ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SCHOOL OF ENGINEERING

SECOND CYCLE MASTER'S DEGREE in
AEROSPACE ENGINEERING
Class LM-20

Design and Implementation of an Advanced Guidance and Control Algorithm for Autonomous Quadcopter Drone

Thesis in:
Athmospheric Flight Dynamics

Supervisor:
Prof. Fabrizio Giulietti

Examiner:
Prof. Paolo Castaldi

Defended by:
Andrea Tini

ACADEMIC YEAR 2022/2023

Abstract

This master's thesis delves into the development and implementation of a precise dynamic model, a fine-tuned flight controller and several optimized guidance algorithms tailored explicitly for quadcopter drones, with a specific focus on their application in intercepting targets (specifically high-performance drones), indicative of potential military purposes. Drawing inspiration from established algorithms designed for missile interception, the study explores the adaptation and optimization of these algorithms to suit the unique dynamics of quadcopter systems. The primary guidance algorithms under consideration include Proportional Navigation (PN), Augmented Proportional Navigation (APN), Optimal Guidance Law (OGL), and PID Guidance.
In the contemporary landscape dominated by unmanned aerial vehicles, quadcopter drones have emerged as integral contributors across diverse sectors, including surveillance and delivery services. In the context of this thesis, Simulink has been employed, a versatile modeling tool, to design and implement robust control systems for quadcopter drones, specifically tailored for intercepting targets.
Acknowledging the sensitive nature of the application, the study considers the intricacies of modeling both the quadcopter drone and the target. The objective extends beyond mere emulation of quadcopter dynamics, aiming to contribute substantively to the broader applications of aerial technology in military contexts.

Contents

List of Figures

Chapter 1

Introduction

The journey into unmanned aerial vehicles (UAVs) and Autonomous Pilotless Reconnaissance Aircraft (APR) dates back to 1849 when the Austrians utilized explosive balloons in the assault on Venice. Fast forward to World War I, the "Aerial Target" in 1916 was a significant development, controlled via radiofrequency. In that same year, the Hewitt-Sperry automatic airplane showcased the concept of unmanned flight, operated through internal gyroscopes.

Post-World War I, technological progress enabled the conversion of aircraft into APRs. Notably, Reginald Denny's Radioplane Company produced over 15,000 remotely piloted helicopters during World War II for anti-aircraft artillery testing. The Italian Army embraced UAVs in the 1960s, employing models like the CL-89. Subsequent advancements include the Mirach 20, FQM 151 A Pointer, and Raven RQ 1A and 1B, reflecting a shift from remote camera reconnaissance to real-time video surveillance with remote piloting capabilities.

Since their inception, initially designed exclusively for military purposes, drones have catalyzed significant strides in terms of miniaturization and accessibility of these technologies to the general public.

They have proliferated to such an extent that they have seamlessly integrated into the fabric of daily life, their proliferation has marked a true democratization of these systems. This democratization has reached a point where individuals can readily access high-performance drones, capable of achieving remarkable feats. This accessibility stems

from a notable reduction in both the cost and dimensions of drones over the years, making them increasingly available to the masses. While this accessibility presents opportunities for diverse applications, such as reconnaissance and precision agriculture, it simultaneously poses a potential threat. The prospect of a criminal obtaining these technologies raises serious concerns for the security of localities and even entire nations, especially when considering the possibility of an armed group acquiring a swarm of explosive drones.
To safeguard the social and political stability of nations vulnerable to such terrorist attacks, there is an undeniable imperative to develop a system or device capable of intercepting and neutralizing these evolving threats. A case in point, widely recognized in contemporary defense strategies, is the Iron Dome (Fig: 1.1)—a formidable anti-missile system developed by Israel.

Figure 1.1: Iron Dome launches interceptor missile

Engineered to intercept and neutralize short-range missile threats, the Iron Dome has demonstrated its efficacy in real conflict scenarios. However, its large size and, notably, the substantial cost associated with each battery equipped with Tamir missiles (approximately 50 million dollars) present challenges that necessitate exploration of alternative solutions. Various alternative concepts have been introduced for the interception of missiles, one notable proposal involves the utilization of drones equipped with nets (Fig: 1.2) designed to ensnare the selected target [10].

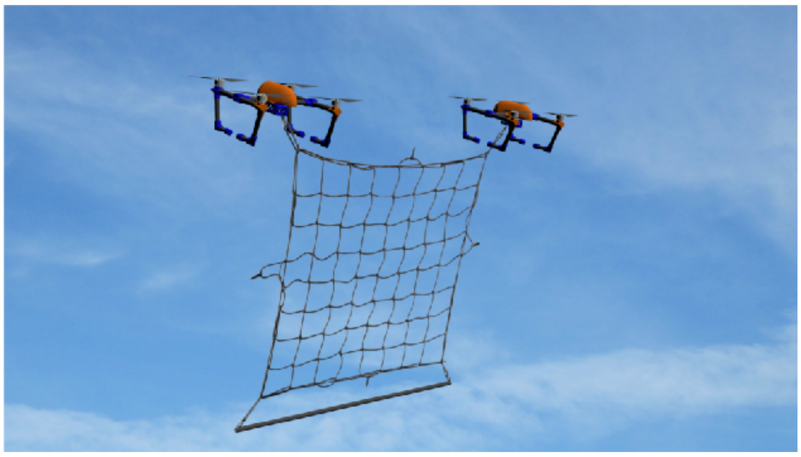


Figure 1.2: Model of the UAV system carrying a net

This brings forth the primary focus of this thesis: the development of an innovative antidrone system based on armed, high-performance drones equipped with explosive charges.
The proposed system aims to address the identified limitations of existing defenses, providing a dynamic and cost-effective solution to counter evolving threats posed by drone technology. The envisioned drone-based system will be tasked not only with intercepting potential threats but also with meeting a critical requirement—obliterating the target in the shortest possible time. This dual emphasis on effectiveness and efficiency is crucial to neutralizing impending dangers before they can strike, thus contributing to a more robust and adaptive security infrastructure.
Throughout the following chapters, this thesis will delve into the conceptualization, development, and comprehensive evaluation of the proposed drone-based antidrone system based on existing missile-based interception algorithm, such as: Proportional Navigation (PN), Augmented Proportional Navigation (APN), Optimal Guidance Law (OGL), and PID Guidance algorithms, shedding light on its potential as a strategic asset in the evolving landscape of security challenges.

Chapter 2

Dynamic model

Within this chapter, the non-linear mathematical model of a multirotor is delineated (Fig: 2.1), commencing with the formulation of general equations governing the kinematics and dynamics of a rigid body. It is imperative to acknowledge that the resultant mathematical model serves as an approximation of actual multirotor dynamics. This approximation is notably influenced by the inherent challenges in comprehensively understanding and accurately modeling certain aerodynamic effects.

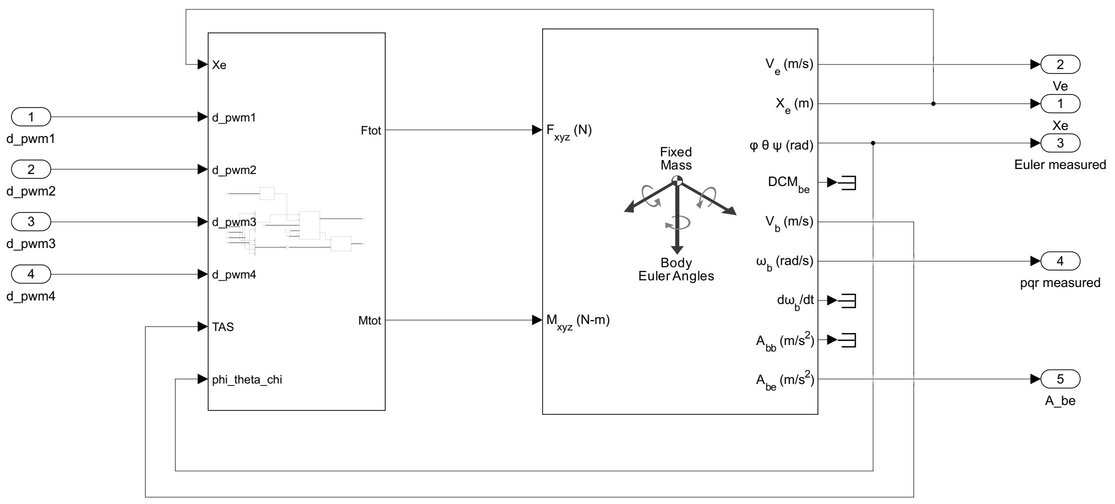


Figure 2.1: Dynamic model developed

2.1 Translational dynamics

The Newton's law applied to the translational motion is:

$$F = m\frac{dV(t)}{dt_i} \tag{2.1}$$

where m is the mass of the drone, V is the velocity vector, $\frac{d}{dt}$ is the inertial time derivative and F is the total force applied to the multirotor. The preceding equation can be extended in its generalized form, termed the Coriolis equation (2.2), wherein due consideration is given to the non-inertial nature of the body-fixed reference system concerning the NED reference system.

$$F = m\frac{dV(t)}{dt_i} = m(\frac{dV(t)}{dt_b} + \Omega_{b/i} \times V) \tag{2.2}$$

in which $\frac{d}{dt_b}$ is the time derivative in the body fixed frame, $\Omega_{b/i} = [p; q; r]$ is the angular velocity of the body fixed frame with respect to the NED frame.
Finally, the formula governing translational dynamics of the drone is as follows:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m}\Big(F_p + F_g + F_a\Big) \tag{2.3}$$

On the left-hand side, the diverse accelerations along the x, y, and z directions are evident. Conversely, the right side incorporates the gyroscopic contribution arising from the non-inertial nature of the body axis reference system, coupled with external forces such as the propulsive F_p, gravitational F_g, and aerodynamic F_a forces. Comprehending these forces is imperative for the design of an efficient control system and the establishment of stable and controlled flight.

Thrust Force

The thrust force is generated by the quadcopter's rotors. It acts in the upward direction, opposing the force of gravity. This force is responsible for lifting the quadcopter off the ground and controlling its altitude.

$$F_i = K_1 \delta_{pwm(i)} \tag{2.4}$$

In which K_1 is a constant that have to be determined experimentally ($K_1 = 0.0033$), δ_{pwm} is the motor command signal and F_i denotes the thrust force generated by the i-th motor.
The final thrust component is:

$$F_p = \begin{bmatrix} 0 \\ 0 \\ -(F_1 + F_2 + F_3 + F_4) \end{bmatrix} \tag{2.5}$$

Below is presented an excerpt of the block architecture designed for computing the propulsive force acting on the drone.

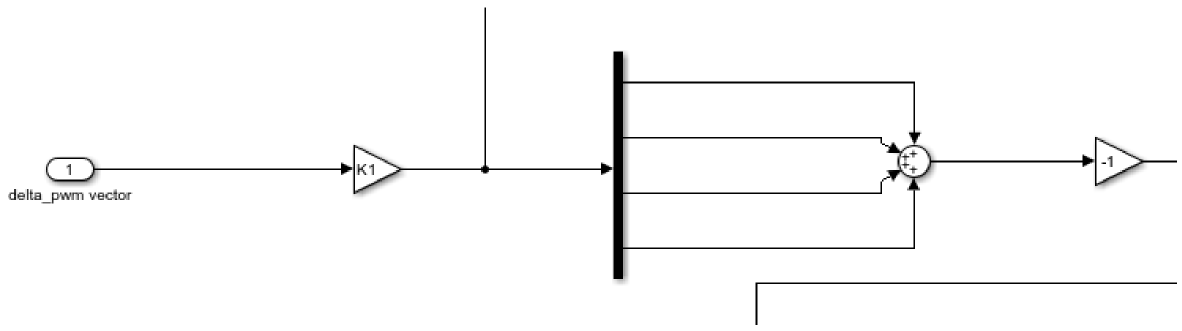


Figure 2.2: Computation of the propulsion force component

Gravitational Force

The weight is the gravitational force acting on the mass of the quadcopter. It acts vertically downward, towards the zenith. Weight is a critical force that influences altitude control and must be counteracted by the thrust force for stable flight. In the calculation of the weight component, it is imperative to factor in the orientation of the body axis in relation to the NED (North-East-Down) frame, as below:

$$F_g = \begin{bmatrix} -mgsin(\theta) \\ mgcos(\theta)sin(\phi) \\ mgcos(\theta)cos(\phi) \end{bmatrix} \tag{2.6}$$

Where m is the drone's mass (set equal to 2 $[kg]$), g is the gravity acceleration, θ is the pitch attitude angle and ϕ is the roll attitude angle. Below is presented an excerpt of the block architecture designed for computing the gravitational force acting on the drone.

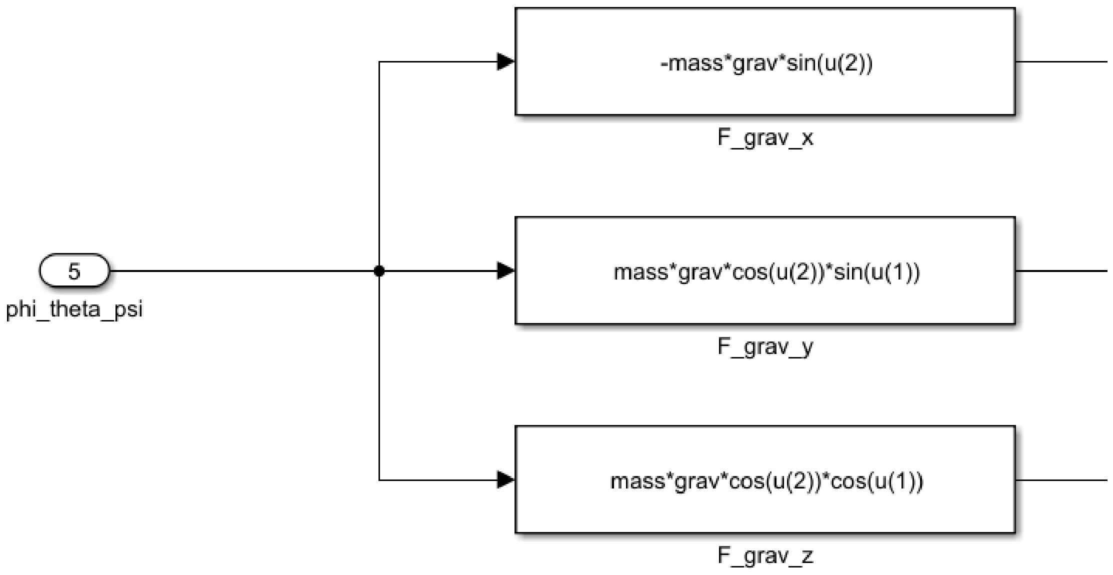


Figure 2.3: Computation of the gravitational force component

Drag Force

As the quadcopter moves through the air, it encounters air resistance, resulting in a drag force. This force acts opposite to the direction of motion and contributes to slowing down the quadcopter's translational motion:

$$F_a = -\frac{1}{2}\rho C_D \begin{bmatrix} A_x u|u| \\ A_y v|v| \\ A_z w|w| \end{bmatrix} \tag{2.7}$$

Where ρ is the air density, C_D is the drone's drag coefficient ($C_D = 0.5$), A_x, A_y and A_z are respectively the frontal area, lateral area, overhead view of the drone (respectively fixed at 0.023, 0.023 and 0.106 $[m^2]$) and u, v and w are the x, y, z components of the True Airspeed without the contribution of the wind speed. Below is presented an excerpt of the block architecture designed for computing the aerodynamic forces acting on the drone.

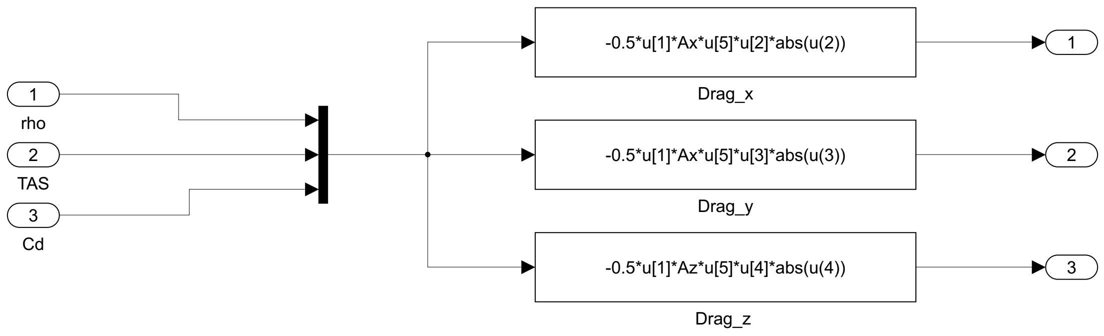

1
rho
2
TAS
3
Cd
-0.5*u[1]*Ax*u[5]*u[2]*abs(u(2))
Drag_x
-0.5*u[1]*Ax*u[5]*u[3]*abs(u(3))
Drag_y
-0.5*u[1]*Az*u[5]*u[4]*abs(u(4))
Drag_z
1
2
3

Figure 2.4: Computation of the aerodynamic force component

2.2 Rotational dynamics

The rotational dynamics of a drone presume the complex interaction of torques and angular velocities, dictating its changes in orientation around the three principal axes: roll, pitch, and yaw. These rotational dynamics are intricately linked to the application of moments and the principles of angular motion. So, in order to obtain the formula governing the rotational dynamics, it is required to start the discussion from the Newton's law applied to the rotational motion:

$$M = \frac{dh}{dt_i} \tag{2.8}$$

where h is the angular momentum and M is the overall applied torque. From the equation of Coriolis, the previous one becomes:

$$M = \frac{dh}{dt_i} = (\frac{dh}{dt_b} + \Omega_{b/i} \times h) \tag{2.9}$$

By setting $h = J\Omega_{b/i}$:

$$M = J\frac{d\Omega_{b/i}}{dt_b} + \Omega_{b/i} \times (J\Omega_{b/i}) \tag{2.10}$$

$$\frac{d\Omega_{b/i}}{dt_b} = J^{-1}(M - \Omega_{b/i} \times (J\Omega_{b/i}) \tag{2.11}$$

Under the assumption of the multirotor's symmetry across all three axes, the constant inertia matrix J can be expressed as:

$$J = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \tag{2.12}$$

where, $J_x = 0.0319[m^2]$, $J_y = 0.0287[m^2]$, $J_z = 0.0633[m^2]$.
The formula governing rotational dynamics is as follows:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{J_y - J_z}{J_x} qr \\ \frac{J_z - J_x}{J_y} pr \\ \frac{J_x - J_y}{J_z} pq \end{bmatrix} + \begin{bmatrix} \frac{1}{J_x} l \frac{\sqrt{2}}{2} \left(F_1 - F_2 - F_3 + F_4 \right) \\ \frac{1}{J_y} l \frac{\sqrt{2}}{2} \left(F_1 + F_2 - F_3 - F_4 \right) \\ \frac{1}{J_z} l \frac{\sqrt{2}}{2} \left(\tau_1 - \tau_2 + \tau_3 - \tau_4 \right) \end{bmatrix} \tag{2.13}$$

The formula above features angular accelerations on the left-hand side and, on the right-hand side, the gyroscopic component due to the non-inertiality of the frame, along with external moment components resulting from propulsive forces (2.4). In quadrotors, as one might anticipate, the moments around the x and y axes arise from an asymmetry in thrust among individual rotors. As for the yaw axis, the resulting external moment is generated by the overall moment produced by the individual blades, which, in pairs, rotate in opposite directions.
The roll dynamics of a drone, denoted by the symbol τ_ϕ , are influenced by the asymmetric distribution of thrust across the left and right rotors. When the drone experiences a difference in thrust between these rotors, a torque is generated, causing the drone to roll about its longitudinal axis. The greater the asymmetry in thrust, the more pronounced the roll motion.

$$\tau_\phi = l \frac{\sqrt{2}}{2} (F_1 - F_2 - F_3 + F_4) \tag{2.14}$$

Where l (fixed at 0.225 [m]) is the distance between CG and the applied forces F_i generated by each rotor blade. Similarly, the pitch dynamics τ_θ result from imbalances in thrust between the front and rear rotors. This discrepancy creates a torque that induces the drone to pitch about its lateral axis. The pitch motion is contingent upon the magnitude of the thrust asymmetry.

$$\tau_\theta = l \frac{\sqrt{2}}{2} (F_1 + F_2 - F_3 - F_4) \tag{2.15}$$

Yaw dynamics τ_ψ, responsible for the drone's rotation around its vertical axis, arise from differences in torque produced by the clockwise and counterclockwise rotating rotors. Changes in rotor speeds create opposing torques, leading to a yawing motion. Precise control of these rotational dynamics is crucial for stable and responsive drone operation.

$$\tau_\psi = \tau_1 - \tau_2 + \tau_3 - \tau_4 \tag{2.16}$$

The following formula is applied to calculate the torque component for each individual motor:

$$\tau_{(i)} = K_2 \delta_{pwm(i)} \tag{2.17}$$

Where also K_2 is determined experimentally and fixed at $K_2 = 3.2689$e-04.Below is presented an excerpt of the block architecture designed for computing the rotational dynamics acting on the drone.

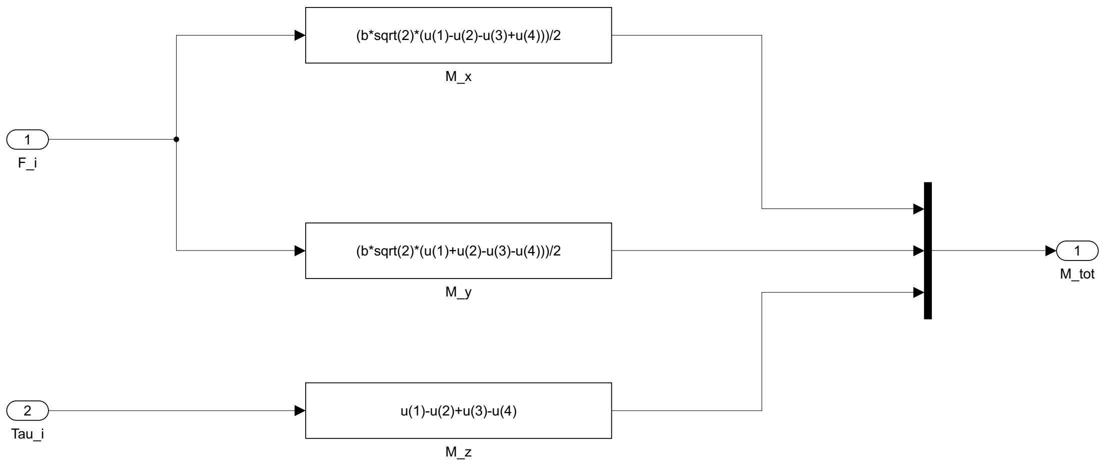

(b*sqrt(2)*(u(1)-u(2)-u(3)+u(4)))/2
M_x
1
F_i
(b*sqrt(2)*(u(1)+u(2)-u(3)-u(4)))/2
M_y
1
M_tot
2
Tau_i
u(1)-u(2)+u(3)-u(4)
M_z

Figure 2.5: Computation of the rotational dynamics acting on the drone

Chapter 3

Flight Controller

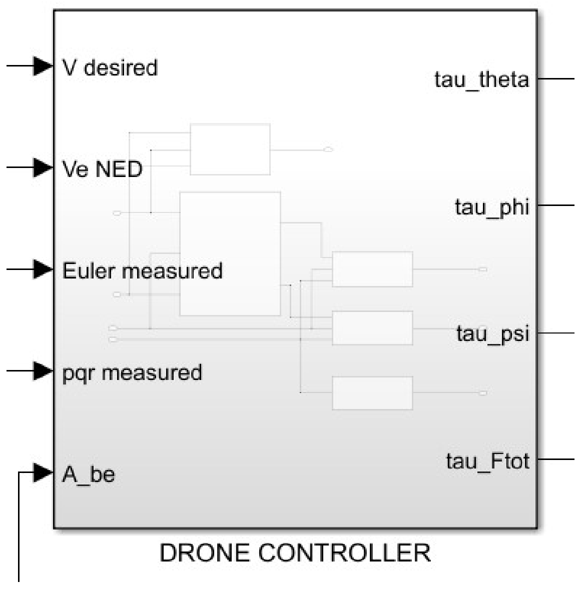


Figure 3.1: Simulink block of the flight controller

Flight controllers serve as the pivotal processing stage within the software framework. The prevalent use of the Proportional-Integral-Derivative (PID) control technique in multirotor applications, owing to its feasibility and simplicity, is noteworthy. This section systematically outlines the structure of the software's flight controllers, organized in a cascade control system [3]. Typically, this architecture incorporates a primary controller and primary dynamics as constituents of the outer loop, with a secondary controller loop integrated as part of this outer loop. The inner loop's set-points are calculated by the outer

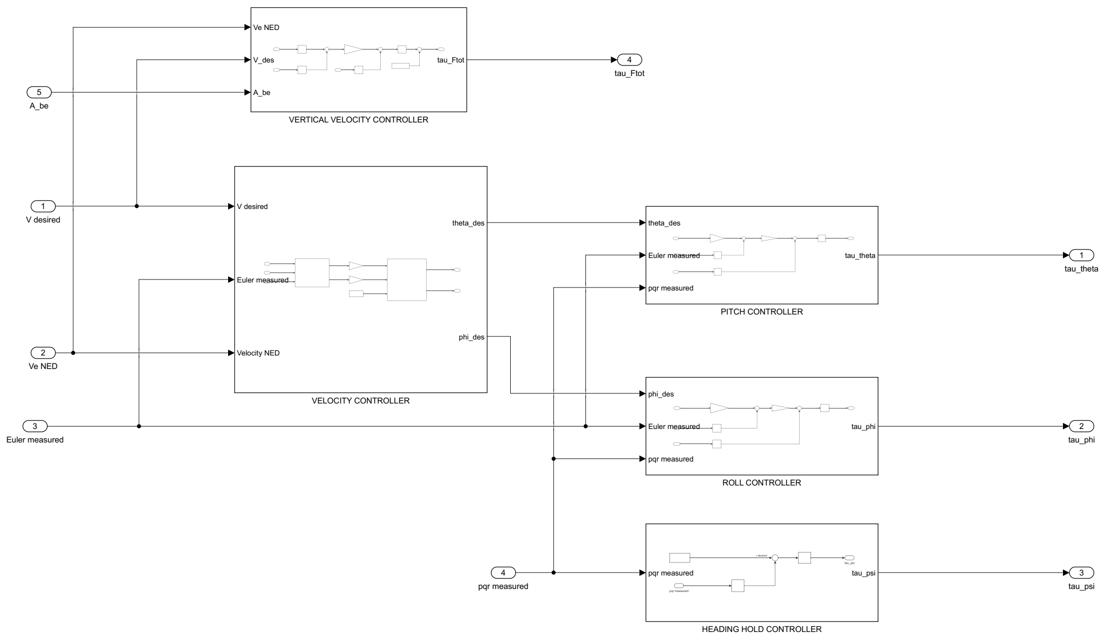


Figure 3.2: Internal architecture of the flight controller

loop, thus earning the nomenclature ”cascade control”. To optimize performance, the inner loop should exhibit considerably faster dynamics compared to the outer loop, mitigating potential interactions and enhancing stability characteristics. Thanks to the application of a cascade control structure, the PID control strategy can be adapted for controlling complex dynamics like rotary-wing aerial vehicles.
In the following sections is showed the detailed design and functionality of each controller, highlighting the strategic approach taken to ensure precise and efficient control in diverse flight scenarios.

3.1 Vertical speed controller

The vertical speed controller allows the user to control multirotor vertical speed and/or vertical acceleration acting on the total force applied by motors. Within the vertical speed controller Fig: 3.3, the inputs are the desired velocity V_{des}, as commanded by the guidance subsystem, the measured velocity in the north-east-down (NED) coordinates V_{be}, and the inertial acceleration in NED coordinates A_{be}. In this controller, the focus lies on the vertical velocity component $\dot{z}$. Therefore, initially, only the z-component is extracted from the respective inputs. The primary objective is to generate an error between the target velocity and the current measured velocity.

$$e_{\dot{z}} = \dot{z_{des}} - \dot{z} \tag{3.1}$$

Subsequently, this error is multiplied by an appropriate gain $K_{p\dot{z}}$ to obtain the desired vertical acceleration $\ddot{z_{des}}$.

$$\ddot{z_{des}} = K_{p\dot{z}} e_{\dot{z}} \tag{3.2}$$

Furthermore, an error is calculated between the desired and measured accelerations.

$$e_{\ddot{z}} = \ddot{z_{des}} - \ddot{z} \tag{3.3}$$

Subsequently, a fine tuned PID controller produces an output delta,

$$\tau_{F_{tot}} = K_{P\ddot{z}} e_{\ddot{z}} - K_{D\ddot{z}} e_{\dot{\ddot{z}}} + K_{I\ddot{z}} \int_0^t e_{\ddot{z}}\, dx + \tau_{hovering} \tag{3.4}$$

which is added to the hovering condition, where it is assumed that the pilot is not actively commanding any signals to the drone (untouched condition). It must be considered that the output $\tau_{F_{tot}}$ lies in a range between 0 (no thrust) and 1 (maximum thrust), the value of $\tau_{F_{tot}} = 0.5$ assigned to the hovering condition.

