Department of Aerospace Engineering

AE - 219: Supervised Learning Project

Cooperative Pursuit and Evasion

Instructor

Prof. Shashi Ranjan Kumar,
Department of Aerospace Engineering

Author

Anjali Rawat 190010007

Contents

1 Abstract

This report contains implementation of aircraft protection guidance using Line of sight approach and Sliding Mode control. Aircraft protection guidance in this report involves formulating and implementing a one way cooperation guidance strategy for the protection of an aircraft from an attacker aircraft using a defender aircraft. In the end this report also touches upon the application of the above approach for orbital pursuit and evasion in space using concepts of Zero Effort Miss and Sliding mode control.

2 Introduction

Aircraft protection guidance is a crucial problem specially for military purposes. Consider the scenario where you have to run away from a bandit and you have a bow and arrow. You will shoot the arrow at the attacker at an angle that it intercepts the path of the attacker and kills it or alternatively try to distract the attacker by introducing disturbances in its path. In this scenario you are the target, the arrow is the defender and the bandit is the attacker. Similar strategies were also previously used for aircraft protection where the aircraft used to launch missiles, chaffs or flares to protect itself. But these methods are not sufficient
An advancement to this type of protection could be if our bow was such that it could change its trajectory on its own and ensure that it hits the bandit. That is an active defender missile which using the data received from the aircraft and limited attacker data (measured and estimated), it could actively change its trajectory to ensure interception in finite time. This is the advancement we will be looking forwards being discussed and worked upon in this report.

3 Literature Review

In order to understand the concept, some papers related to Pursuit and Evasion were read. It was found that there could be a lot of ways of solving the problem of Pursuit and Evasion. It can be solved using optimal control, multi objective optimization, game theory etc. [1] views the problem as an aircraft protection problem where it makes use of simple geometry and Sliding Mode Control to achieve the desired output. [2] views this problem as a three body game where each body has its own objective related to minimizing or maximizing distance from one another. It makes use of multi objective theory, Nash equilibrium and Riccati equations. [3] views it as a differential game where it is solved using variational techniques.
Due to complexity of other approaches like game theory and lack of background in control or optimal control to start with, it was considered better to shift the focus towards [1] in order to learn and understand the concepts of such problems.

4 Sliding Mode Control

The first step was to understand Sliding Mode control due to lack of background in controls. Several material was read in order to understand SMC.
Sliding mode control (SMC) is a nonlinear control technique with great properties of accuracy, robustness, and easy tuning and implementation. In SMC the system states are driven onto a particular surface in the state space, named sliding surface. Once the sliding surface is reached, sliding mode control keeps the states on the close neighbourhood of the sliding surface. Hence the sliding

mode control is a two part controller design. The first part involves the design of a sliding surface so that the sliding motion satisfies design specifications. The second is concerned with the selection of a control law that will make the switching surface attractive to the system state. SMC allows for controlling nonlinear processes subject to external disturbances and heavy model uncertainties.

There are two main steps to setting up a sliding mode control:
Step 1: Defining a Sliding surface
Step 2: Designing a control law that drives the controlled variable to its reference value.

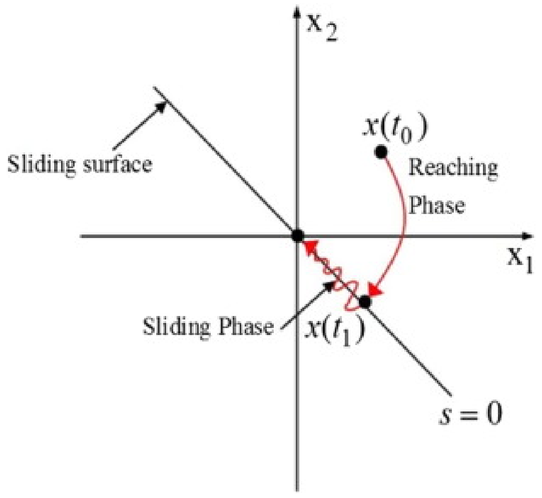


Figure 1: Sliding mode control

Main drawback of sliding mode control is chattering due to high frequency switching and it gets difficult to implement in mechanical systems due to high frequency switching of control input.

4.1 Implementing Sliding Mode Control

Problem:
Define a sliding surface and appropriate control input u to achieve the desired state:

$$x \to 0$$
$$\dot{x} \to 0$$

where x is the error in position and $\dot{x}$ is the velocity and $\ddot{x} = u$

State space equations:
Let,

$$x_1 = x$$
$$x_2 = \dot{x}$$

Therefore,

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = u$$

Step 1: Define a sliding surface
Define an S such that when $S \to 0$ then $x_1, x_2 \to 0$

Let us consider the simplest sliding surface-

$$S = x_2 + cx_1 \tag{1}$$

$$S = \dot{x} + cx$$

where c is arbitrary constant and c>0.

Justification for choosing this sliding surface:
When $S = 0$

$$\dot{x} + cx = 0$$

Solution to this ODE is:

$$x = x(0)e^{-ct}$$

where c will define the rate of exponential decay and x(0) is the initial value of x.

Step 2: Designing a control law

The sliding-mode controller comprises two parts:

$$u = u^{eq} + u^{disc} \tag{2}$$

i.e. an equivalent and a discontinuous part.
Equivalent part can be derived from the equation:

$$\dot{S} = 0 \tag{3}$$

$$u^{eq} = -cx_2 \tag{4}$$

Discontinous part for the sliding mode control can be defined as:

$$u^{disc} = -Msgn(S) \tag{5}$$

Therefore our control law for sliding mode control is:

$$u = -cx_2 - Msgn(S) \tag{6}$$

Our control law u should be such that it drives the sliding surface S to 0.
Applying the concept of Lypanov stability, γ represents energy.

$$\gamma = \frac{S^2}{2} \tag{7}$$

For the system to be stable,

$$\dot{\gamma} < 0 \implies \dot{S}S < 0 \tag{8}$$

Analysing the above equation,
If $S < 0, \dot{S} > 0 \implies$ it is driving $S \to 0$
If $S > 0, \dot{S} < 0 \implies$ it is driving $S \to 0$
In both conditions $S \to 0$ which is our desired result.

If S $< -$ve number $\implies$ faster convergence
Substituting,

$$\begin{aligned} S &= x_2 + cx_1 \\ \dot{S} &= \dot{x_2} + c\dot{x_1} \\ &= u + cx_2 \end{aligned}$$

$$(x_2 + cx_1)(u + cx_2) < 0$$

Substitute, equation(6)

$$S(-Msgn(S)) < 0$$

$$-M|S| < 0$$

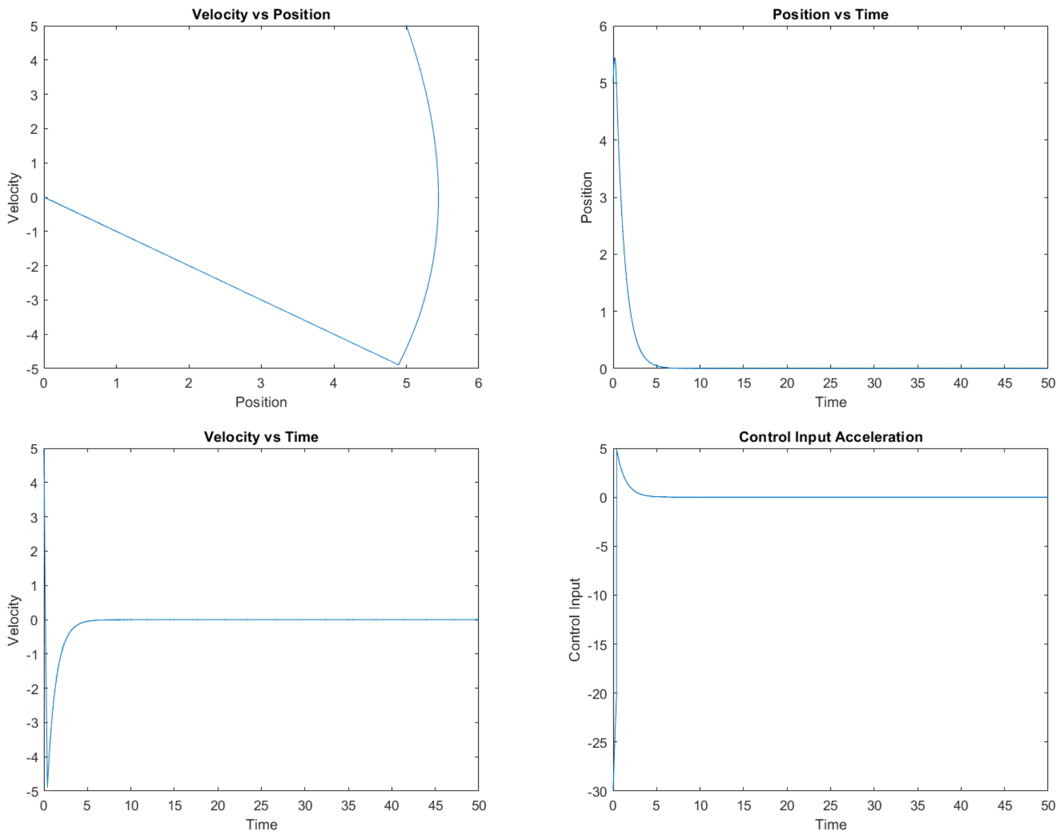
$$\implies M > 0 \ \ (\forall S \neq 0)$$

