping coefficients
iterations = 50; % Define the number of iterations for the simulation
for i = 1:iterations % Loop over 50 iterations
    sim('AsteroidImpact.slx');
    % Compute the acceleration (ddx) by differentiating velocity (ast_dx) with respect to time (simulation_time)
    ddx = diff(ast_dx) ./ diff(simulation_time);
    m_ast = 10000; % Weight of asteroid
    c = (m_ast * ddx) ./ (-ast_dx(2:end)); % Compute damping coefficients c from the dynamic equation: m·ddx + c·dx = 0
    c_x(i) = mean(c);  % Calculate the average damping coefficient for 50 iterations
end
c_final = mean(c_x) % After all iterations - compute the overall average damping coefficient
```

Figure 1: Code snippet of C value estimation

Control Scheme (See this chapter for changes for GA2)

Different control techniques were utilised for the different scenarios at hand. For each milestone listed below, the control logic utilized to achieve successful interception will be discussed.

Milestone 2 (Scenarios 1 & 2)

In this milestone, two key control techniques were used to create successful tracking and stabilization of the rocket towards the asteroid. The tracking of the rocket toward the asteroid was achieved using proportional navigation methods. The stability control of the rocket was achieved using PID control methods. Each of these methods are further elaborated on below:

Proportional Navigation

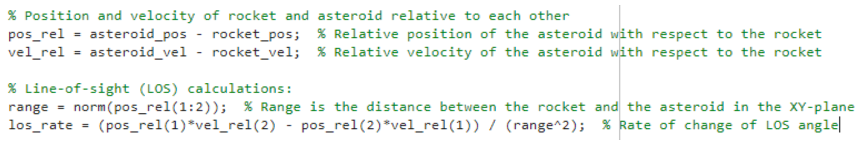
Proportional navigation is a guidance law often used by missiles and for autonomous system control. It directs the system to intercept a moving target by adjusting its direction based on how the line-of-sight (LOS) angle to the target object is changing. The line-of-sight angle is the angle between some form of reference direction (y-axis) to the straight line connecting the two moving objects together. The technical objective to using PN navigation is that the system should accelerate in a direction proportional to the rate of change of the line-of-sight angle. This technique aims to null the LOS rate, leading to efficient paths chosen for asteroid interception – as well as having a very simple computational method, making it a suitable choice for rocket-asteroid interception control.

The equation for the rate of change of the LOS angle is:

$$\dot{\boldsymbol{\lambda}} = \frac{\boldsymbol{x} \cdot \boldsymbol{v_y} - \boldsymbol{y} \cdot \boldsymbol{v_x}}{\boldsymbol{x^2} + \boldsymbol{y^2}} \quad \ldots \boldsymbol{eqn}\ (\mathbf{12})$$

Where $\dot{\lambda}$ is the LOS rate, x and y represent the relative position coordinates of the asteroid w.r.t to the rocket, v_x and v_y represent the relative velocity components of the asteroid w.r.t to the rocket.

The following code snippet indicates the calculation of the rate of change of the LOS angle:



```
% Position and velocity of rocket and asteroid relative to each other
pos_rel = asteroid_pos - rocket_pos;  % Relative position of the asteroid with respect to the rocket
vel_rel = asteroid_vel - rocket_vel;  % Relative velocity of the asteroid with respect to the rocket