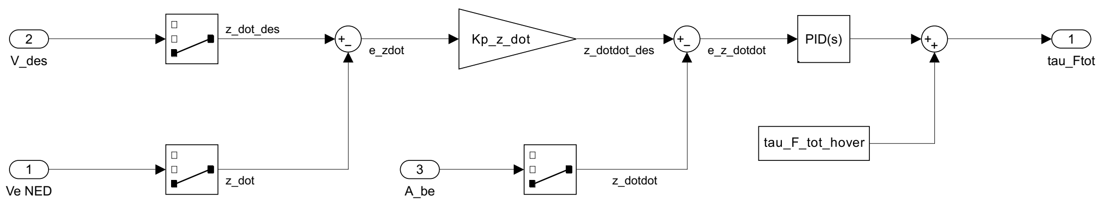


Figure 3.3: Vertical speed controller block diagram

3.2 Velocity controller

An effective approach to controlling Unmanned Aerial Vehicles (UAVs) involves employing speed commands. This section delineates the modeling of the velocity controller within the flight control software for rotary-wing UAVs. Positioned within the primary cascade control scheme, this controller is formulated as a feedback proportional controller. It receives forward/right velocity references, denoted as $v_{f_{des}}$, and subsequently generates desired roll/pitch angles for the inner control loop, namely ϕ_{des} and θ_{des}. In the velocity controller designed Fig: 3.4, the inputs are the desired velocity, the measured Euler angles, and the velocity measured in inertial coordinates. These inputs are processed within a "MATLAB function" block to calculate the respective velocity errors in the forward-right-downward coordinates e_{v_f}, e_{v_r} which is the similar to the vehicle frame but rotated by the heading angle ψ. This approach is justified by the intuitive nature of commanding the drone with instructions like forward/backward or right/left, making the control process more user-friendly. Within this block, the fundamental calculation involves the following:

$$\begin{bmatrix} e_{v_f} \\ e_{v_r} \end{bmatrix} = \begin{bmatrix} cos\psi & sin\psi \\ -sin\psi & cos\psi \end{bmatrix} \begin{bmatrix} u_{des} - u \\ v_{des} - v \end{bmatrix} = \begin{bmatrix} v_{f_{des}} - v_f \\ v_{r_{des}} - v_r \end{bmatrix} \tag{3.5}$$

To the left of the equality sign, there are the respective errors along the forward and right directions. In the middle, is possible to find the differences in errors for the corresponding velocity components along the x and y axes in inertial coordinates. These differences are then multiplied by a rotation matrix, transforming them into the forward-right-downward coordinate system. On the right, the velocity differences directly expressed in the abovementioned coordinates. Subsequently, each component is multiplied by pure gains, which yield the desired acceleration along the forward and right directions (3.6).

$$\begin{bmatrix} a_{f\,des} \\ a_{r\,des} \end{bmatrix} = \begin{bmatrix} K_{v_f} e_{v_f} \\ K_{v_r} e_{v_r} \end{bmatrix} \tag{3.6}$$

Following this, the obtained accelerations are utilized to calculate the respective θ_{des} and ϕ_{des} according to the formula:

$$\theta_{des} = -arctan(\frac{a_{f\,des}}{g}) \tag{3.7}$$

$$\phi_{des} = arctan(\frac{a_{r\,des} cos_{\theta des}}{g}) \tag{3.8}$$

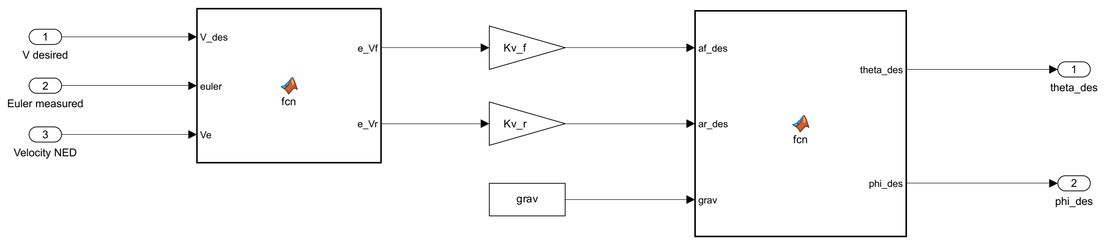


Figure 3.4: Velocity controller block diagram

3.3 Attitude controller

The attitude controller constitutes the innermost layer of the control hierarchy within the software, governing the fundamental piloting aspects of the multirotor vehicle. Consistent with the overarching philosophy employed in the system, the attitude controller is structured as a cascade control system. The outer loop of the attitude controller relies on Euler angles ϕ and θ, leveraging attitude measurements obtained from the dynamic model in two separate controllers, namely "Roll controller" and "Pitch controller". In this context, a simplified proportional gain is employed. Meanwhile, the inner loop is dedicated to managing angular velocities through a PID control system. Considering the symmetrical nature of the drone along the pitch and roll channels, it is evident that the respective controllers (roll and pitch controllers which constitutes the Attitude controller) share an almost identical architecture. The only difference lies in the input variables; in fact, these are the measured Euler angles, measured angular rates, and, respectively, the θ_{des} for pitch controller (Fig: 3.5) and ϕ_{des} for the roll controller (Fig: 3.6). These desired values are, in fact, the outputs from the velocity controller. The desired Euler angle input is multiplied by a pure gain, which takes into account the maximum value attainable by the drone—specifically, ϕ_{max} and θ_{max}, both set to a maximum of 20° for safety reasons. In this scenario as well, the respective error between the desired and measured values is calculated.

$$e_\phi = \phi_{des} - \phi \tag{3.9}$$

$$e_\theta = \theta_{des} - \theta \tag{3.10}$$

Subsequently, another gain $K_{p\phi}$ and $K_{p\theta}$ are applied to obtain the desired angular velocity (p_{des} for the roll controller, q_{des} for the pitch controller).

$$p_{des} = K_{p\phi} e_\phi \tag{3.11}$$

$$q_{des} = K_{p\theta} e_{\theta} \tag{3.12}$$

Once the error is generated with the measured angular velocity, this error is passed through a tuned PID controller, providing the respective τ_{ϕ} and τ_{θ} values.

$$\tau_{\phi} = K_{Pp} e_p - K_{Dp} e_{\dot{p}} + K_{Ip} \int_0^t e_p \, dx \tag{3.13}$$

$$\tau_{\theta} = K_{Pq} e_q - K_{Dq} e_{\dot{q}} + K_{Iq} \int_0^t e_q \, dx \tag{3.14}$$

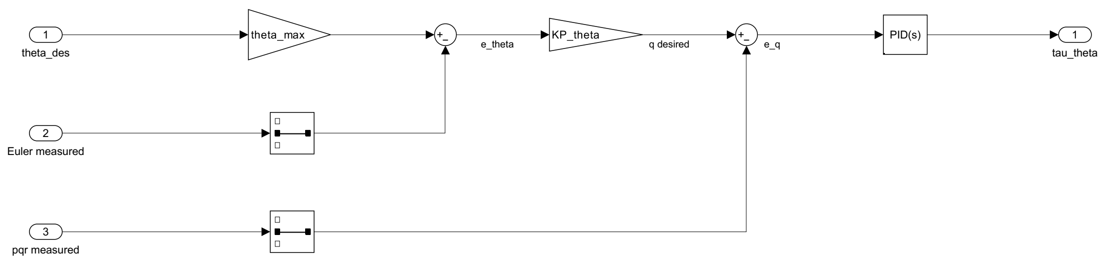


Figure 3.5: Pitch controller block diagram

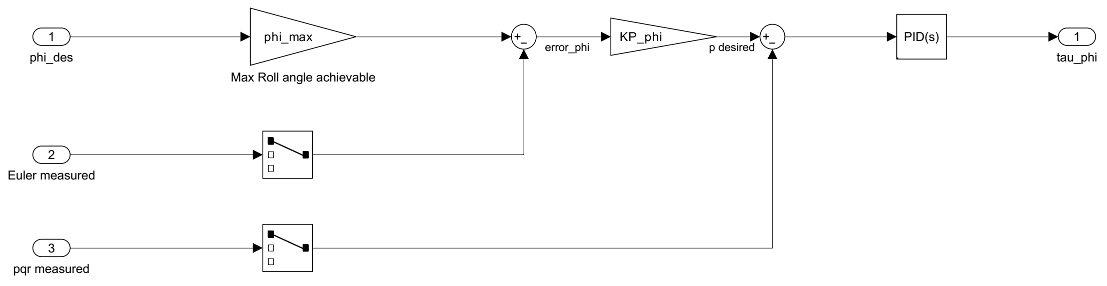


Figure 3.6: Roll controller block diagram

3.4 Heading hold controller

The heading hold controller Fig: 3.7 features a much simpler design. In fact, the initial step involves calculating the difference between the initial yaw rate r_{ini} and the measured yaw rate, which is provided as input to the controller. Once the error is calculated, it is directly passed to the PID controller, which then produces the necessary output τ_ψ to control the yaw channel.

$$e_r = r_{ini} - r \tag{3.15}$$

$$\tau_\psi = K_{Pr} e_r - K_{Dq} e_{\dot{r}} + K_{Ir} \int_0^t e_r \, dx \tag{3.16}$$

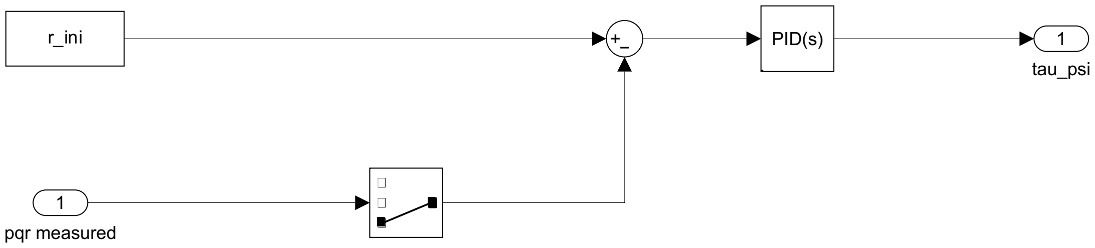


Figure 3.7: Heading hold controller block diagram

3.5 Tuning

The iterative tuning process, conducted through a systematic trial-and-error approach, was initiated with a focus on the vertical speed controller, considering its limited influence on the pitch and roll channels of the drone. Following the confirmation of functionality and a reliable response to external disturbances, attention was then directed towards parameter acquisition for the heading hold controller. Once these controllers were optimized, the tuning process transitioned to the pitch channel. Significantly, the acquisition of coefficients for the pitch

channel facilitated a straightforward copy-and-paste methodology for the roll channel controller, capitalizing on the inherent symmetry of the quadcopter, resulting in identical behavior on both channels.
Subsequent to obtaining pure gains for the velocity controller, a Simulink model was meticulously constructed to accurately represent both the plant and the controller. The initial configuration involved setting up a PID Controller block, with parameters strategically determined based on preliminary estimates or default values. Navigating through the PID Tuner, a suite of tools and visualizations was employed, guided by a comprehensive open-loop analysis.
The iterative tuning process unfolded dynamically, with continual adjustments to proportional, integral, and derivative gains based on insights gained into the system's behavior. Transient response analyses, facilitated by the PID Tuner, played a pivotal role in refining the controller's performance and optimizing transient dynamics. Robustness analysis emerged as a cornerstone in ensuring stability under varying conditions. Through interactive engagement with tuning sliders, gains were fine-tuned to strike a delicate balance between responsiveness, stability, and robustness.
Validation, conducted through extensive simulations incorporating diverse input scenarios and disturbances, became imperative. This validation process led to the transfer of tuned parameters back to the Simulink model, ensuring the effectiveness of the refined controller in real-world scenarios.
Here is a list with the coefficient values implemented in each controller:

- Velocity controller: $Kv_f = 15$, $Kv_r = 15$
- Vertical speed controller: $Kp_{\dot{z}} = 10$, $K_{P\ddot{z}} =$ -1, $K_{I\ddot{z}} =$ -0.82, $K_{D\dddot{z}}$ = 0.0038, $N_z = 33.7$
- Pitch controller: $K_{P\theta} = 8$, $K_{Pq} = 10$, $K_{Dq} = 0$, $K_{Iq} = 0.2$, $N_q =$ 100
- Roll controller: $K_{P\phi} = 8$, $K_{Pp} = 10$, $K_{Dp} = 0$, $K_{Ip} = 0.2$, $N_p =$ 100
- Heading hold controller: $K_{Pr} = 1$, $K_{Dr} =$ -0.01, $K_{Ir} = 0.01$, N_r = 1.72

3.6 Motor mixing

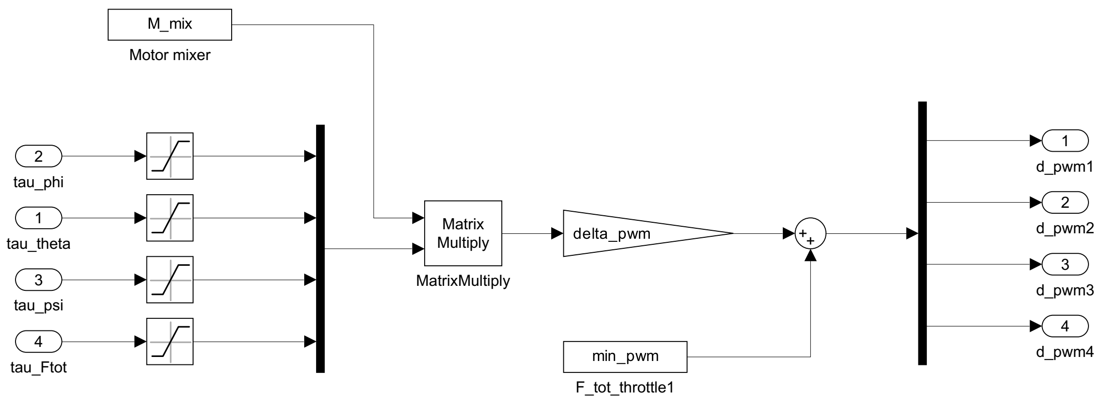


Figure 3.8: Motor mixing procedure within the Simulink environment

However, before sending these signals ($\tau_\phi, \tau_\theta, \tau_\psi$ and $\tau_{F_{tot}}$) to the dynamic model, it is essential to perform the operation known as ”Motor Mixing” (Fig: 3.8). This operation involves controlling the individual motor speeds to obtain the desired flight characteristics. In such systems, the ability to independently adjust the speed of each motor enables the vehicle to modify its orientation, stabilize, and execute various maneuvers. Basically, the motor mixing consists into a simple matrix multiplication:

$$\begin{bmatrix} \delta_{pwm1} \\ \delta_{pwm2} \\ \delta_{pwm3} \\ \delta_{pwm4} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \\ \tau_{F_{tot}} \end{bmatrix} \tag{3.17}$$

As evident from the above equation, the motor mixing matrix multiplies various contributions to obtain the different delta δ_{pwm} values. The matrix is constructed by taking into account the rotation directions of individual blades, each of which distinctly influences the behavior of the drone. The concept of δ_{pwm} can be described as a method of encoding analog information in a digital signal. In the context of quadcopters, PWM is often used to control the speed of the motors. It works by varying the width of pulses in a train of digital pulses. The average voltage is determined by the duty cycle, which is the ratio of the pulse width to the total time period of the signal. After obtaining the PWM values through the motor mixing procedure, they can be utilized within the model as inputs for the translational and rotational blocks.

3.7 Testing

To individually assess each controller, a straightforward methodology was employed. Each controller under evaluation was isolated from the others, and an arbitrary input value was systematically assigned to observe their respective responses to a step change. As evident from the ensuing graphs, the controllers exhibited consistently satisfactory outcomes.

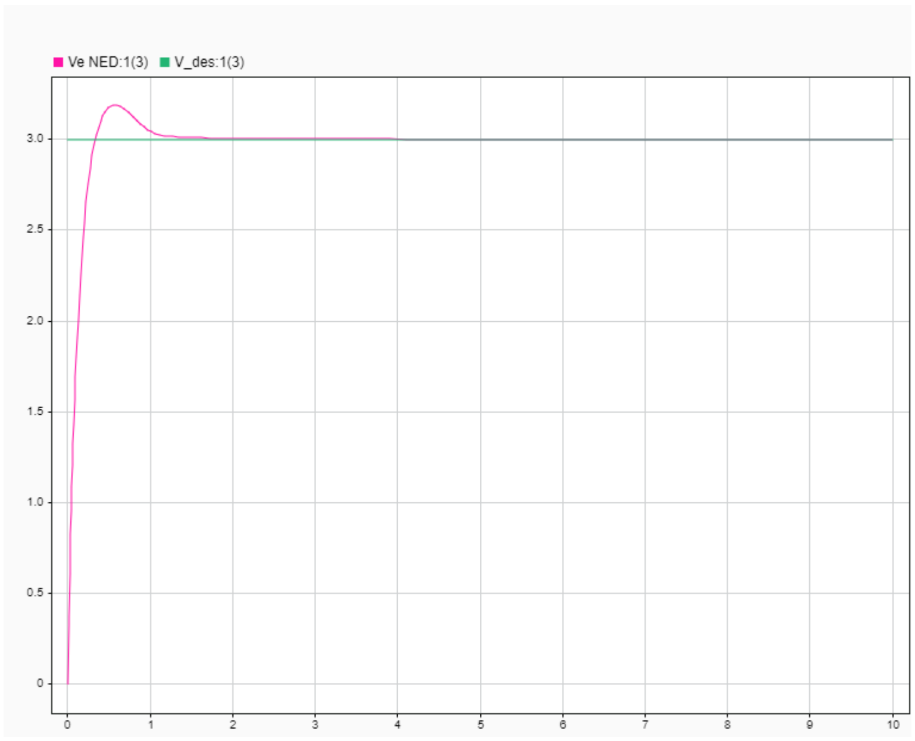


Figure 3.9: Response of the **Vertical speed controller** to step change of $V_{zdes} = 3\frac{m}{s}$ vs time

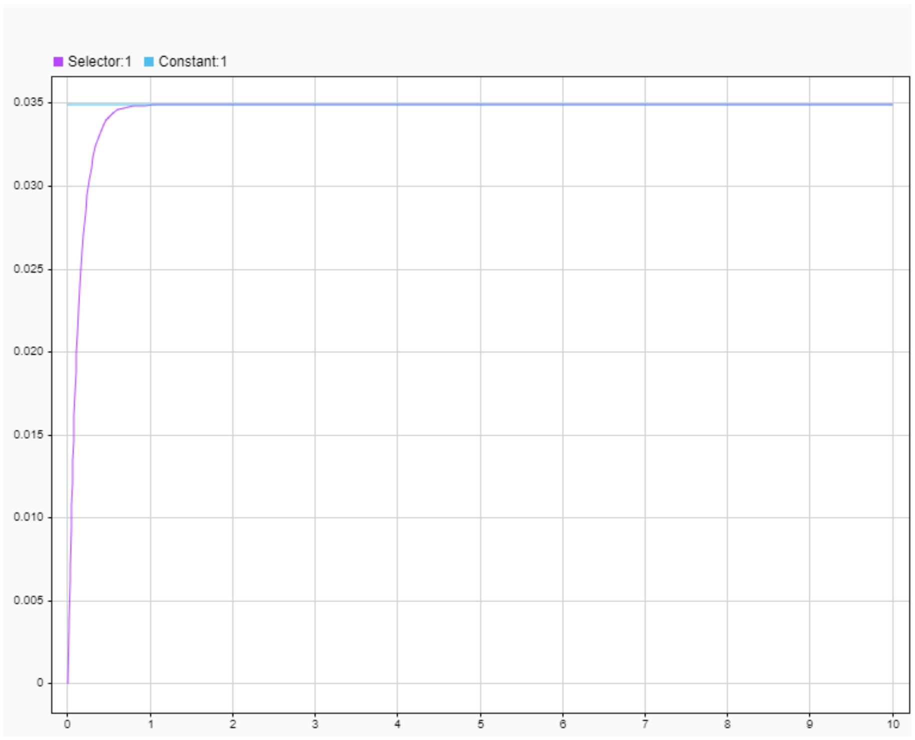


Figure 3.10: Response of the **Pitch controller** to step change of $\theta_{des} = 2$ vs time

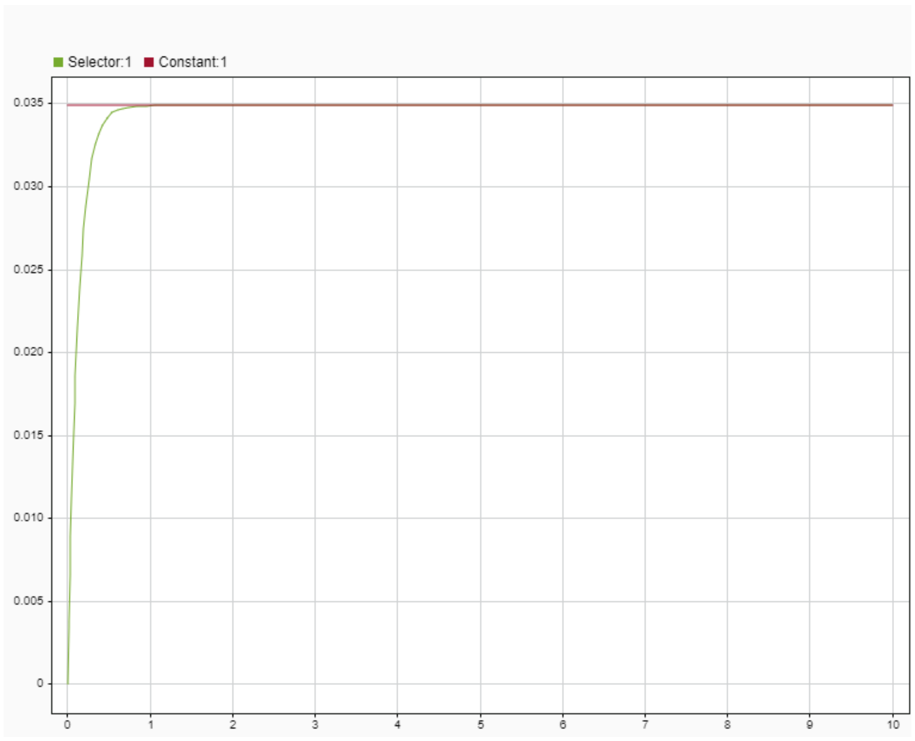


Figure 3.11: Response of the **Roll controller** to step change of $\phi_{des} = 2$ vs time

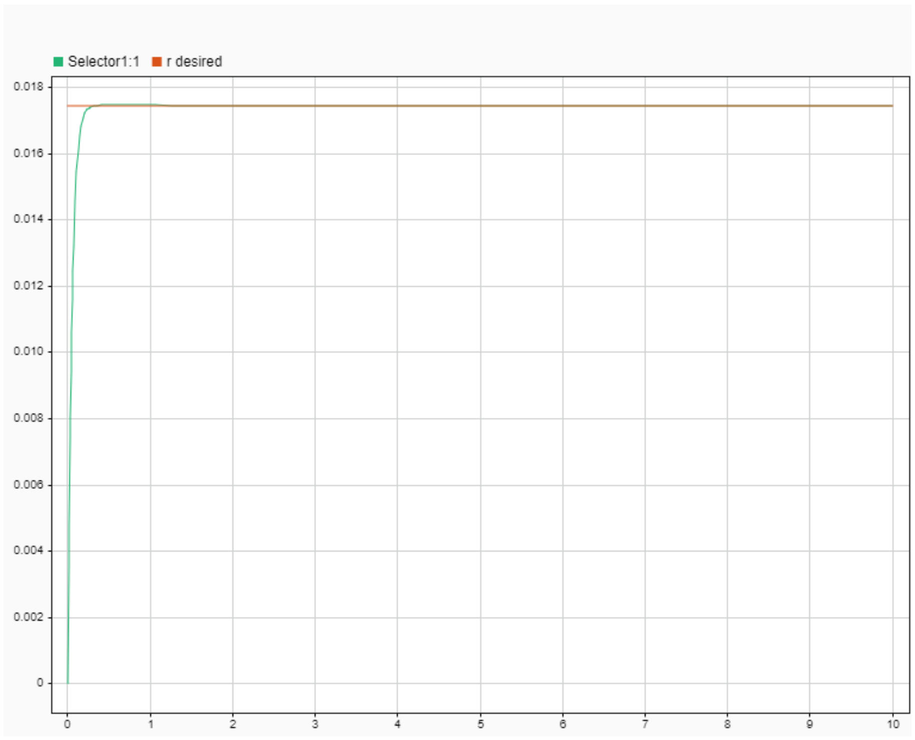


Figure 3.12: Response of the **Heading hold controller** to step change of $r_{des} = 1\frac{}{s}$ vs time

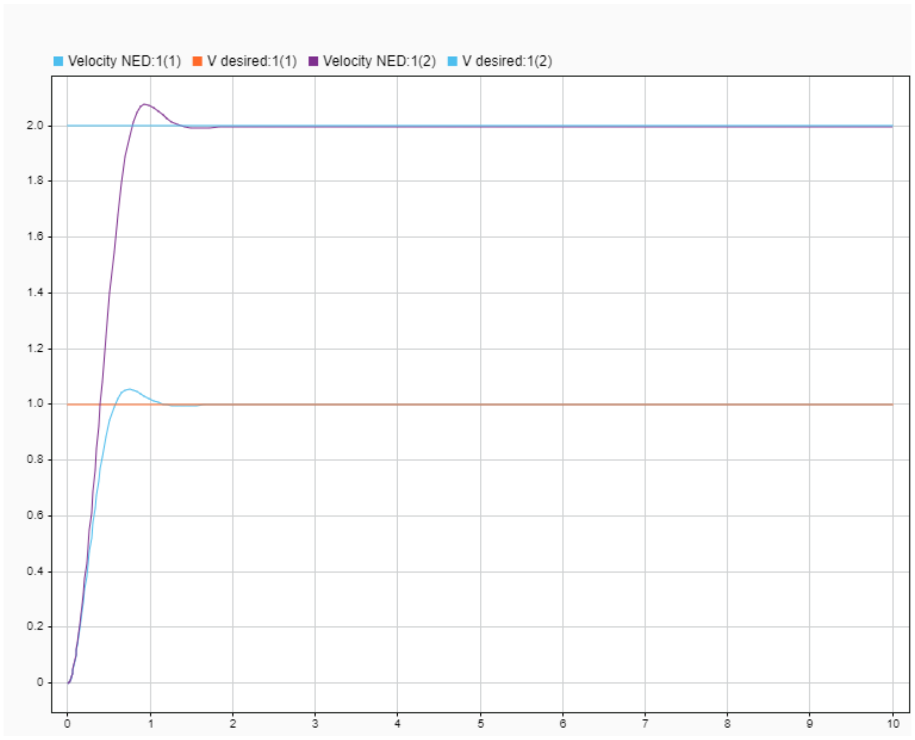


Figure 3.13: Response of the **Velocity controller** to step change of $V_{des} = [1; 2; 0]\frac{m}{s}$ vs time

Overall flight control system response

To comprehensively assess the developed flight controller, testing was conducted using a holistic approach. The input to the controller encompassed not only all the measured values from the dynamic system but also a velocity expressed in the NED frame. This velocity input originates from the guidance system since it is an output of the latter. Consequently, to evaluate the effectiveness and efficiency of the flight controller, arbitrary velocity commands were arbitrarily issued. This process involved bypassing the input from the guidance system, aiming to ascertain whether the drone could autonomously attain the specified velocity conditions.
In this testing phase, the flight controller was presented with a predefined desired velocity, denoted as $V_{des} = [1; 2; 3]\frac{m}{s}$, to observe and analyze its ability to guide the drone autonomously towards the specified speed conditions. This methodological approach allowed for a thorough examination of the flight controller's performance in response to

user-defined velocity inputs, facilitating insights into its capacity to effectively navigate and regulate the drone's movement.

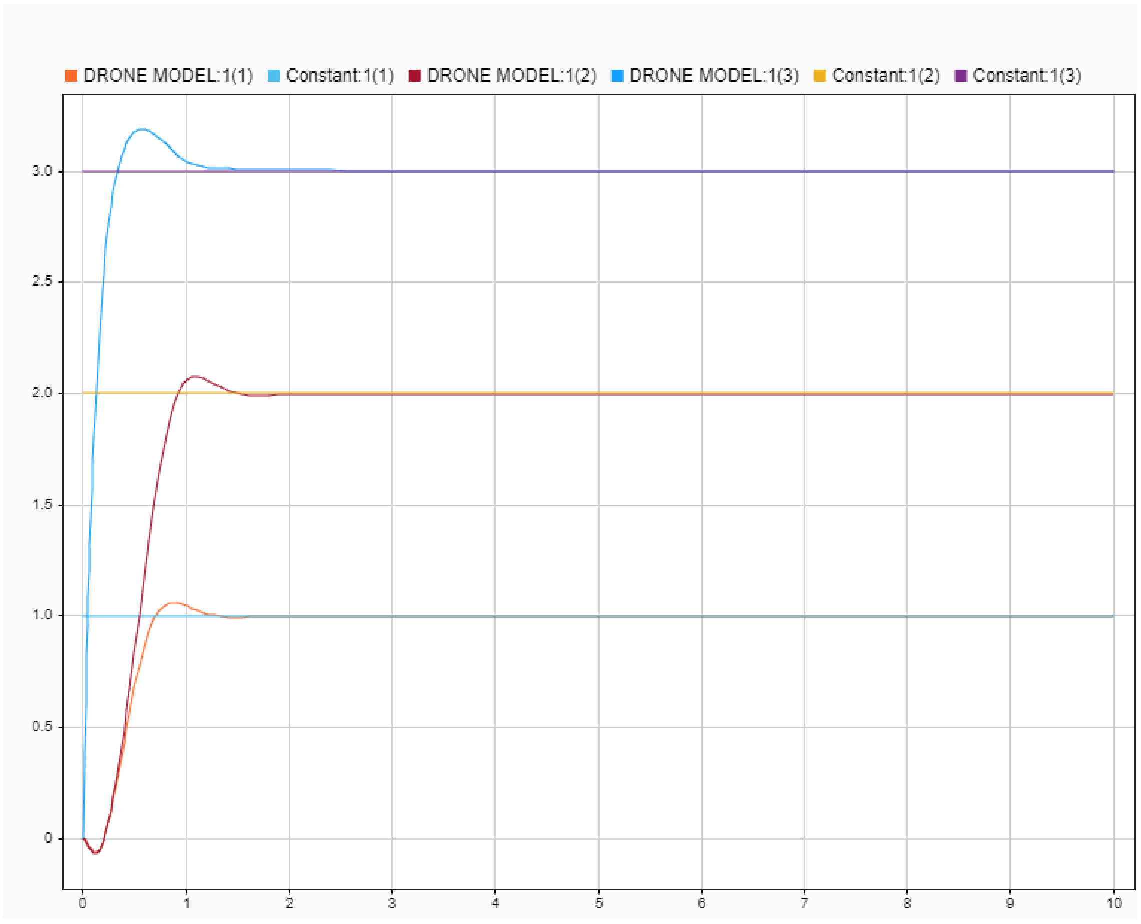


Figure 3.14: Response of the entire system to step change of $V_{des} = [1; 2; 3]\frac{m}{s}$ vs time

Chapter 4

Guidance controllers

In the ensuing sections of this chapter, a comprehensive exploration of the diverse methodologies underpinning the development of the guidance controller for the interceptor drone will be undertaken. The intricate nature of a guidance system is underscored by its pivotal role in issuing precise commands to the flight controller, thereby orchestrating a nuanced adjustment of the drone's orientation. This orchestration is further complemented by the reception of pertinent information applicable to the targeted entity.
It is imperative to recognize that the guidance system operates as an integral subsystem within the drone, serving as the conduit through which commands are judiciously disseminated in accordance with the dynamic state of the target being pursued. In essence, the efficacy of the guidance system is paramount in dictating the drone's responsiveness and adaptability to the evolving conditions of the mission.
In the subsequent discussion, a meticulous dissection of the four distinct guidance systems, will be undertaken. The following paradigms will be scrutinized: Proportional Navigation, Augmented Proportional Navigation, Optimal Guidance Law, and PID Guidance. Within this analytical framework, an in-depth examination of the original algorithms integral to missile guidance, which serve as the foundational inspiration for these respective technologies, will be conducted.
Moreover, an in-depth analysis on the assumptions intrinsic to these algorithms will ensue, shedding light on the underlying principles that

govern their functionality. This comprehensive analysis will culminate in a detailed delineation of the systematic progression from the very early stages of each technology's original algorithm to the final, operational iteration effectively implemented on the drone. The ensuing synthesis aims to provide a profound understanding of the intricacies involved in adapting missile guidance algorithms to the unique operational capabilities of the interceptor drone.