4.2 Results

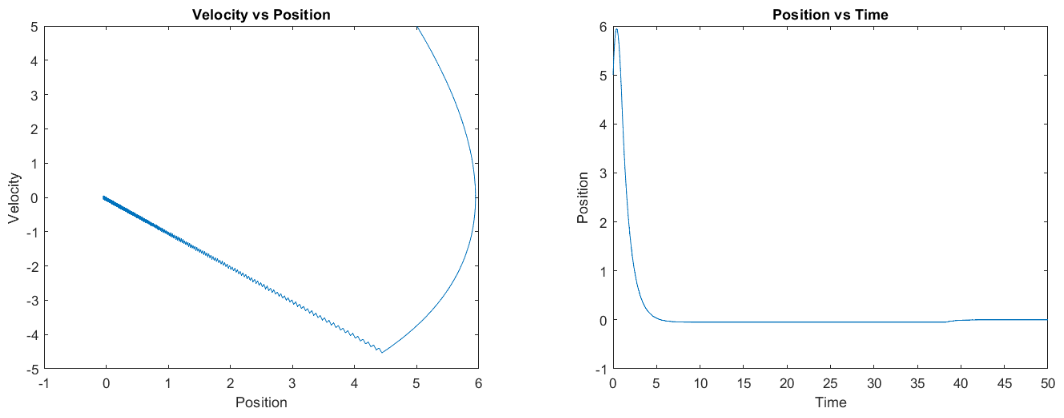
(MATLAB code 1 attached in Appendix)

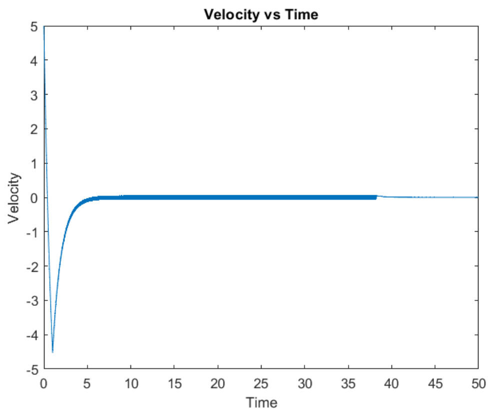
Initial conditions:
$x_1 = 5$
$x_2 = 5$
Time step: h = 0.01
Controller gain M = 25
Sliding surface constant: c = 1
Final time: $t_f = 50$

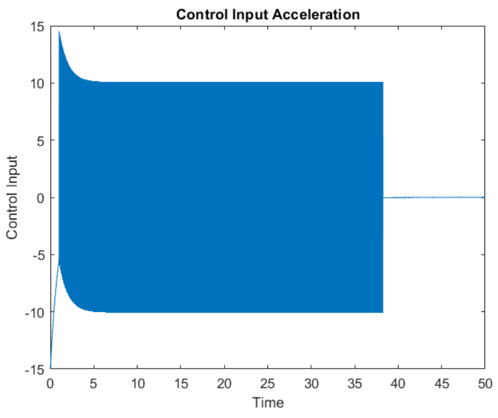
Below are graphs obtained:



If M = 10:
We can observe chattering as seen below:







4.3 Observations

- In the Velocity vs Position graphs we can see that the value of the states first reach the sliding surface from their initial values and then moves along the sliding surface to converge to 0. By varying the value of c we can vary the slope of the sliding surface
- In the Position vs time graph we can see the exponential decay of the position as predicted by the solution to the ODE
- In the Velocity vs time graph we can see the convergence and stabilization of the velocity to 0 value
- For gain M = 25, chattering is considerably reduced for this system therefore we can say that M =25 is an optimal gain for this system. While for other values of gain like M= 10 we can clearly observe a lot of chattering making it difficult to implement practically
- The greater the gain M, faster is the convergence

4.4 Reduction of chattering

We can reduce chattering by:

- Selecting the optimal value of gain
- By replacing the discontinuous function sign(M) by a smooth approximate function like tanh(M) or sat(M) (saturation function).

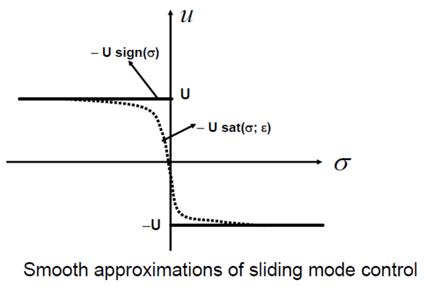


Figure 2: Smooth approximate function

5 Aircraft Protection Guidance using LOS Approach

The aim is to design a guidance strategy for the successful interception of the attacker aircraft using defender aircraft before it reaches the target aircraft. In this report we perform this using the Line of Sight approach with the help of Sliding Mode Control. Our focus is only on one way cooperation in which the aircraft can perform its own maneuvers irrespective of the maneuvers of the defender. Though all information is conveyed to the defender.
In two way cooperation the aircraft and defender perform maneuvers that will minimize control action of both vehicles.

5.1 Line of Sight Approach

The line joining two bodies is called the Line of Sight of those two bodies. If the defender is able to reach the line of sight of the attacker and the target and maintain its position between the attacker and the target on their LOS for a certain period of time, we can guarantee interception of attacker before it reaches the target. This is called the LOS approach.
For reaching the LOS and maintaining that state till interception occurs, we have used SMC.

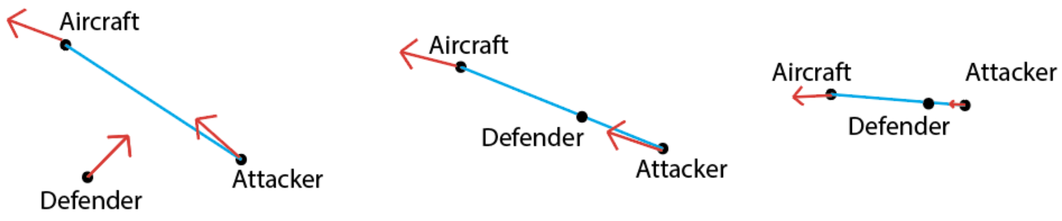


Figure 3: LOS approach

5.2 Problem Formulation

Let us consider a 3 body engagement of point masses in a 2D plane as shown in the figure below. There is an aircraft being chased by an attacker aircraft/missile. To protect itself from the attacker, the aircraft launches a defender missile which will intercept the path of the attacker and destroy it before it reaches the aircraft. The coordinate system used is the inertial Cartesian coordinate system.

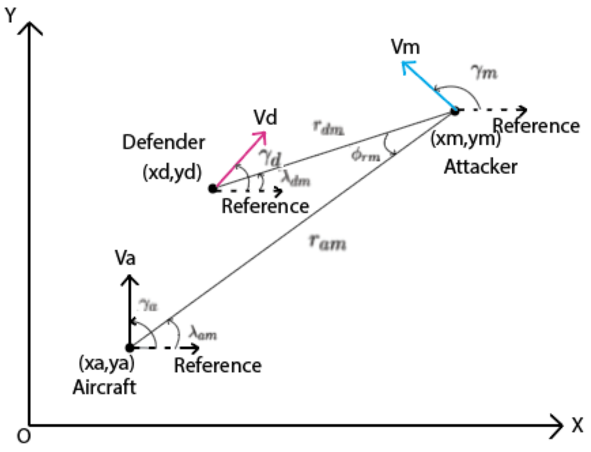


Figure 4: Three body engagement

Notations used:
a, d, m : Subscripts for aircraft(target), defender, attacker
$(x_a, y_a), (x_d, y_d), (x_m, y_m)$: x and y coordinates of aircraft, defender, attacker
V_a, V_d, V_m : Velocities of target, defender, and attacker
$\gamma_a, \gamma_d, \gamma_m$: Flight path angles of target, defender, and attacker
$\lambda_{am}, \lambda_{dm}$: LOS angle between aircraft and attacker, defender and attacker
r_{am}, r_{dm} : Relative distance between aircraft and attacker, defender and attacker
ϕ_{rm} : Difference between the LOS angles

Below are some key points and assumptions considered in this formulation:

- There is a bound to the maximum acceleration of all the three bodies
- The acceleration of the attacker or the guidance strategy of the attacker is unknown to the defender or the aircraft. Only the bound on the acceleration of attacker is known
- The velocities of the three bodies are constant since there is no forwards acceleration. Only lateral acceleration is present
- $V_d \approx V_m > V_a$, which is a practical assumption as the attacker and defender are usually missiles which can reach very high speed while aircraft contains a human pilot who cannot withstand such high speeds
- The lethal radius of defender missile is non zero $\implies r_{dm}$ is non zero at interception

Now we will design a guidance strategy for the aircraft and defender team using non linear Sliding Mode Control.