% Line-of-sight (LOS) calculations:
range = norm(pos_rel(1:2));  % Range is the distance between the rocket and the asteroid in the XY-plane
los_rate = (pos_rel(1)*vel_rel(2) - pos_rel(2)*vel_rel(1)) / (range^2);  % Rate of change of LOS angle
```

Figure 2: LOS calculation

The equation for the acceleration required to hit the moving target is shown below:

$$\boldsymbol{a_n} = \boldsymbol{N} \times \boldsymbol{v_c} \times \boldsymbol{LOS_{rate}} \quad \ldots \boldsymbol{eqn}(\mathbf{13})$$

Where:

a = normal acceleration of rocket; N = Navigation gain;

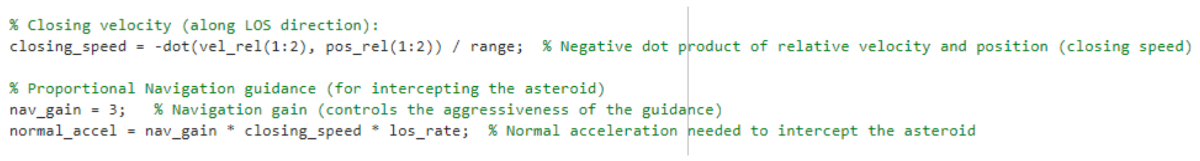
V$_c$ = Closing velocity → $v_c = -\frac{\overrightarrow{v_{\text{rel}}} \cdot \overrightarrow{r_{\text{rel}}}}{|\overrightarrow{r_{\text{rel}}}|} \quad \ldots eqn(14)$

The closing velocity determines how fast the rocket is closing in on the asteroid, and the navigation gain is used as a scalar quantity to adjust how aggressively the rocket turns to intercept the asteroid. Through the modelling theory of PN systems whereby a PN system behaves like a 2nd order system, the damping ratio has a relationship with the navigation gain, where $\zeta \approx \frac{1}{\sqrt{N}}$. Three different navigation gain values were tested and associated damping ratios calculated. The results of the different gain values used were summarized and discussed.

Nav Gain Value	**Damping**	**Result**
N = 2	$\zeta \approx 0.707$	Critically damped, yet response too lethargic - asteroid often missed.
N = 3	$\zeta = 0.577$	Slightly below critical damping, yet response increases in aggression and optimal correction toward asteroid achieved.
N = 4	$\zeta = 0.5$	Moving to more underdamped behaviour. Adjustments slightly more aggressive, successful asteroid collisions decreased.

Table 1: PN performance under different navigation gains

The code snippet below shows the chosen navigation gain (N=3) along with the closing velocity calculation and the fundamental PN equation:



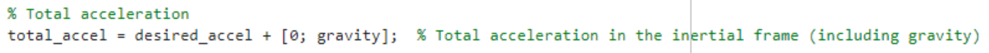
```
% Closing velocity (along LOS direction):
closing_speed = -dot(vel_rel(1:2), pos_rel(1:2)) / range;  % Negative dot product of relative velocity and position (closing speed)

% Proportional Navigation guidance (for intercepting the asteroid)
nav_gain = 3;   % Navigation gain (controls the aggressiveness of the guidance)
normal_accel = nav_gain * closing_speed * los_rate;  % Normal acceleration needed to intercept the asteroid
```

Figure 3: Closing velocity calculation and Navigation gain logic

Additional considerations and calculations:

- **Gravity consideration:**
  The desired normal acceleration and its direction does not account for any external forces acting on the rocket. Therefore, a gravity compensation was added to the total acceleration of the rocket to counter the downward effect of gravitational acceleration on the rocket.