4.1 Origins of PN Guidance

Proportional navigation, though recognized by the Germans at Peenemünde during World War II, saw no reported applications on Hs. 298 or R-1 missiles utilizing this guidance principle. The pioneering implementation of proportional navigation occurred with the Lark missile, achieving its first successful test in December 1950. Subsequently, proportional navigation guidance has been ubiquitously employed in the guidance systems of tactical radar, infrared (IR), and television (TV) guided missiles worldwide [1]. The widespread adoption of this interceptor guidance law is attributed to its inherent simplicity, effectiveness, and ease of implementation.

Notably, the initial study of proportional navigation is credited to C. Yuan and others at the RCA Laboratories during World War II, conducted under the auspices of the U.S. Navy [8]. The guidance law was conceptualized based on physical reasoning and the available equipment at that time. Extensive research on proportional navigation was carried out at Hughes Aircraft Company [2], where it was implemented in a tactical missile employing a pulsed radar system. Further development occurred at Raytheon, leading to the implementation of proportional navigation in a tactical continuous-wave radar homing missile [5].

Post-World War II, the U.S. work on proportional navigation was declassified and first appeared in the Journal of Applied Physics [13]. Mathematical derivations affirming the "optimality" of proportional navigation emerged more than 20 years later [11]. In alignment with

the historical context of proportional navigation's origins, this chapter refrains from presenting mathematical proofs in deriving the guidance law. Instead, the focus will be on demonstrating to the reader the efficacy of the guidance technique. Subsequently, an exploration of certain properties of the guidance law will be undertaken, both observed and derived. Finally, the chapter will elucidate how this classical guidance law serves as the foundational basis for more advanced techniques in interceptor guidance.
The studied algorithms are developed for the interception of generic targets using ballistic missiles. These algorithms rely on several stringent assumptions, such as a missile possessing a non-zero velocity vector and significantly outperforming the target. Consequently, the magnitude of the missile's velocity will be greater than that of the target.

4.2 PN Guidance Law

Theoretically, the proportional navigation guidance law dictates acceleration commands that are perpendicular to the instantaneous line-of-sight between the missile and the target. These commands are proportionally determined by the line-of-sight rate and closing velocity. In mathematical terms, the guidance law is articulated as follows [14]:

$$n_c = N' V_c \dot{\lambda} \tag{4.1}$$

where:

- n_c denotes the acceleration command (in m/s),
- N represents a unitless designer-chosen gain (typically in the range of 3–5), known as the effective navigation ratio,
- V_c signifies the closing velocity between the missile and the target (in m/s),
- $\dot{\lambda}$ indicates the line-of-sight rate (in rad/s).

In the context of tactical radar homing missiles employing proportional

navigation guidance, the seeker provides an effective measurement of the line-of-sight rate, while closing velocity information is obtained from a Doppler radar. For tactical infrared missile applications utilizing proportional navigation guidance, the line-of-sight rate is directly measured, whereas the closing velocity required by the guidance law is approximated. In the case of tactical endoatmospheric missiles, the implementation of proportional navigation guidance commands typically involves the manipulation of fins or other control surfaces to generate the necessary lift. Exoatmospheric strategic interceptors utilize thrust vector control, lateral divert engines, or squibs to achieve the desired acceleration levels.
In elucidating the overall PN algorithm, it is considered the 2D case to better understand how the PN algorithm works, as showed in Fig: 4.1

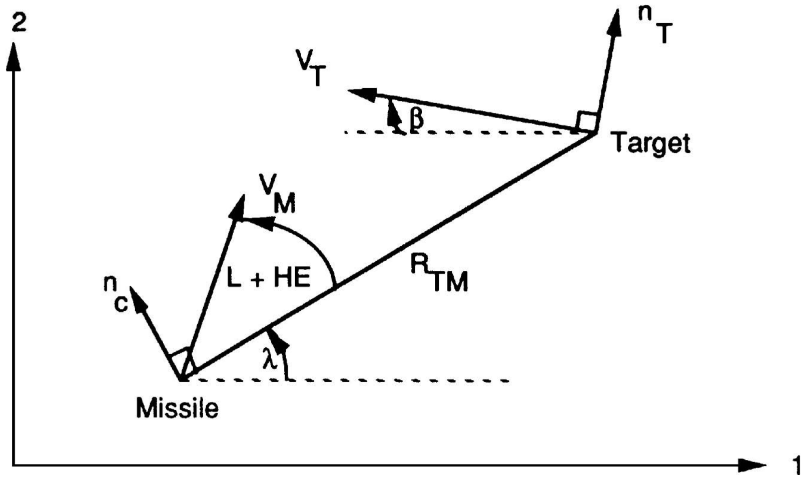


Figure 4.1: Sketch of the 2D missile-target engagement scenario

In this inertial coordinate system fixed to a flat-Earth model, the 1-axis aligns with the downrange direction, and the 2-axis may represent either altitude or crossrange. Utilizing the inertial coordinate system of Fig: 4.1 enables the integration of acceleration and velocity components along the 1 (abscissa) and 2 (ordinate axis) directions without introducing additional terms due to the Coriolis effect. The model assumes constant velocity for both the missile and target, with gravitational and drag effects neglected for simplicity. In Fig: 4.1, the missile, with velocity magnitude V_M, is oriented at an angle of $L+HE$ relative to the line of sight, where L denotes the missile lead angle, representing the theoretically correct angle for the missile to form a collision triangle with the target. The angle HE is identified as the heading error, representing the initial deviation of the missile from the collision triangle. The line connecting the missile and target in Fig: 4.1 is termed the line of sight, making an angle of λ with respect to the fixed reference. The length of the line of sight, denoted as R_{TM}, signifies the instantaneous separation between the missile and target. From a guidance standpoint, the objective is to minimize the range between the missile and target at the expected intercept time. The closest point of approach is referred to as the miss distance. The closing velocity V_c is defined as the negative rate of change of the distance between the missile and the target.

$$V_c = -\dot{R_{TM}} \tag{4.2}$$

Hence, at the end of the engagement, when the missile and target are closest, the sign of V_c changes. The desired acceleration command n_c, derived from the proportional navigation guidance law, is perpendicular to the instantaneous line of sight. In the engagement model of Fig: 4.1, the target can maneuver evasively with acceleration magnitude n_T. The angular velocity of the target can be expressed as

$$\beta = \frac{n_T}{V_T} \tag{4.3}$$

where V_T is the magnitude of the target velocity.
The components of the target velocity vector in the Earth or inertial coordinate system can be determined by integrating the differential equation for the target's flight-path angle β and substituting in the following formula:

$$V_{T1} = -V_T cos(\beta) \tag{4.4}$$

$$V_{T2} = V_T sin(\beta) \tag{4.5}$$

Target position components in the Earth fixed coordinate system are found by directly integrating the target velocity components.

$$\dot{R_{T1}} = V_{T1} \tag{4.6}$$

$$\dot{R_{T2}} = V_{T2} \tag{4.7}$$

Similarly, the missile's velocity and position differential equations, with components a_{M1} and a_{M2} representing the missile acceleration in Earth coordinates, are provided by the respective formulas.

$$\dot{V_{M1}} = a_{M1} \tag{4.8}$$

$$\dot{V_{M2}} = a_{M2} \tag{4.9}$$

$$\dot{R_{M1}} = V_{M1} \tag{4.10}$$

$$\dot{R_{M2}} = V_{M2} \tag{4.11}$$

To find the missile acceleration components, the components of the relative missile-target separation must be defined first.

$$R_{TM1} = R_{T1} - R_{M1} \tag{4.12}$$

$$R_{TM2} = R_{T2} - R_{M2} \tag{4.13}$$

The line-of-sight angle, obtained using trigonometry in terms of the relative separation components, can be expressed using the formula:

$$\lambda = atan \frac{R_{TM2}}{R_{TM1}} \tag{4.14}$$

The relative velocity components in Earth coordinates are defined,

$$V_{TM1} = V_{T1} - V_{M1} \tag{4.15}$$

$$V_{TM2} = V_{T2} - V_{M2} \tag{4.16}$$

and the line-of-sight rate is calculated through direct differentiation:

$$\dot{\lambda} = \frac{R_{TM1}V_{TM2} - R_{TM2}V_{TM1}}{R_{TM}^2} \tag{4.17}$$

The relative separation R_{TM} is expressed in terms of its inertial components using the distance formula,

$$R_{TM} = \sqrt{R_{TM1}^2 + R_{TM2}^2} \tag{4.18}$$

differentiating the (4.18), the closing velocity formula is obtained:

$$V_c = -\dot{R_{TM}} = \frac{R_{TM1}V_{TM1} + R_{TM2}V_{TM2}}{R_{TM}} \tag{4.19}$$

The magnitude of the missile guidance command n_c is determined from the definition of proportional navigation:

$$n_c = N'V_c\dot{\lambda} \tag{4.20}$$

Since the acceleration command is perpendicular to the instantaneous line of sight, the missile acceleration components in Earth coordinates are obtained through trigonometry, as given by the provided formula:

$$a_{M1} = -n_c sin(\lambda) \tag{4.21}$$

$$a_{M2} = n_c cos(\lambda) \tag{4.22}$$

All the necessary differential equations for modeling a complete missile-target engagement in two dimensions have been listed. However, additional equations for the initial conditions on the differential equations are required to complete the engagement model. A missile employing proportional navigation guidance is not directed at the target but is fired in a direction to lead the target. The initial angle of the missile velocity vector with respect to the line of sight, known as the missile lead angle L, is determined by the law of sines.

$$L = asin\frac{V_T sin(\beta + \lambda)}{V_M} \tag{4.23}$$

In practice, the missile is usually not launched exactly on a collision triangle due to uncertainties in predicting the expected intercept point. Any initial angular deviation of the missile from the collision triangle is termed a heading error HE. The initial missile velocity components can be expressed in terms of the theoretical lead angle L and actual heading error HE, as presented in the formula.

$$V_{M1}(0) = V_M cos(L + HE + \lambda) \tag{4.24}$$

$$V_{M2}(0) = V_M sin(L + HE + \lambda) \tag{4.25}$$

4.2.1 Linearization

To date, insights into the effectiveness of proportional navigation have been derived exclusively from numerical simulations of the two-dimensional engagement model. In the interest of analytical analysis, comprehension, and the formulation of design relationships, it becomes essential to temporarily deviate from the nonlinear missile-target simulation and establish a simplified model. The linearization of the two-dimensional engagement model is pursued to deepen our understanding.

It is noteworthy that the introduction of the linearized model does not entail the abandonment of the nonlinear engagement model. On the contrary, the nonlinear engagement model will consistently serve as a benchmark to validate insights generated by robust analytical techniques applied to the linearized model.

The linearization process of the missile-target geometry can be seamlessly executed through the definition of new relative quantities, as delineated in Figure 2.6. Here, y signifies the relative separation between the missile and target perpendicular to the fixed reference. The representation of relative acceleration, denoting the disparity between missile and target acceleration, is elucidated through examination of Fig: 4.2.

$$\ddot{y} = n_T cos(\beta) - n_c cos(\lambda) \tag{4.26}$$

In the case of small angle approximation the 4.26 becomes:

$$\ddot{y} = n_T - n_c \tag{4.27}$$

The expression for the line-of-sight angle λ can also be linearized using the small-angle approximation, yielding

$$\lambda = \frac{y}{R_{TM}} \tag{4.28}$$

Therefore, in the context of a linearized analysis, the closing velocity is considered a positive constant. Given that the closing velocity is defined as the negative derivative of the range from the missile to the target, and acknowledging that the range must converge to zero at the conclusion of the flight, the range equation can be linearized with the time-varying relationship.

$$R_{TM} = V_c(t_F - t) = V_c t_{go} \tag{4.29}$$

Here, t denotes the current time, and t_F represents the total flight time for the engagement which is now regarded as a constant, and t_{go} which is the time to go until intercept. The quantity t_F–t signifies the time remaining until the end of the flight. Consequently, the range from the missile to the target is expressed as the product of the closing velocity and the time remaining until intercept.
In light of the range diminishing to zero at the flight's conclusion, a reevaluation of the miss distance definition is imperative. The linearized miss distance is defined as the relative separation between the missile and the target y at the termination of the flight, as indicated in the formula.

$$MISS = y(t_F) \tag{4.30}$$

It is noteworthy that the linearized miss distance, derived in this manner, is an approximation to the actual miss, as it does not adhere to the distance formula. However, it will become evident shortly that this approximation for the miss distance proves to be highly accurate.

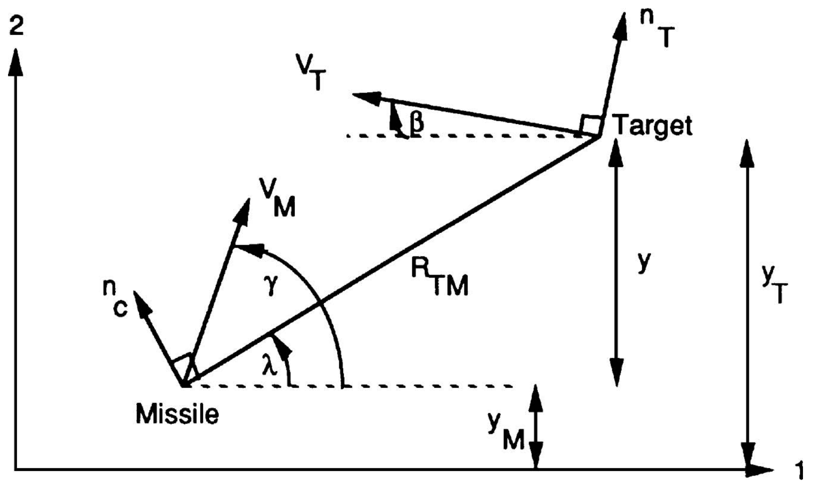


Figure 4.2: Engagement model for linearization

4.3 Early iterations

In this section, a comprehensive exploration unfolds, delving into the iterative journey undertaken during the developmental phases of the algorithm. The objective is to meticulously dissect each stage, scrutinizing the assumptions made, identifying encountered errors, and delineating the evolutionary steps that ultimately culminated in the refined and finalized code. The genesis of this process was rooted in a commitment to validate theoretical principles and assumptions drawn from authoritative sources such as the [14] book.
The initial impetus behind this endeavor was to subject the algorithm to rigorous testing, a crucial step in ensuring its theoretical soundness and practical efficacy. As a strategic approach, the translation of the algorithm into Simulink served as a practical bridge between conceptualization and implementation. The intricate interplay of theoretical constructs and algorithmic assumptions laid the groundwork for subsequent iterations.

4.3.1 First iteration

As mentioned initially, to conduct experiments and thoroughly assess the theory presented in [14], with a particular focus on the assumptions made to define the algorithm, the approach adopted involved a direct translation of the algorithm into Simulink. In Fig: 4.3, there is a depiction of the developed code. This strategy was chosen to facilitate rigorous testing and validation, ensuring that the theoretical foundations and algorithmic assumptions could be effectively examined in a simulated environment.
The translation process aimed at capturing the essence of the algorithm, preserving its key principles while adapting it for implementation within the Simulink framework. This methodological choice not only facilitates a systematic exploration of the theoretical constructs but also provides a practical avenue for verifying the algorithm's performance under various conditions.

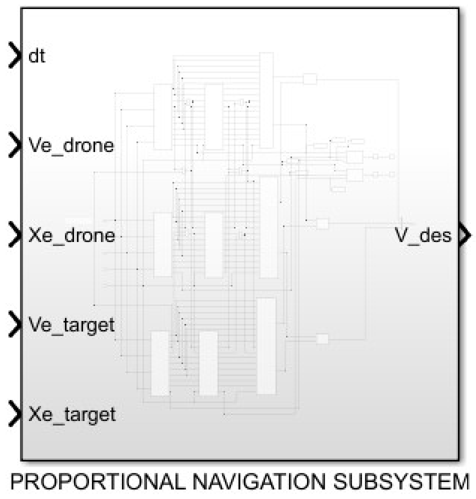


Figure 4.3: Simulink block of the PN Guidance subsystem (first iteration)

As evident from the illustration, the inputs to the block consist of several parameters:

- The timestep δt, generated by Simulink, assumes a crucial role for the integration process within the block, it has been computed according to the Fig: 4.4

- the measured velocity (Ve_{drone}) of the drone in the NED frame,
- the measured position (Xe_{drone}) of the drone in the NED frame,
- the measured velocity (Ve_{target}) of the drone in the NED frame,
- the measured position (Xe_{target}) of the drone in the NED frame.

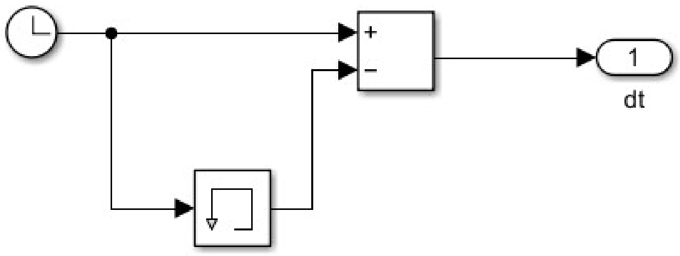


Figure 4.4: Computation of δt

Finally, the output of the commanded velocity (V_{des}) in NED frame which will be introduced within the controller. Upon detailing the inputs and outputs of the block, an intricate examination of its internal structure becomes imperative. The visual representation in the Fig: 4.5 underscores a notable complexity in the developed code, presenting a potential challenge for a programmer unfamiliar with its intricacies. This observation underscores an initial error in the design – the utilization of an excessive number of inputs for each block. A noteworthy discrepancy is revealed, where the number of inputs provided in a single MATLAB Simulink block surpassed 15. This abundance of inputs not only contributes to the overall code's convolution but also raises concerns regarding its accessibility and comprehensibility, particularly for programmers who may encounter the code for the first time. This initial misstep highlights the importance of a streamlined and transparent coding structure to enhance the code's readability and facilitate a more straightforward understanding, ultimately contributing to the efficiency of the overall implementation.

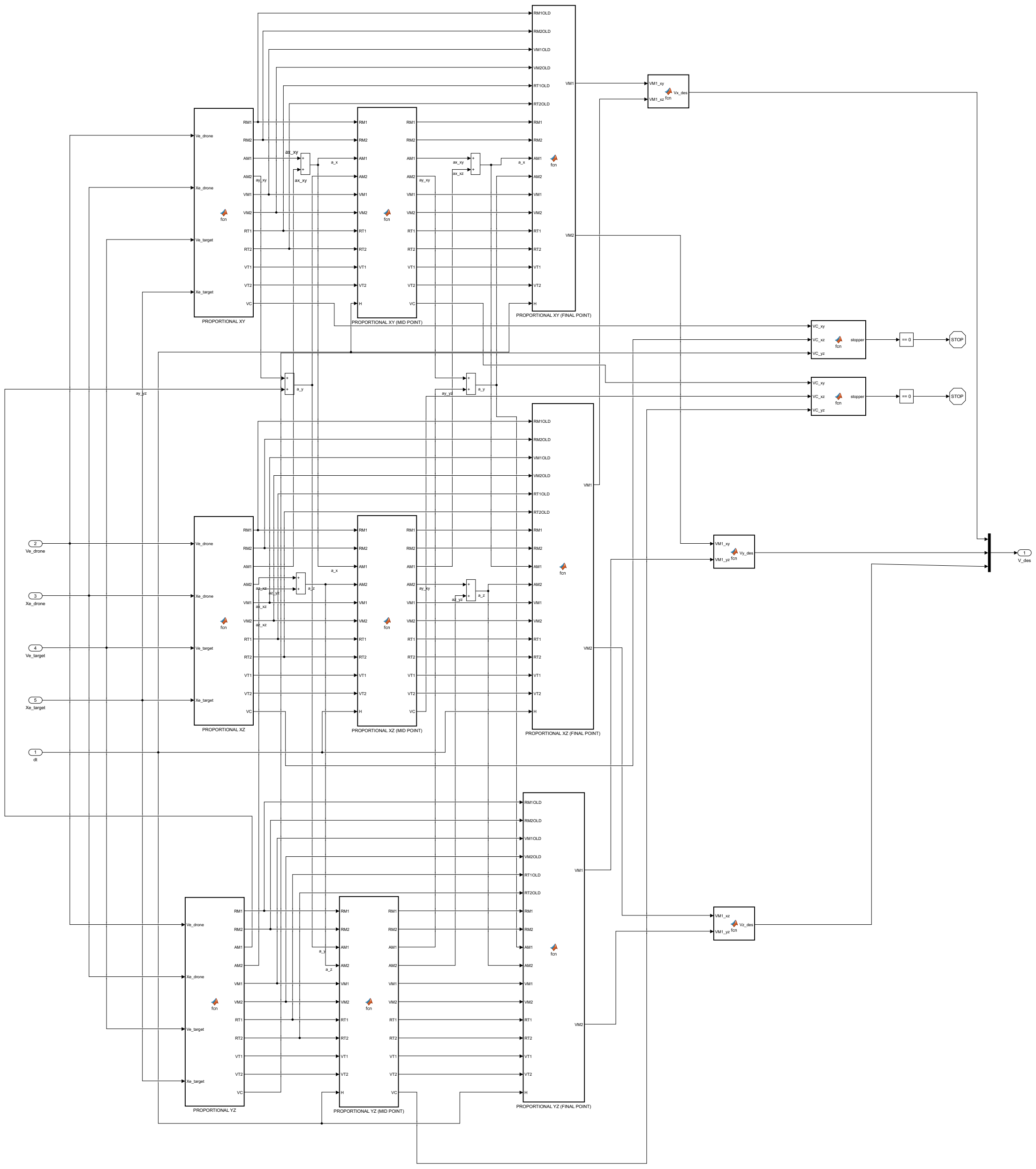

PROPORTIONAL XY
PROPORTIONAL XY (MID POINT)
PROPORTIONAL XY (FINAL POINT)
PROPORTIONAL XZ
PROPORTIONAL XZ (MID POINT)
PROPORTIONAL XZ (FINAL POINT)
PROPORTIONAL YZ
PROPORTIONAL YZ (MID POINT)
PROPORTIONAL YZ (FINAL POINT)
Ve_drone
Xe_drone
Ve_target
Xe_target
dt
V_des
Vx_des
Vy_des
Vz_des
stopper
STOP
>= 0
fcn

Figure 4.5: Internal architecture of the PN Guidance

The primary task at hand involved the adaptation of the initially proposed algorithm in a two-dimensional (2D) context for ballistic missiles to a three-dimensional (3D) framework. According to [6], the prescribed course of action was to systematically partition the 2D problem into three distinct 2D sub-problems, each corresponding to a specific plane in the xyz coordinate system—namely, the S_{xy}, S_{xz}, and S_{yz} planes (Fig: 4.6).

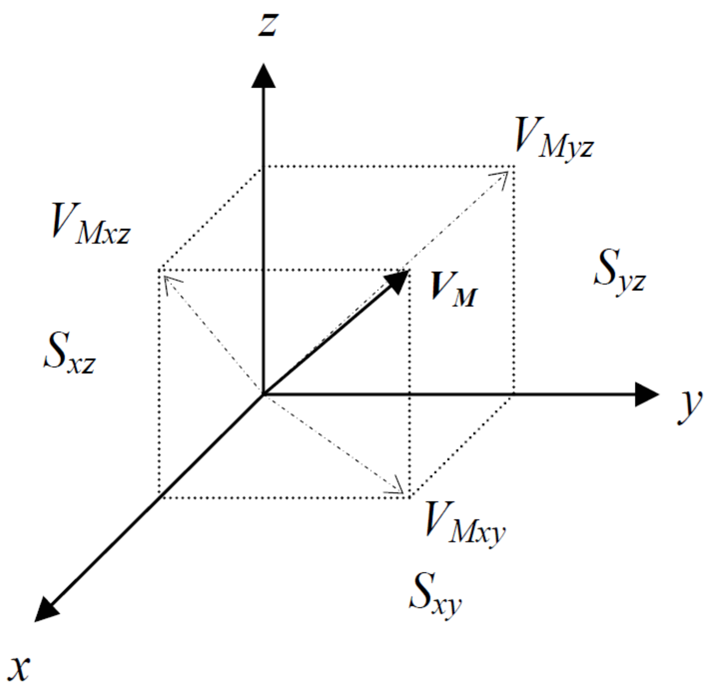


Figure 4.6: 3D problem

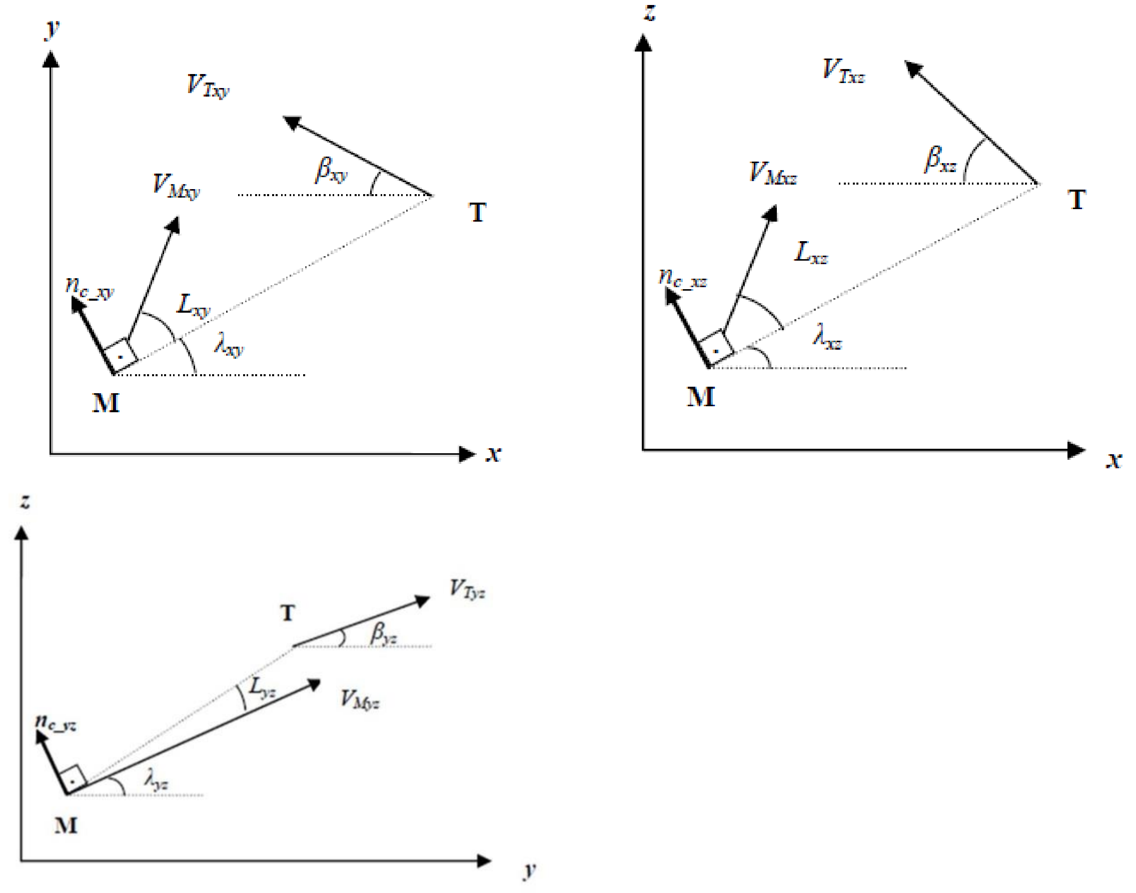


Figure 4.7: The projections of drone's M and target's T relative motion onto 3 planes, reprinted from [7]

From a mathematical perspective, the algorithm becomes as follows. Maintaining the notation seen in [14], the relative distance R_{TM} between interceptor and target becomes:

$$R_{TM} = \sqrt{R_{TMX}^2 + R_{TMY}^2 + R_{TMZ}^2} \tag{4.31}$$

The calculation of relative distances in each direction is feasible:

$$R_{TMX} = R_{TX} - R_{MX} \tag{4.32}$$

$$R_{TMY} = R_{TY} - R_{MY} \tag{4.33}$$

$$R_{TMZ} = R_{TZ} - R_{MZ} \tag{4.34}$$

The LOS angles are:

$$\lambda_{XY} = atan\frac{R_{TMY}}{R_{TMX}} \tag{4.35}$$

$$\lambda_{XZ} = atan\frac{R_{TMZ}}{R_{TMX}} \tag{4.36}$$

$$\lambda_{YZ} = atan\frac{R_{TMZ}}{R_{TMY}} \tag{4.37}$$

Target velocity vector projections onto S_{xy}, S_{xz}, and S_{yz} planes are:

$$V_{TMX} = V_{TX} - V_{MX} \tag{4.38}$$

$$V_{TMY} = V_{TY} - V_{MY} \tag{4.39}$$

$$V_{TMZ} = V_{TZ} - V_{MZ} \tag{4.40}$$

The LOS rate projections onto each plane are:

$$\dot{\lambda_{XY}} = \frac{R_{TMX}V_{TMY} - R_{TMY}V_{TMX}}{R_{TM}^2} \tag{4.41}$$

$$\dot{\lambda_{XZ}} = \frac{R_{TMX}V_{TMZ} - R_{TMZ}V_{TMX}}{R_{TM}^2} \tag{4.42}$$

$$\dot{\lambda_{YZ}} = \frac{R_{TMY}V_{TMZ} - R_{TMZ}V_{TMY}}{R_{TM}^2} \tag{4.43}$$

The closing velocity V_c:

$$V_{CXY} = -\dot{R_{TMXY}} = \frac{R_{TMX}V_{TMX} + R_{TMY}V_{TMY}}{\sqrt{R_{TMX}^2 + R_{TMY}^2}} \tag{4.44}$$

$$V_{CXZ} = -\dot{R_{TMXZ}} = \frac{R_{TMX}V_{TMX} + R_{TMZ}V_{TMZ}}{\sqrt{R_{TMX}^2 + R_{TMZ}^2}} \tag{4.45}$$

$$V_{CYZ} = -\dot{R_{TMYZ}} = \frac{R_{TMY}V_{TMY} + R_{TMZ}V_{TMZ}}{\sqrt{R_{TMY}^2 + R_{TMZ}^2}} \tag{4.46}$$

Hence, the commanded missile accelerations onto each plane from the PN guidance law are:

$$n_{C_{XY}} = N'V_{CXY}\dot{\lambda}_{XY} \tag{4.47}$$

$$n_{C_{XZ}} = N'V_{CXZ}\dot{\lambda}_{XZ} \tag{4.48}$$

$$n_{C_{YZ}} = N'V_{CYZ}\dot{\lambda}_{YZ} \tag{4.49}$$

Missile acceleration components for x, y, and z-axis are generated by combining two acceleration commands sharing the same axis. Fig: 4.7 suggests that the acceleration component of a particular axis interacts with the acceleration commands of two planes. Through the application of trigonometric functions, the unified acceleration components of the missile along the x, y and z axes can be computed as follows (Fig: 4.8):

$$a_{MX} = a_{X_{xy}} + a_{X_{xz}} = -n_{C_{XY}}sin(\lambda_{XY}) - n_{C_{XZ}}sin(\lambda_{XZ}) \tag{4.50}$$

$$a_{MY} = a_{Y_{xy}} + a_{Y_{yz}} = n_{C_{XY}}cos(\lambda_{XY}) - n_{C_{YZ}}sin(\lambda_{YZ}) \tag{4.51}$$

$$a_{MZ} = a_{Z_{xz}} + a_{Z_{yz}} = n_{C_{XZ}}cos(\lambda_{XZ}) + n_{C_{YZ}}cos(\lambda_{YZ}) \tag{4.52}$$

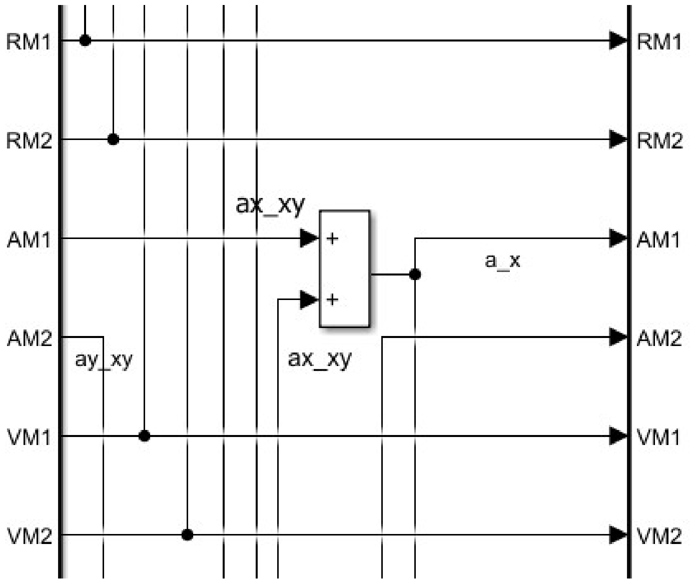


Figure 4.8: Computation of a_{MX} (The calculus of the other components relies according to the formulas 4.51 and 4.52)

After computing the respective acceleration components, the subsequent step involved the implementation of second-order Runge-Kutta. The rationale behind this step is rooted in the fact that the flight controller is commanded in velocity, compelling the calculation of the velocity component associated with the given acceleration.
Following the computation of velocity, verification blocks were introduced, as depicted in Fig: 4.9, to halt the simulation upon the drone reaching the target. This condition was established by setting the closing velocity in each of its components equal to zero. Naturally, in numerical terms, setting V_c equal to zero is not feasible; therefore, it was arbitrarily defined the simulation to cease when each component of V_c became less than zero.