Defining relative distances in terms of Cartesian coordinates:

$$r_{am} = \sqrt{(x_a - x_m)^2 + (y_a - y_m)^2} \tag{9}$$

$$r_{dm} = \sqrt{(x_d - x_m)^2 + (y_d - y_m)^2} \tag{10}$$

Defining LOS angles in terms of Cartesian coordinates:

$$\tan \lambda_{am} = \frac{y_m - y_a}{x_m - x_a} \tag{11}$$

$$\lambda_{am} = tan^{-1} \frac{y_m - y_a}{x_m - x_a} \tag{12}$$

$$Similarly, \lambda_{dm} = tan^{-1} \frac{y_m - y_d}{x_m - x_d} \tag{13}$$

Expression for range rates:

$$\dot{r}_{dm} = V_m cos(\gamma_m - \lambda_{dm}) - V_d cos(\gamma_d - \lambda_{dm}) \tag{14}$$

$$\dot{r}_{am} = V_m cos(\gamma_m - \lambda_{am}) - V_a cos(\gamma_a - \lambda_{am}) \tag{15}$$

Expressions for LOS rates:

$$r_{dm}\dot{\lambda}_{dm} = V_m sin(\gamma_m - \lambda_{dm}) - V_d sin(\gamma_d - \lambda_{dm}) \tag{16}$$

$$r_{am}\dot{\lambda}_{am} = V_m sin(\gamma_m - \lambda_{am}) - V_d sin(\gamma_a - \lambda_{am}) \tag{17}$$

Expression for heading angles: (where i = a,d,m)

$$\dot{\gamma}_i = \frac{a_i}{V_i} \qquad where, |a_i| \leq a_i^{max} \tag{18}$$

The objective is to reach and maintain the position of the defender on the LOS of the attacker and target. This can be achieved if ϕ_{rm} reaches 0 and remains 0 for a certain time.

$$\phi_{rm} = \lambda_{am} - \lambda_{dm} \tag{19}$$

This is achieved using SMC.
Step 1: Sliding surface can be defined as below:

$$S = \dot{\phi}_{rm} + \eta\phi_{rm} \qquad where, \eta > 0 \tag{20}$$

Step 2: Designing a control law:
For finding the equivalent control input put $\dot{S} = 0$

$$\dot{S} = \ddot{\phi}_{rm} + \eta\dot{\phi}_{rm} \tag{21}$$

$$\dot{\phi}_{rm} = \dot{\lambda}_{am} - \dot{\lambda}_{dm} \tag{22}$$

$$\ddot{\phi}_{rm} = \ddot{\lambda}_{am} - \ddot{\lambda}_{dm} \tag{23}$$

$$\ddot{\lambda}_{dm} = \frac{-2\dot{r}_{dm}\dot{\lambda}_{dm}}{r_{dm}} - \frac{cos(\gamma_d - \lambda_{dm})a_d}{r_{dm}} + \frac{cos(\gamma_m - \lambda_{dm})a_m}{r_{am}} \tag{24}$$

$$\ddot{\lambda}_{am} = \frac{-2\dot{r}_{am}\dot{\lambda}_{am}}{r_{am}} - \frac{cos(\gamma_a - \lambda_{am})a_a}{r_{am}} + \frac{cos(\gamma_m - \lambda_{am})a_m}{r_{am}} \tag{25}$$

Using equations (21) to (25):

$$\ddot{\phi}_{rm} = \frac{-2\dot{r}_{am}\dot{\lambda}_{am}}{r_{am}} + \frac{2\dot{r}_{dm}\dot{\lambda}_{dm}}{r_{dm}} - \frac{cos(\gamma_a - \lambda_{am})a_a}{r_{am}} + \frac{cos(\gamma_d - \lambda_{dm})a_d}{r_{dm}} + d$$

$$d = (\frac{cos(\gamma_m - \lambda_{ad})}{r_{am}} - \frac{cos(\gamma_m - \lambda_{dm})}{r_{dm}})a_m \tag{26}$$

$$|d| = |(\frac{cos(\gamma_m - \lambda_{ad})}{r_{am}} - \frac{cos(\gamma_m - \lambda_{dm})}{r_{dm}})a_m| \tag{27}$$

$$|d| \leq (\frac{cos(\gamma_m - \lambda_{ad})}{r_{am}} - \frac{cos(\gamma_m - \lambda_{dm})}{r_{dm}})a_m^{max} \tag{28}$$

$$|d| \leq (\frac{1}{r_{am}} - \frac{1}{r_{dm}})a_m^{max} \tag{29}$$

$$|d| \leq d_{max} \tag{30}$$

The terms containing a_m are termed as disturbance since the value of a_m is unknown to the defender and the aircraft. Only a_m^{max} is known.

$$\dot{S} = \frac{-2\dot{r}_{am}\dot{\lambda}_{am}}{r_{am}} + \frac{2\dot{r}_{dm}\dot{\lambda}_{dm}}{r_{dm}} - \frac{cos(\gamma_a - \lambda_{am})a_a}{r_{am}} + \frac{cos(\gamma_d - \lambda_{dm})a_d}{r_{dm}} + d + \eta\dot{\phi}_r m \tag{31}$$

We ignore the disturbance term d, since that is not known to us.

$$a_d^{eq} = \frac{r_{dm}}{cos(\gamma_d - \lambda_{dm})}[\frac{-2\dot{r}_{am}\dot{\lambda}_{am}}{r_{am}} + \frac{2\dot{r}_{dm}\dot{\lambda}_{dm}}{r_{dm}} - \frac{cos(\gamma_a - \lambda_{am})a_a}{r_{am}} + \eta(\dot{\lambda}_a m - \dot{\lambda}_d m)] \tag{32}$$

We choose the discontinuous control to be:

$$a_d^{disc} = -\frac{M}{cos(\gamma_d - \lambda_{dm})}sign(S) \qquad where, M > 0 \tag{33}$$

Using the equation $S\dot{S} < 0$ from Lyapunov function we arrive at the following bound on M:

$$M > r_{dm}d_{max} \tag{34}$$

Hence the acceleration of the defender can be defined as:

$$a_d = a_d^{eq} + a_d^{disc} \tag{35}$$

$$a_d = -\left(\frac{2\dot{r}_{am} - \eta r_{dm}}{cos(\gamma_d) - \lambda_{dm}}\right)\dot{\lambda}_{dm} + \frac{r_{dm}}{r_{am}}\left(\frac{2\dot{r}_{am} - \eta r_{dm}}{cos(\gamma_d) - \lambda_{dm}}\right)\dot{\lambda}_{am} + \frac{r_{dm}}{r_{am}}\frac{cos(\gamma_a - \lambda_{am})}{cos(\gamma_d - \lambda_{dm})}a_a - \frac{M_1}{cos(\gamma_d - \lambda_{dm})}sign(S) \tag{36}$$

By analysing this equation we can see that the defender totally relies on the aircraft information at the beginning as $r_{dm} = r_{am}$. As it goes far away from the attacker $\frac{r_{dm}}{r_{am}}$ will tend to zero which means that the defender doesn't give much weight to the aircraft information during the end and relies on its own information.

5.3 Results

We need to propagate 9 variables using RK4:
x_a, y_a, γ_a
x_m, y_m, γ_m
x_d, y_d, γ_d

For the purpose of simulation and propagation of attacker we need to consider a guidance strategy for the attacker aircraft which will not be known to the defender or aircraft. Here we will consider that the attacker moves according to PN guidance.

PN guidance: It dictates that the missile velocity vector should rotate at a rate propotional to the rotation rate of the LOS and in same direction.

$$a_n = N\dot{\lambda}V \tag{37}$$

N is multiplied since the velocity vector should rotate faster than the LOS.

