EEE4119F Milestone 4

Project Report

Prepared by:
Luke van der Walt - VWLLUK003

Prepared for:
EEE4119F
Department of Electrical Engineering
University of Cape Town

May 20, 2025

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.

May 20, 2025

Luke van der Walt

Date

Contents

1 Introduction

1.1 Problem Definition

An asteroid is heading towards Earth and is projected to impact near a major city. A rocket-based interception strategy has been proposed to prevent it from hitting the city. This problem is divided into three scenarios with varying parameters, constraints, and objectives.

1.2 Problem Objectives

The first objective is to determine a model of the dynamics of the rocket and the asteroid. The next objectives are to use this model to design control schemes that direct the rocket to intercept the asteroid for each scenario. The specific objectives for each scenario are as follows:

- **Scenario 1:** The rocket must detonate within $150m$ of the asteroid before it lies within $200m$ of the city.
- **Scenario 2:** Same as scenario 1, but with a constraint where the rocket can not fly straight up.
- **Scenario 3:** Same as scenario 1, but the rocket must detonate near the asteroid at an angle of $-30° \leq \phi \leq 30°$ or $150° \leq \phi \leq 210°$ from the asteroid's front angle θ_{ast}

1.3 Scenario Parameters

In all the scenarios, the rocket starts at $x = 0m$, $y = 0m$ and the city is at $x = 2500m$, $y = 0m$. The starting parameters for the asteroid in each of the three scenarios are:

Scenario	$x_0\ (m)$	$y_0\ (m)$	$\theta_0\ (rad)$	$\dot{x}\ (m/s)$	$\dot{y}\ (m/s)$	$\dot{\theta}_0\ (rad/s)$
1	-3000	5000	0	182	0	0
2	$-2000 \pm \text{rand}(0, 100)$	$4000 \pm \text{rand}(0, 100)$	0	170	0	0.1
3	-2500	3000	0	160	100	$\pm\,\text{rand}(0, 20)$

Table 1: Scenario Starting Parameters

2 Modelling

2.1 Rocket Modelling

The rocket is equipped with a single thruster that produces a force F adjustable within the range $0 \leq F \leq 37kN$, with a thrust angle α constrained between $-\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$ radians. It is modelled as a rectangular box with a width of $w = 5m$ and a height of $h = 15m$. The rocket weighs $m_R = 1000kg$ and its centre of mass is located $offset = 3.5m$ from its base.

The rocket's position is represented by the 2D coordinates (x, y), measured from its initial starting point at $(0, 0)$, which is slightly underground. Its heading is measured as θ_r anticlockwise from the vertical. Aerodynamic drag effects are considered negligible. This is summarised in Figure 1. To get the rocket dynamics, the equations of motion for the rocket are derived using Newton's Second Law and the Lagrangian method.

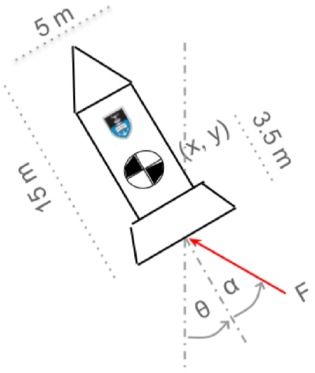

Figure 1: Rocket Free Body Diagram

2.1.1 Generalised Coordinates

The rocket is modelled as a 2D rigid body with three generalised coordinates:

$$q = [x, y, \theta]^T, \quad \dot{q} = [\dot{x}, \dot{y}, \dot{\theta}]^T, \quad \ddot{q} = [\ddot{x}, \ddot{y}, \ddot{\theta}]^T$$

where x, y represent the position of the rocket's centre of mass (COM) in the inertial frame, and θ represents the rocket's orientation.