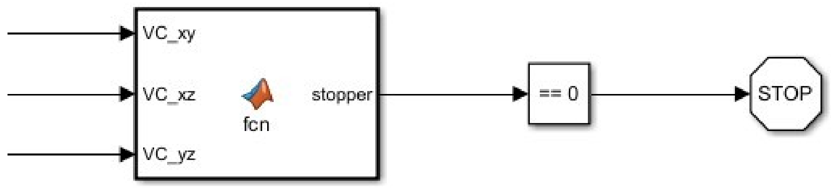


Figure 4.9: Stop condition for the simulation

Following the integration of the subsystem with other components of the complete model, the subsequent phase involved the execution of simulations to assess the algorithm's feasibility. However, this undertaking was accompanied by several challenges, notably a pronounced slowness in simulations due to the code's intricate complexity and the substantial volume of inputs supplied to each MATLAB function block.
The observed impediments, encompassing both the protracted simulation duration and the lack of code clarity, prompted a critical evaluation of the existing codebase. It became apparent that the code's inefficiency and lack of readability hindered its utility. This realization underscored the potential for the development of a more streamlined and comprehensible code structure, one that would facilitate a more

seamless comprehension for programmers approaching the code for the first time.

4.3.2 Second iteration

Therefore, upon recognizing the aforementioned challenges within the code, the subsequent iteration focused on the refinement of the codebase to address these issues. Specifically, the emphasis was placed on developing a code structure that would enhance readability for programmers, all while ensuring a certain degree of continuity and resemblance to the initially proposed code outlined in the previous section.

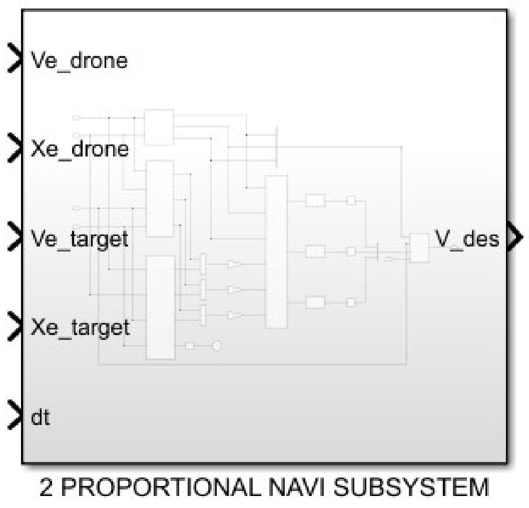


Figure 4.10: Simulink block of the PN Guidance subsystem (second iteration)

In Fig: 4.10 is possible to appreciate that the inputs and outputs of the proposed code are the same.
Within this block, the Simulink code developed can be observed in Fig: 4.11, exhibiting a notable enhancement in conciseness and comprehensibility for the reader. The Simulink implementation, by design, embodies a more streamlined structure that facilitates a clearer understanding of its intricacies.

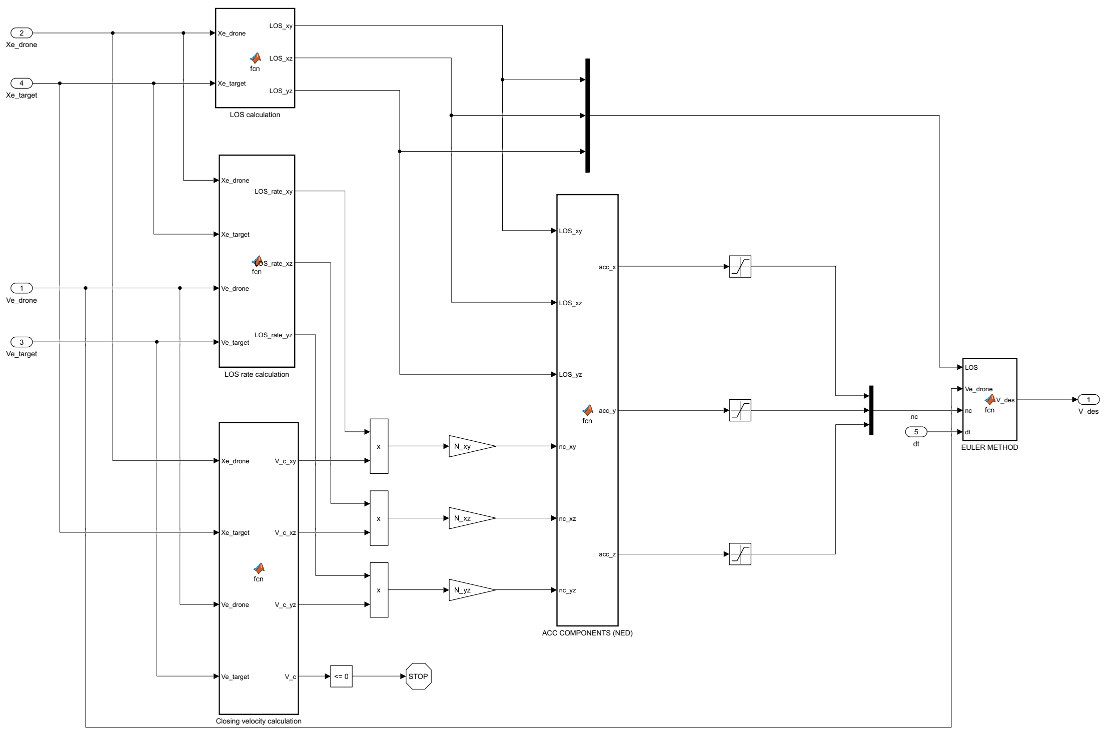


Figure 4.11: Internal architecture of the PN Guidance (second iteration)

As mentioned earlier, the computations performed meticulously adhere to the previously articulated theoretical framework. Indeed, Fig: 4.12 elucidates the dedicated block responsible for the computation of Line-of-Sight (LOS) within their respective planes S_{xy}, S_{xz}, and S_{yz}. The visual representation in Fig: 4.12 serves as a testament to the rigorous alignment of the executed calculations with the established theoretical principles outlined earlier.

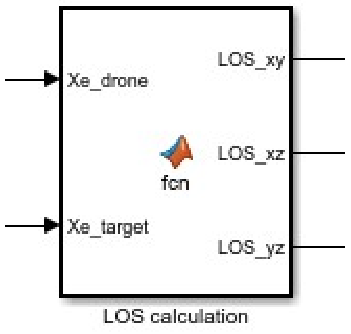


Figure 4.12: Simulink blocks employed for the LOS calculation (second iteration)

The closing velocity and LOS rate Simulink blocks Fig: 4.13; it can be observed that the fourth output of the "Closing velocity calculation" block also yields the total closing velocity V_c:

$$V_C = \sqrt{V_{CXY}^2 + V_{CXZ}^2 + V_{CYZ}^2} \tag{4.53}$$

which serves as a termination criterion for the simulation upon reaching negative values, as articulated in the initial iteration.

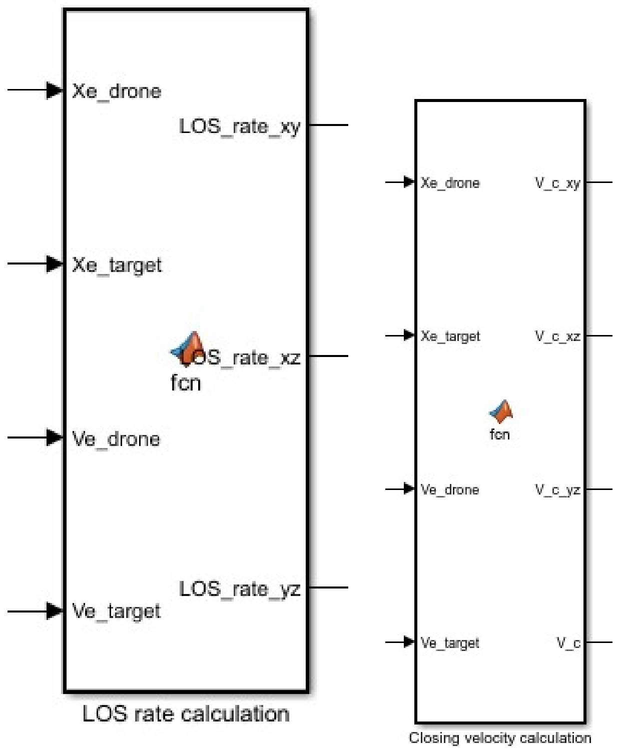


Figure 4.13: Simulink block employed for the $\dot{\lambda}$ and V_c calculation (second iteration)

The subsequent step involved the computation of the respective commanded acceleration components $n_{C_{XY}}$, $n_{C_{XZY}}$ and $n_{C_{YZ}}$, as specified in formulas 4.47, 4.48, and 4.49. Subsequently, these components were decomposed and aggregated along the corresponding x, y, and z (according to 4.50, 4.51, and 4.52) axes within the NED frame Fig: 4.14.

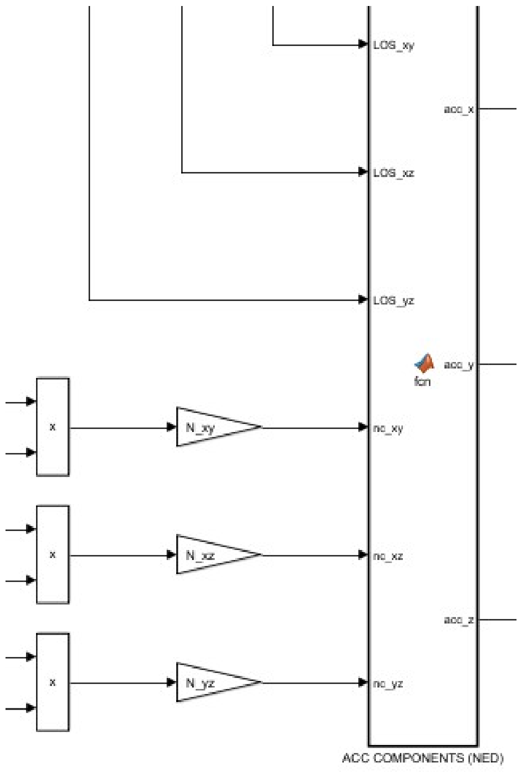


Figure 4.14: Portion of the Simulink code employed for the respective acceleration along the x, y and z directions in NED (second iteration)

The next phase of the process closely resembles the initial iteration, but with a distinct optimization approach. Specifically, efforts were made to streamline and improve the efficiency of the code by reducing the number of integration blocks. Instead of utilizing a second-order Runge-Kutta method, a more computationally efficient yet effective numerical integration method, namely the Euler method, was implemented, as depicted in the accompanying Fig: 4.15.

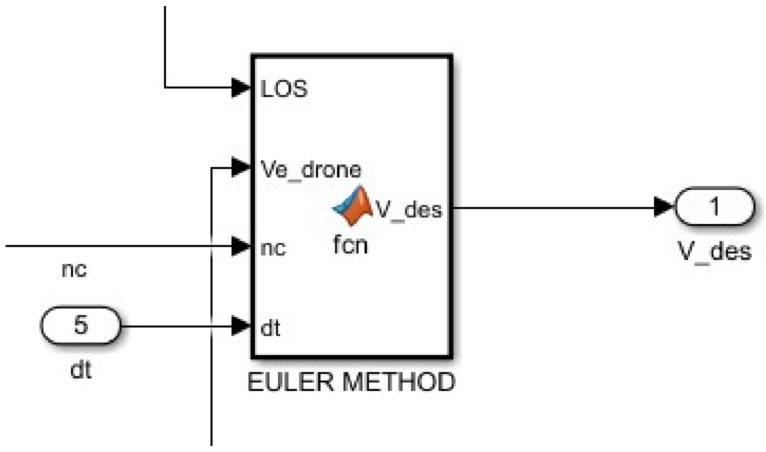


Figure 4.15: Implementation of the Euler integration method (second iteration)

4.3.3 Third iteration

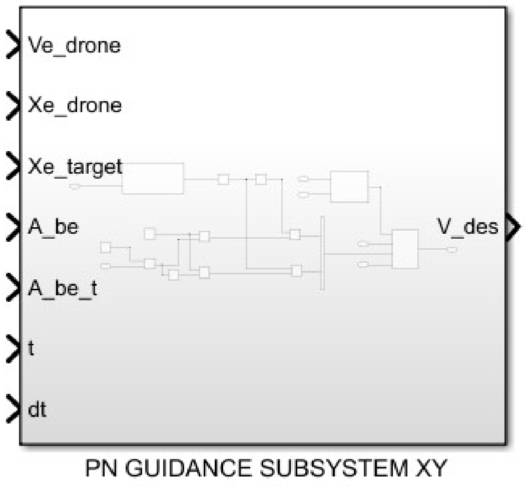


Figure 4.16: Simulink block of the PN Guidance subsystem (third iteration)

In this instance, an alternative approach was chosen, drawing inspiration from the architecture outlined in reference [9]. Grounded in a linearized problem, this architecture ensures a more streamlined and comprehensible framework. The subsystem illustrated in the figure corresponds to the S_{xy}, plane (those relevant to the S_{xz}, and S_{yz} planes remain identical, with the only distinction being the components em-

ployed). In this context, the subsystem takes as inputs, alongside the previously mentioned parameters—drone position, target position, and drone velocity—the drone's acceleration in NED axes A_{be}, the inertial acceleration of the target A_{be_t}, time t, and time step δt.
As evident from Fig: 4.17,
the architecture now exhibits a more streamlined and straightforward design. Additionally, the underlying algorithm has undergone a notable transformation. In this instance, a PN guidance law is employed, grounded in the concept of time-to-go t_{go}. The total flight time of the engagement t_F has been arbitrarily set to 10 seconds.

$$t_{go} = t_F - t \tag{4.54}$$

This concept has proven crucial in determining the respective gains, namely K_v, K_r, which represent the gains associated with relative velocity and relative position, respectively (calculated following the table in Fig: 4.29). These gains are subsequently multiplied by their respective x and $\dot{x}$ values (relative distance and relative velocity), these values have been computed through a double integration of the acceleration difference between target and drone, this information has been calculated within the MATLAB function block labeled "ACCELERATION MAGNITUDE DIFFERENCE" in Fig: 4.18. Within this block, the inputs consist of the corresponding inertial axis accelerations A_{be} and A_{be_t}.
Upon the calculation of the commanded acceleration, the subsequent step involves determining the Line-of-Sight and applying the Euler method (Fig: 4.19), as previously discussed in the preceding section.

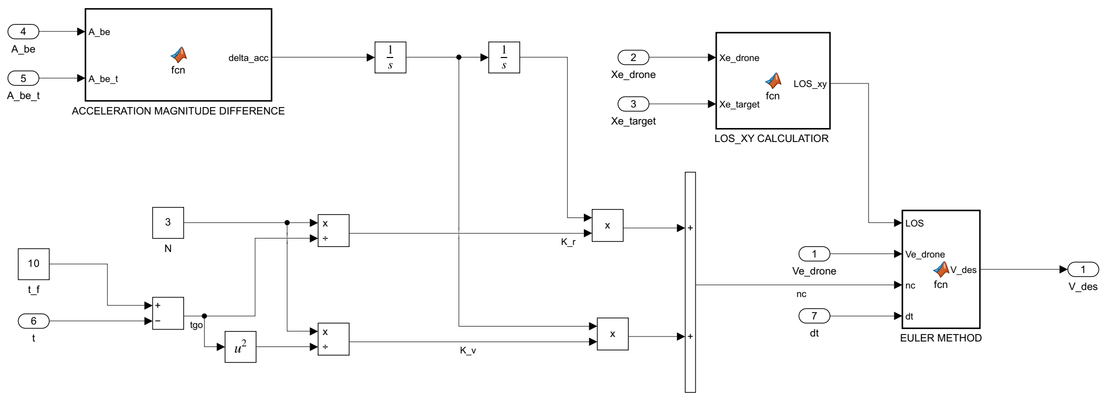


Figure 4.17: Internal design of the x-component PN Guidance subsystem (third iteration)

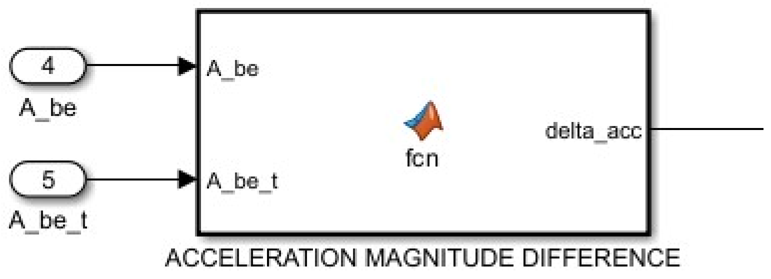


Figure 4.18: Simulink block employed for the δa in terms of magnitude (third iteration)

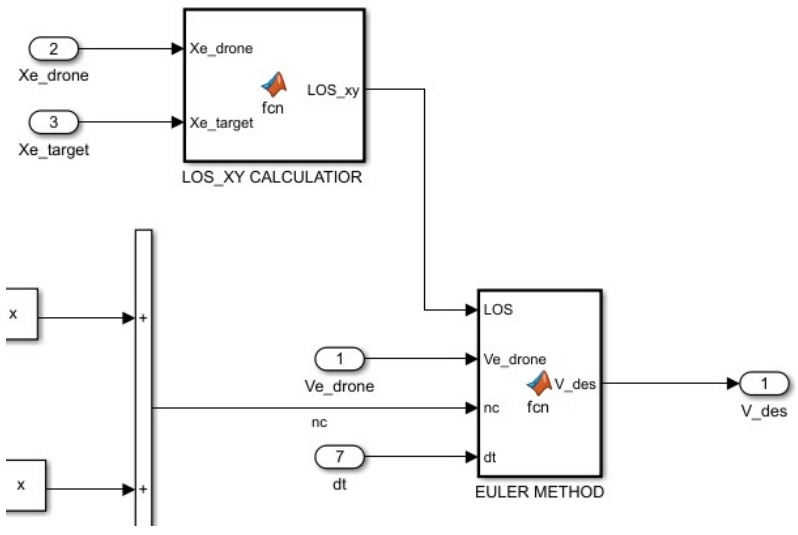


Figure 4.19: MATLAB function blocks employed for the calculation of the LOS and the subsequent integration through Euler method (third iteration)

4.4 Review of the original assumptions

Before introducing the final version of the Proportional Navigation (PN) algorithm, it is necessary to engage in a discussion regarding the assumptions proposed in the original algorithm. Indeed, with a more refined eye, one can identify the assumptions made. This contemplation arises from the observation that the codes presented earlier, once executed, yielded perplexing results and were entirely incapable of reaching the target, let alone intercepting it. Initially, the presumption was that the issue lay in an incorrect implementation of the algorithm. However, upon a thorough review of the theory, it became apparent that the assumptions made by the original missile algorithm were not pertinent to the specific scenario for which the guidance subsystem for drones was being developed. This realization prompted a critical examination of the foundational assumptions, which, when incongruent with the characteristics of drone guidance, posed significant challenges to achieving the intended objectives. Therefore, a nuanced consideration of these assumptions became imperative for the refinement of the guidance algorithms tailored for drone applications.
The assumptions made are indeed suitable for an interceptor missile; however, they do not hold for a drone.
The identified assumptions include the premise that the missile cannot have zero velocity—a reasonable assertion for intercepting missiles powered by solid propellants. Once initiated, such missiles cannot come to a halt; their sole objective is to impact the target. Moreover, the assumption stipulating that the interceptor must possess superior performance in terms of velocity magnitude and propulsive thrust is closely linked to the third and most robust assumption. This third assumption asserts that the commanded acceleration, denoted as n_c, does not aim to alter the missile's velocity magnitude, a parameter predetermined by the thrust profile and the propellant regression rate. Instead, its sole purpose is to induce a change in the missile's trajectory, positioning it for an impact scenario commonly referred to as the collision triangle (Fig: 4.20).

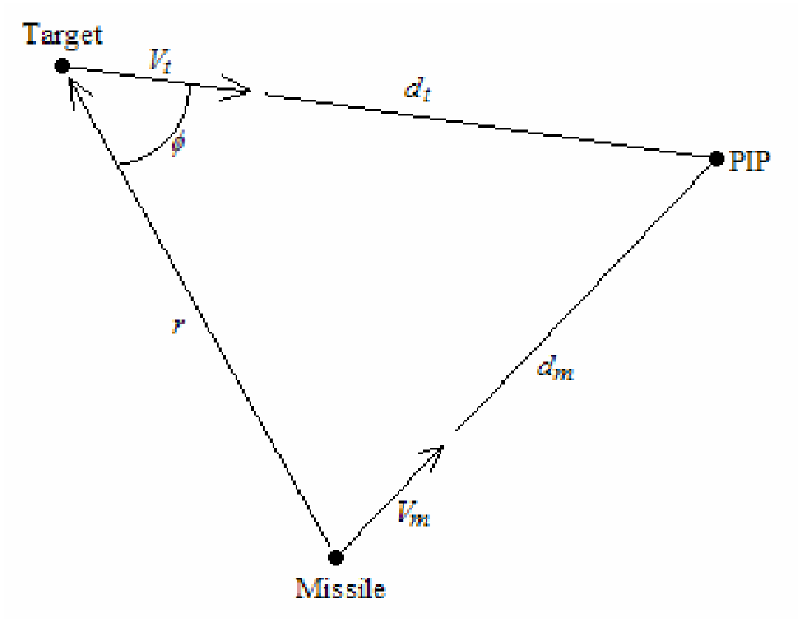


Figure 4.20: Collision triangle representation [4]

To underscore the inadequacy of this assumption, an example is presented—one that reveals a singularity rendering interception impossible for the drone. This underscores the critical need to reassess and refine these assumptions for the development of a guidance subsystem tailored explicitly for drone applications.
The guided missile's control system, as illustrated in Fig: 4.1, mandates that the commanded acceleration is consistently perpendicular to the relative distance between the missile and the target. In the specific scenario of an interceptor drone and target, consider a hypothetical condition where the target is positioned at a relative distance with a non-zero component solely along the x-direction R_{TMX} (with R_{TMY} and R_{TMZ} components being zero). Furthermore, assume that both entities share identical magnitudes of velocity along the x direction (both velocities along the y and z directions are null), resulting in a relative velocity V_{TMX} of zero. In such a case, the acceleration vector n_c would indeed be perpendicular to the relative distance, as per theoretical expectations. However, it would also be perpendicular to the velocity vector, failing to ensure any increase in the velocity magnitude. This contradicts the intended functionality of the guidance

algorithm, which should ideally facilitate such velocity increments in this particular scenario.
Given this analysis, it becomes imperative to reformulate the algorithm to address this specific challenge in the interceptor-drone and target scenario. The algorithm must be adapted to ensure alignment with the unique dynamics and requirements of this particular situation. This necessitates a nuanced revision of the guidance algorithm, optimizing it specifically to the interception conditions involving drones and targets.
Consequently, the algorithm's realignment is crucial for achieving the desired outcomes and enhancing the overall effectiveness of the guidance system in scenarios characterized by varying relative distances and velocities.

4.5 PN controller

Following the comprehensive examination of the assumptions embedded in the original algorithm, the next phase entails the practical implementation of the code, considering the previously elucidated assumptions. Throughout the code development process, a concerted effort has been made to closely align theoretical principles with practical considerations, ensuring that modifications to the original algorithm are judicious and essential. Striking a balance between theoretical foundation and real-world applicability, the code has been refined with precision, adhering strictly to what is indispensably necessary.
It is noteworthy that certain assumptions, pivotal in rendering the initial iterations ineffective, had to be selectively disregarded. However, this deliberate omission was confined to aspects directly influencing the efficacy of the algorithm, leaving the section concerning the definition of relevant gains untouched. This strategic decision stems from the recognition of the significance of maintaining continuity in certain foundational elements, such as gain definitions, while adapting others to enhance the algorithm's operational effectiveness.
Below in Fig: 4.21, the Proportional Navigation subsystem is visually depicted. The inputs showcased in the illustration encompass the respective positions of the target along the North-East-Down (NED) axes, the drone's position in inertial axes, and the inertial acceleration of the target.
In Fig: 4.22, the internal structure is presented, characterized by its streamlined and easily comprehensible design. Each directional component necessitates a dedicated channel within the algorithm, underscoring its ability to independently manage commands along the x, y, and z directions. This meticulous allocation of channels ensures a modular and efficient organization, allowing for the nuanced control of each axis without undue complexity.
Taking the x-component as an illustrative sample, identical in internal structure to the others, as depicted in Fig: 4.23, the development of this code drew inspiration from Source [9].

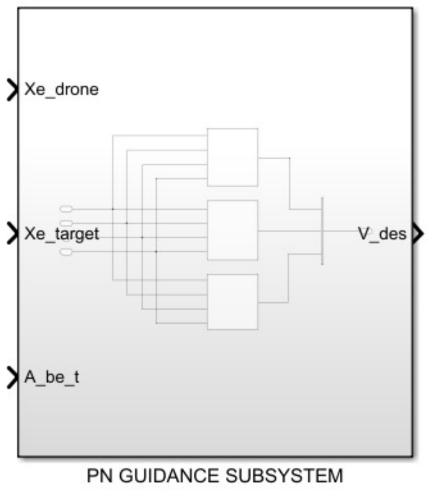


Figure 4.21: Simulink block of the PN Guidance subsystem

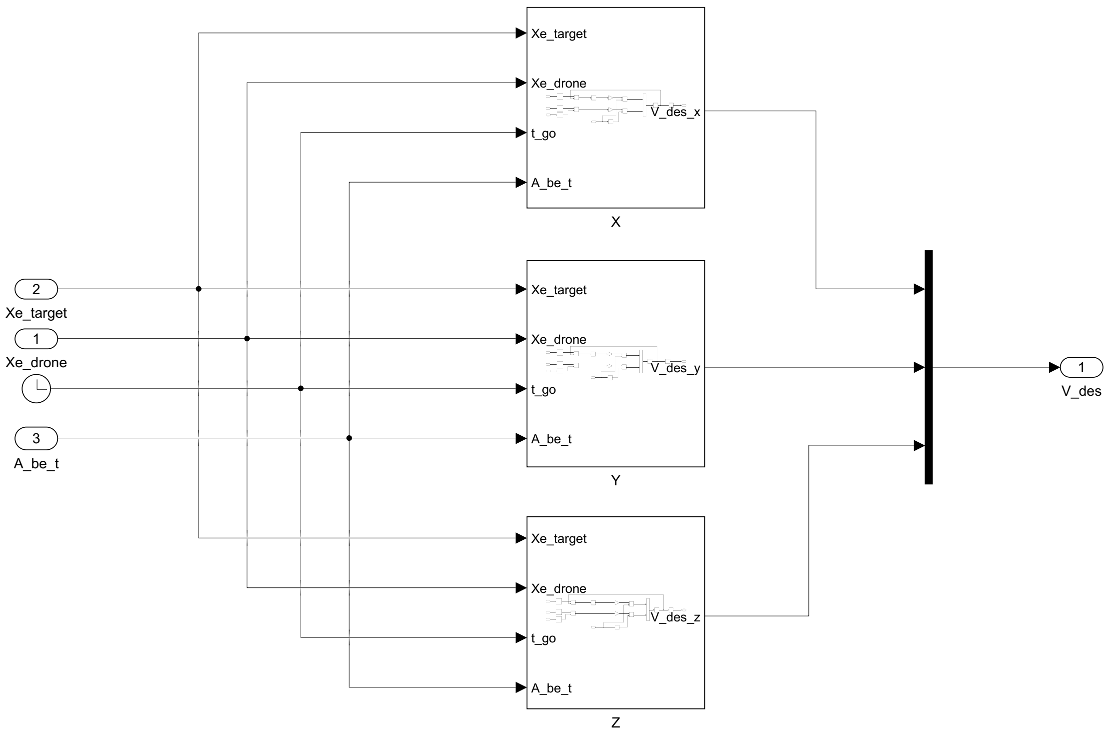


Figure 4.22: Internal architecture of the PN Guidance subsystem

Before delving into the intricacies of the code development, it is pertinent to briefly elucidate the role of Time-to-Go (t_{go}), as it proves to be a fundamental element in the algorithms devised. A succinct

understanding of t_{go}, as expounded in (4.54), is pivotal for comprehending the developed algorithms. t_{go}'s definition, as derived from the definition of the total flight time of the engagement (t_F), hinges on the time required to intercept the target without necessitating additional maneuvers.
In accordance with [9], t_{go} has to be estimated separately. The initial assumption guiding this endeavor was to define t_{go} as t, a choice consistent with the dynamics of the considered gains. Additionally, this assumption aligns logically with the simulation's initial stages, where one can reasonably anticipate high gains that gradually diminish as the drone approaches the target. This logic stems from the fact that the gains, namely K_r and K_v, are functions of $\frac{1}{t_{go}^2}$ and $\frac{1}{t_{go}}$ respectively (according to Fig: 4.29), reflecting a natural reduction in their values as the intercepting drone draws closer to the target.
Another crucial aspect is the strategic approach taken to circumvent singularities that may arise during the computation of gains. This challenge becomes particularly pronounced when the gains are functions involving $\frac{1}{t_{go}}$ and $\frac{1}{t_{go^2}}$, as these expressions become undefined when t_{go} equals zero, leading to computational errors. To address this, a standardized time offset of 0.1 has been systematically introduced across simulations for the Optimal Guidance Law (OGL), Proportional Navigation (PN), and Augmanted Proportional Navigation (APN).
Furthermore, it is imperative to note that the aforementioned assumption will also be taken into account in the algorithms of Augmented Proportional Navigation (APN) and Optimal Guidance Law (OGL).
As depicted in the figure, the initial step involves the computation of the error between the respective acceleration components A_{be_t} and n_c. Subsequently, this delta is processed through an integrator block, yielding the relative velocity $\dot{x}$ between the target and the drone along the x-component. It is paramount to meticulously configure the initial conditions within this block to ensure the accuracy of the launched simulations.
Simultaneously, in conjunction with this branch, the calculation of

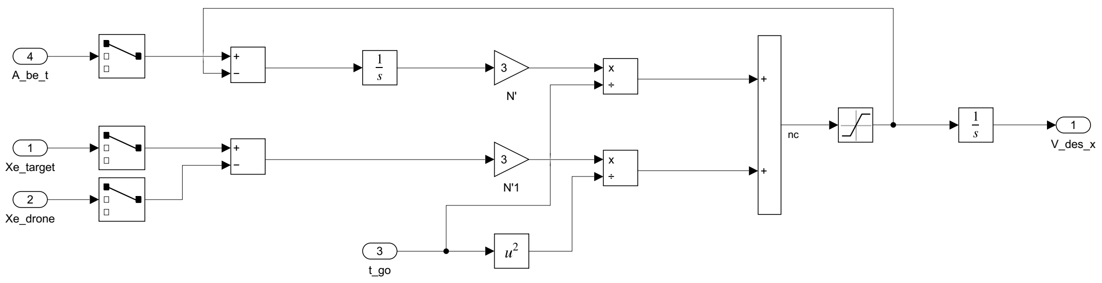


Figure 4.23: Simulink internal architecture of the x-component PN Guidance subsystem

relative position x unfolds. These computed values are then multiplied by their respective gains for relative velocity and relative position.