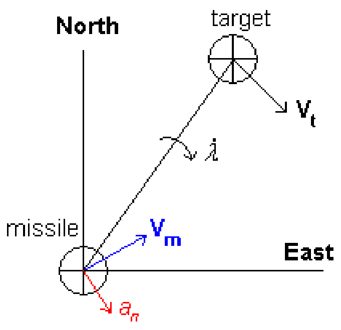


Figure 5: Propotional Navigation (PN)

For attacker aircraft we consider the following PN guidance:

$$a_a = 3V_{cm}\dot{\lambda}_m \tag{38}$$

$$a_a = 3\dot{r}_m\dot{\lambda}_m \tag{39}$$

MATLAB code 2 is given in Appendix. It compares guidance strategies when aircraft acceleration is 0 and when aircraft moves according to Propotional Navigation.

Parameters:
$M = 0.25$
$\eta = 10$
$V_a = 100$
$V_d = 200$
$V_m = 200$
$a_d^{max} = a_m^{max} = 20g$
$a_A^{max} = 10g$
$(x_a, y_a) = (0, 0)$
$(x_d, y_d) = (0, 0)$
$(x_m, y_m) = (5000, 0)$
$\lambda_{am} = 0$
$\gamma_m = 150$
$\gamma_a = 45$
$\gamma_d = 0$
$N = 3$

Below are the graphs obtained:

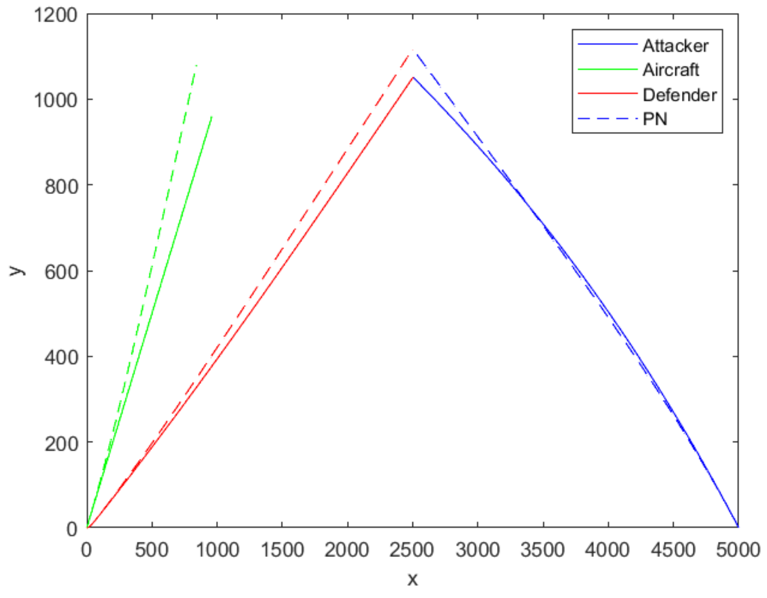
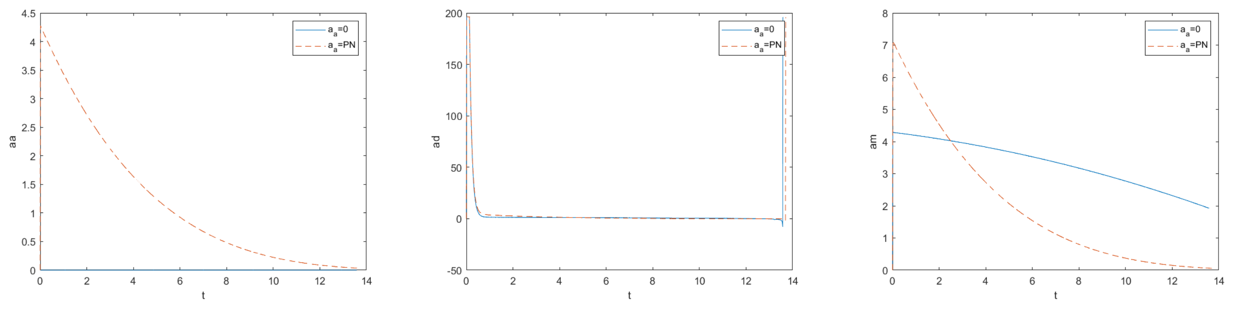


Figure 6: Interception of the attacker, planar arrangement



Graph when aircraft has non zero acceleration:

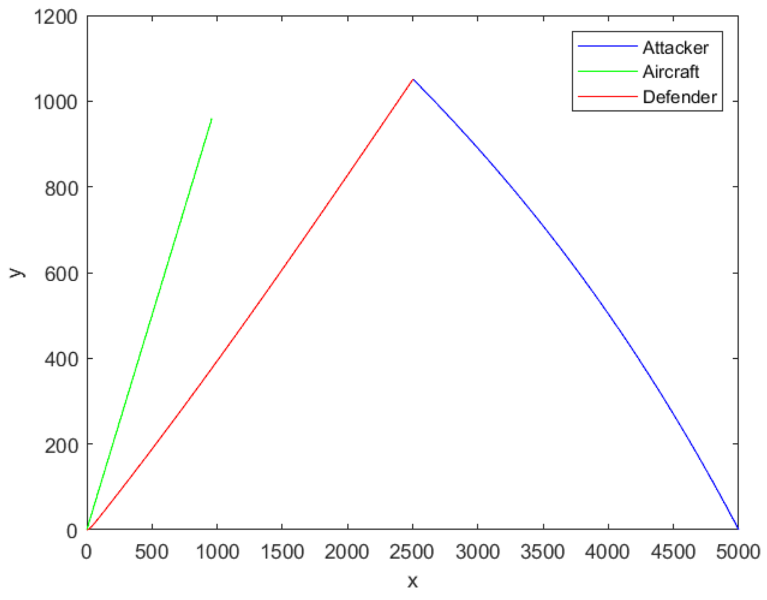


Figure 7: Interception of the attacker, $aa = 2m/s^2$

We can see that the defender is successful in intercepting the attacker before it reaches the aircraft.

6 Variations of the problem

The different ways in which we can extend this problem is:

- Using a controller other than SMC for the LOS approach
- Hitting at an angle for greater impact i.e. maintaining a certain value of λ_{dm} and λ_{am} which is obtained by maintaining certain value of sliding surface S

- Other geometric ways to intercept apart from LOS approach, for eg. a circle or ellipse. LOS approach requires more maneuver capabilities hence it is a subset of a circle which is a subset of an ellipse
- Obstacle avoidance
- Multi body aircraft pursuit and evasion
- Time constrained problem
- Orbital pursuit and evasion

7 Orbital Pursuit and Evasion

Let us take the same scenario in space and replace aircrafts with spacecrafts. The target spacecraft is moving in a circular orbit and is considered to be non maneuverable. The attacker spacecraft follows the target spacecraft in the same orbit with the help of a tangential and inwards radial thrust. The defender is in a smaller circular orbit and has to reach the attacker before the attacker reaches the target. The defender can only measure and estimate the data of the attacker but doesn't know the attackers guidance strategy. The defender makes use of Zero effort Miss and Sliding mode control to intercept the attacker.

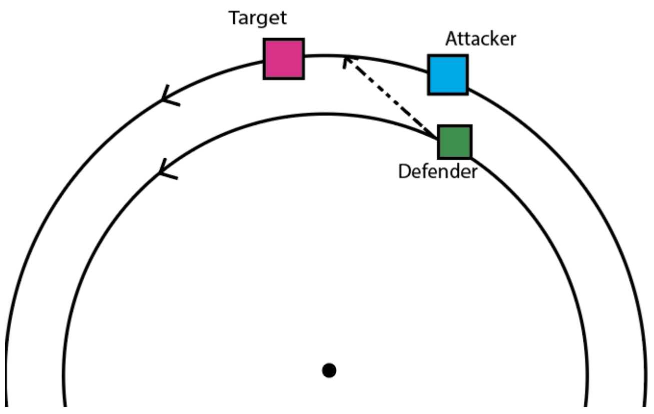


Figure 8: Rough schematic of orbital pursuit and evasion

The dynamics of the system become non linear in space, hence we need to make certain assumptions:

- The target, attacker and the defender are defined to close to each other so that linearization of dynamics is possible and Clohessy–Wiltshire equations hold.
- The target is in a circular orbit
- The gravity is considered to be constant
- Neglect perturbations

7.1 Orbital Frame

Orbital frame is used when the orbit of a body around another body is assumed to be circular. The origin is fixed to the orbiting body. The x axis points in the direction of velocity of velocity of orbiting body, z axis is directed from the orbiting body to the center of the body around which it orbits, and the y axis completes the right handed co-ordinate system (thus forming normal vector to the orbital plane). It is a rotating frame hence a non-inertial frame.