2.1.2 Rocket Kinematics

The position of the rocket's COM in the inertial frame is:

$$\mathbf{r}_R = [x, y, 0]^T$$

The velocity of the COM is obtained through differentiation:

$$\dot{\mathbf{r}}_R = \frac{\partial \mathbf{r}_R}{\partial q}\dot{q} = [\dot{x}, \dot{y}, 0]^T$$

2.1.3 Rocket Rotational Kinematics

The angular velocity of the rocket is the time derivative of its orientation:

$$\omega_{01} = \dot{\theta}$$

2.1.4 Energy Analysis

The total kinetic energy of the rocket is the sum of its translational and rotational components:

$$T_R = \frac{1}{2}m_R(\dot{x}^2 + \dot{y}^2) + \frac{1}{2}J_{Rzz}\dot{\theta}^2$$

where J_{Rzz} is its mass moment of inertia about the z-axis. J_{Rzz} is calculated as:

$$J_{Rzz} = \frac{1}{12}m_R(w^2 + h^2) + m_R(offset)^2 = \frac{1}{12}(1000)(5^2 + 15^2) + 1000(3.5)^2$$

$$\therefore J_{Rzz} = 33083.33 \text{ kg} \cdot \text{m}^2$$

The potential energy of the rocket, considering only gravitational effects from $g = 9.81$ and is expressed as:

$$V_R = m_R g y$$

2.1.5 Generalised Forces

The generalised forces Q are determined by analysing the thrust force vector $\mathbf{F}$ applied at an angle α. Each one is found by summing the product of the partial derivative of the position and its corresponding force for each coordinate:

$$Q = \begin{bmatrix} -F\sin(\theta+\alpha) \\ F\cos(\theta+\alpha) - (m_R)(g) \\ -F(offset)\sin(\alpha) \end{bmatrix}$$

2.1.6 Mass Matrix Calculation

The mass matrix $M(q)$ is obtained from the Hessian of the kinetic energy with respect to $\dot{q}$:

$$M(q) = \frac{\partial^2 T_R}{\partial \dot{q}^2} = \begin{bmatrix} m_R & 0 & 0 \\ 0 & m_R & 0 \\ 0 & 0 & J_{Rzz} \end{bmatrix}$$

2.1.7 Coriolis Matrix Calculation

The Coriolis matrix is derived as:

$$C(q,\dot{q}) = \dot{M}(q)\dot{q} - \frac{\partial T_R}{\partial q} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

2.1.8 Gravity Matrix Calculation

The gravity matrix is calculated from the potential energy:

$$G(q) = \frac{\partial V_R}{\partial q} = \begin{bmatrix} 0 \\ m_R g \\ 0 \end{bmatrix}$$

2.1.9 Equations of Motion

Combining the derived components, the equations of motion are obtained through the manipulator equation:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = Q$$

which simplifies to:

$$\ddot{q} = M^{-1}(Q - C - G)$$

Substituting each term and solving for the accelerations results in the final expression:

$$\ddot{q} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} -\frac{F}{m_R}\sin(\alpha+\theta) \\ \frac{F}{m_R}\cos(\alpha+\theta) - g \\ -\frac{F}{J_{Rzz}}(offset)\sin(\alpha) \end{bmatrix} = \begin{bmatrix} -\frac{F}{10000}\sin(\alpha+\theta) \\ \frac{F}{10000}\cos(\alpha+\theta) - 9.81 \\ -\frac{F}{9452}\sin(\alpha) \end{bmatrix} \tag{1}$$