- K_r, the position-relative gain,

$$K_r = \frac{N'}{t_{go}^2} \tag{4.55}$$

- K_v, the velocity-relative gain,

$$K_v = \frac{N'}{t_{go}} \tag{4.56}$$

Where, the Effective Navigation Ratio N' has been set equal to 3.
This interconnected process underscores the intricate dynamics involved in the algorithm, where the precise determination of initial conditions plays a pivotal role in ensuring the fidelity and reliability of simulations.
Subsequently, the respective components are aggregated to derive the commanded acceleration n_c.

$$n_c = \frac{N'}{t_{go}^2}(x + \dot{x}t_{go}) \tag{4.57}$$

As a judicious measure, a saturator has been arbitrarily configured with saturation limits set at 2G and -2G. This deliberate choice aims

to align the algorithm with the realistic operational capabilities of the drone in which the guidance code is intended to be deployed. By imposing these constraints, the algorithm is tailored to adhere closely to the physical constraints of the specific drone system.
The integration of acceleration is meticulously executed, with due consideration to the initial conditions. This integration process yields the desired velocity component along the x-axis (V_{des_x}), a pivotal parameter within the broader guidance framework. Notably, this computed velocity component is then transmitted directly to the controller, signifying a seamless integration of the algorithmic outputs into the control architecture.
Externally to the considered subsystem, a termination block has been incorporated to govern the simulation cessation (Fig: 4.24). In initial iterations, the stopping condition was embedded within the guidance subsystem, dictating simulation halt when the closing velocity's modulus fell below 0. However, in the current configuration, this condition is external, predicated simply on the relative distance between the target and the drone. The simulation halts once the distance becomes as an arbitrarily set threshold. The decision to set the threshold at 1 [m] is arbitrary, chosen to represent a distance at which the drone, armed with an explosive charge, is capable of effectively neutralizing the pursued target.
This modification in the stopping condition's placement external to the guidance subsystem reflects an intentional shift in strategy. It introduces a more generalized criterion for simulation termination, focusing on the overarching goal of achieving a specific relative distance rather than the intricacies of closing velocity. The threshold, though arbitrary, aligns with practical considerations, emphasizing the drone's capability to engage and neutralize the target within the defined spatial proximity. This strategic refinement enhances the versatility and applicability of the simulation, catering to a broader spectrum of interception scenarios.
In summary, as evident from both the illustrative figure and the aforementioned explanation, the segment concerning the decomposition of

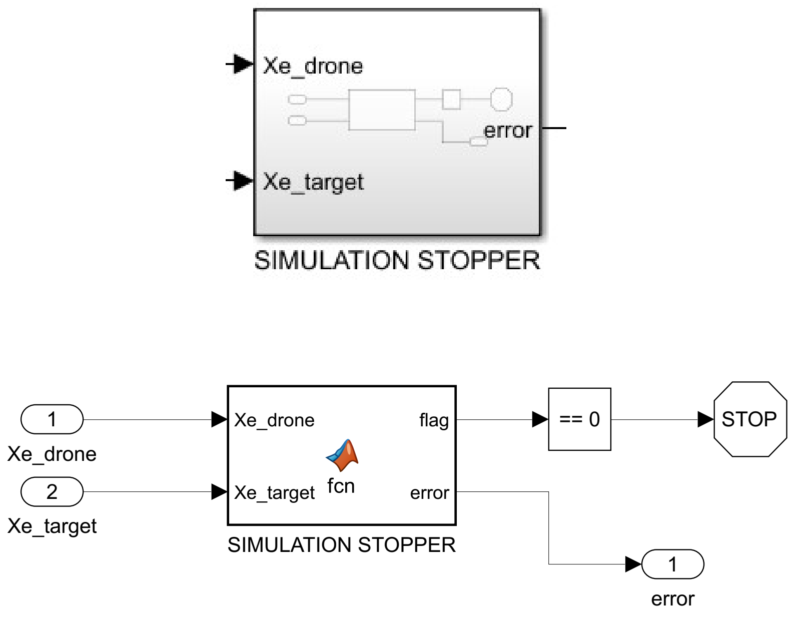


Figure 4.24: Simulink block delved to halt the simulation once the target is reached

the commanded acceleration (n_c) through a dedicated block for LOS calculation, followed by subsequent projection along the x, y, and z axes into their respective components, is notably absent. This absence arises from the direct utilization of acceleration and position components at the outset of the code. Importantly, it assumes that the commanded acceleration is not constrained to be perpendicular to the relative distance. Instead, it can assume any direction depending on the operational scenario in which the drone is deployed. Therefore, the commanded acceleration (n_c) serves not only to alter the direction of velocity but also the magnitude of the velocity itself.
This strategic departure from the conventional assumption of perpendicularity broadens the applicability of the algorithm, allowing it to adapt to diverse operational contexts where the directional influence of the commanded acceleration extends beyond a strict perpendicular orientation. This nuanced consideration of the acceleration's role introduces a versatile element to the algorithm, providing the flexibility needed to navigate varied scenarios effectively.

Furthermore, it is crucial to emphasize that this code does not presuppose a constant drone velocity strictly greater than zero. This acknowledgment of diverse velocity scenarios aligns the algorithm with the practical realities of drone operations, where velocity conditions may vary dynamically. Additionally, the intricate integration processes involving Euler's method or second-order Runge-Kutta have been supplanted by a simplified integrator, reinforcing the algorithm's computational efficiency without compromising accuracy or reliability. This streamlined approach contributes to the algorithm's agility and adaptability in addressing the complexities inherent in diverse operational environments.

4.6 APN Guidance Law

The Augmented Proportional Navigation Guidance Law is deduced from the linearized PN algorithm, in fact the line-of-sight λ is defined as:

$$\lambda = \frac{y}{R_{TM}} = \frac{y}{V_c(t_F - t)} \tag{4.58}$$

We can find the line-of-sight rate by taking the derivative of the preceding expression,

$$\dot{\lambda} = \frac{y + \dot{y}t_{go}}{V_c t_{go}^2} \tag{4.59}$$

Therefore, it is possible to highlight the expression of the commanded acceleration

$$n_c = N' V_c \dot{\lambda} = N' \frac{y + \dot{y}t_{go}}{t_{go}^2} \tag{4.60}$$

The content within the parentheses of the preceding equation denotes the anticipated separation between the missile and the target in future instances. In simpler terms, this expression, enclosed in parentheses, quantifies the miss distance that would ensue if the missile were to

refrain from further corrective acceleration and the target abstained from maneuvering. This expression is commonly referred to as the zero effort miss ZEM. Consequently, Proportional Navigation can be conceptualized as a guidance law wherein acceleration commands are inversely proportional to the square of the time remaining until intercept and directly proportional to the zero effort miss.
In the event of target maneuvers, the zero effort miss necessitates augmentation by an additional term. The updated equation for the zero effort miss, accounting for a constant target maneuver, is succinctly expressed as follows:

$$ZEM = y + \dot{y}t_{go} + 0.5n_T t_{go}^2 \tag{4.61}$$

where n_T is the target maneuver acceleration. Then, the n_c formula is displayed:

$$n_c = \frac{N'ZEM}{t_{go}^2} = N'V_c\dot{\lambda} + 0.5N'n_T \tag{4.62}$$

It is possible to appreciate that the augmented proportional navigation, is basically a proportional navigation law with an extra term to account for the maneuvering target (Fig: 4.25).

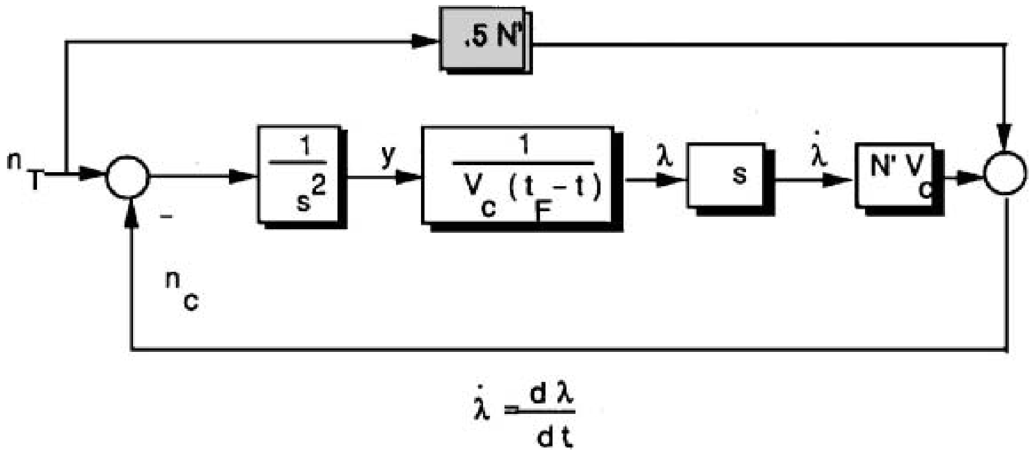


Figure 4.25: Augmented proportional navigation block architecture

4.7 APN Controller

In the course of developing the Augmented Proportional Navigation (APN) controller, a deliberate decision was made to adhere to the same conceptual framework expounded upon during the elucidation of the final model crafted for the Proportional Navigation (PN) controller. In parallel fashion, the assumptions delineated in the earlier discussion—those for instance, concerning the requisite perpendicular orientation of the commanded acceleration concerning the target-drone vector—were intentionally set aside. Instead, the focus shifted towards the novel assumptions introduced in the dedicated section.

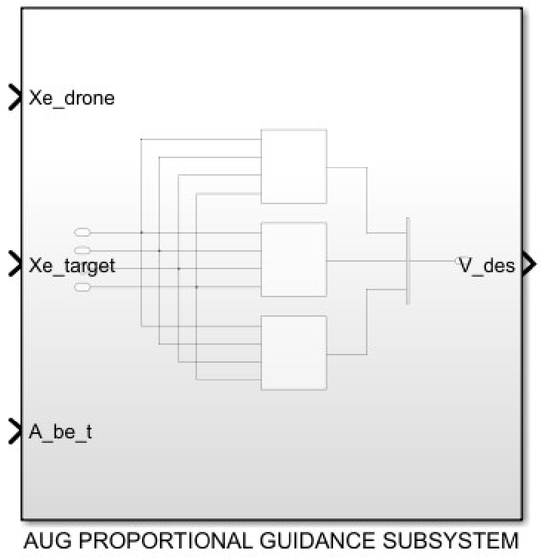


Figure 4.26: Simulink block of the APN Guidance subsystem

So, building upon the insights derived from the preceding section on PN controller development, the evolution of APN in Fig: 4.26 and Fig: 4.27 adheres to the same conceptual framework articulated earlier. In essence, the assumptions inherent in the PN subsystem form the foundational basis for the APN subsystem (as explained in the specific paragraph). Consequently, the key distinction between a PN subsystem and an APN subsystem lies in the singular utilization of the target's acceleration n_T in the computation of commanded acceleration n_c (Fig: 4.28):

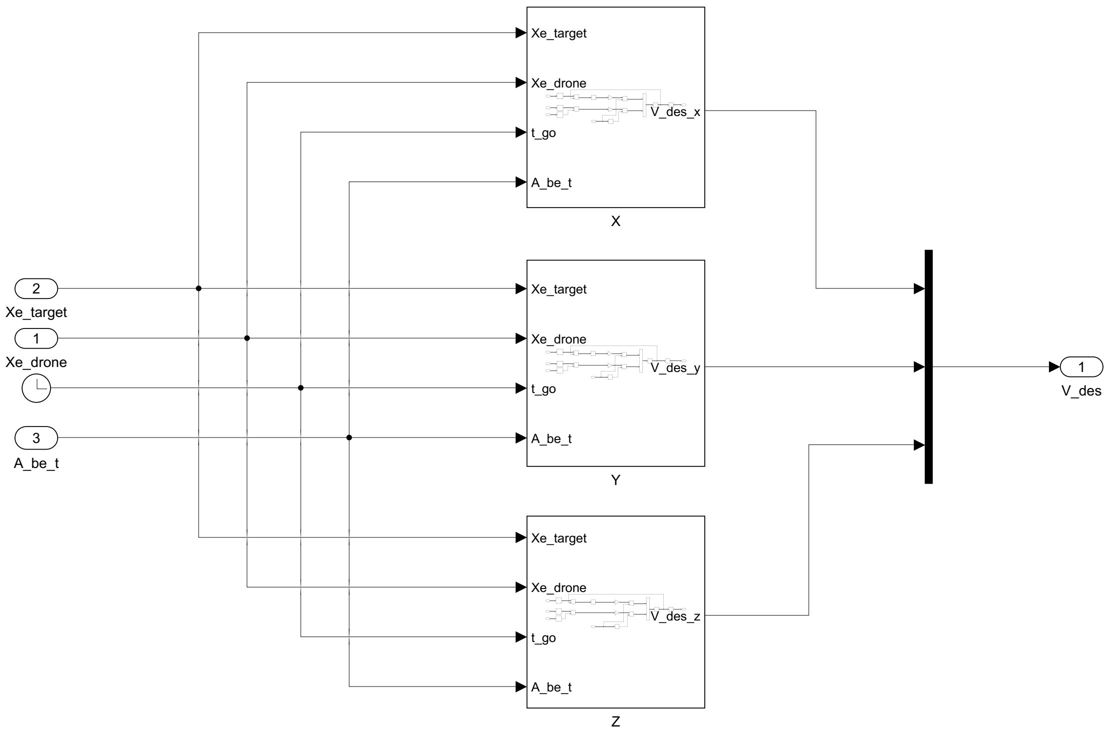


Figure 4.27: Simulink architecture of the APN Guidance subsystem

$$n_c = \frac{N'}{t_{go}^2}(x + \dot{x}t_{go} + 0.5n_T t_{go}^2) \tag{4.63}$$

Where, considering an Effective Navigation Ratio N' equal to 3, the computed gains are:

- K_r, the position-relative gain,

$$K_r = \frac{N'}{t_{go}^2} \tag{4.64}$$

- K_v, the velocity-relative gain,

$$K_v = \frac{N'}{t_{go}} \tag{4.65}$$

- K_{at}, the acceleration gain for the target,

$$K_{at} = \frac{N'}{2} \tag{4.66}$$

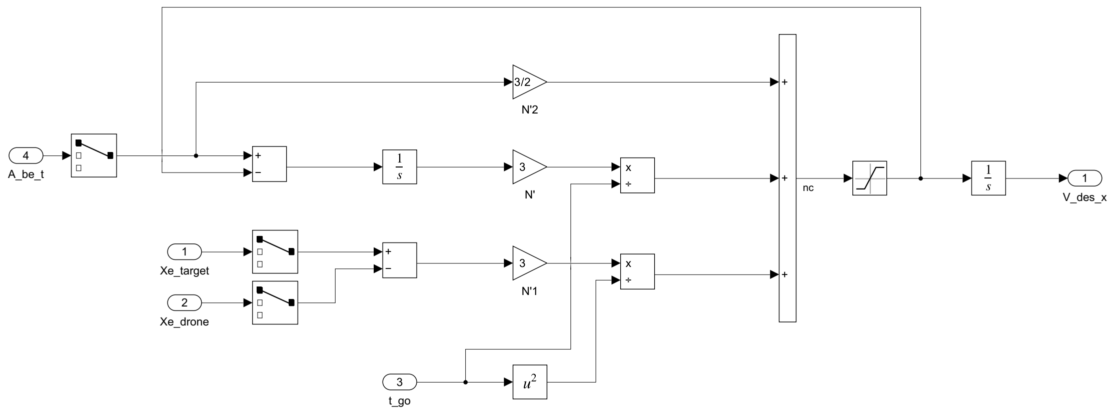


Figure 4.28: Simulink internal architecture of the x-component APN Guidance subsystem

By extending the principles established in PN, the augmentation in APN lies in the enhanced consideration of target dynamics. The incorporation of the target's acceleration as a pivotal parameter infuses an additional layer of sophistication into the guidance algorithm. This augmentation acknowledges and capitalizes on the evolving dynamics of the target, allowing the guidance system to dynamically adapt its response in light of the target's acceleration profile.
It is imperative to underscore that the fundamental assumptions, governing principles, and mathematical foundations set forth in the PN paradigm remain integral to the APN framework. The continuity in approach ensures a seamless transition from one subsystem to the other, facilitating a coherent and unified guidance strategy. The deliberate choice to augment the guidance algorithm with the target's acceleration introduces a nuanced responsiveness that aligns with the intricacies of dynamic target scenarios.
In essence, the development of the APN subsystem is an evolutionary step forward, leveraging the established principles of PN while

strategically incorporating the target's acceleration to enhance the algorithm's adaptability and efficacy. This nuanced integration underscores a methodical approach to guidance algorithm evolution, wherein each augmentation is rooted in a meticulous understanding of the underlying dynamics and operational exigencies.

4.8 Optimal Guidance Law

In the following section, an algorithm is expounded upon, showcasing a method to intercept targets with even greater efficiency through the deployment of missiles. The algorithm, meticulously detailed, accounts for the inherent lag resulting from the dynamics of the system, a factor that significantly influences the actual acceleration implemented in practice.

The utilization of target acceleration information has proven to be instrumental in the derivation of a guidance law aimed at mitigating missile acceleration requirements. It has been established that miss distance is intricately linked to guidance system lags, with larger guidance system time constants (T) generally corresponding to augmented miss distances. This correlation, although subject to certain parasitic effects and specific types of noise disturbances, underscores the critical role of minimizing guidance system lags for enhanced interception efficiency.

According to this, a more pragmatic assumption is introduced, where the missile is considered to respond to an acceleration command through a first-order lag transfer function [9].

$$\frac{n_M}{n_c} = \frac{1}{Ts+1} \tag{4.67}$$

where n_M is the effective missile acceleration. Consequently, the allusion to a non-ideal missile response is established. The time constant, denoted as T, serves as a composite (roll-up) function derived from the missile's response under a particular flight condition. It predominantly relies on the aerodynamic characteristics of the missile and the design of its flight control system. Before commencing the LQ optimization problem, it is necessary to define the state vector x:

$$\vec{x} = \begin{bmatrix} y \\ \dot{y} \\ n_T \\ n_M \end{bmatrix} \tag{4.68}$$

The input vector has been defined as:

$$\vec{u} = n_M \tag{4.69}$$

So, once the input and state vector have been defined, it is possible to set the LQ optimization problem, by defining the cost function J which must be minimized in order to achieve the final goal $(y(t_F) = 0$ subject to minimizing $(\int_{t_0}^{t_F} u^2\, dt))$.

$$minJ(\vec{x_0}(t_0), \vec{u_0}(t_0), x_0) = 0.5\|\vec{x}(t_F)\|^2 + 0.5 \int_{t_0}^{t_F} u(t)^2\, dt \tag{4.70}$$

The derivative of the state vector $\vec{x}$ is as following:

$$\dot{\vec{x}} = A\vec{x} + B\vec{u} \tag{4.71}$$

$$\dot{\vec{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{T} \end{bmatrix} \vec{x} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{T} \end{bmatrix} \vec{u} \tag{4.72}$$

In this context, in adherence to [9], a terminal velocity penalty is omitted to streamline the overall complexity of the guidance law. This choice results in the formation of the terminal penalty matrix denoted as $Q_F = diag[b, 0, 0, 0]$. However, the role of this matrix will be elucidated later in the following paragraph, where an in-depth analysis of the function of this parameter, denoted as b, will be undertaken. This exploration aims to provide a comprehensive understanding of the matrix's significance within the broader context of the discussed subject matter. However, before obtaining the generalized optimal guidance law, it is possible to appreciate that in the case of zero lag between the commanded and effective missile acceleration ($n_c = n_M$), the optimal guidance law becomes exactly as the APN law with a value of the navigation constant $N' = 3$.

$$ZEM_{OPT} = y + \dot{y}t_{go} + 0.5n_T t_{go}^2 - T^2(\frac{t_{go}}{T} + e^{\frac{-t_{go}}{T}} - 1)n_M \tag{4.73}$$

Which is:

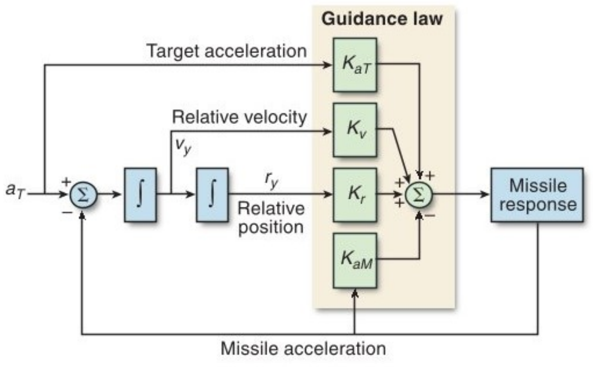
$$ZEM_{OPT} = ZEM_{APN} - T^2(\frac{t_{go}}{T} + e^{\frac{-t_{go}}{T}} - 1)n_M \tag{4.74}$$

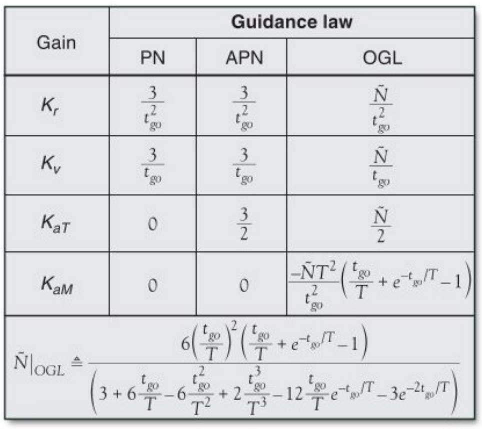
Instead, in the context of a Non-Ideal Missile Response Assumption, the generalized optimal guidance law is time-varying:

$$n_c = \frac{N'}{t_{go}^2}(y + \dot{y}t_{go} + 0.5n_T t_{go}^2 - T^2(z + e^{-z} - 1)n_M) \tag{4.75}$$

$$N' = \frac{6z^2(e^{-z} - 1 + z)}{2z^3 + 3 + 6z - 6z^2 - 12ze^{-z} - 3e^{-2z}} \tag{4.76}$$

Where $z = \frac{t_{go}}{T}$.
Hereafter on the left, a summarizing diagram is presented, illustrating the implementation of the various architectures explained thus far in a conventional block diagram. On the right, a summary table is provided, encompassing all the gains associated with the different guidance laws.





Gain	Guidance law		
	PN	APN	OGL
K_r	$\frac{3}{t_{go}^2}$	$\frac{3}{t_{go}^2}$	$\frac{\tilde{N}}{t_{go}^2}$
K_v	$\frac{3}{t_{go}}$	$\frac{3}{t_{go}}$	$\frac{\tilde{N}}{t_{go}}$
K_{aT}	0	$\frac{3}{2}$	$\frac{\tilde{N}}{2}$
K_{aM}	0	0	$\frac{-\tilde{N}T^2}{t_{go}^2}\left(\frac{t_{go}}{T}+e^{-t_{go}/T}-1\right)$
$\tilde{N}\big\rvert_{OGL} \triangleq \frac{6\left(\frac{t_{go}}{T}\right)^2\left(\frac{t_{go}}{T}+e^{-t_{go}/T}-1\right)}{\left(3+6\frac{t_{go}}{T}-6\frac{t_{go}^2}{T^2}+2\frac{t_{go}^3}{T^3}-12\frac{t_{go}}{T}e^{-t_{go}/T}-3e^{-2t_{go}/T}\right)}$			

Figure 4.29: The feedback structure of the PN, APN, and OGL guidance laws.
The varying complexity of these guidance laws becomes evident as additional assumptions are introduced concerning engagement and missile response characteristics. The diagram underscores the notable escalation in complexity when transitioning from ideal to non-ideal interceptor response assumptions.

4.9 OGL Controller

The formulation of code based on the Optimal Guidance Law (OGL) has presented a considerable degree of difficulty. This complexity has been notably pronounced in discerning the specific roles of various parameters and their subsequent impact on the drone's behavior post-implementation.
To commence our exploration, Fig: 4.30 visually encapsulates the coding block, delineating the inputs and outputs. Strikingly, these parameters closely echo those observed in preceding guidance subsystems. However, navigating through the intricacies of the OGL algorithm demanded a nuanced understanding of the unique parameters and their intricate relationships.
Here in Fig: 4.31, an implementation of the Optimal Guidance Law algorithm is observed for each axis - x, y, and z. The assumptions previously articulated remain pertinent to the development of this code.

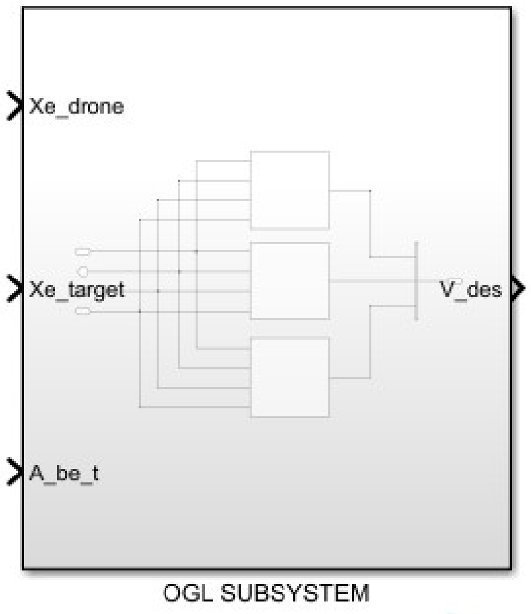


Figure 4.30: Simulink block of the OGL subsystem

In this context, the algorithm is instantiated separately for each axis, addressing the distinctive dynamics associated with motion along x, y, and z. This tailored approach acknowledges the unique characteristics and requirements of each spatial dimension, contributing to a more precise and effective application of the OGL algorithm within the multidimensional framework.

During the implementation, a concerted effort has been made to adhere closely to the theoretical framework outlined in [9]. Fig: 4.32 illustrates the proposed internal architecture, focusing exclusively on the x-component, as the other components mirror the one depicted. Commencing from the top, the inertial acceleration input A_{be_t} from the target is considered, specifically focusing on the x-component within the relevant block.

Subsequently, the error is computed between the x-component of the target's inertial acceleration and the actual acceleration of the drone n_d. The latter takes into account the dynamic response times of the drone, recognizing that the commanded acceleration n_c will not be instantaneously executed but will rather be subject to a certain delay (as showed in 4.77).