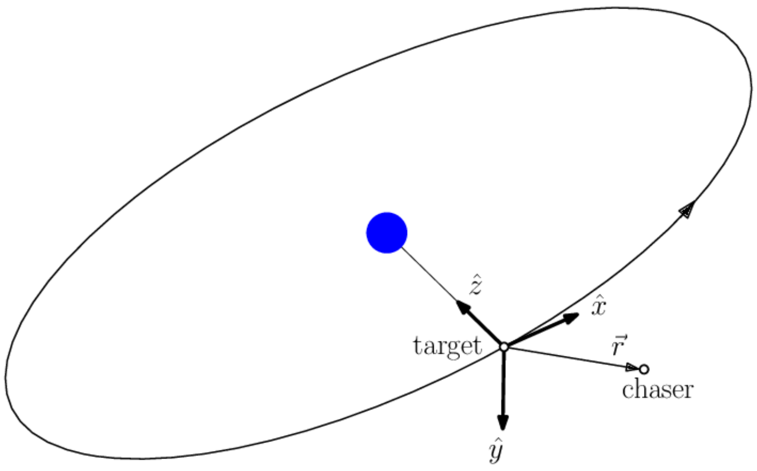


Figure 9: Orbital frame

7.2 ZEM

ZEM is the perpendicular distance by which the defender will miss the attacker if no control input is applied. Once we know this error in distance, we can give the required control input to reduce this miss distance.

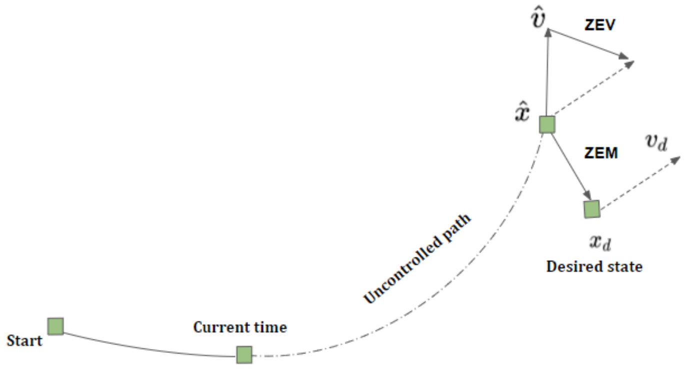


Figure 10: ZEM/ZEV illustration

7.3 Problem Formulation

It is equivalent to 3 body engagement in space. A low continous thrust is applied for manuevers. C-W equations are used for dynamics of target, defender and attacker satellite in space.

$$\ddot{x} = 3n^2x + 2n\dot{y} + Tu_x \tag{40}$$

$$\ddot{y} = -2n\dot{x} + Tu_y \tag{41}$$

$$\ddot{z} = -n^2z + Tu_z \tag{42}$$

$$where, n = \sqrt{\frac{\mu}{a_t^3}} \tag{43}$$

T is the maximum thrust
u_x, u_y, u_z are the control variables in x,y,z ranging from 0 to 1.
a_t is the semi major axis of the target.
x, y, z, $\dot{x}, \dot{y}, \dot{z}$ are the position and velocity vector of the chaser with respect to the target in the target's orbit frame.

Future task: he formulation of ZEM and Sliding mode control for this system is the immediate future task of this project.

8 References

References used in making this report are:

1. S. R. Kumar and D. Mukherjee
   Cooperative Active Aircraft Protection Guidance Using Line-of-Sight Approach IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS VOL. 57, NO. 2, APRIL 2021

2. THE LADY, THE BANDIT AND THE BODY-GUARD GAME
   Ilan Rusnak, RAFAEL, P.O.Box 2250, Haifa 31021, Israel

3. Differential Games and Optimal Pursuit-Evasion Strategies
   Y. C. HO, Senior Member, IEEE, A.E. Bryson Jr., and S. Baron
   IEEE TRASSACTIONS ON AUTOMATIC CONTROL, Vol. Ac-10, No. 4, October, 1965

4. Pursuer's Control Strategy for Orbital Pursuit-Evasion-Defense Game with Continuous Low Thrust Propulsion
   Junfeng Zhou, Lin Zhao, Jianhua Cheng *, Shuo Wang and Yipeng Wang

9 Acknowledgements

I would like to express my special thanks of gratitude to my project guide Prof. Shashi Ranjan Kumar who helped me throughout this project and made this learning experience an enriching one.

10 Appendix

10.1 MATLAB code 1: Sliding Mode Control

main_SMC.m

```
x1 = 5;  %Initial value
x2 = 5;  %Initial value

h = 0.01;  %Time step
M = 25;    %Gain, greater gain -faster convergence - but greater control input required
c = 1;     %SMC constant, defines the slope of the sliding surface

t_f = 50;  %Total time
x1_array = double.empty(t_f/h,0);
x2_array = double.empty(t_f/h,0);

i=1;
for loop_index = h:h:t_f        % Run Integrator, step-size = h_RK4, final time = t_f
    x1_array(i) = x1;
    x2_array(i) = x2;
    S(i) = x2 + c*x1;  %Basic sliding surface
    u(i) = -c*x2 - M*sign(S(i));   %Control input
    [x2_new,x1_new] = ma_RK4(x2, x1, h, S(i), M, c);
    x1 = x1_new;                 %updating (i - th) state with (i+1 - th) state
    x2 = x2_new;
    i = i+1;  %Updating the array index
end

t = linspace(0,t_f, t_f/h);

figure(1);
plot(t, x1_array);
title('Position vs Time');
xlabel('Time');
ylabel('Position');


figure(2);
plot(t, x2_array);
title('Velocity vs Time');
xlabel('Time');
ylabel('Velocity');



figure(3);
plot(x1_array, x2_array);
title('Velocity vs Position');
```

```
xlabel('Position');
ylabel('Velocity');


figure(4);
plot(t, u);
title('Control Input Acceleration');
xlabel('Time');
ylabel('Control Input');


figure(5);
plot(t, S);
title('Sliding surface');
xlabel('Time');
ylabel('Sliding surface');
```

RK4.M

```
function [x2_new,x1_new] = ma_RK4(x2, x1, h, S, M, c)
% Runge-Kutta Fourth Order Integrator
        v_a = x2;
        u_a = -c*x2 - M*sign(S);

        v_b = x2 + (h/2)*u_a;
        u_b = -c*v_b - M*sign(S);

        v_c = x2 + (h/2)*u_b;
        u_c = -c*v_c - M*sign(S);

        v_d = x2 + h*u_c;
        u_d = -c*v_d - M*sign(S);

        x2_new = x2 + (h/6) * (u_a + 2*u_b + 2*u_c + u_d);
        %calculating velocity after small time h_RK4
        x1_new = x1 + (h/6) * (v_a + 2*v_b + 2*v_c + v_d);
        %calculating position after small time h_RK4
end
```