```
% Total acceleration
total_accel = desired_accel + [0; gravity];  % Total acceleration in the inertial frame (including gravity)
```

Figure 4: Gravity compensation

- **Thrust force calculation:**
  After accounting for the presence of the force of gravity, the desired thrust force for the rocket could be calculated using Newton's 2nd Law.
  $$\overrightarrow{F_{thrust}} = m_{rocket}\overrightarrow{a_{tot}} \; \ldots eqn(15)$$

The above technique alone does not allow for a stable rocket model, hence the integration of PID control methods which are further discussed under the following topic.

PID Control (Proportional-Integral-Derivative)

In order to create a stable rocket model that does not spin uncontrollably through the air whilst applying PN navigation techniques, a PID control method was introduced into the control model to minimise the effects of the thrust force and angle on the rocket's angular orientation. The desired orientation of the rocket is to have the longitudinal axis of the rocket body align itself with the trajectory path toward the asteroid that is determined through proportional navigation.

PID Design:

The control equation for the PID controller is

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \ldots . eqn(16)$$

Where $u(t)$ is the control output, $K_p e(t)$ is the proportional term, $K_i \int_0^t e(\tau)d\tau$ is the integral term and $K_d \frac{d}{dt} e(t)$ is the derivative term.

- The proportional term gives a correction to the current angular difference, $e(t)$,between the current rocket orientation and the desired orientation of the rocket taken from the PN navigation.
- The integral term then adds up the past errors over time through $\int_0^t e(\tau)d\tau$. This improves small errors that are still existing after the proportional term has been applied.
- The derivative term tracks how fast the error is changing, $\frac{d}{dt} e(t)$, and adjusts the rockets angular velocity accordingly.
  The Simulink figure below shows how the controller function block uses feedback to constantly update the previous angle error used for the derivative term.

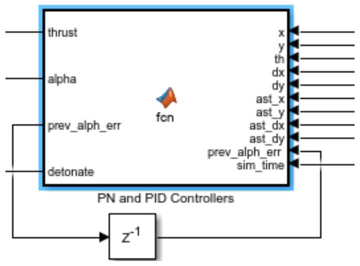


Figure 5: Feedback for previous angle error

(See Appendix A for complete Simulink Model with feedback)

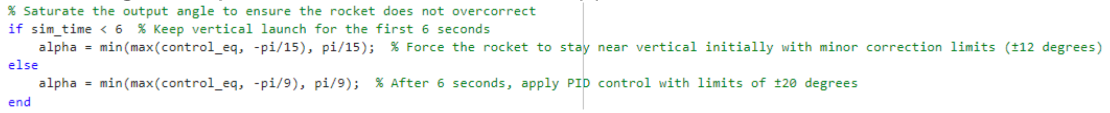
Controller Gains:

Due to high sensitivity of the rocket to control inputs, large gains can easily cause instability, oscillations, or even overshoot and loss of control. Therefore, careful consideration of the chosen gain values was taken.

- $K_p = 0.0005$ → A low proportional gain results in slow, precise corrections. Choosing a high gain can result in an overreaction to small angular errors. Initially, this value was set to 0.005, yet this was still too high and resulted in slight overshooting. Dropping this value by x10^{-1} significantly improved this control.
- $K_i = 0.00001$ →The risk of making the integral gain too large is that a short-lived error can lead to a large, accumulated value. Therefore, K_i was initially set to 0, and slowly increased by miniscule amounts and it was found that the 0.00001 achieved optimal rocket corrections.
- $K_d = 0.0001$ → The desired derivative gain was also established to be low to prevent overdamping from occurring. The initial value chosen was 0.01, which resulted in a lethargic response to angle changes. Through analysis of dropping the value by x10^{-1}, the optimal gain value was found to be 0.0001.

Additional considerations and calculations:

- **Thrust angle saturation:**
  In order to prevent the rocket from overcorrecting due to excessive angle commands from the PID control equation output, the thrust angle of the rocket was saturated/clamped between a lower and upper angular limit. During launch (first 6 seconds), the angular variance range is small (±12 degrees) to allow the rocket to get off the ground. Once in the air, the steering limit is relaxed but still limited (±20 degrees) to allow the rocket to orientate itself to intercept the asteroid whilst still maintaining stability control, shown in the code snippet below.



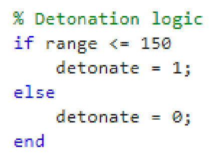
```
% Saturate the output angle to ensure the rocket does not overcorrect
if sim_time < 6  % Keep vertical launch for the first 6 seconds
    alpha = min(max(control_eq, -pi/15), pi/15);  % Force the rocket to stay near vertical initially with minor correction limits (±12 degrees)
else
    alpha = min(max(control_eq, -pi/9), pi/9);  % After 6 seconds, apply PID control with limits of ±20 degrees
end
```

Figure 6: Thrust angle saturation

In order to fulfil scenario 2, the rocket could not launch completely vertically as interception with the asteroid would not be possible, hence the minor flexibility of angle adjustments at launch time.

Detonation Logic

In order to fulfil scenario 1's detonation requirement, where the rocket must detonate within 150m of the asteroid, a detonation output was applied based off the distance between the rocket and the asteroid. The logic is shown below along with its integration into the Simulink model. (See Appendix A for detonation integration in Simulink Model)