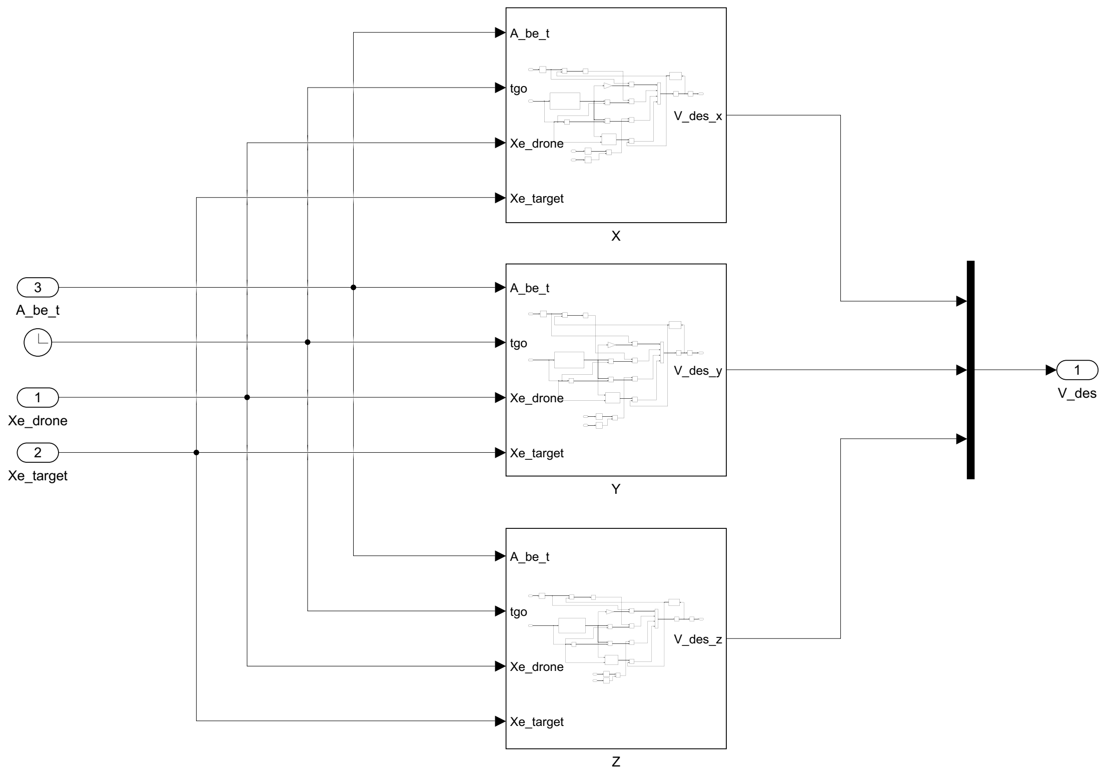


Figure 4.31: Internal architecture of the OGL subsystem

$$\frac{n_d}{n_c} = \frac{1}{Ts + 1} \tag{4.77}$$

This delay is represented by the constant T, arbitrarily set at 0.01. This value appears acceptable, given the assumed context of a highly responsive drone, poised to operate with targets capable of executing potentially abrupt maneuvers.
It is noteworthy that the temporal dynamics inherent in the drone's response are critical considerations in the formulation of the control strategy. The introduction of the delay constant T accounts for the realistic dynamics of the drone's actuation, contributing to a more accurate representation of the control system's behavior. This temporal aspect becomes particularly significant in scenarios where the drone needs to respond promptly to dynamic changes in the target's motion. Following the computation of the error, it undergoes integration through

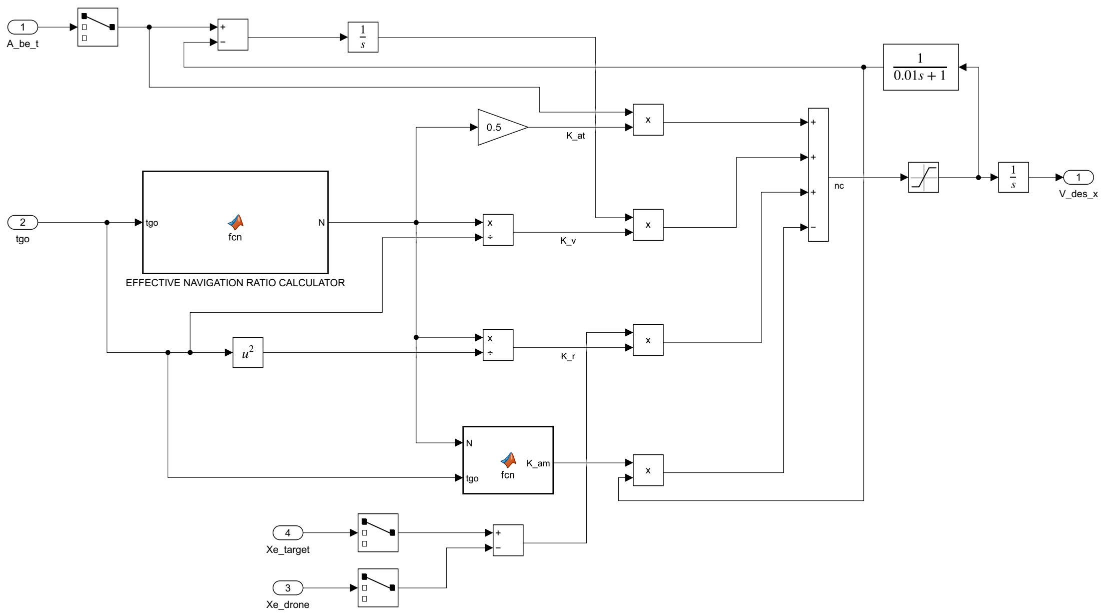


Figure 4.32: Internal architecture of the x-component OGL subsystem

an integrator, facilitating the derivation of the relative velocity $\dot{x}$ between the target and the drone. In conjunction with this branch, an additional Simulink branch takes input from the positions of the target and drone in the NED frame. Within this subsystem, the error between the two positions is calculated, resulting in the determination of the relative position x.

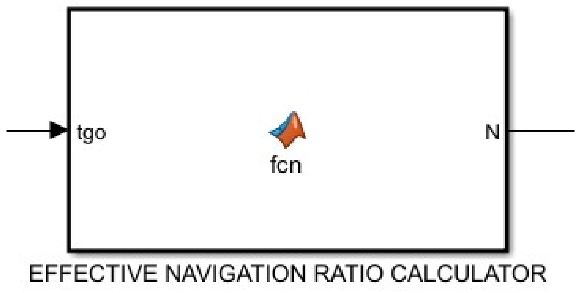


Figure 4.33: MATLAB function block employed in the calculation of the Effective Navigation Ratio

Meanwhile, the computation of the Effective Navigation Ratio N' coefficient takes place within a dedicated MATLAB Simulink block (Fig:

4.33), as evident from Formula (4.78),

$$N' = \frac{6z^2(e^{-z} - 1 + z)}{2z^3 + 3 + 6z - 6z^2 - 12ze^{-z} - 3e^{-2z} + \frac{6}{bT^3}} \tag{4.78}$$

$$z = \frac{t_{go}}{T} \tag{4.79}$$

revealing its clear dependence on time. When the original Optimal Guidance Law (OGL) for missile guidance was expounded upon in the dedicated chapter, the formula for calculating the navigation coefficient featured a parameter b belonging to the penalty matrix $Q_F = diag[b; 0; 0; 0]$.
According to Source [9], this parameter was made to tend towards infinity in order to achieve, in theory, the definitive OGL, which is:

$$N' = \frac{6z^2(e^{-z} - 1 + z)}{2z^3 + 3 + 6z - 6z^2 - 12ze^{-z} - 3e^{-2z}} \tag{4.80}$$

Consequently, in the initial iteration of this algorithm, the parameter b was set to tend towards infinity to align with the theoretical underpinnings.
However, the role of this parameter will be thoroughly examined in a subsequent paragraph, specifically underscoring its significance during the optimization process of the pertinent guidance subsystem. This nuanced exploration will shed light on the theoretical motivations behind the initial infinity tendency of b and, in turn, elucidate its practical implications for the optimization dynamics.
Subsequent to the determination of the effective navigation ratio, the calculation of gains is computed as follows:

- K_{at}, the acceleration gain for the target,

$$K_{at} = \frac{N'}{2} \tag{4.81}$$

- K_r, the position-relative gain,

$$K_r = \frac{N'}{t_{go}^2} \tag{4.82}$$

- K_v, the velocity-relative gain,

$$K_v = \frac{N'}{t_{go}} \tag{4.83}$$

- K_{am}, the acceleration gain for the actual drone motion,

$$K_{am} = \frac{N'T^2}{t_{go}^2}(\frac{t_{go}}{T} + e^{-\frac{t_{go}}{T}} - 1) \tag{4.84}$$

Once these gains have been computed, the desired commanded acceleration is:

$$n_c = \frac{6z^2(e^{-z} - 1 + z)}{t_{go}^2} \frac{(x + \dot{x}t_{go} + 0.5n_T t_{go}^2 - T^2(z + e^{-z} - 1)n_d)}{(2z^3 + 3 + 6z - 6z^2 - 12ze^{-z} - 3e^{-2z}) + \frac{6}{bT^3}} \tag{4.85}$$

$$z = \frac{t_{go}}{T} \tag{4.86}$$

In the provided context, x represents the relative distance between the target and the drone along the x direction, $\dot{x}$ denotes the relative velocity, n_T signifies the acceleration of the target, and n_d characterizes the actual acceleration of the drone.
It is noteworthy that the formula expressing the commanded acceleration is presented in its generalized form, wherein the parameter b is explicitly featured. It is important to recall that, in the initial iterations, the parameter b was initially considered to tend towards infinity, aligning with the theoretical proposition outlined in reference [9].
Upon the computation of the commanded acceleration, a saturator was introduced with an upper limit set at 2G and a lower limit set at -2G. Subsequently, this acceleration was integrated to derive the desired

velocity component along the x-axis V_{des_x}, which is then forwarded to the controller.

4.9.1 Optimization

As mentioned previously, the initial approach was guided by the principles outlined in [9], which advocates for the parameter b within the penalty matrix Q_F to tend towards infinity. Consequently, simulations were initiated with the theoretical optimal value of the effective navigation ratio N' (4.80). Upon launching the simulation, it became apparent that the drone successfully reached the target. This assessment was conducted through a comprehensive visualization process, involving plots of the x, y, and z components of both the target and the drone. Each component was displayed in a separate scope (Fig: 4.34), enabling a clear assessment not only of the target's attainment but also of the drone's trajectory.

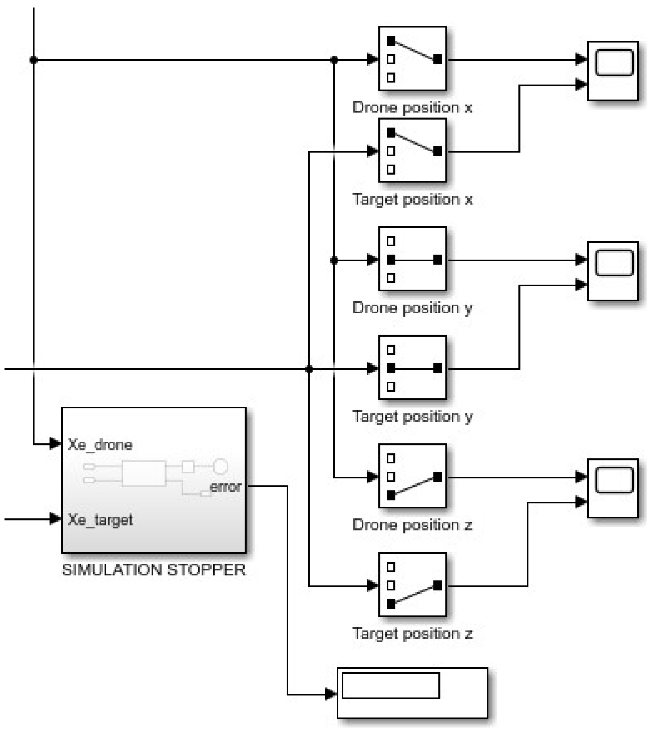


Figure 4.34: Scope blocks to visualize each position component (x, y, z) of the target in comparison to the drone's position

Upon scrutiny, it was observed that, considering the parameter b tending towards infinity, the x and y components performed commendably, efficiently reaching the target as showed in Fig: 4.35.

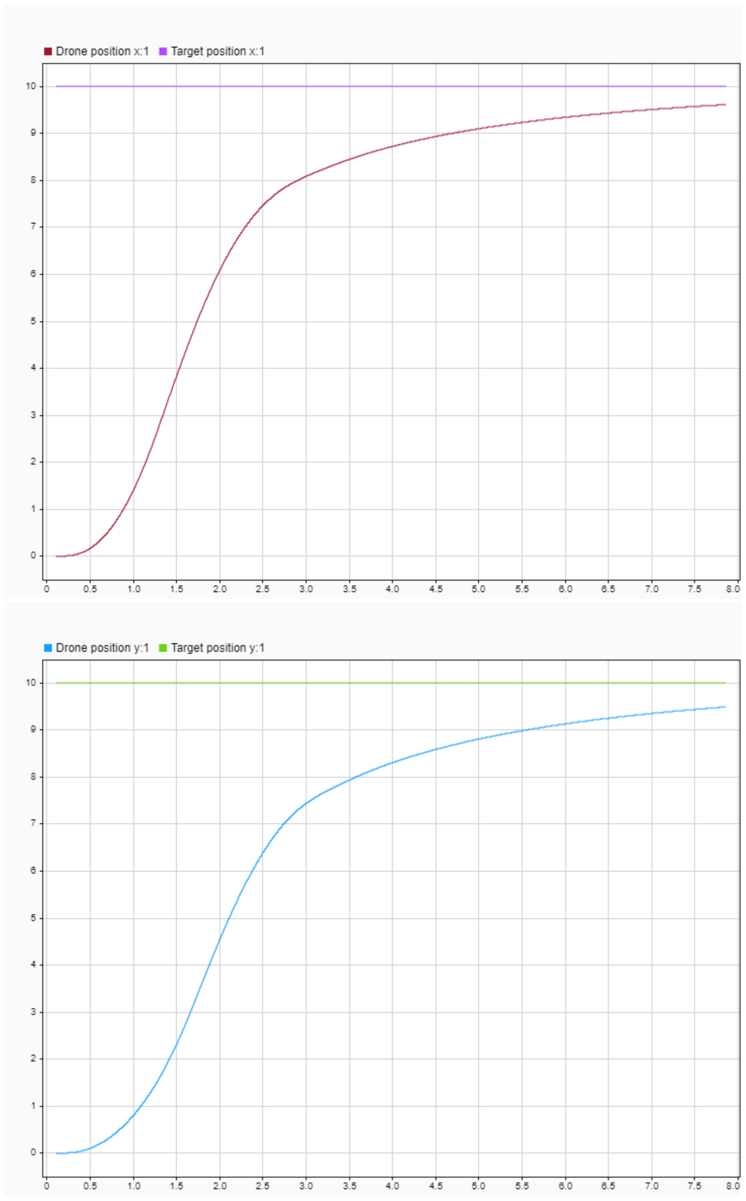


Figure 4.35: Plot illustrating the fixed x and y-components of the target at 10 [m], contrasted with the drone's position

However, an intriguing observation surfaced with the z component. While effective in achieving the target, as indicated by the fulfillment of the task, an excessive overshoot was noted when examining the drone's altitude during the simulation (Fig: 4.36). This phenomenon significantly compromised a crucial aspect—the efficiency of the algorithm.

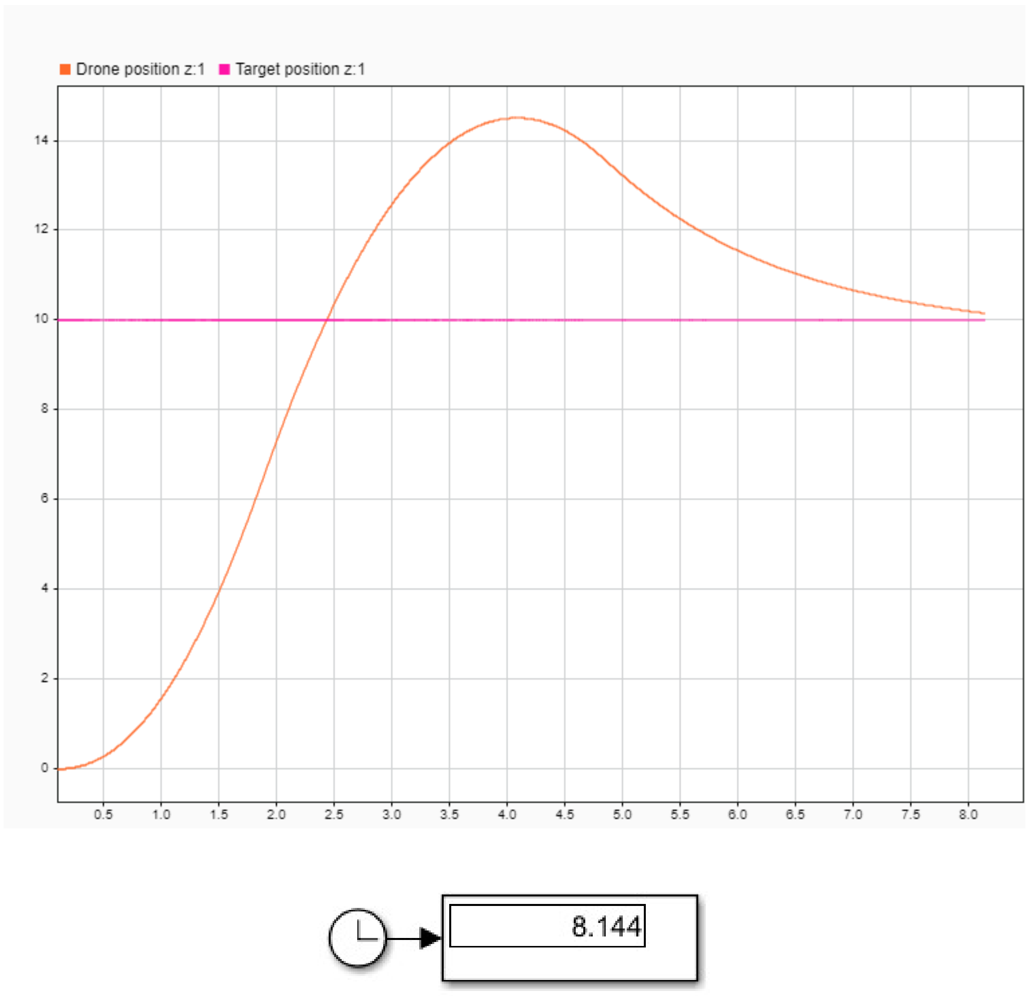


Figure 4.36: Plot illustrating the fixed z-component of the target at an altitude of 10 [m], contrasted with the drone's position and the corresponding time needed to reach the target

The efficiency of the algorithm is integral to meeting specific requirements, particularly the imperative to reach the target in the shortest possible time. The observed overshoot phenomenon undeniably extended interception times, contravening the objective of minimizing the time required for target acquisition.
This realization prompted a reassessment of the initial inclination towards an infinite value for b, as the harmful impact on efficiency became apparent.
In the subsequent phases of simulation iterations, an exhaustive exploration of diverse values for the parameter b was undertaken through

a meticulous trial-and-error procedure. This iterative tuning process aimed to strike a delicate equilibrium, rectifying the overshooting issue while remaining closely aligned with theoretical principles. Within this ongoing refinement, the final selected value for b settled at 0.1.
The determination of this specific value represented a conscientious choice, informed by a dual objective. On one hand, the value of 0.1 was sufficiently diminutive to notably curtail overshooting tendencies, addressing a critical concern identified in prior simulations. On the other hand, it was imperative to ensure that the chosen value did not excessively compromise the time required to reach the target, thus safeguarding the algorithm's overall efficiency.
Hereafter is presented the plot in Fig: 4.37 depicting the positional relationship between the target and the drone in the ultimate iteration of the algorithm, following the optimization process. As discernible, the time required to reach the target has been notably reduced, and there is an absence of overshooting, indicative of the enhanced performance achieved through the optimization endeavors.
The final rendition of the algorithm stands as a testament to the iterative refinement process, where meticulous adjustments, particularly concerning the parameter b and associated gains, have yielded a more expedient and precise trajectory tracking. The visual representation of the target-drone positions showcases a harmonized convergence without the lingering oscillations seen in earlier versions.

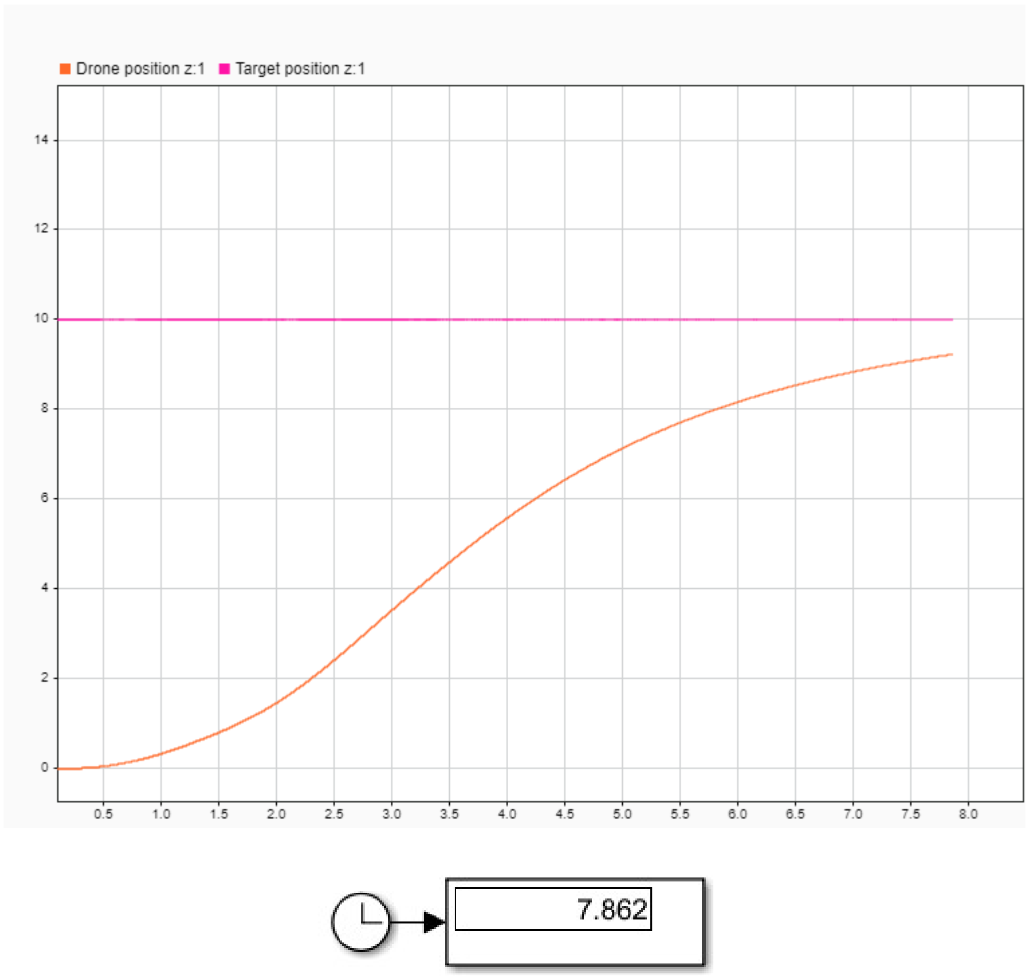


Figure 4.37: Plot illustrating the fixed z-component of the target at an altitude of 10 [m], contrasted with the drone's position and the corresponding time needed to reach the target (after the optimization)

4.10 PID Guidance Controller

Ultimately, there was a deliberate endeavor to develop a guidance code from scratch, indicating the initiation of the coding process without reliance on external sources. The objective was to conceive a functional and efficient code, entirely conceived without recourse to existing references or frameworks and to compare the relative results associated with the corresponding of the PN, APN and OGL guidance controllers. The employed strategy relies on a Proportional-Integral-Derivative (PID) control strategy. Essentially, the fundamental concept is derived from the interception nature of the drone, aiming to strike a target with arbitrary initial conditions. It is logical, therefore, to utilize the respective positions of the target and the interceptor drone as input data. Consequently, whenever there exists a relative distance between the drone and the target, the controller will not receive null commands.

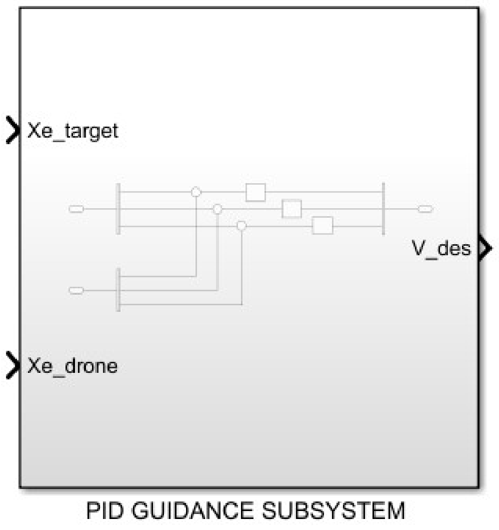


Figure 4.38: Simulink block of the PID Guidance subsystem

As evident from the Fig: 4.38, the output aligns with the anticipated outcome, namely the desired velocity V_{des}. The internal architecture in Fig: 4.39 is notably straightforward and streamlined. Following the decomposition of the position vectors for both the target and the drone, the subsequent step involves computing the relative error, which is then fed directly into the PID controller in order to obtain the desired velocity.

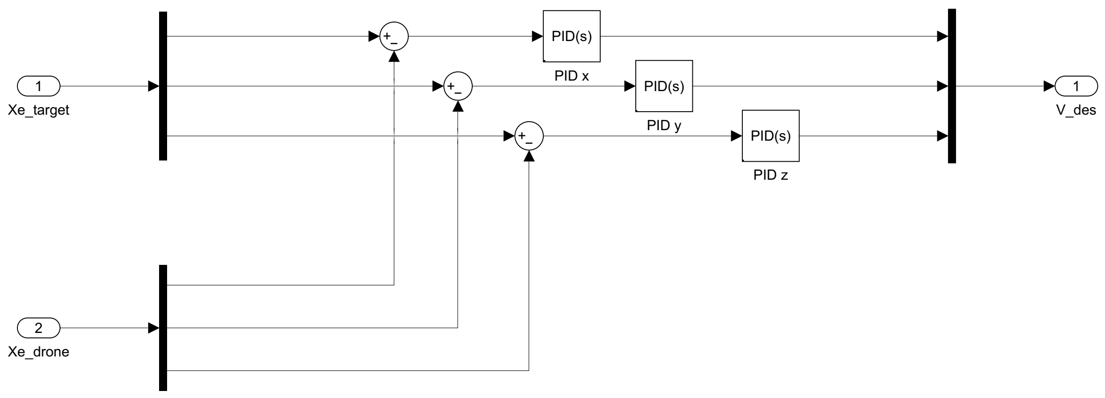


Figure 4.39: Internal architecture of the PID Guidance subsystem

4.10.1 Tuning

The optimization of the PID guidance subsystem necessitated a meticulous tuning process, wherein all the Proportional-Integral-Derivative gains were manually adjusted through a trial-and-error procedure. This tuning endeavor is imperative to ensure the effectiveness and stability of the guidance system. The fine-tuning of these gains involves striking a delicate balance to achieve optimal performance in response to variations and disturbances while maintaining the system's robustness. A judicious selection of PID gains is paramount in minimizing overshooting, reducing settling time, and enhancing the overall responsiveness of the guidance subsystem. The tuning process, grounded in a rigorous analytical approach, contributes to the precision and reliability of the guidance algorithm, aligning the system with the desired specifications and operational requirements. Furthermore, this calibration endeavor is undertaken in accordance with established principles of control theory, emphasizing the significance of achieving an optimal trade-off between responsiveness and stability within the

PID guidance subsystem.

Here is a list with the obtained coefficient values:

- X-channel: $K_{p_x} = 2$, $K_{i_x} = 0.1$, $K_{d_x} = 1.5$. $N_x = 60$,
- Y-channel: $K_{p_y} = 1.5$, $K_{i_y} = 0.1$, $K_{d_y} = 1.5$, $N_y = 60$,
- Z-channel: $K_{p_z} = 0.9$, $K_{i_z} = 0.1$, $K_{d_z} = 1.2$, $N_z = 60$.

Chapter 5

Results

The present chapter delves into the outcomes derived from an extensive series of simulations conducted to rigorously assess the efficacy of the developed guidance algorithms. The algorithms under scrutiny encompass Proportional Navigation (PN), Augmented Proportional Navigation (APN), Optimal Guidance Law (OGL), and PID Guidance. The simulations encompass diverse scenarios, namely intercepting a static target, pursuing a moving target, and a meticulous comparative analysis of their respective performances. This empirical exploration endeavors to objectively examine and quantify the algorithms' capabilities across varied operational conditions. The ensuing sections will elucidate the nuanced dynamics revealed in the simulations, providing a comprehensive insight into the comparative effectiveness and adaptability of the distinct guidance algorithms in addressing different target interception scenarios.
Before specifying the different results in these several scenarios for both the target and the drone, meticulous attention was devoted to defining the initial conditions for the North-East-Down (NED) reference system. It is noteworthy that these NED reference system parameters remain consistent across all simulations for various guidance subsystems. Thus, the parameters of latitude, longitude, and altitude for the origin were meticulously specified as follows: $Lat_{ref} = 44.200626, Lon_{ref} = 12.063822, h_{ref} = 0$.

5.1 Static target

The initial phase of testing involved simulating a scenario where the target remains static and is positioned at a specific distance from the interceptor drone. The objective of this test, conducted across various guidance systems, is to assess the effectiveness of the drone in intercepting a target situated at a designated distance. Additionally, the test aims to observe the interception times, facilitating a comparative analysis among different guidance controllers. The initial conditions of the target were arbitrarily set, corresponding to:

- Initial velocity in NED frame, $V_{0t} = [0; 0; 0]\frac{m}{s}$,
- Initial position in NED frame, $Xe_{0t} = [10; 20; 30]m$,
- Initial attitude, $Euler_{0t} = [0; 0; 0]rad$,
- Initial angular velocity, $\omega_{0t} = [0; 0; 0]\frac{rad}{s}$,

The initial conditions of the drone were:

- Initial velocity in NED frame, $V_0 = [0; 0; 0]\frac{m}{s}$,
- Initial position in NED frame, $Xe_0 = [0; 0; 0]m$,
- Initial attitude, $Euler_0 = [0; 0; 0]rad$,
- Initial angular velocity, $\omega_0 = [0; 0; 0]\frac{rad}{s}$,

Proportional Navigation

It is imperative to underscore that, concerning the Proportional Navigation (PN) algorithm, once the code was established, simulations were conducted without undergoing any form of optimization. This deliberate decision aimed to maintain a practical adherence to theoretical concepts, refraining from extensive code modifications. This approach was not only grounded in the desire to uphold a practical fidelity to theory but was also influenced by the limited number of parameters available for arbitrary adjustments within the PN algorithm. A notable contrast can be drawn with the Optimal Guidance Law code, where a dedicated optimization process, as expounded in the "Optimization" paragraph, was undertaken, driven by the presence of discernible parameters, such as the b parameter.
A potential avenue for tuning can be identified in adjusting the Time-

to-Go (t_{go}) with a more pronounced offset, specifically, 1 second. However, it is crucial to discern that establishing the initial simulation time offset at 1 second entails a fundamentally different concept than the offset discussed in preceding chapters. Unlike the 0.1 second offset, which exclusively aimed to eliminate the singularity associated with $t_{go} = 0$, setting the initial t_{go} to 1 second effectively employs this value as a tuning parameter.
Simulations conducted with an initial offset of 1 second yielded noteworthy outcomes, notably the reduction of the characteristic overshooting inherent in the original algorithm. This overshooting, attributed to the substantial values of gains during the initial phases of interception, is visibly decreased, as depicted in Fig: 5.1.
The discernible absence of overshooting in these simulations underscores the efficacy of the tuning strategy associated with the 1-second offset. However, as emphasized earlier, a deliberate decision was made to adhere to a standard of fidelity to the assumptions posited in preceding chapters. Consequently, despite the demonstrable reduction in overshooting achieved with an initial t_{go} offset of 1 second, the chosen approach retained the 0.1-second offset. This decision, although leading to a conspicuous overshooting effect, aligns with the commitment to maintaining consistency with the initial premise, where the primary intent of the 0.1-second offset was singularly focused on nullifying the singularity at $t_{go} = 0$.
After this necessary discussion concerning the role of t_{go}, which extends to the Augmented Proportional Navigation and Optimal Guidance Law algorithms as they are also dependent on t_{go}, we can proceed to evaluate the performance of the Proportional Navigation code in simulating the previously outlined static scenario. As depicted in Fig: 5.2, the results are highly satisfactory, with the target reached in a relatively short time. However, as mentioned earlier and evident in the discussed plot, there is a noticeable overshooting attributed to the elevated values of the gains during the initial phases of interception.