2.2 Asteroid Modelling

The asteroid's position is represented by coordinates in the x and y directions, measured from the rocket starting point at $(0,0)$. The initial positions of the asteroid are listed in Table 1, and all the trajectories result in an impact within $\pm 500m$ of the city. The asteroid is influenced by gravitational forces $g = 9.81\ m/s^2$ and air resistance F_{drag} and has a mass of $m_A = 10\ 000kg$. The drag force is proportional to the magnitude of its velocity by c, the drag coefficient, where:

$$F_{drag} \propto c\sqrt{\dot{x}^2 + \dot{y}^2}$$

The asteroid's motion is modelled based on the x component velocity data returned from the simulation. The velocity is first smoothed using a low-pass filter to remove high-frequency noise, and then numerical differentiation is applied to compute the acceleration. This is used to calculate the drag coefficient c. Its equation of motion in the x direction, which also includes process noise due to unmodelled atmospheric effects, is defined as follows:

$$\ddot{x} = \frac{F_{drag_x}}{m} + noise_x$$

2.2.1 Velocity Smoothing

A low-pass filter is applied to the velocity data to remove high-frequency noise as follows:

$$v_x = \text{lowpass}(\text{raw data}, f_c, f_s)$$

Where f_s is the sampling frequency from the simulation, and f_c is the cutoff frequency chosen as $f_c = 0.1$ Hz.

2.2.2 Acceleration Estimation

The acceleration component is estimated by performing numerical differentiation on the smoothed velocity data:

$$\ddot{x} = a_x = \frac{\partial v_x}{\partial t}$$

This is computed using the gradient method:

$$a_x = \text{gradient}(v_x, \Delta t)$$

2.2.3 Drag Coefficient Estimation

The drag coefficient c is estimated from the horizontal dynamics of the asteroid. Since the drag force is proportional to the asteroid's speed, the acceleration in the x direction can be represented as:

$$m_A a_x = F_{\text{drag},x}$$

where the drag force is modelled as:

$$F_{\text{drag},x} = -cv_x$$

Substituting into the acceleration equation:

$$a_x = -\frac{c}{m_A} v_x$$

$$\therefore c = -m_A \frac{a_x}{v_x}$$

2.2.4 Final Drag Coefficient

The final drag coefficient is calculated as the average of the dataset:

$$c_{avg} = \frac{1}{N} \sum_{i=1}^{N} c_i$$

where N is the number of valid data points. However, since there is a lot of noise in the system, the value for c fluctuates between datasets. To account for this, this entire process is repeated several times to find the average calculated c value. The final c was found to be approximately:

$$c_{final} \approx 94.19$$

3 Control Scheme

Two methods are explored to control the rocket depending on the scenario. For Scenario 1, a simple proportional feedback controller is used that restrains the rocket to vertical motion only. For scenarios 2 and 3, a more robust proportional navigation controller is used, which allows accurate tracking in x and y.

3.1 Scenario 1

As seen in Table 1, the asteroid starts far away and high above the rocket. This allows for a simple control method where the rocket only has to fly vertically up to intercept the asteroid. The controller for Scenario 1 computes the rocket thrust force F based on feedback from the rocket's vertical dynamics and the current asteroid height. This control method is structured as a linear state feedback controller with a reference tracking term. The rocket dynamics are linearised by restricting $\alpha = 0$ and $\theta = 0$ and accounting for gravity.

3.1.1 System Overview

The system is expressed in the state space from:

$$\dot{X} = AX + Bu \ , \quad Y = CX + Du$$

The rocket EOM are non-linear equations as seen in Equation 1. Restricting $\alpha = 0$ and $\theta = 0$ gives the state vector and vertical motion as:

$$X = \begin{bmatrix} \dot{y} \\ y \end{bmatrix}, \quad \ddot{y} = \frac{F}{m_r} - g \tag{2}$$

3.1.2 Linearisation of the System

To cancel out the non-linear gravity term in Equation 2, the input u, can be expressed as:

$$u = F - g \times m_R$$

The linearised state-space representation of the vertical dynamics is:

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{m_R} \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad D = 0$$