10.2 MATLAB code 2: Aircraft Protection Guidance

main_compare.m

```
    c = 10; %sliding mode coefficient
M = 0.25; % will have to check at each interval if gain M> (1+ r_dm/r_am)*a_m_max

%Defining initial values of all parameters
%Constant velocities
V_a = 100;
```

```
V_d = 200;
V_m = 200;
V = [V_a; V_d; V_m];

%Initial position coordinates
x_a = 0;
y_a = 0;
x_d = 0;
y_d = 0;
x_m = 5000;
y_m = 0;

%Initial heading angles
g_a = 45*pi/180;
g_d = 0;
g_m = 150*pi/180;

p = [x_a, y_a ; x_d, y_d ; x_m, y_m]; %2D position array
g = [g_a; g_d; g_m]; %Heading angle array

%LOS angles
l_am = atan((y_m-y_a)/(x_m-x_a));
l_dm = atan((y_m-y_d)/(x_m-x_d));
l = [l_am; l_dm]; %lambda array

%LOS distances
r_am = ((x_a-x_m)^2 + (y_a-y_m)^2)^0.5;
r_dm = ((x_d-x_m)^2 + (y_d-y_m)^2)^0.5;
r = [r_am; r_dm]; %Los distance array

% Rate of change of LOS distance
r_dot_am = V_m*cos(g_m-l_am) - V_a*cos(g_a-l_am);
r_dot_dm = V_m*cos(g_m - l_dm) - V_d*cos(g_d - l_dm);
r_dot = [r_dot_am; r_dot_dm];

% Rate of change of LOS angle
l_dot_am =( V_m*sin(g_m - l_am) - V_a*sin(g_a - l_am) )/r_am;
l_dot_dm =( V_m*sin(g_m - l_dm) - V_d*sin(g_d - l_dm) )/r_dm;
l_dot = [l_dot_am ; l_dot_dm];

%Defining the angle between two LOS
phi = l_am - l_dm;
phi_dot = l_dot_am - l_dot_dm;
x2 = phi_dot;
x1 = phi;

%Defining the sliding surface
S = x2 + c*x1;
```

```
%Lateral Accelerations
a_a = -3*r_dot_am*l_dot_am; %Aircraft accelrarion considered to be 0


% begin propagating
t_f = 15;
h = 0.01;

xm_PN(1) = x_m;
ym_PN(1) = y_m;


n = floor(t_f/h);
time_PN(1) = 0;
time_PN(2) = h;


for ii = 2:n        % Run Integrator, step-size = h_RK4, final time = t_f
    [p, g, l, l_dot, r, r_dot, aa_new, ad_new, am_new, S, phi, phi_dot] =
    RK4_PN_target(p, g, l, l_dot, r, r_dot, a_a, h, c, M, S, V);

    if abs(ad_new) >= 20*9.81
        ad_new = 20*9.81 * sign(ad_new);
    end

    aaa_PN(ii) = aa_new;
    amm_PN(ii) = am_new;
    add_PN(ii) = ad_new;

    Sarray_PN(ii) = S;

    phi_array_PN(ii) = phi;
    phi_dot_array_PN(ii) = phi_dot;

    xm_PN(ii) = p(3,1);
    ym_PN(ii) = p(3,2);
    xa_PN(ii) = p(1,1);
    ya_PN(ii) = p(1,2);
    xd_PN(ii) = p(2,1);
    yd_PN(ii) = p(2,2);


    if r(2)< 1
        break;
    end

    time_PN(ii+1) = time_PN(ii) + h;
```

```
end


%%%%%%%%%%%%%%%%%%%%%%


%Resetting all values for PN target simulation

%Defining initial values of all parameters
%Constant velocities
V_a = 100;
V_d = 200;
V_m = 200;
V = [V_a; V_d; V_m];

%Initial position coordinates
x_a = 0;
y_a = 0;
x_d = 0;
y_d = 0;
x_m = 5000;
y_m = 0;

%Initial heading angles
g_a = 45*pi/180;
g_d = 0;
g_m = 150*pi/180;

p = [x_a, y_a ; x_d, y_d ; x_m, y_m]; %2D position array
g = [g_a; g_d; g_m]; %Heading angle array

%LOS angles
l_am = atan((y_m-y_a)/(x_m-x_a));
l_dm = atan((y_m-y_d)/(x_m-x_d));
l = [l_am; l_dm]; %lambda array

%LOS distances
r_am = ((x_a-x_m)^2 + (y_a-y_m)^2)^0.5;
r_dm = ((x_d-x_m)^2 + (y_d-y_m)^2)^0.5;
r = [r_am; r_dm]; %Los distance array

% Rate of change of LOS distance
r_dot_am = V_m*cos(g_m-l_am) - V_a*cos(g_a-l_am);
r_dot_dm = V_m*cos(g_m - l_dm) - V_d*cos(g_d - l_dm);
r_dot = [r_dot_am; r_dot_dm];

% Rate of change of LOS angle
```

```
l_dot_am =( V_m*sin(g_m - l_am) - V_a*sin(g_a - l_am) )/r_am;
l_dot_dm =( V_m*sin(g_m - l_dm) - V_d*sin(g_d - l_dm) )/r_dm;
l_dot = [l_dot_am ; l_dot_dm];

%Defining the angle between two LOS
phi = l_am - l_dm;
phi_dot = l_dot_am - l_dot_dm;
x2 = phi_dot;
x1 = phi;

%Defining the sliding surface
S = x2 + c*x1;

%Lateral Accelerations
a_a = 0; %Aircraft accelrarion considered to be 0


% begin propagating
t_f = 15;
h = 0.01;

xm(1) = x_m;
ym(1) = y_m;


n = floor(t_f/h);
time(1) = 0;
time(2) = h;


for ii = 2:n        % Run Integrator, step-size = h_RK4, final time = t_f
    [p, g, l, l_dot, r, r_dot, aa_new, ad_new, am_new, S, phi, phi_dot] =
    RK4_new(p, g, l, l_dot, r, r_dot, a_a, h, c, M, S, V);

    if abs(ad_new) >= 20*9.81
        ad_new = 20*9.81 * sign(ad_new);
    end

    aaa(ii) = aa_new;
    amm(ii) = am_new;
    add(ii) = ad_new;

    Sarray(ii) = S;

    phi_array(ii) = phi;
    phi_dot_array(ii) = phi_dot;

    xm(ii) = p(3,1);
    ym(ii) = p(3,2);
    xa(ii) = p(1,1);
    ya(ii) = p(1,2);
    xd(ii) = p(2,1);
    yd(ii) = p(2,2);


    if r(2)< 1
        break;
    end

    time(ii+1) = time(ii) + h;

end



figure();
plot(xm, ym, 'b');
hold on
plot(xa, ya, 'g');
plot(xd, yd, 'r');
plot(xm_PN, ym_PN, 'Color','b', 'LineStyle','--');
plot(xa_PN, ya_PN, 'Color','g','LineStyle', '--');
plot(xd_PN, yd_PN, 'Color','r','LineStyle', '--');
hold off
xlabel('x');
ylabel('y');
legend('Attacker','Aircraft', 'Defender', 'PN')

figure();
plot(time,Sarray);
hold on
plot(time_PN,Sarray_PN, 'LineStyle','--');
hold off
xlabel('t');
ylabel('S');

figure();
plot(time,phi_dot_array);
hold on
plot(time_PN,phi_dot_array_PN, 'LineStyle','--');
hold off
xlabel('t');
ylabel('phi_dot');

figure();
plot(phi_array, phi_dot_array);
```

```
hold on
plot(phi_array_PN, phi_dot_array_PN, 'LineStyle','--');
hold off
xlabel('phi');
ylabel('phi_dot');

figure();
plot(time,amm);
hold on
plot(time_PN,amm_PN,'LineStyle', '--');
hold off
xlabel('t');
ylabel('am');
legend('a_a=0','a_a=PN')

figure();
plot(time,aaa);
hold on
plot(time_PN,aaa_PN , 'LineStyle','--');
hold off
xlabel('t');
ylabel('aa');
legend('a_a=0','a_a=PN')

figure();
plot(time,add);
hold on
plot(time_PN,add_PN,'LineStyle', '--');
hold off
xlabel('t');
ylabel('ad');
legend('a_a=0','a_a=PN')
```