```
% Detonation logic
if range <= 150
    detonate = 1;
else
    detonate = 0;
end
```

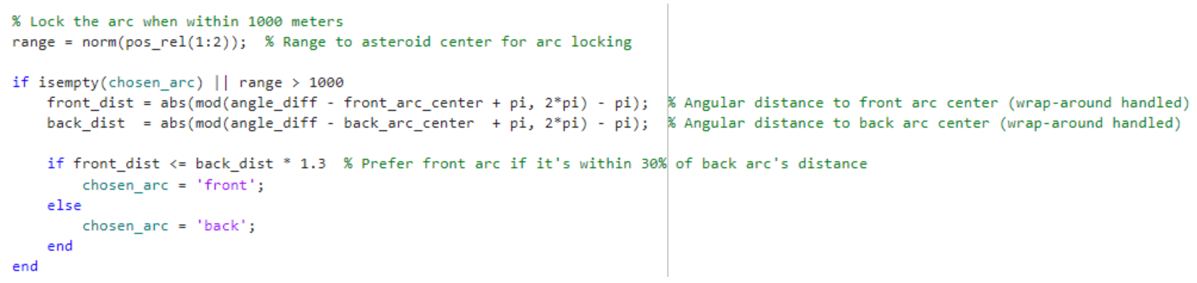
Figure 7: Code snippet of detonation logic

Milestone 3 (Scenario 3)

In order to satisfy the requirements for scenario 3, technical adjustments were made to the control scheme used in scenarios 1 & 2 to control the point of impact that the rocket makes with the asteroid. Whilst still utilising PN navigation in cohesion with PID control, the technical additions that were made are elaborated on below.

Impact point selection (Arc selection)

In order to successfully strike the front (-30 to 30 degrees) or back of the asteroid (150 to 210 degrees), arc selection logic was introduced to allow the rocket to determine which side of the asteroid is closer and lock on to that impact point. The PN logic then guides the rocket to strike that specific impact point, once the chosen arc has been locked.



```
% Lock the arc when within 1000 meters
range = norm(pos_rel(1:2));  % Range to asteroid center for arc locking

if isempty(chosen_arc) || range > 1000
    front_dist = abs(mod(angle_diff - front_arc_center + pi, 2*pi) - pi);  % Angular distance to front arc center (wrap-around handled)
    back_dist  = abs(mod(angle_diff - back_arc_center  + pi, 2*pi) - pi);  % Angular distance to back arc center (wrap-around handled)

    if front_dist <= back_dist * 1.3  % Prefer front arc if it's within 30% of back arc's distance
        chosen_arc = 'front';
    else
        chosen_arc = 'back';
    end
end
```

Figure 8: Arc selection logic

It was discovered that the rocket was always bias to targeting the back arc of the asteroid. In order to mitigate the issue, a multiplier bias was introduced for the front arc. The chosen arc is only selected once the rocket is within 1000m of the asteroid.

Navigation Gain adjustments

In order to create aggressive adjustments to the chosen target arc when the rocket comes near the asteroid, the navigation gain was increased linearly from 4 → 6 once the rocket came within 1500m of the asteroid. The changing gain value is summarized below.

Distance to Asteroid (m)	**Gain Value**
>1500	N = 4
150 < Distance < 1500	$\mathrm{N} = 4 + 2 \cdot \frac{1500 - \mathrm{range}}{1350} \quad \ldots eqn(17)$
<150	N = 6

Table 2: Navigation gain adjustments at different distances from the asteroid

For scenario 3, it was discovered that an initial gain of N = 4 was better suited compared to N = 3 in Milestone 2, where more aggressive responses where desired for arc alignment.

Thrust force multipliers

For the purpose of improving interception and control of the rocket near impact, thrust multipliers were applied to adjust the thrust force at different distances away from the asteroid.

Distance to asteroid (m)	**Thrust Multiplier**
>2000	F_{thrust} x 1.15
1000 < Distance < 2000	F_{thrust} x 1.0
<1000	F_{thrust} x 0.9

Table 3: Thrust multiplier adjustments at different distances from asteroid

PID Gain adjustments

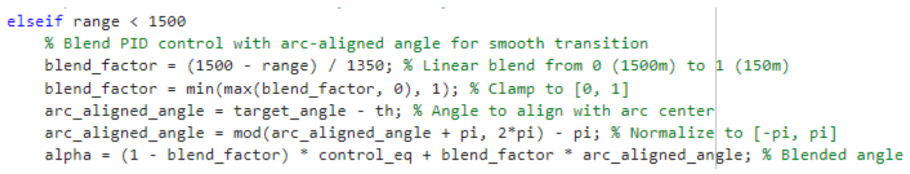
Initially, the PID gains remained the same as in milestone 2 for rocket stability. However, due to the requirement of arc alignments and the goal of the front of the rocket targeting a desired arc of the asteroid, certain gain values were adjusted.

PID Gain	**Reason**
K_p = 0.001	Increased to allow rocket orientation to adjust faster to angle misalignment with arc.
K_i = 0.0001	Remained the same (Prevent large error accumulation)
K_d = 0.0002	Increased to aid in preventing overshoot.

Table 4: PID gain adjustments for Milestone 3

Blending PID control with arc alignment

In order to align the rocket correctly with the chosen arc, a blend factor calculation was introduced to slowly transition the orientation of the rocket from PID control to asteroid arc alignment once the rocket was in 1500m of the asteroid.