3.1.3 Controller Design

The rocket is setup in closed loop with a proportional feedback gain K_{pp}. A reference gain K_{rr} is calculated to ensure unit steady-state gain from the reference input:

$$K_{rr} = \frac{1}{D - C(A - BK_{pp})^{-1}B}$$

The input control signal is then:

$$u = K_{rr} \cdot y_{\text{ref}} - K_{pp} \cdot X$$

where y_{ref} is the target vertical position of the asteroid, but since $u = F - g \times m_R$:

$$F = K_{rr} \cdot y_{\text{ref}} - K_{pp} \cdot X + g \times m_R$$

The proportional feedback gain K_{pp} is selected using pole placement as:

$$K_{pp} = \begin{bmatrix} k_v & k_p \end{bmatrix} = \begin{bmatrix} 180 & 20 \end{bmatrix}$$

This gives a reference gain of:

$$K_{rr} = 20$$

3.2 Scenario 2

Scenario 2 requires a more robust control method than Scenario 1. Proportional navigation is used in this scenario since it is a reliable guidance control method already used in many homing missiles [1]. Proportional navigation steers a body by rotating its velocity vector in proportion to the Line-Of-Sight (LOS) rate between it and a target. By doing this, it maintains a constant LOS angle to ensure collision [2].

3.2.1 System Overview

This system is expressed in the state space, just as in subsubsection 3.1.1. Proportional navigation controls the rocket's acceleration according to its heading angle and the target. The state vector and inputs for this system are therefore:

$$X = \begin{bmatrix} \dot{\theta} \\ \theta \end{bmatrix}, \quad u = \begin{bmatrix} F \\ \alpha \end{bmatrix}$$

This, when substituted with Equation 1, gives a dynamic equation of:

$$\dot{X} = \begin{bmatrix} \ddot{\theta} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -\frac{F}{9452}\sin(\alpha) \\ \dot{\theta} \end{bmatrix}$$

and an output of:

$$Y = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} -\frac{F}{10000}\sin(\alpha+\theta) \\ \frac{F}{10000}\cos(\alpha+\theta) - 9.81 \end{bmatrix}$$

3.2.2 Linearisation of the System

The non-linear state equations are linearised around the equilibrium point $X = 0$, $F = m_R \times g$ and $\alpha = 0$ using a first-order Taylor Approximation. The linearised state-space matrices are computed using partial differentiation with respect to (w.r.t) the state or inputs, and then evaluated at the equilibrium point, as show below :

$$A = \frac{\partial \dot{X}}{\partial X}\bigg|_{X=0,u=[m_R g,0]} \quad \text{and} \quad B = \frac{\partial \dot{X}}{\partial u}\bigg|_{X=0,u=[m_R g,0]}$$

$$C = \frac{\partial Y}{\partial X}\bigg|_{X=0,u=[m_R g,0]} \quad \text{and} \quad D = \frac{\partial Y}{\partial u}\bigg|_{X=0,u=[m_R g,0]}$$

3.2.3 Augmentation of the Model

To improve tracking accuracy, an augmented state-space model is constructed. The augmented matrices introduce new states. The augmented states include the integral of the tracking error (which is the velocity error) in both the x and y directions and are calculated as follows:

$$A_{\text{aug}} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix}, \quad B_{\text{aug}} = \begin{bmatrix} B \\ -D \end{bmatrix}$$

3.2.4 LQR Controller Design

A Linear Quadratic Regulator (LQR) is applied to design the controller for the scenario. The LQR gain matrix is computed as:

$$K_{aug} = \text{lqr}(A_{aug}, B_{aug}, Q, R) \tag{3}$$

The penalty weights for Scenario 2 are chosen as:

$$Q = \begin{bmatrix} 5 \times 10^1 & 0 & 0 & 0 \\ 0 & 8 \times 10^5 & 0 & 0 \\ 0 & 0 & 2.8 & 0 \\ 0 & 0 & 0 & 1.1 \times 10^8 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 9 \times 10^7 \end{bmatrix}$$

These weights give heavy penalties to rapid changes in direction, whilst promoting large amounts of thrust to compensate for the tracking differences in the y direction.