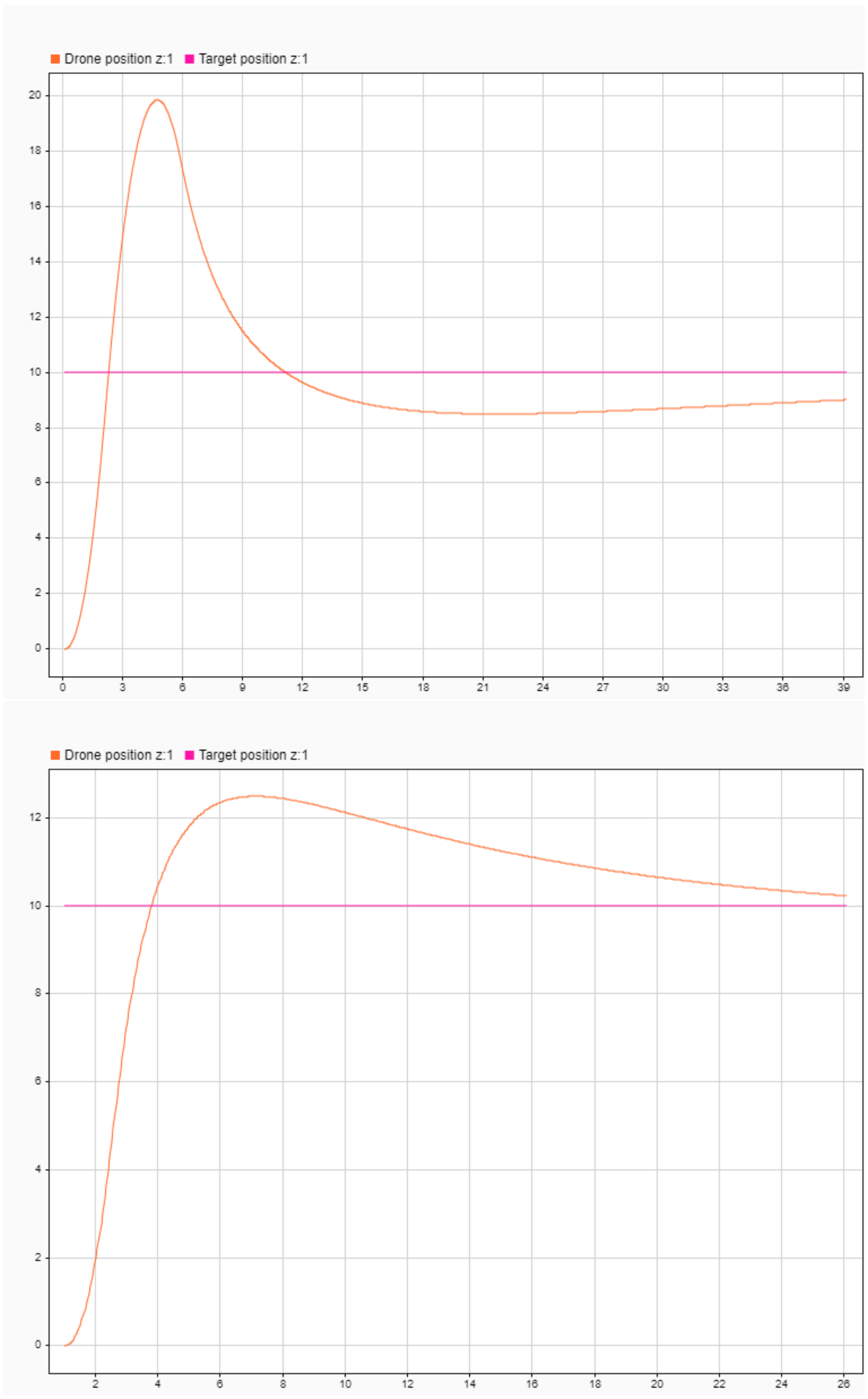
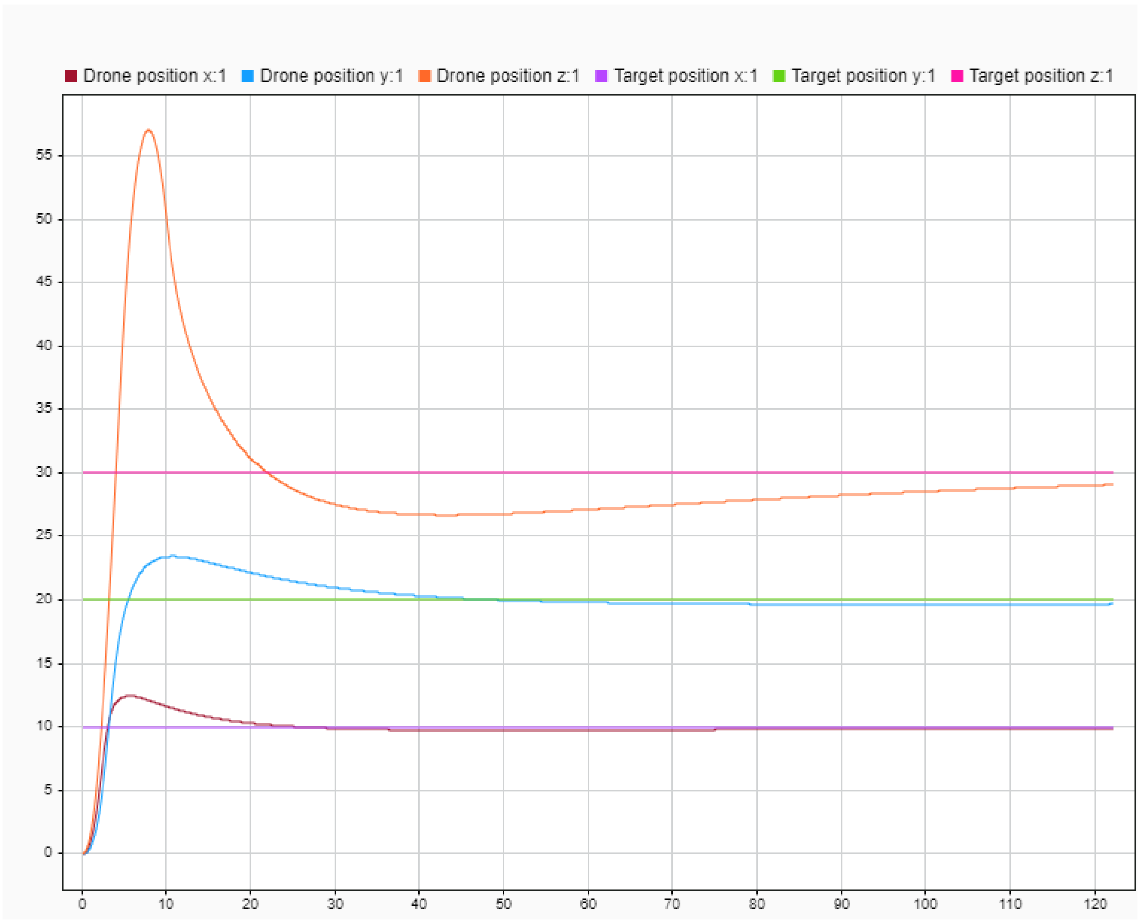


Figure 5.1: Plot illustrating the reduction of the overshooting due to a more pronounced offset of the t_{go}



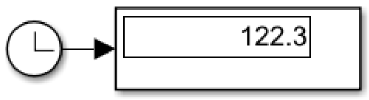
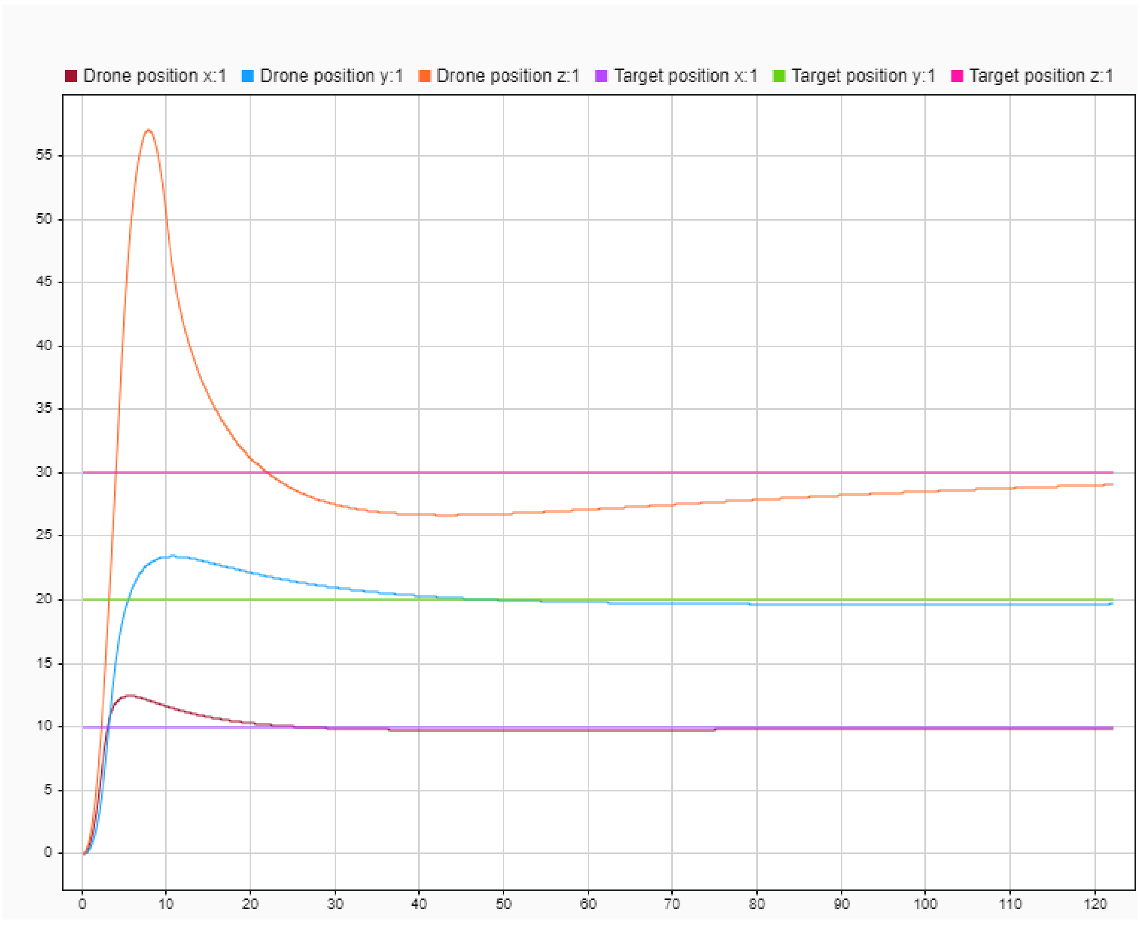


Figure 5.2: Plot illustrating the response of the drone equipped with a Proportional Navigation controller as it pursues a stationary target located at a designated distance Xe_{0t}, along with the associated time required to reach the target

Augmented Proportional Navigarion

The same considerations apply to the simulation of the model's behavior with the Augmented Proportional Navigation (APN) controller implemented. In this instance, the discussions regarding t_{go} in the paragraph concerning the simulation of a static target with Proportional Navigation remain applicable. As evident from the plot (Fig: 5.3), in this case as well, there is a notable overshooting, significantly impacting the performance in terms of target interception times. Contrary to intuition, the interception time for the Augmented Proportional Navigation controller is identical to its Proportional Navigation counterpart. The rationale behind this lies in the fact that the commanded acceleration in the case of APN is essentially a PN commanded acceleration with the addition of a contribution arising from the target's acceleration, which, in the scenario of a static target, is zero. Therefore, in the case of a static target, the APN controller and the PN controller are equivalent in terms of performance.



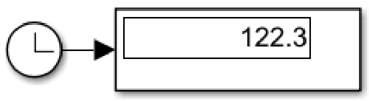
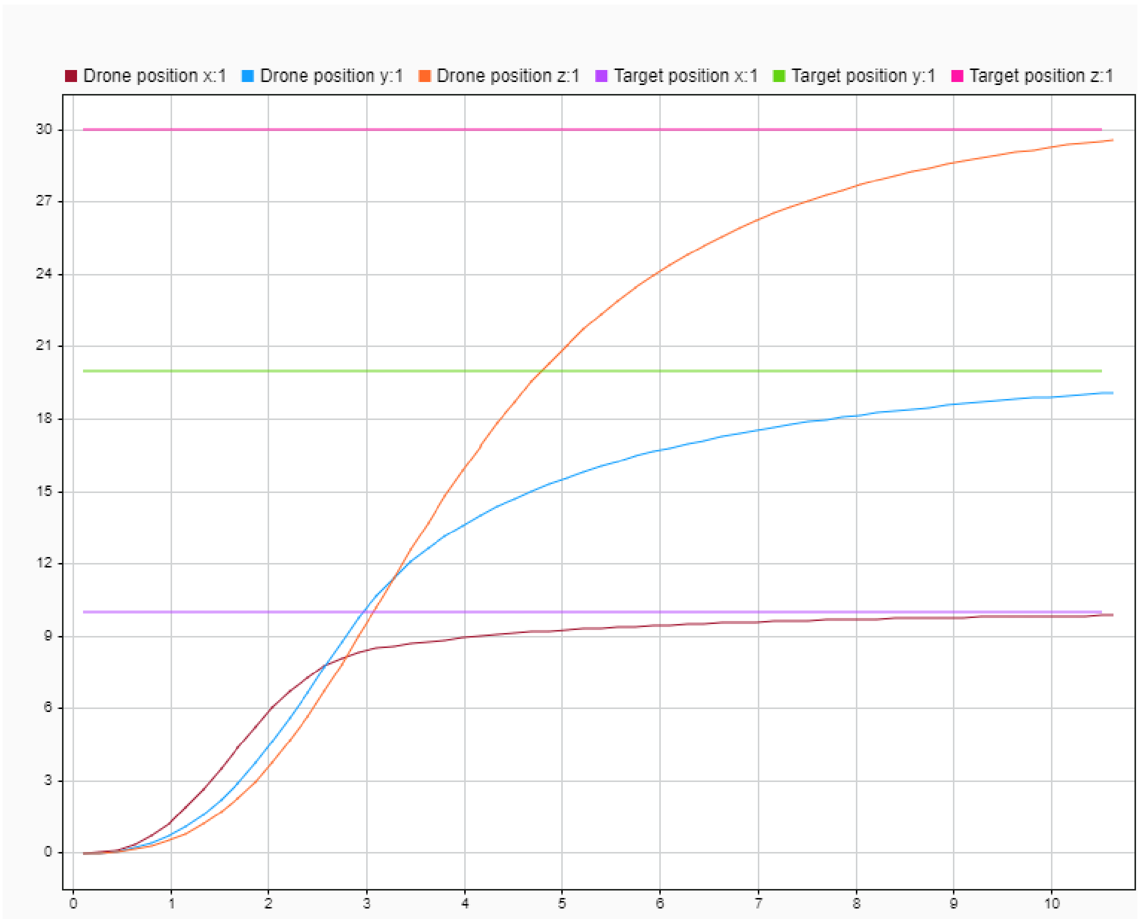


Figure 5.3: Plot illustrating the response of the drone equipped with an Augmented Proportional Navigation controller as it pursues a stationary target located at a designated distance Xe_{0t}, along with the associated time required to reach the target

Optimal Guidance Law

Regarding the performance analysis of the Optimal Guidance Law (OGL) controller (Fig: 5.4), the situation is markedly different. Indeed, while maintaining the considerations regarding t_{go} made earlier, the performance of this controller surpasses those observed previously with the first two. Through the optimization implemented, the target is reached in approximately one twelfth of the time required by the Augmented Proportional Navigation and Proportional Navigation controllers. Additionally, the drone's trajectory during interception is significantly improved, eliminating the typical overshooting that, understandably, compromises overall performance.



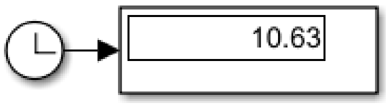
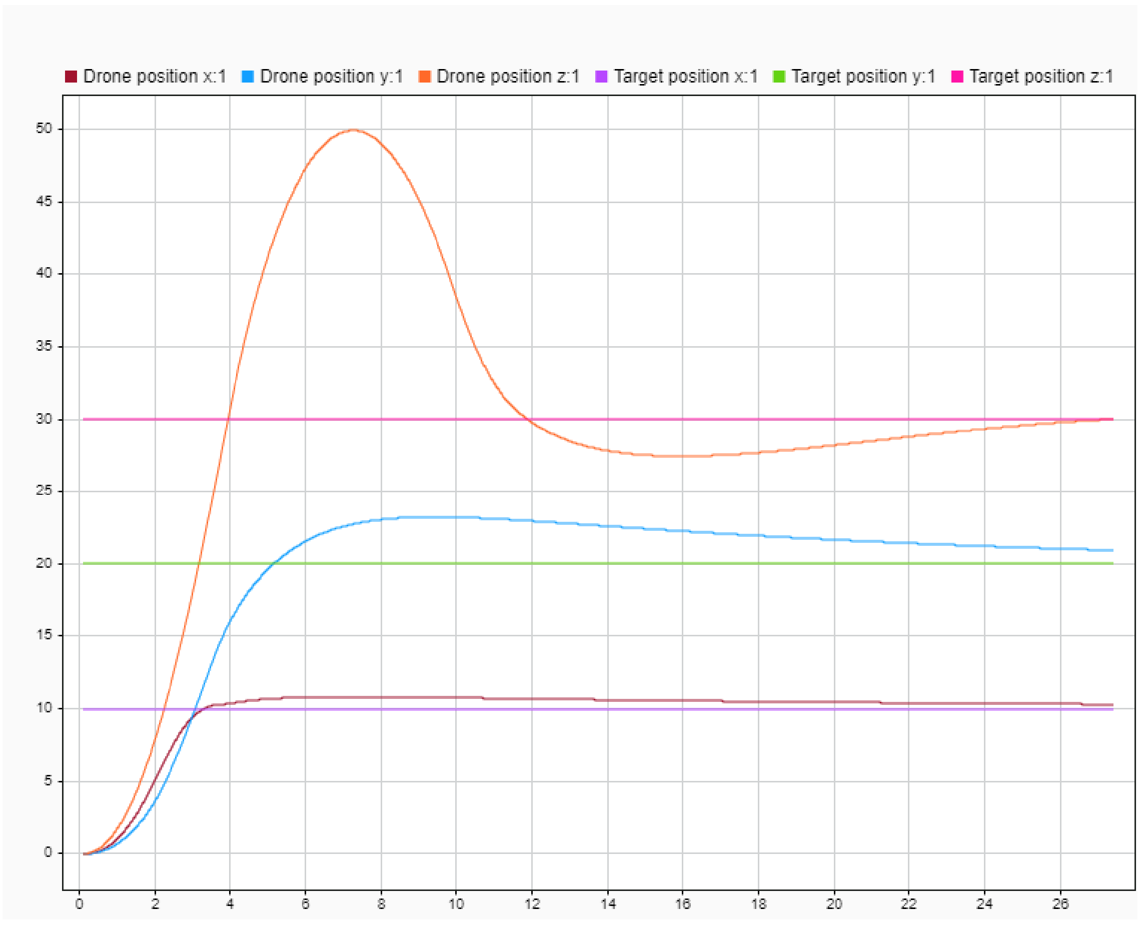


Figure 5.4: Plot illustrating the response of the drone equipped with an Optimal Guidance Law controller as it pursues a stationary target located at a designated distance Xe_{0t}, along with the associated time required to reach the target

PID Guidance

Finally, the assessment of the performance of the guidance controller based on a PID control strategy is conducted. As observed, the tuning procedure employed enables the controller to efficiently intercept the target in a relatively short time. Nevertheless, particularly concerning the z-component, an excessive overshooting is observed. Despite the optimization conducted, this overshooting persists, albeit significantly reduced through the optimization procedure.



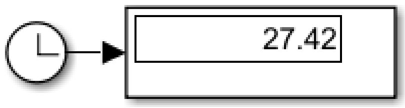


Figure 5.5: Plot illustrating the response of the drone equipped with an PID guidance controller as it pursues a stationary target located at a designated distance Xe_{0t}, along with the associated time required to reach the target

5.2 Moving target

Subsequently, the performance of the various guidance controllers was tested in the case of a moving target. The results, as depicted in the subsequent graphs, indicate the successful interception of the target, with the simulation halting as expected when the relative distance between the target and the drone reaches 1 meter. This distance is deemed sufficiently minimal to ensure a secure obliteration of the target when the interceptor drone activates the explosive charge. A MATLAB script has been devised to generate random initial conditions for the target, encompassing both position and velocity. This approach aims to facilitate an examination of the drone's behavior within a wholly randomized scenario. The initial conditions specified for the target are as follows:

- Initial velocity in NED frame, $V_{0t} = [12.8344; 5.8090; 0.1785]\frac{m}{s}$,
- Initial position in NED frame, $Xe_{0t} = [9.4417; 12.1738; 2.5397]m$,
- Initial attitude, $Euler_{0t} = [0; 0; 0]rad$,
- Initial angular velocity, $\omega_{0t} = [0; 0; 0]\frac{rad}{s}$,

The initial conditions of the drone were:

- Initial velocity in NED frame, $V_0 = [0; 0; 0]\frac{m}{s}$,
- Initial position in NED frame, $Xe_0 = [0; 0; 0]m$,
- Initial attitude, $Euler_0 = [0; 0; 0]rad$,
- Initial angular velocity, $\omega_0 = [0; 0; 0]\frac{rad}{s}$,

Proportional Navigation

As evident from the graph (Fig: 5.7), the drone efficiently intercepts the target. It is noteworthy to observe the consistent overshooting along the z-component of the trajectory in Fig: 5.6. It is also noteworthy that the generated overshooting causes the drone to assume an altitude very close to the ground or, in any case, the launch altitude, presumably where the simulation commences. Nevertheless, no singularities have been identified due to the assumption of negative altitude values by the drone, thus rendering the simulation more faithful to reality.Nevertheless, despite this phenomenon, the target is successfully

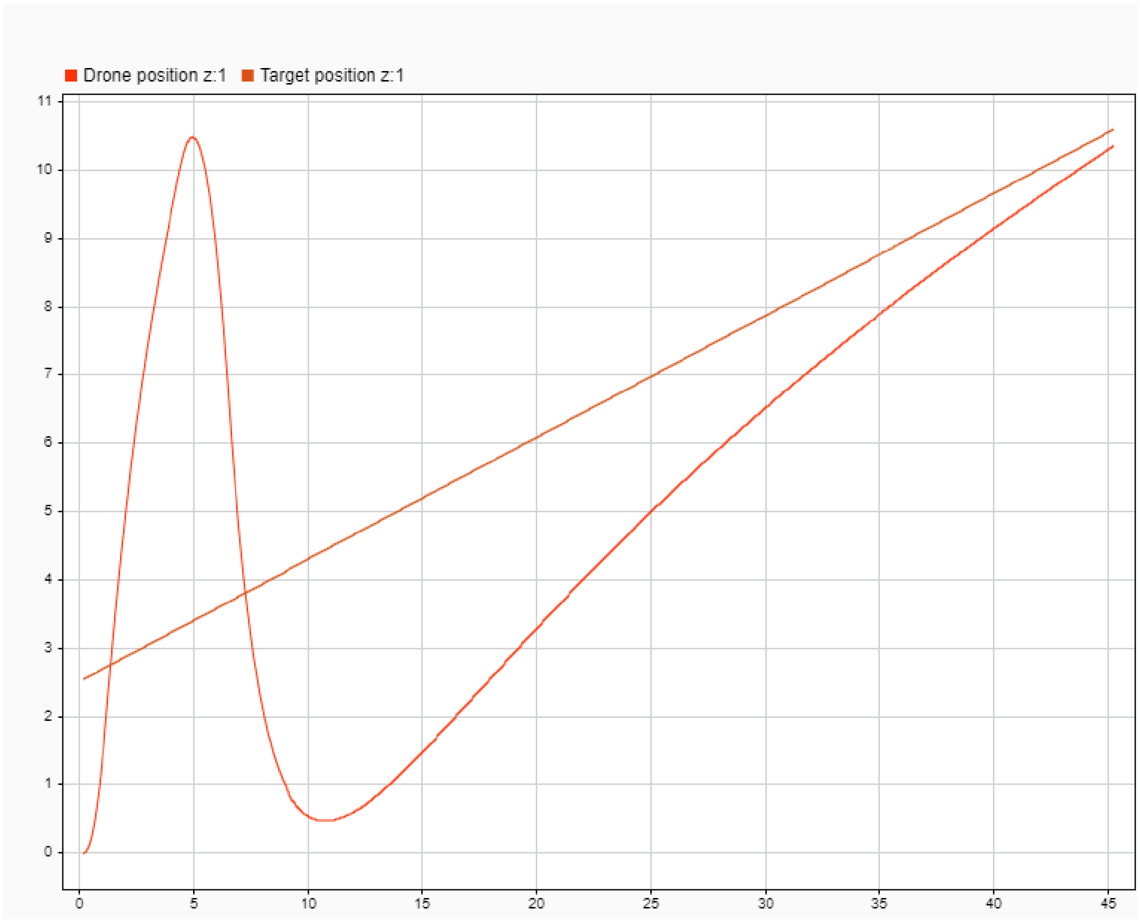
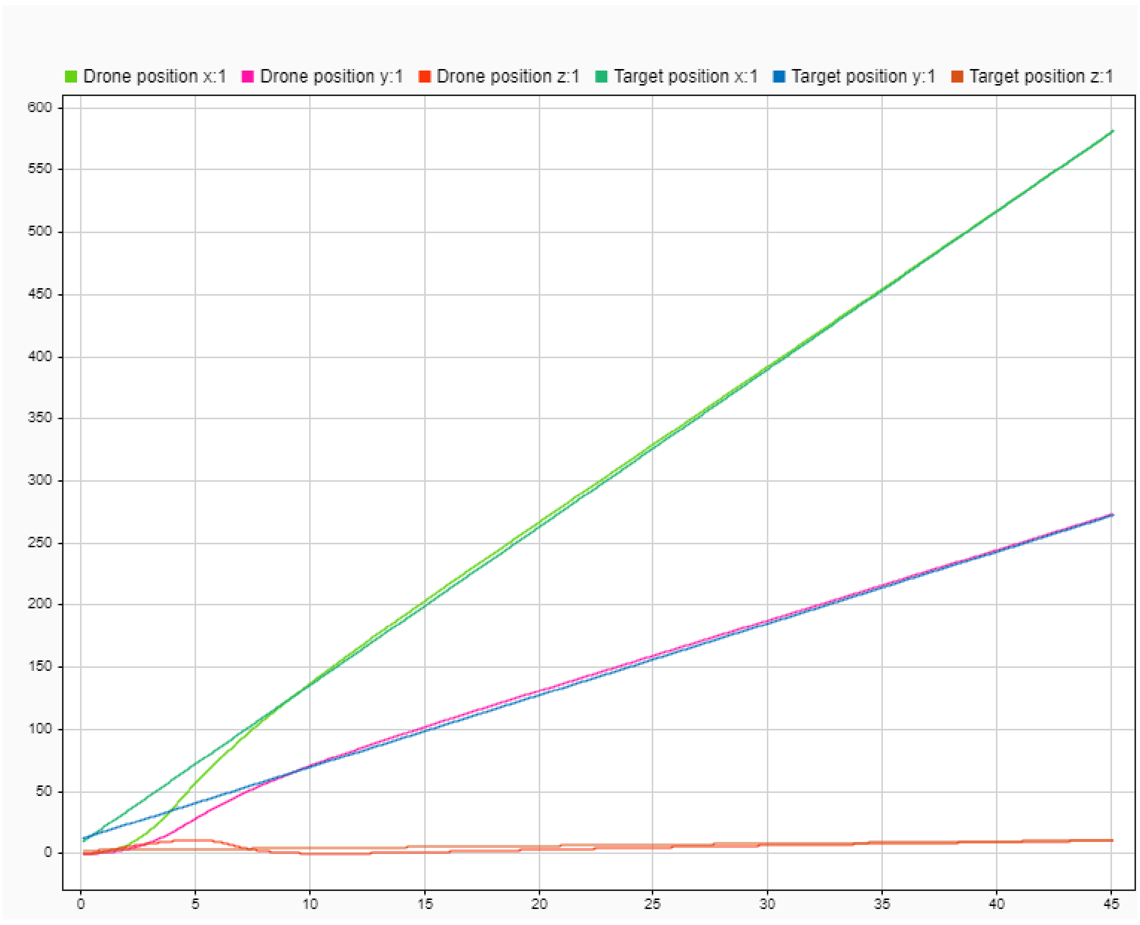


Figure 5.6: Plot illustrating the response of the drone, equipped with a Proportional Navigation controller, in pursuit of a moving target along the z-component, depicting the generated overshooting

obliterated.



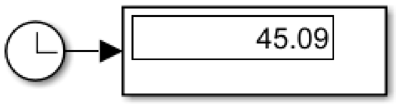
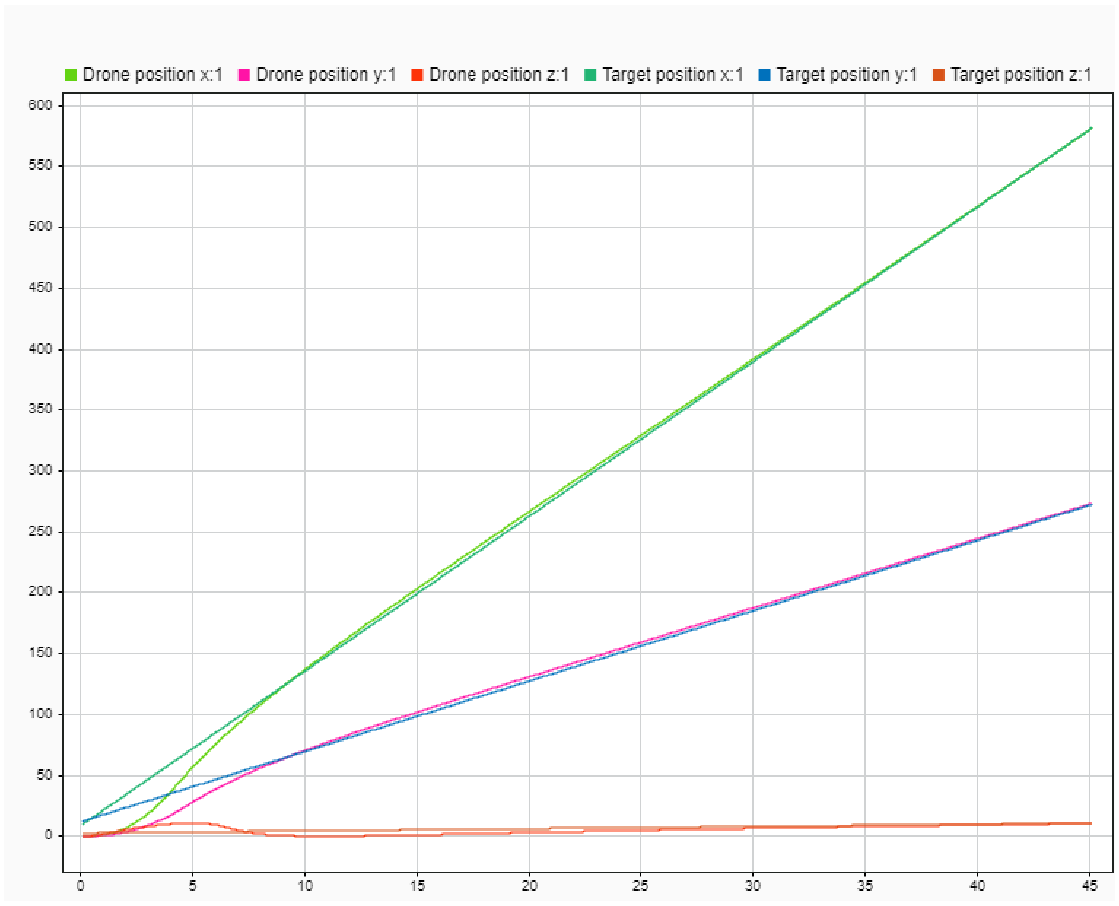


Figure 5.7: Plot illustrating the response of the drone equipped with a Proportional Navigation controller as it pursues a moving target, along with the associated time required to reach the target

Augmented Proportional Navigation

In the case of a drone equipped with an Augmented Proportional Navigation (APN) controller, an observation revealed behavior precisely identical to that of the Proportional Navigation case. This aligns with expectations, as the APN controller essentially functions as a PN controller with the additional element of target acceleration. Consequently, similar to the static case, no discernible difference in efficiency was noted between the two, as the target in this scenario was modeled to maintain a constant velocity directly set in the initial conditions.