```
elseif range < 1500
    % Blend PID control with arc-aligned angle for smooth transition
    blend_factor = (1500 - range) / 1350; % Linear blend from 0 (1500m) to 1 (150m)
    blend_factor = min(max(blend_factor, 0), 1); % Clamp to [0, 1]
    arc_aligned_angle = target_angle - th; % Angle to align with arc center
    arc_aligned_angle = mod(arc_aligned_angle + pi, 2*pi) - pi; % Normalize to [-pi, pi]
    alpha = (1 - blend_factor) * control_eq + blend_factor * arc_aligned_angle; % Blended angle
```

Figure 9: Technique for blending PID control and arc alignment

This code snippet shows the linear transition from full PID control at 1500m away from the asteroid to full arc alignment at 150m from the asteroid, using the equation

$$\boldsymbol{blend\ factor} = \frac{\mathbf{1500} - \boldsymbol{range}}{\mathbf{1350}} \ldots \boldsymbol{eqn}(\mathbf{18})$$

The new thrust angle was then calculated using the blend factor to slowly adjust alpha from using PID control to arc alignment.

$$\boldsymbol{\alpha} = (\mathbf{1} - \boldsymbol{blend\_factor}) \cdot \boldsymbol{control\_eqn} + \boldsymbol{blend\_factor} \cdot \boldsymbol{arc\_aligned\_angle} \ldots \boldsymbol{eqn}\ (\mathbf{19})$$

Blend factor = 0, alpha = control_eqn → Blend factor = 1, alpha = arc_aligned_angle

The rocket thrust angles were still saturated as done in milestone 2, to prevent the rocket from overcorrecting. However minor saturation adjustments were made and summarised.

Thrust Angle Saturation	**Time from launch (s) /Distance to asteroid (m)**	**Reason**
$\alpha = 0°$	First 6 seconds	Allow rocket to gain altitude.
$-30° \leq \alpha \leq 30°$	After first 6s until 500m of asteroid	Allow rocket to target asteroid but prevent excessive angle commands.

$-36° \leq \alpha \leq 36°$	Distance < 500m	More aggressive adjustments closer to asteroid to target arc.

Table 5: Thrust angle saturation adjustments

Detonation logic

The same range for the detonation logic was kept (i.e. 150m) to satisfy the detonation requirement, except angular requirements were introduced such that the rocket would only detonate when it was within 150m of the asteroid and its angular position satisfied the criteria required for scenario 3. (See Appendix B for Milestone 3 detonation logic)

Results and Discussion

Modelling

The modelling of the rocket and asteroid was successful in mathematically representing their dynamics along with predicting an accurate drag coefficient, which was consistently in the range of 90-95.

Control Schemes

A summary of the results for each scenario is shown below:

Scenario	**Number of trials**	**Number of successful trials**
1	10	3
2	10	10
3	10	10