3.2.5 Controller Implementation

The control law for Scenario 2 combines state feedback with proportional navigation guidance, using the augmented LQR gain K_{aug} computed using Equation 3. The controller fist gets the relative position and velocity between the asteroid and rocket as follows:

$$\mathbf{R} = \begin{bmatrix} x_a - x_r \\ y_a - y_r \end{bmatrix}, \quad \mathbf{V}_r = \begin{bmatrix} \dot{x}_a - \dot{x}_r \\ \dot{y}_a - \dot{y}_r \end{bmatrix}$$

Using these vectors, the LOS angular rate ω_z is calculated as:

$$\omega_z = \frac{R_x V_{ry} - R_y V_{rx}}{R_x^2 + R_y^2}$$

This rate is used to compute the necessary acceleration vector that steers the rocket to collide with the asteroid [1]. The acceleration vector is calculated as:

$$\mathbf{a} = N \cdot (\mathbf{V}_r \times \omega_z)$$

Here N is the navigation constant and is chosen to be 5. The tracking error, which is the velocity error, is then estimated using the current simulation time t as follows:

$$\mathbf{v}_{error} = \mathbf{a} \cdot t - \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \end{bmatrix}$$

Finally, the full control law uses the augmented LQR gains to compute the input vector:

$$u = u_{hover} - K_{state} \cdot \begin{bmatrix} \dot{\theta} \\ \theta \end{bmatrix} + K_{velocity} \cdot \mathbf{v}_{error}$$

with:

$$u_{hover} = \begin{bmatrix} m_R g \\ 0 \end{bmatrix}$$

3.3 Scenario 3

As seen in subsection 1.2, Scenario 3 requires an additional constraint on the angle at which the rocket detonates near the asteroid. To achieve this, the same control scheme used in Scenario 2 is used, however, it uses a unique set of LQR weights and includes extra code that adjusts the acceleration according to the *impact angle* by computing the angular difference between the rocket's approach vector and the asteroid's orientation. If this difference falls outside the acceptable bounds of $\pm 30°$ from the front or rear of the asteroid, a corrective term is applied to the acceleration vector. This term steers the rocket sideways by generating a perpendicular compensator vector, scaled proportionally to the angular difference. This adjustment helps redirect the rocket to an allowable impact zone and can be seen in Listing 1.

```
if ~angle_impact
    turn = sign(angle_diff);  % Turning direction (- = left, + = right)
    comp = turn*([0 -1; 1 0]*relative_pos)/norm(relative_pos);% Get perpendicular compensator vector
    scale = min(abs(angle_diff_deg) / 30, 1); % Scale based on the angular difference
    a = a + (scale * 5 * comp); % Compute the final adjustment vector and apply it to acceleration
end
```

Listing 1: Scenario 3 Angle Compensator Code

4 Results and Discussion

To test the controllers, the simulation was repeated 100 times with different seeds for the random number generator applied to the values in Table 1.

4.1 Scenario 1

The controller for Scenario 1 was fairly robust and had a failure rate of approximately 16%. The rocket mostly reached within $150m$ of the asteroid, but would occasionally overshoot the asteroid and miss it. This is likely because the rocket had to respond quickly, with a lot of thrust, to the asteroid's height if it were to intercept it by only flying vertically.

4.2 Scenario 2

The controller for Scenario 2 was extremely robust and had a failure rate below 2%. The rocket repeatedly flew within $150m$ of the asteroid despite the varying starting conditions and process noise. Proportional navigation is a very robust and commonly used control law, so this is to be expected. These results also indicate that the LQR weights were chosen well for the given scenario.

4.3 Scenario 3

The controller for Scenario 3 was not robust and had a failure of approximately 69%. The rocket would fly within $150m$ of the asteroid approximately 93% but would seldom be able to correct itself enough to arrive within the target angle. These requirements are difficult to achieve since the rocket, under a feedback control scheme, has a very slow response time compared to the possible rotation speeds of the asteroid.