RK4_PN_target

```
    function [p, g, l, l_dot, r, r_dot, aa_new, ad_new, am_new, S, phi, phi_dot] =
    RK4_new(p, g, l, l_dot, r, r_dot, a_aa, h, c, M, S, V)
    aa = -3*l_dot(1)*r_dot(1);
    am = -5*l_dot(1)*r_dot(1);
    ad = -(2*r_dot(2) - c*r(2))*l_dot(2)/cos(g(2) - l(2))
    +  (r(2)/r(1))*(2*r_dot(1) - c*r(1))*l_dot(1)/cos(g(2)-l(2))
    +  (r(2)/r(1))*cos(g(1) - l(1))*aa/cos(g(2)-l(2))
    -  M*sign(S)/cos(g(2)-l(2));

    a_a = [aa ; ad; am];
    a = [aa ;  ad ;  am];
```

```
g_b = g + (h/2)*[a(1)/V(1); a(2)/V(2) ; a(3)/V(3)];
p_b = p + (h/2)*[V(1)*cos(g_b(1)) , V(1)*sin(g_b(1)) ;
V(2)*cos(g_b(2)), V(2)*sin(g_b(2)) ; V(3)*cos(g_b(3)) , V(3)*sin(g_b(3)) ];
l = [atan((p_b(3,2)-p_b(1,2))/(p_b(3,1)-p_b(1,1))) ;
atan((p_b(3,2)-p_b(2,2))/(p_b(3,1)-p_b(2,1)))];
r = [((p_b(1,1)-p_b(3,1))^2 + (p_b(1,2)-p_b(3,2))^2)^0.5 ;
((p_b(2,1)-p_b(3,1))^2 + (p_b(2,2)-p_b(3,2))^2)^0.5];
r_dot = [V(3)*cos(g_b(3)-l(1)) - V(1)*cos(g_b(1)-l(1)) ;
V(3)*cos(g_b(3) - l(2)) - V(2)*cos(g_b(2) - l(2))];
l_dot = [ ( V(3)*sin(g_b(3) - l(1)) - V(1)*sin(g_b(1) - l(1)) )/r(1) ;
( V(3)*sin(g_b(3) - l(2)) - V(2)*sin(g_b(2) - l(2)) )/r(2)];
%S = l_dot(1) - l_dot(2) + c*(l(1) - l(2));

a(1) = -3*l_dot(1)*r_dot(1);
a(3) = -5*l_dot(1)*r_dot(1);
a(2) = -(2*r_dot(2) - c*r(2))*l_dot(2)/cos(g_b(2) - l(2))  +
(r(2)/r(1))*(2*r_dot(1) - c*r(1))*l_dot(1)/cos(g_b(2)-l(2))  +
(r(2)/r(1))*cos(g_b(1) - l(1))*a(1)/cos(g_b(2)-l(2))
-  M*sign(S)/cos(g_b(2)-l(2)) ;

a_b = [a(1); a(2) ; a(3)];

g_c = g + (h/2)*[a(1)/V(1); a(2)/V(2) ; a(3)/V(3)];
p_c = p + (h/2)*[V(1)*cos(g_c(1)) , V(1)*sin(g_c(1)) ; V(2)*cos(g_c(2)) ,
V(2)*sin(g_c(2)) ; V(3)*cos(g_c(3)) , V(3)*sin(g_c(3)) ];
l = [atan((p_c(3,2)-p_c(1,2))/(p_c(3,1)-p_c(1,1))) ;
atan((p_c(3,2)-p_c(2,2))/(p_c(3,1)-p_c(2,1)))];
r = [((p_c(1,1)-p_c(3,1))^2 + (p_c(1,2)-p_c(3,2))^2)^0.5 ;
((p_c(2,1)-p_c(3,1))^2 + (p_c(2,2)-p_c(3,2))^2)^0.5];
r_dot = [V(3)*cos(g_c(3)-l(1)) - V(1)*cos(g_c(1)-l(1)) ;
V(3)*cos(g_c(3) - l(2)) - V(2)*cos(g_c(2) - l(2))];
l_dot = [ ( V(3)*sin(g_c(3) - l(1)) - V(1)*sin(g_c(1) - l(1)) )/r(1) ;
( V(3)*sin(g_c(3) - l(2)) - V(2)*sin(g_c(2) - l(2)) )/r(2)];
%S = l_dot(1) - l_dot(2) + c*(l(1) - l(2));

a(1) = -3*l_dot(1)*r_dot(1);
a(3) = -5*l_dot(1)*r_dot(1);
a(2) = -(2*r_dot(2) - c*r(2))*l_dot(2)/cos(g_c(2) - l(2))  +
(r(2)/r(1))*(2*r_dot(1) - c*r(1))*l_dot(1)/cos(g_c(2)-l(2))  +
(r(2)/r(1))*cos(g_c(1) - l(1))*a(1)/cos(g_c(2)-l(2))  -
M*sign(S)/cos(g_c(2)-l(2));

a_c = [a(1); a(2) ; a(3)];


g_d = g + h*[a(1)/V(1); a(2)/V(2) ; a(3)/V(3)];
p_d = p + h*[V(1)*cos(g_d(1)) , V(1)*sin(g_d(1)) ; V(2)*cos(g_d(2)) ,
V(2)*sin(g_d(2)) ; V(3)*cos(g_d(3)) , V(3)*sin(g_d(3)) ];
```

```
l = [atan((p_d(3,2)-p_d(1,2))/(p_d(3,1)-p_d(1,1))) ;
atan((p_d(3,2)-p_d(2,2))/(p_d(3,1)-p_d(2,1)))];
r = [((p_d(1,1)-p_d(3,1))^2 + (p_d(1,2)-p_d(3,2))^2)^0.5 ;
((p_d(2,1)-p_d(3,1))^2 + (p_d(2,2)-p_d(3,2))^2)^0.5];
r_dot = [V(3)*cos(g_d(3)-l(1)) - V(1)*cos(g_d(1)-l(1)) ;
V(3)*cos(g_d(3) - l(2)) - V(2)*cos(g_d(2) - l(2))];
l_dot = [ ( V(3)*sin(g_d(3) - l(1)) - V(1)*sin(g_d(1) - l(1)) )/r(1) ;
( V(3)*sin(g_d(3) - l(2)) - V(2)*sin(g_d(2) - l(2)) )/r(2)];
%S = l_dot(1) - l_dot(2) + c*(l(1) - l(2));

a(1) = -3*l_dot(1)*r_dot(1);
a(3) = -5*l_dot(1)*r_dot(1);
a(2) = -(2*r_dot(2) - c*r(2))*l_dot(2)/cos(g_d(2) - l(2))  +
(r(2)/r(1))*(2*r_dot(1) - c*r(1))*l_dot(1)/cos(g_d(2)-l(2))  +
(r(2)/r(1))*cos(g_d(1) - l(1))*a(1)/cos(g_d(2)-l(2))  -
M*sign(S)/cos(g_d(2)-l(2));

a_d = [a(1); a(2) ; a(3)];

%Final values to be returned
go = g + (h/6)*( [a_a(1)/V(1) ; a_a(2)/V(2) ; a_a(3)/V(3)] + 2*[a_b(1)/V(1) ;
a_b(2)/V(2) ; a_b(3)/V(3)] + 2*[a_c(1)/V(1) ; a_c(2)/V(2) ; a_c(3)/V(3)] +
[a_d(1)/V(1) ; a_d(2)/V(2) ; a_d(3)/V(3)]);
p = p + (h/6)*( [V(1)*cos(g(1)) , V(1)*sin(g(1)) ; V(2)*cos(g(2)), V(2)*sin(g(2)) ;
V(3)*cos(g(3)) , V(3)*sin(g(3))]+ 2*[V(1)*cos(g_b(1)) , V(1)*sin(g_b(1)) ;
V(2)*cos(g_b(2)), V(2)*sin(g_b(2)) ; V(3)*cos(g_b(3)) , V(3)*sin(g_b(3)) ] +
2*[V(1)*cos(g_c(1)) , V(1)*sin(g_c(1)) ; V(2)*cos(g_c(2)), V(2)*sin(g_c(2)) ;
V(3)*cos(g_c(3)) , V(3)*sin(g_c(3))]  + [V(1)*cos(g_d(1)) , V(1)*sin(g_d(1)) ;
V(2)*cos(g_d(2)), V(2)*sin(g_d(2)) ; V(3)*cos(g_d(3)) , V(3)*sin(g_d(3))] );

g = go;

l = [atan((p(3,2)-p(1,2))/(p(3,1)-p(1,1))) ;
atan((p(3,2)-p(2,2))/(p(3,1)-p(2,1)))];
r = [((p(1,1)-p(3,1))^2 + (p(1,2)-p(3,2))^2)^0.5 ; ((p(2,1)-p(3,1))^2 +
(p(2,2)-p(3,2))^2)^0.5];
r_dot = [V(3)*cos(g(3)-l(1)) - V(1)*cos(g(1)-l(1)) ; V(3)*cos(g(3) - l(2)) -
V(2)*cos(g(2) - l(2))];
l_dot = [ ( V(3)*sin(g(3) - l(1)) - V(1)*sin(g(1) - l(1)) )/r(1) ;
( V(3)*sin(g(3) - l(2)) - V(2)*sin(g(2) - l(2)) )/r(2)];

phi_dot = l_dot(1) - l_dot(2);
phi = l(1) - l(2);
S = phi_dot + c*phi;

aa_new = -3*l_dot(1)*r_dot(1);
am_new = -5*l_dot(1)*r_dot(1);
ad_new = -(2*r_dot(2) - c*r(2))*l_dot(2)/cos(g_d(2) - l(2))  +
    (r(2)/r(1))*(2*r_dot(1) - c*r(1))*l_dot(1)/cos(g_d(2)-l(2))  +
    (r(2)/r(1))*cos(g_d(1) - l(1))*aa_new/cos(g_d(2)-l(2))  -
    M*sign(S)/cos(g_d(2)-l(2)) ;

end
```