Table 6: Results of scenarios 1,2 and 3

Milestone 2

In simulation, using the control logic from milestone 2 led to inconsistent success in scenario 1. Modifying the model to prevent initial vertical flight (for scenario 2) caused scenario 1 performance to fluctuate. Greater success was seen in scenario 1 when the rocket was allowed to launch vertically for a short time before adjusting its thrust angle. While the same control logic couldn't fully satisfy both scenarios, scenario 2 achieved 100% success, with accurate asteroid interception well before the 200 m city range limit. Thrust angle saturation also played a key role in stabilizing the rocket's control.

Milestone 3

Scenario 3 was also a consistent success when introducing arc selection and adjusting the detonation criteria of the rocket. The variation of the rocket's thrust proved useful to the navigation and angle alignment when the rocket reached critically close distances to the asteroid. Introducing a blending technique to adjust the rockets behaviour from PID control to arc alignment also proved very effective in achieving success for the scenario required.

Conclusion

The use of Lagrangian mechanics alongside Newtonian dynamics enabled the development of accurate and reliable models for both the rocket and the asteroid. The implementation and iterative refinement of proportional navigation (PN) techniques proved essential for achieving successful simulation outcomes across multiple scenarios. Integrating PID control into the PN framework further contributed to a more realistic and practically implementable control system.

The robustness and adaptability of the PN-PID strategy were demonstrated in Scenario 3, where the control architecture from Milestone 2 provided a strong foundation, removing the need for a complete redesign. However, some limitations remained, as control objectives were not fully achieved across all scenarios. Refining the logic from Milestone 2 could improve performance in Scenario 1, while exploring adaptive strategies, such as thrust multipliers or distance-based navigation gains, may enhance overall system effectiveness across multiple scenarios.

References

[1] EEE4119F Course Staff, “EEE4119F Project Milestone 1 Brief,” Dept. of Electrical Engineering, Univ. of Cape Town, Cape Town, South Africa, 2025.

[2] A. Vally, “Lecture 2 & 3 Summary: 2D Lagrange Mechanics,” African Robotics Unit, University of Cape Town, Feb. 24, 2025.

[3] EEE4119F Course Staff, “EEE4119F Project Milestone 2 Brief,” Dept. of Electrical Engineering, Univ. of Cape Town, Cape Town, South Africa, 2025.