5 Conclusion

This report uses the Lagrangian method to present a full dynamic model of a 2D rocket. The equations of motion were derived and expressed in the standard manipulator form, enabling control-oriented formulation. The asteroid dynamics were modelled to estimate the drag coefficient c using filtered simulation data and numerical differentiation. The final estimated drag coefficient was approximately $c \approx 94.19$.

Three control strategies were developed and tested. Scenario 1 used a simple linear state-feedback controller constrained to vertical motion. While it was relatively effective, its limited flexibility resulted in a 16% failure rate, primarily due to overshooting. Scenario 2 introduced a more advanced proportional navigation control method, combined with an LQR-based state-feedback controller in an augmented state space. This approach proved highly robust, with a failure rate below 2%, demonstrating accurate tracking and adaptability to noisy conditions. This validates the use of proportional navigation and LQR design for dynamic interception tasks with non-linear conditions. Scenario 3 added angular constraints to the impact conditions. It extended the Scenario 2 controller with directional compensators, but it struggled to consistently satisfy the angular requirements, leading to a 69% failure rate.

5.1 Recommendations

Scenario 1 could have benefited from some form of derivative control, where the rocket pre-emptively slows down when it starts nearing the height of the asteroid, however, this would have been difficult since the rocket thrusters can only produce forward thrust and can not reverse. Additionally, if the proportional navigation method used in Scenario 2 had been implemented for this scenario, it would have likely achieved better results.

Scenario 3 could have been improved using a predictive control method such as Model Predictive Control (MPC) or Trajectory Optimisation. Both of these methods could be used to predict where the asteroid would face to find a suitable path for the rocket to follow.

References

[1] Wikipedia contributors, "Proportional navigation — Wikipedia, The Free Encyclopedia," 2024, [Online; accessed 19-May-2025]. [Online]. Available: https://en.wikipedia.org/wiki/Proportional_navigation#References

[2] N. F. Palumbo, R. A. Blauwkamp, and J. M. Lloyd, "Basic principles of homing guidance," *Johns Hopkins APL Technical Digest*, vol. 29, no. 1, pp. 25–41, 2010.

Appendix

A Appendix: Simulink Controller Code

A.1 Scenario 1

```
function [F, alpha, Krr] = fcn(y, dy, th, ast_y)

%F = 1000000;
alpha = 0;
mR = 1000;
g = 9.81;

X = [dy;y];

% Linearised state space

A = [0 0; 1 0];
B = [1/mR; 0];
C = [0 1];
D = 0;

Kpp = [180   20];

Acl_k = (A-B*Kpp);
Krr = 1/(D-C*(Acl_k \ B));

Kpp_x = Kpp*X;

F_array = Krr * ast_y - Kpp_x;
F = F_array(1)+g*mR;

end
```