%%%%%%%%%

RK4_new

```
    function [p, g, l, l_dot, r, r_dot, aa_new, ad_new, am_new, S, phi, phi_dot] =
    RK4_new(p, g, l, l_dot, r, r_dot, a_aa, h, c, M, S, V)
    aa = a_aa;
    am = -3*l_dot(1)*r_dot(1);
    ad = -(2*r_dot(2) - c*r(2))*l_dot(2)/cos(g(2) - l(2))  +
    (r(2)/r(1))*(2*r_dot(1) - c*r(1))*l_dot(1)/cos(g(2)-l(2))  +
    (r(2)/r(1))*cos(g(1) - l(1))*a_aa/cos(g(2)-l(2))
    -  M*sign(S)/cos(g(2)-l(2));

    a_a = [aa ; ad; am];
    a = [aa ;  ad ;  am];

    g_b = g + (h/2)*[a(1)/V(1); a(2)/V(2) ; a(3)/V(3)];
    p_b = p + (h/2)*[V(1)*cos(g_b(1)) , V(1)*sin(g_b(1)) ; V(2)*cos(g_b(2)),
    V(2)*sin(g_b(2)) ; V(3)*cos(g_b(3)) , V(3)*sin(g_b(3)) ];
    l = [atan((p_b(3,2)-p_b(1,2))/(p_b(3,1)-p_b(1,1))) ;
    atan((p_b(3,2)-p_b(2,2))/(p_b(3,1)-p_b(2,1)))];
    r = [((p_b(1,1)-p_b(3,1))^2 + (p_b(1,2)-p_b(3,2))^2)^0.5 ;
    ((p_b(2,1)-p_b(3,1))^2 + (p_b(2,2)-p_b(3,2))^2)^0.5];
    r_dot = [V(3)*cos(g_b(3)-l(1)) - V(1)*cos(g_b(1)-l(1)) ;
    V(3)*cos(g_b(3) - l(2)) - V(2)*cos(g_b(2) - l(2))];
    l_dot = [ ( V(3)*sin(g_b(3) - l(1)) - V(1)*sin(g_b(1) - l(1)) )/r(1) ;
    ( V(3)*sin(g_b(3) - l(2)) - V(2)*sin(g_b(2) - l(2)) )/r(2)];
    %S = l_dot(1) - l_dot(2) + c*(l(1) - l(2));

    a(1) = a_aa;
    a(3) = -3*l_dot(1)*r_dot(1);
    a(2) = -(2*r_dot(2) - c*r(2))*l_dot(2)/cos(g_b(2) - l(2))  +
    (r(2)/r(1))*(2*r_dot(1) - c*r(1))*l_dot(1)/cos(g_b(2)-l(2))  +
    (r(2)/r(1))*cos(g_b(1) - l(1))*a_aa/cos(g_b(2)-l(2))
    -  M*sign(S)/cos(g_b(2)-l(2)) ;

    a_b = [a(1); a(2) ; a(3)];

    g_c = g + (h/2)*[a(1)/V(1); a(2)/V(2) ; a(3)/V(3)];
    p_c = p + (h/2)*[V(1)*cos(g_c(1)) , V(1)*sin(g_c(1)) ; V(2)*cos(g_c(2)) ,
    V(2)*sin(g_c(2)) ; V(3)*cos(g_c(3)) , V(3)*sin(g_c(3)) ];
    l = [atan((p_c(3,2)-p_c(1,2))/(p_c(3,1)-p_c(1,1))) ;
```

```
atan((p_c(3,2)-p_c(2,2))/(p_c(3,1)-p_c(2,1)))];
r = [((p_c(1,1)-p_c(3,1))^2 + (p_c(1,2)-p_c(3,2))^2)^0.5 ;
((p_c(2,1)-p_c(3,1))^2 + (p_c(2,2)-p_c(3,2))^2)^0.5];
r_dot = [V(3)*cos(g_c(3)-l(1)) - V(1)*cos(g_c(1)-l(1)) ;
V(3)*cos(g_c(3) - l(2)) - V(2)*cos(g_c(2) - l(2))];
l_dot = [ ( V(3)*sin(g_c(3) - l(1)) - V(1)*sin(g_c(1) - l(1)) )/r(1) ;
( V(3)*sin(g_c(3) - l(2)) - V(2)*sin(g_c(2) - l(2)) )/r(2)];
%S = l_dot(1) - l_dot(2) + c*(l(1) - l(2));

a(1) = a_aa;
a(3) = -3*l_dot(1)*r_dot(1);
a(2) = -(2*r_dot(2) - c*r(2))*l_dot(2)/cos(g_c(2) - l(2))  +
(r(2)/r(1))*(2*r_dot(1) - c*r(1))*l_dot(1)/cos(g_c(2)-l(2))  +
(r(2)/r(1))*cos(g_c(1) - l(1))*a_aa/cos(g_c(2)-l(2))  -
M*sign(S)/cos(g_c(2)-l(2));

a_c = [a(1); a(2) ; a(3)];


g_d = g + h*[a(1)/V(1); a(2)/V(2) ; a(3)/V(3)];
p_d = p + h*[V(1)*cos(g_d(1)) , V(1)*sin(g_d(1)) ; V(2)*cos(g_d(2)) ,
V(2)*sin(g_d(2)) ; V(3)*cos(g_d(3)) , V(3)*sin(g_d(3)) ];
l = [atan((p_d(3,2)-p_d(1,2))/(p_d(3,1)-p_d(1,1))) ;
atan((p_d(3,2)-p_d(2,2))/(p_d(3,1)-p_d(2,1)))];
r = [((p_d(1,1)-p_d(3,1))^2 + (p_d(1,2)-p_d(3,2))^2)^0.5 ;
((p_d(2,1)-p_d(3,1))^2 + (p_d(2,2)-p_d(3,2))^2)^0.5];
r_dot = [V(3)*cos(g_d(3)-l(1)) - V(1)*cos(g_d(1)-l(1)) ;
V(3)*cos(g_d(3) - l(2)) - V(2)*cos(g_d(2) - l(2))];
l_dot = [ ( V(3)*sin(g_d(3) - l(1)) - V(1)*sin(g_d(1) - l(1)) )/r(1) ;
( V(3)*sin(g_d(3) - l(2)) - V(2)*sin(g_d(2) - l(2)) )/r(2)];
%S = l_dot(1) - l_dot(2) + c*(l(1) - l(2));

a(1) = a_aa;
a(3) = -3*l_dot(1)*r_dot(1);
a(2) = -(2*r_dot(2) - c*r(2))*l_dot(2)/cos(g_d(2) - l(2))  +
(r(2)/r(1))*(2*r_dot(1) - c*r(1))*l_dot(1)/cos(g_d(2)-l(2))  +
(r(2)/r(1))*cos(g_d(1) - l(1))*a_aa/cos(g_d(2)-l(2))
-  M*sign(S)/cos(g_d(2)-l(2)) ;

a_d = [a(1); a(2) ; a(3)];

%Final values to be returned
go = g + (h/6)*( [a_a(1)/V(1) ; a_a(2)/V(2) ; a_a(3)/V(3)] + 2*[a_b(1)/V(1) ;
a_b(2)/V(2) ; a_b(3)/V(3)] + 2*[a_c(1)/V(1) ; a_c(2)/V(2) ; a_c(3)/V(3)] +
[a_d(1)/V(1) ; a_d(2)/V(2) ; a_d(3)/V(3)]);
p = p + (h/6)*( [V(1)*cos(g(1)) , V(1)*sin(g(1)) ; V(2)*cos(g(2)), V(2)*sin(g(2)) ;
V(3)*cos(g(3)) , V(3)*sin(g(3))]+ 2*[V(1)*cos(g_b(1)) , V(1)*sin(g_b(1)) ;
V(2)*cos(g_b(2)), V(2)*sin(g_b(2)) ; V(3)*cos(g_b(3)) , V(3)*sin(g_b(3)) ] +
    2*[V(1)*cos(g_c(1)) , V(1)*sin(g_c(1)) ; V(2)*cos(g_c(2)), V(2)*sin(g_c(2)) ;
    V(3)*cos(g_c(3)) , V(3)*sin(g_c(3))]  + [V(1)*cos(g_d(1)) , V(1)*sin(g_d(1)) ;
    V(2)*cos(g_d(2)), V(2)*sin(g_d(2)) ; V(3)*cos(g_d(3)) , V(3)*sin(g_d(3))] );

    g = go;

    l = [atan((p(3,2)-p(1,2))/(p(3,1)-p(1,1))) ;
    atan((p(3,2)-p(2,2))/(p(3,1)-p(2,1)))];
    r = [((p(1,1)-p(3,1))^2 + (p(1,2)-p(3,2))^2)^0.5 ; ((p(2,1)-p(3,1))^2 +
    (p(2,2)-p(3,2))^2)^0.5];
    r_dot = [V(3)*cos(g(3)-l(1)) - V(1)*cos(g(1)-l(1)) ; V(3)*cos(g(3) - l(2)) -
    V(2)*cos(g(2) - l(2))];
    l_dot = [ ( V(3)*sin(g(3) - l(1)) - V(1)*sin(g(1) - l(1)) )/r(1) ; ( V(3)*sin(g(3)
    - l(2)) - V(2)*sin(g(2) - l(2)) )/r(2)];

    phi_dot = l_dot(1) - l_dot(2);
    phi = l(1) - l(2);
    S = phi_dot + c*phi;

    aa_new = a_aa;
    am_new = -3*l_dot(1)*r_dot(1);
    ad_new = -(2*r_dot(2) - c*r(2))*l_dot(2)/cos(g_d(2) - l(2))  +
    (r(2)/r(1))*(2*r_dot(1) - c*r(1))*l_dot(1)/cos(g_d(2)-l(2))  +
    (r(2)/r(1))*cos(g_d(1) - l(1))*a_aa/cos(g_d(2)-l(2))
    -  M*sign(S)/cos(g_d(2)-l(2)) ;

end
```