[4] M. Hodžić and N. Prljača, “Missile Guidance using Proportional Navigation and Machine Learning,” *Journal of Engineering Research and Sciences*, vol. 3, no. 3, pp. 19–26, Mar. 2024, doi: https://doi.org/10.55708/js0303003.

[5] “The PID Controller & Theory Explained,” *Ni.com*, 2025. https://www.ni.com/en/shop/labview/pid-theory-explained.html?srsltid=AfmBOoqY3u-dl0PvvD3Kh7zWO9TZv6nWVhu4Tx1BG2UM0aQhgFXnsot4.

[6] EEE4119F Course Staff, “EEE4119F Project Milestone 3 Brief,” Dept. of Electrical Engineering, Univ. of Cape Town, Cape Town, South Africa, 2025.

Appendix A

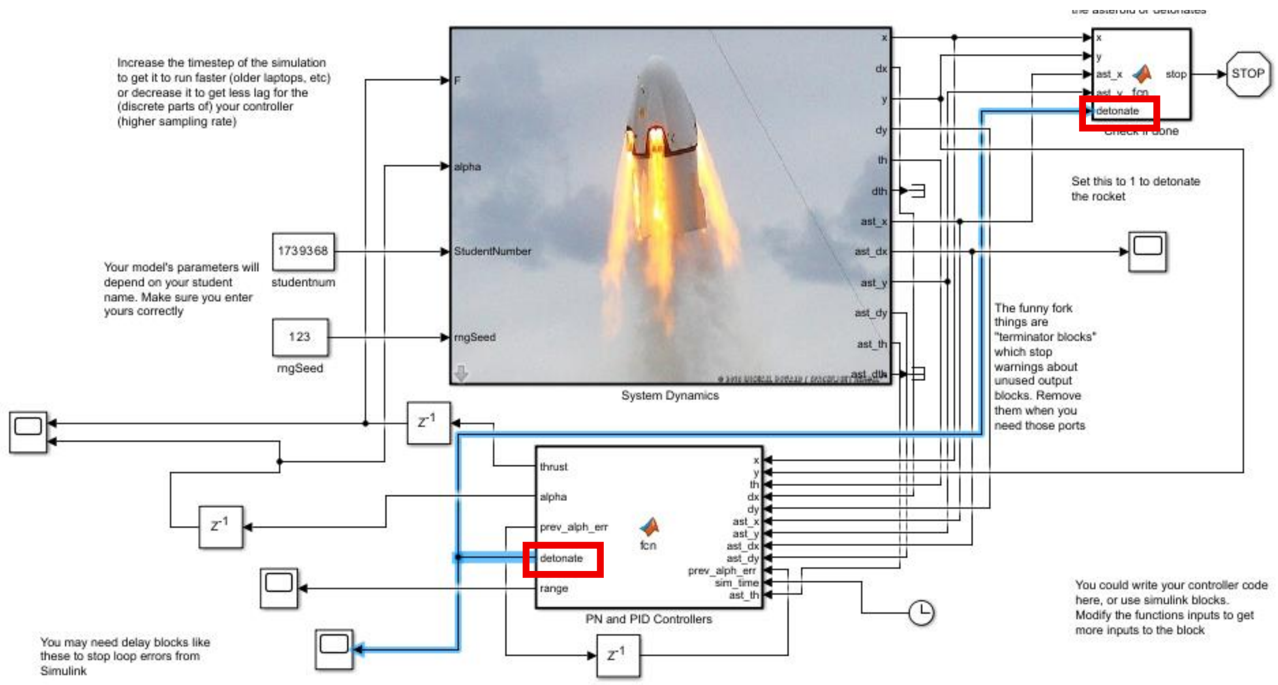
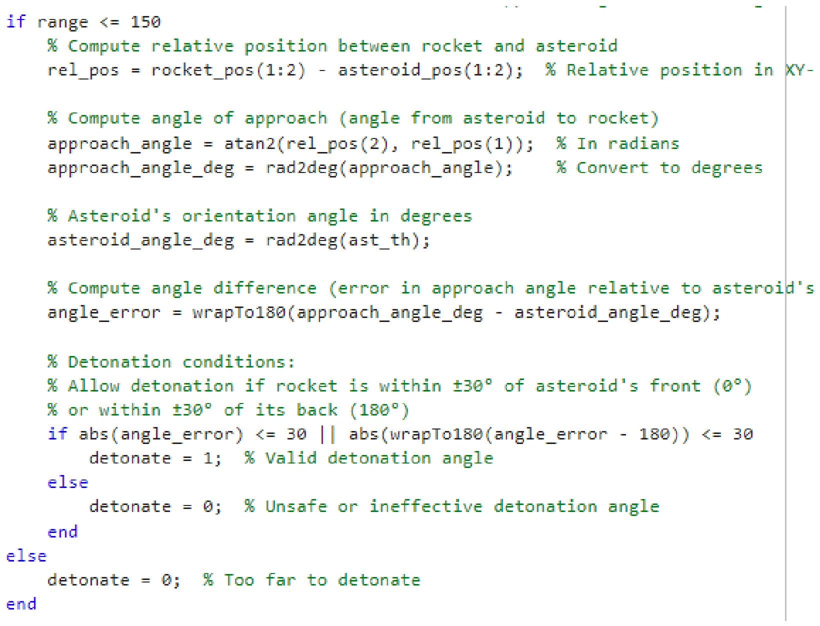

Increase the timestep of the simulation to get it to run faster (older laptops, etc) or decrease it to get less lag for the (discrete parts of) your controller (higher sampling rate)
F
alpha
StudentNumber
rngSeed
1739368
studentnum
123
rngSeed
Your model's parameters will depend on your student name. Make sure you enter yours correctly
x
dx
y
dy
th
dth
ast_x
ast_dx
ast_y
ast_dy
ast_th
ast_dth
System Dynamics
x
y
ast_x
ast_y
fcn
stop
detonate
STOP
Check if done
Set this to 1 to detonate the rocket
The funny fork things are "terminator blocks" which stop warnings about unused output blocks. Remove them when you need those ports
thrust
alpha
prev_alph_err
detonate
range
fcn
x
y
th
dx
dy
ast_x
ast_y
ast_dx
ast_dy
prev_alph_err
sim_time
ast_th
PN and PID Controllers
You could write your controller code here, or use simulink blocks. Modify the functions inputs to get more inputs to the block
You may need delay blocks like these to stop loop errors from Simulink
z⁻¹

Figure 10: Complete Simulink Model showing detonation integration

Appendix B



```
if range <= 150
    % Compute relative position between rocket and asteroid
    rel_pos = rocket_pos(1:2) - asteroid_pos(1:2);  % Relative position in XY-

    % Compute angle of approach (angle from asteroid to rocket)
    approach_angle = atan2(rel_pos(2), rel_pos(1));  % In radians
    approach_angle_deg = rad2deg(approach_angle);    % Convert to degrees

    % Asteroid's orientation angle in degrees
    asteroid_angle_deg = rad2deg(ast_th);

    % Compute angle difference (error in approach angle relative to asteroid's
    angle_error = wrapTo180(approach_angle_deg - asteroid_angle_deg);

    % Detonation conditions:
    % Allow detonation if rocket is within ±30° of asteroid's front (0°)
    % or within ±30° of its back (180°)
    if abs(angle_error) <= 30 || abs(wrapTo180(angle_error - 180)) <= 30
        detonate = 1;  % Valid detonation angle
    else
        detonate = 0;  % Unsafe or ineffective detonation angle
    end
else
    detonate = 0;  % Too far to detonate
end
```

Figure 11: Detonation logic for Milestone 3