Listing 2: Scenario 1 Controller

A.2 Scenario 3

```
function [F, alpha] = PropNavController(X_r, X_ast, sim_time, Scenario)

    K_aug_local = [0,0,0,0;
                   0,0,0,0];

%   Rocket values
    x = X_r(1);
    y = X_r(3);
    th = X_r(5);
    dx = X_r(2);
    dy = X_r(4);
    dth = X_r(6);
```

```
13
14  %   Asteroid values
15      ast_x = X_ast(1);
16      ast_y = X_ast(3);
17      ast_th = X_ast(5);
18      ast_dx = X_ast(2);
19      ast_dy = X_ast(4);
20      ast_dth = X_ast(6);
21
22  %   F = 1000000;
23  %   alpha = 0;
24      mR = 1000;
25      g = 9.81;
26
27  %   K_aug_local is the augemnent proportional gain determined in
28  %   EEE4119F_ControllerDesign_VWLLUK003.m using LQR in an augmentate state
29  %   space
30
31      if Scenario == 1
32          K_aug_local = [1.50363726552080e-15 1.93590543037203e-13  2.57292235584690e-16
        ↪ -28.2842712474619;
33                         -1.19751777876960 -0.727169682213816  -0.00122474487139158
        ↪ -8.15024582273976e-17];
34
35      elseif Scenario == 2
36          K_aug_local = [2.79560157053078e-12 1.09287997693191e-12  2.71996135563450e-15
        ↪ -10488.0884817015;
37                         -0.445301430405698  -0.102126970188803  -0.000176383420737637
        ↪ -8.92746691656357e-16];
38
39      elseif Scenario == 3
40
41          K_aug_local = [1.50363726552080e-15 1.93590543037203e-13  2.57292235584690e-16
        ↪ -28.2842712474619;
42                         -1.19751777876960 -0.727169682213816  -0.00122474487139158
        ↪ -8.15024582273976e-17];
43
44      end
45
46  %   Range from missile to target
47
48      R = [ast_x; ast_y] - [x; y];
49
50  %   Target velocity relative to missile velocity
51
52      Vr = [ast_dx; ast_dy] - [dx; dy];
53
54  %   The Line Of Sight (LOS) rotation rate is found by dividing the 2D cross
55  %   product of R and Vr, by the dot product of R with itself. Returns a
56  %   scalar in the z-axis [0  0 omega_z].
57
58      omega_z = (R(1)*Vr(2) - R(2)*Vr(1)) / dot(R, R);
59
```

```
%   The acceleration normal to the instantaneous velocity is found by taking
%   the cross product of the relative velocity with the LOS rotation rate
%   and scaling it with a proportionality constant.

    a = 5 * [-Vr(2)*omega_z; Vr(1)*omega_z];

%   The impact angle is accounted for by determining the angle difference
%   between rocket heading and the the asteroid front.

    relative_pos = [x - ast_x; y - ast_y];
    relative_angle = atan2(relative_pos(2), relative_pos(1));
    angle_diff = wrapToPi(relative_angle - ast_th);
    angle_diff_deg = rad2deg(angle_diff);

%   The angle difference is compare to the acceptable impact angles:
%     30   front,   30   back =  180    30

    angle_front = abs(angle_diff_deg) <= 30;
    angle_back = abs(wrapTo180(angle_diff_deg - 180)) <= 30;
    angle_impact = angle_front || angle_back;

%   If the angle impact is not within an acceptable values, adjust the
%   path of the rocket by applying a compensator term to the acceleration
%   term.

if ~angle_impact

    % Determine the turning direction (- = left, + = right)
    turn = sign(angle_diff);

    % Compute the perpendicular compensator vector and normalize it
    comp = turn * ([0 -1; 1 0] * relative_pos) / norm(relative_pos);

    % Scale the adjustment based on the angular difference (max of 1)
    scale = min(abs(angle_diff_deg) / 30, 1);

    % Compute the final adjustment vector and apply it to acceleration
    a = a + (scale * 5 * comp);
end

%   Estimated velocity error

    v_error = a .* sim_time - [dx; dy];

%   Control law system

    u_hover = [mR * g; 0];
    state_feedback = K_aug_local(:,1:2) * [dth; th];
    velocity_feedback = K_aug_local(:,3:4) * v_error;

    u_ctrl = u_hover - state_feedback + velocity_feedback;

    F_ctrl = u_ctrl(1);
```

```
113     alpha_ctrl = u_ctrl(2);
114
115
116 %   Ensuring it still operates within its limits. The maximum thrust
117 %   force was observed from the maximum slope of the rocket's velocity.
118
119 if  F_ctrl > 37000
120         F_ctrl = 37000;
121
122 elseif  F_ctrl < 0
123         F_ctrl = 0;
124
125 end
126 max_alpha = 1.5;
127 if  alpha_ctrl > max_alpha
128         alpha_ctrl = max_alpha;
129
130 elseif  alpha_ctrl < -max_alpha
131         alpha_ctrl = -max_alpha;
132
133 end
134
135     F = F_ctrl;
136     alpha = alpha_ctrl;
137
138
139 end
```

Listing 3: Scenario 3 Controller