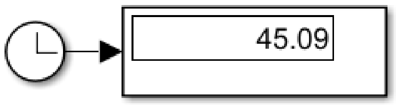


Figure 5.8: Plot illustrating the response of the drone equipped with a Augmented Proportional Navigation controller as it pursues a moving target, along with the associated time required to reach the target

Optimal Guidance Law

As anticipated, this algorithm demonstrated superior performance, attributed to the optimization process undertaken. The interception time is significantly lower compared to that observed in all other algorithms, achieving target interception in approximately half the time and without experiencing overshooting.

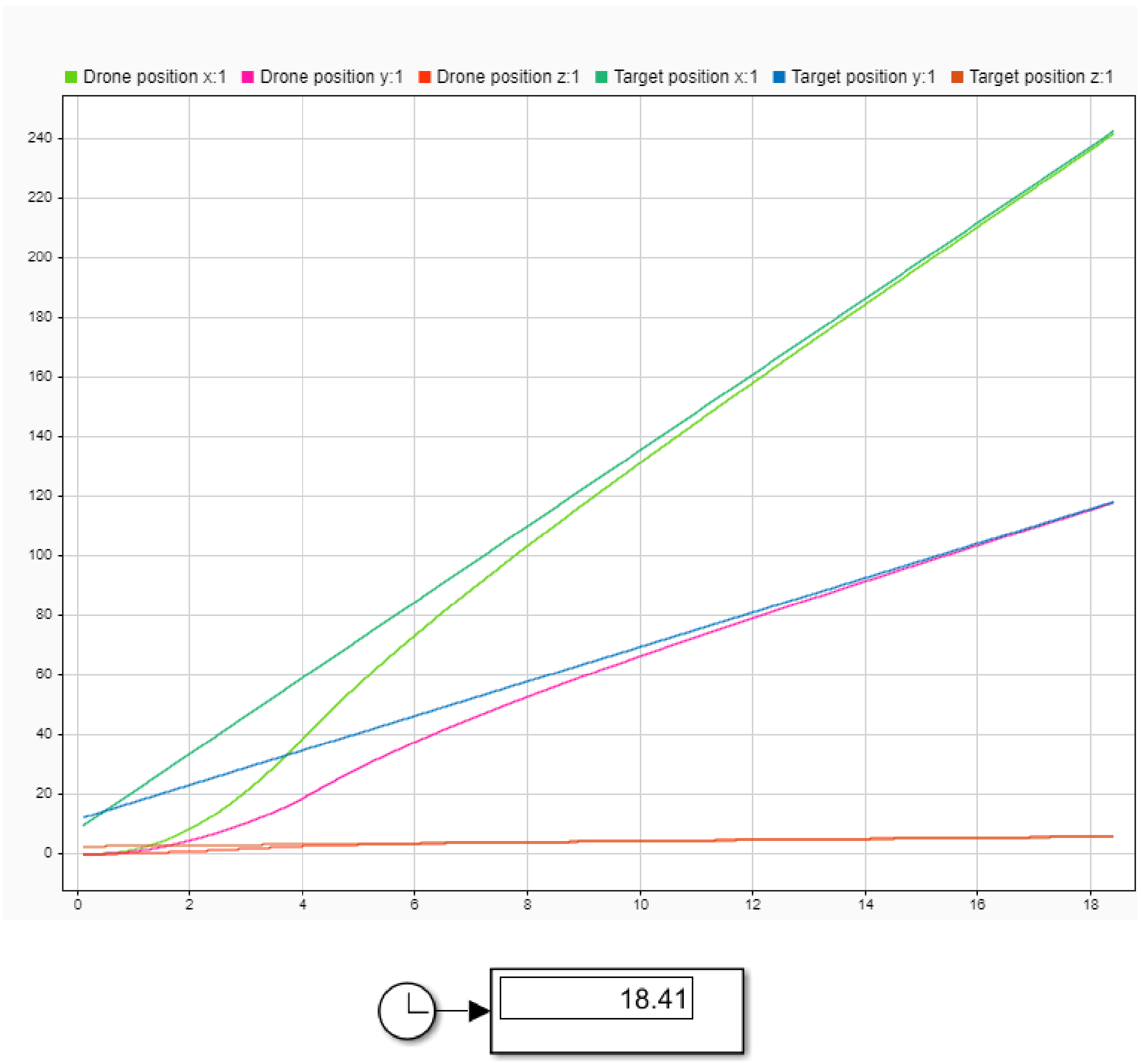


Figure 5.9: Plot illustrating the response of the drone equipped with a Optimal Guidance Law controller as it pursues a moving target, along with the associated time required to reach the target

PID Guidance

Regarding the final guidance algorithm (Fig: 5.11), the PID guidance controller exhibited a significant improvement compared to the proportional controllers (APN and PN), albeit to a lesser extent than the OGL. This improvement stems from effective tuning, optimizing the PID-based controller to extract the maximum performance inherent in the PID technology. In this instance, substantial overshooting was also observed (Fig: 5.10), albeit considerably reduced in comparison to APN and PN controllers.

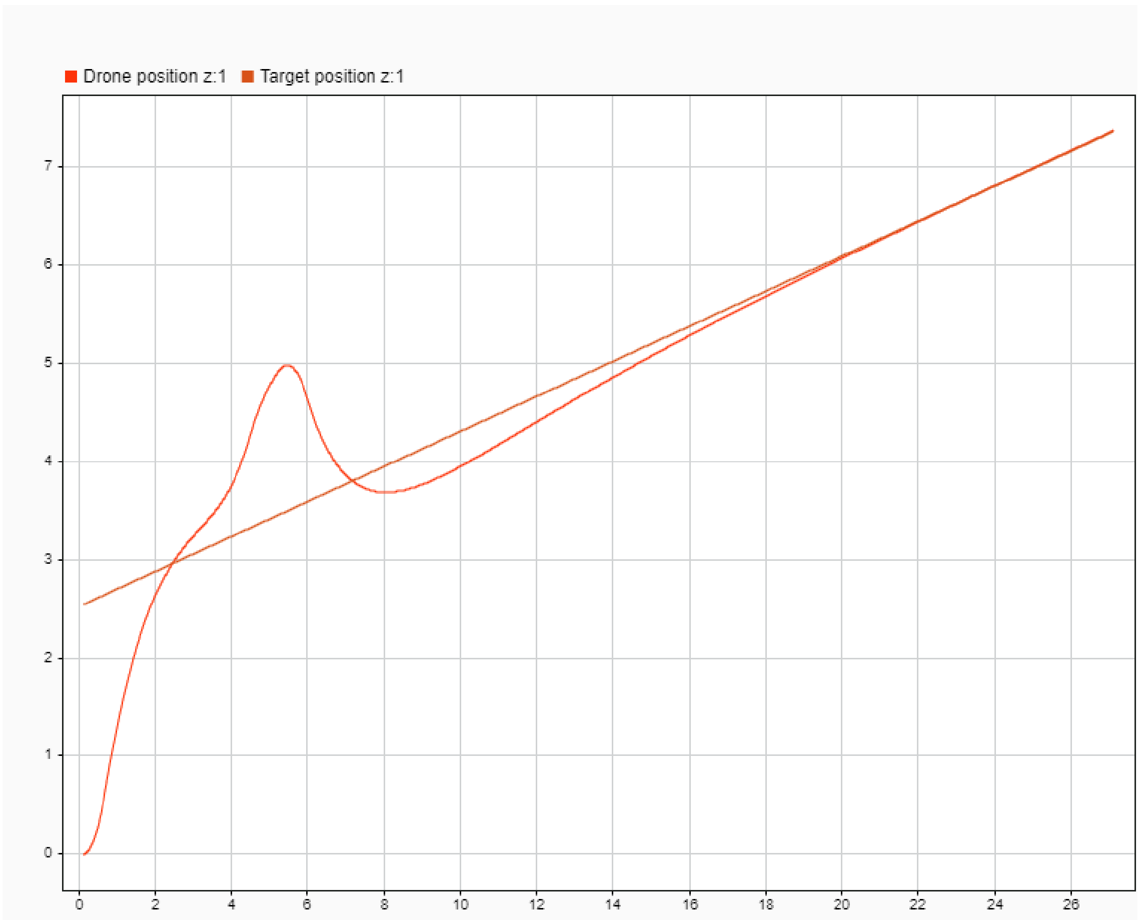


Figure 5.10: Plot illustrating the response of the drone, equipped with a PID controller, in pursuit of a moving target along the z-component, depicting the generated overshooting

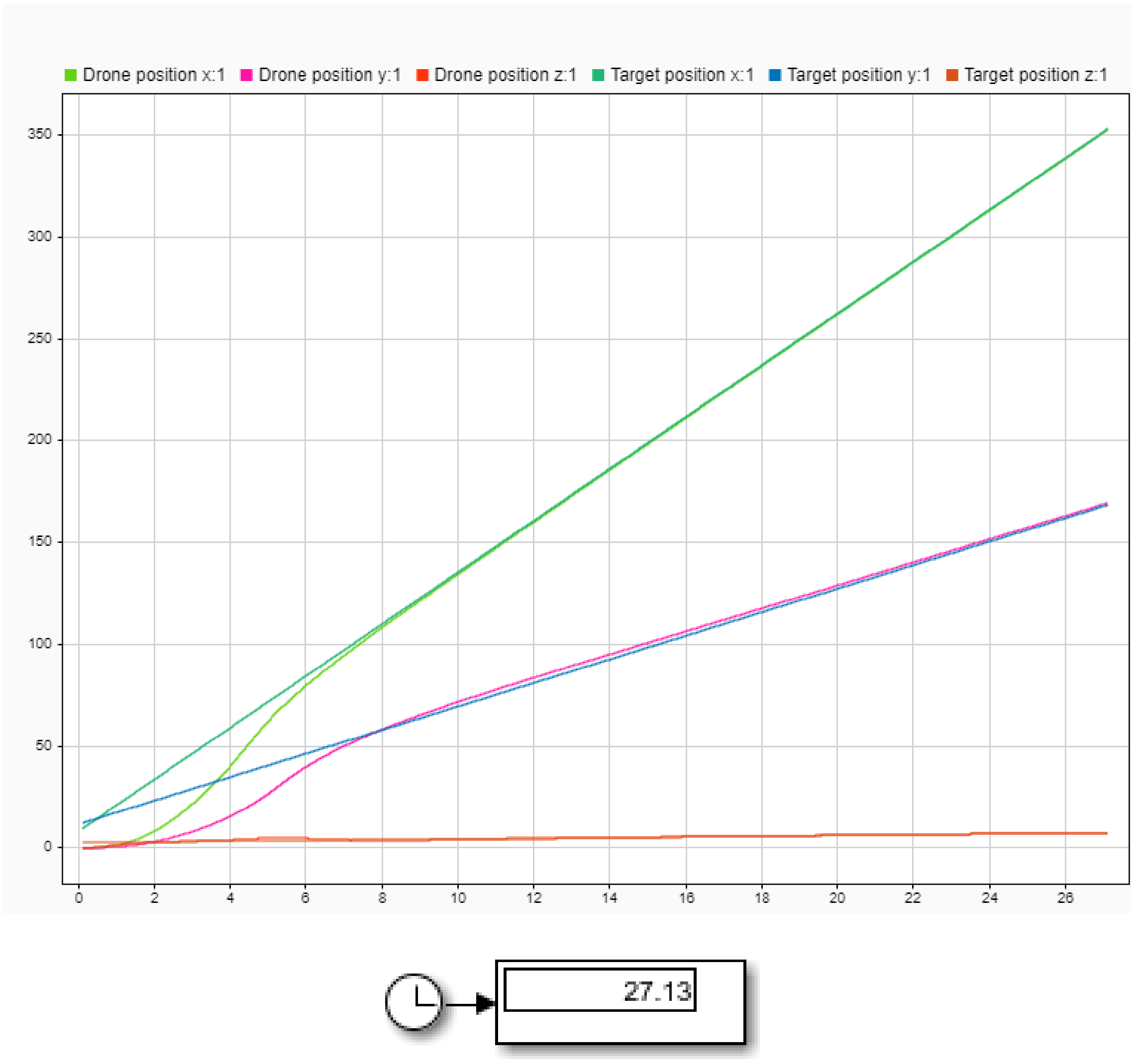


Figure 5.11: Plot illustrating the response of the drone equipped with a PID guidance controller as it pursues a moving target, along with the associated time required to reach the target

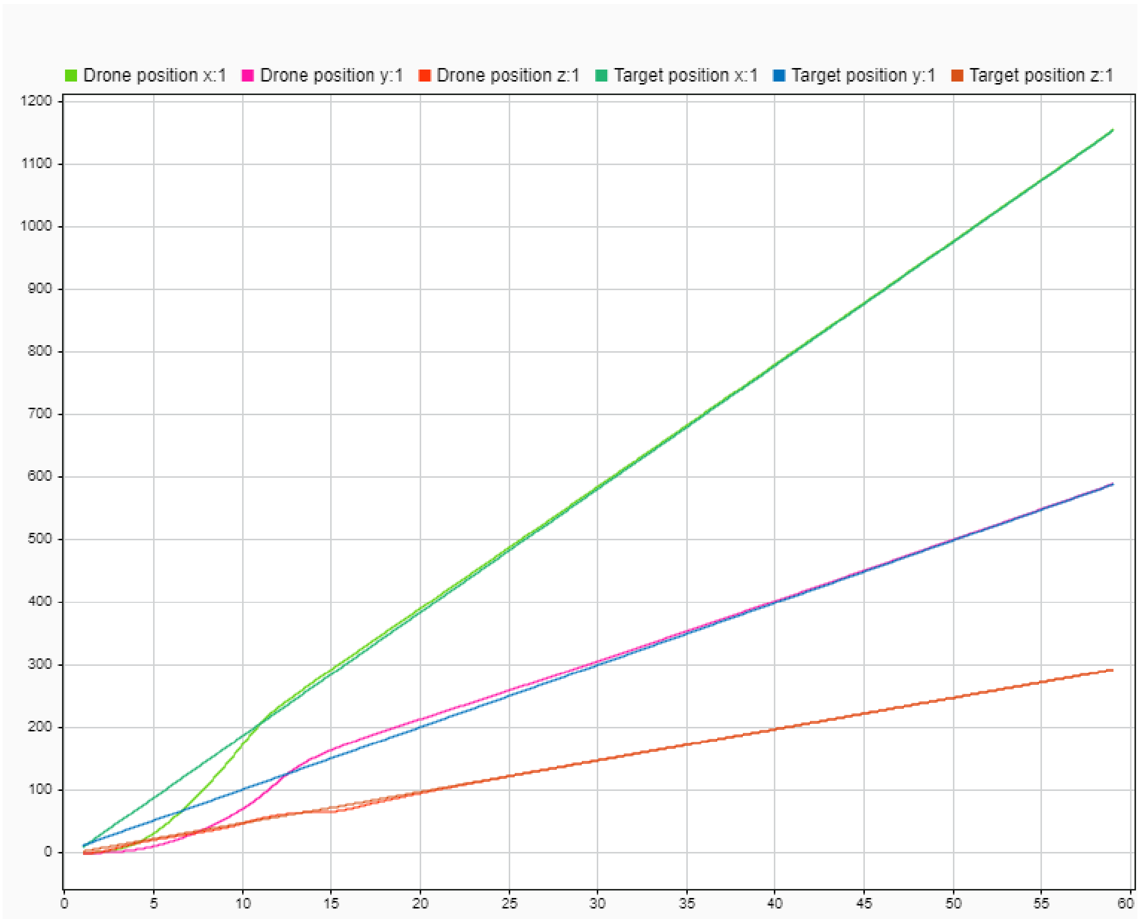
5.3 Limitations

The studied algorithms have yielded highly compelling results, with simulations providing a comprehensive insight into their behaviors and performance. Furthermore, an examination of the outcomes of optimizations applied to the developed guidance controllers has been insightful. However, upon conducting more in-depth tests, certain limitations within some algorithms became apparent. Specifically, the PN and APN algorithms demonstrated less accuracy in scenarios involving targets with a high magnitude of velocity.
Through extensive testing, a discernible threshold for ensuring complete and efficient target interception was identified, approximately at speeds of 55/60 km/h (15.2778/16.6667 m/s). Following numerous simulations, hypotheses, and trial demonstrations, the cause of this apparent inefficiency—contrary to theoretical expectations—was determined to stem from the lack of optimization in the PN and APN algorithms. In contrast, the PID controller maintained high efficacy standards due to optimization (Fig: 5.12).
Regarding the OGL controller, it was observed that to uphold a high standard of efficacy in scenarios involving targets with notably high velocities, a slight adjustment to the interception threshold was necessary, setting it at 1.5. In the illustrated scenario (refer to Fig: 5.13), the target exhibited velocities of $V_t = [10; 20; 10]$, corresponding to a magnitude of approximately 90 km/h, which is relatively high. Despite the elevated target speed, the OGL controller efficiently achieved interception within a considerably brief time frame. This adjustment in the interception threshold showcases the adaptability of the OGL algorithm to varied scenarios, thereby contributing to its robust performance across a spectrum of target velocities.
To substantiate the claim expressed previously for the PN and APN controllers, a reevaluation of the assumptions made about Time to Go was conducted. Previous chapters highlighted that introducing a more pronounced offset to t_{go} could enhance interception effectiveness and efficiency, effectively utilizing t_{go} as a tuning parameter. However,

to adhere faithfully to missile interception theory and the initially assumed conditions, the initial t_{go} was kept at 0.1.

To demonstrate this, simulations were performed with an initial t_{go} set to 1 second. As evident in the Fig: 5.14, this singularity was eliminated, and a notable increase in the efficiency of the APN and PN algorithms became apparent (only one graph is presented since, as previously observed, the APN and PN algorithms yielded identical results).

In summary, the developed algorithms have showcased excellent interception capabilities and notable efficiency. However, these attributes are not assured for high target velocities, unless potential modifications to the original algorithm and, possibly, the underlying assumptions—beyond the scope of this thesis—are implemented. Therefore, to maintain consistency with these findings, it is advised against employing the aforementioned algorithms for scenarios involving very high target velocities, as effective and efficient interception cannot be guaranteed.



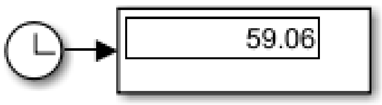
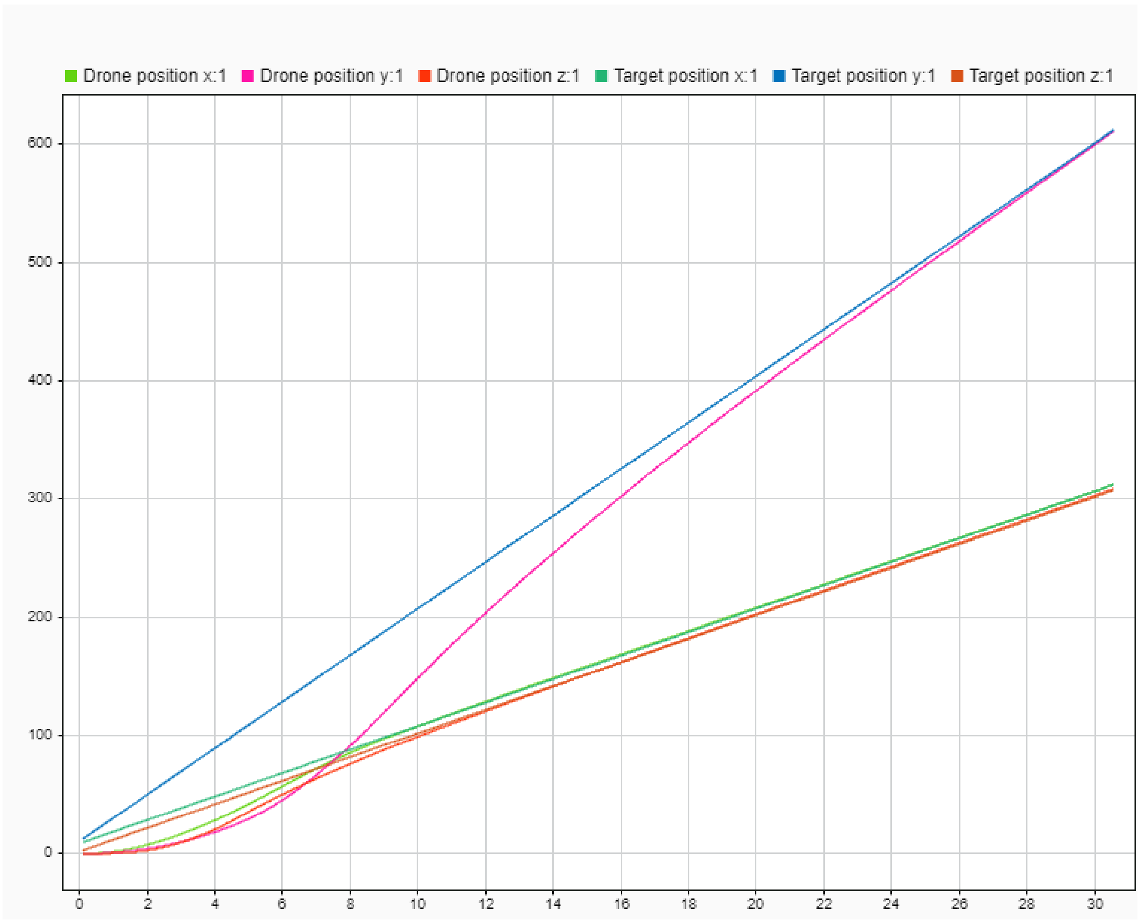


Figure 5.12: Plot illustrating the behavior of the interceptor drone equipped with the PID controller with a high-speed target ($||V_t|| = 83\frac{km}{h}$), along with the associated time required to reach the target



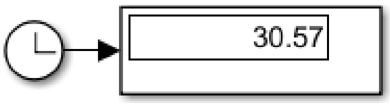
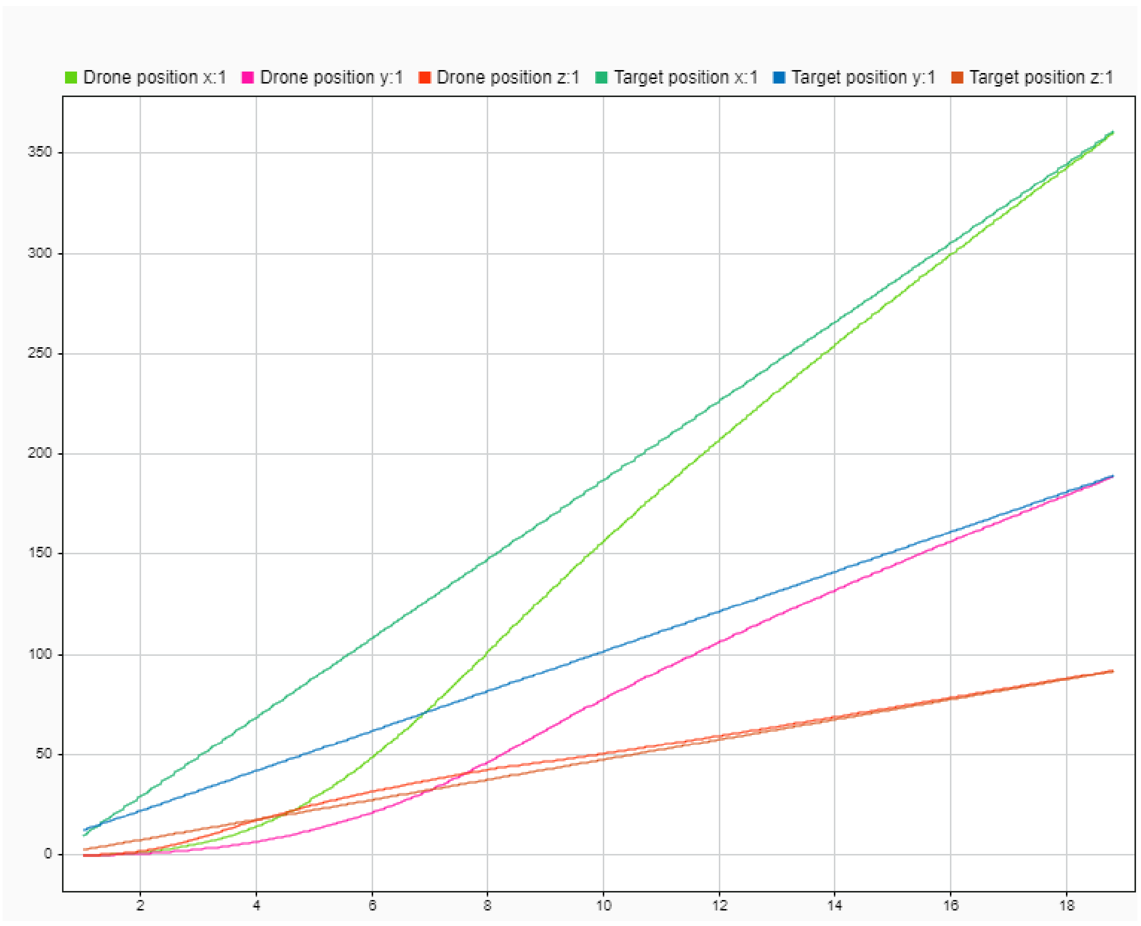


Figure 5.13: Plot illustrating the behavior of the interceptor drone equipped with the OGL controller with a high-speed target ($||V_t|| = 88\frac{km}{h}$), along with the associated time required to reach the target



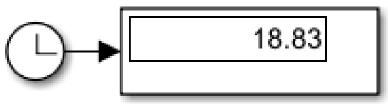


Figure 5.14: Plot illustrating the behavior of the interceptor drone equipped with the PN controller with a high-speed target ($||V_t|| = 83\frac{km}{h}$), along with the associated time required to reach the target

5.4 3D Simulations

Upon completion of the comparative analysis of diverse guidance controllers, a series of simulations was conducted to evaluate the drone's capacity for intercepting another drone under randomly generated flight conditions. The chosen scenario, within which the initial conditions for the target were generated for testing, represents a standard situation where the interception system identifies a mobile object (a hostile drone) operating in a pseudo-cruising flight state. This state is typified by a negligible vertical velocity component relative to its horizontal progression in the S_{xy} plane.
The rationale for selecting the target's cruising flight condition is rooted in the commonly observed practice of deploying a standard interception system to engage a target during its cruise phase, typically launched at a safe distance from the hostile group. The deliberate decision to maintain a non-zero vertical velocity, typically absent during normal cruising conditions, aims to scrutinize the interceptor drone's behavior along the vertical axis when exposed to a non-zero vertical velocity.
To achieve this objective, a specialized script was developed to randomly generate initial conditions for the target to be intercepted, while the interceptor maintains the aforementioned observed initial conditions:

- Initial velocity in NED frame, $V_0 = [0; 0; 0]\frac{m}{s}$,
- Initial position in NED frame, $Xe_0 = [0; 0; 0]m$,
- Initial attitude, $Euler_0 = [0; 0; 0]rad$,
- Initial angular velocity, $\omega_0 = [0; 0; 0]\frac{rad}{s}$,

Proportional Navigation

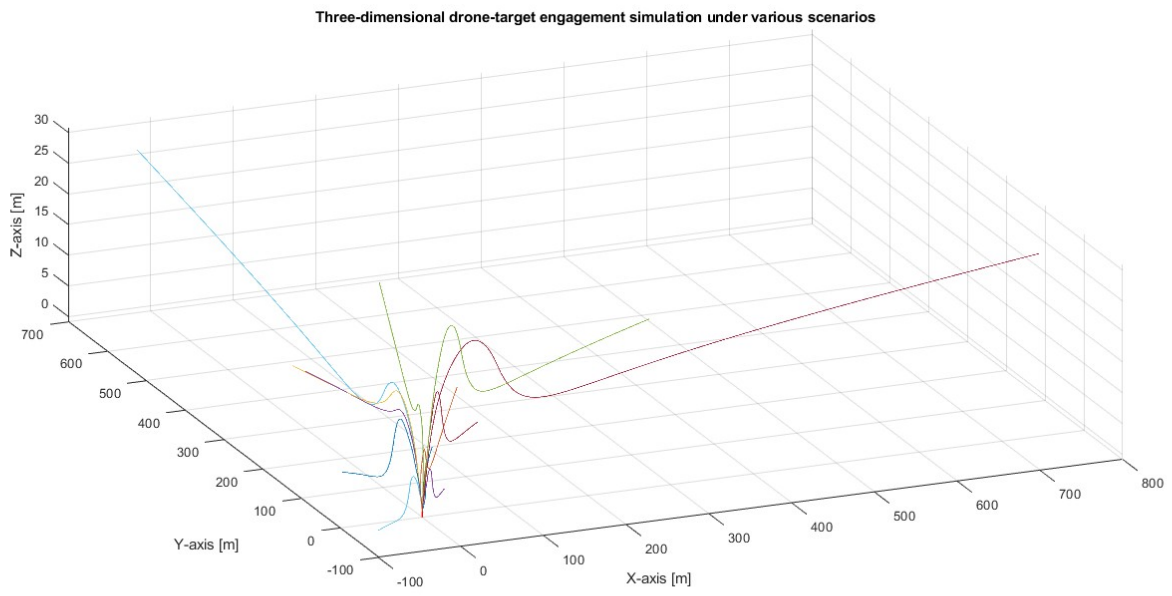


Figure 5.15: Plot illustrating the behavior of the interceptor drone equipped with the PN controller in various randomly generated interception scenarios.

Augmented Proportional Navigation

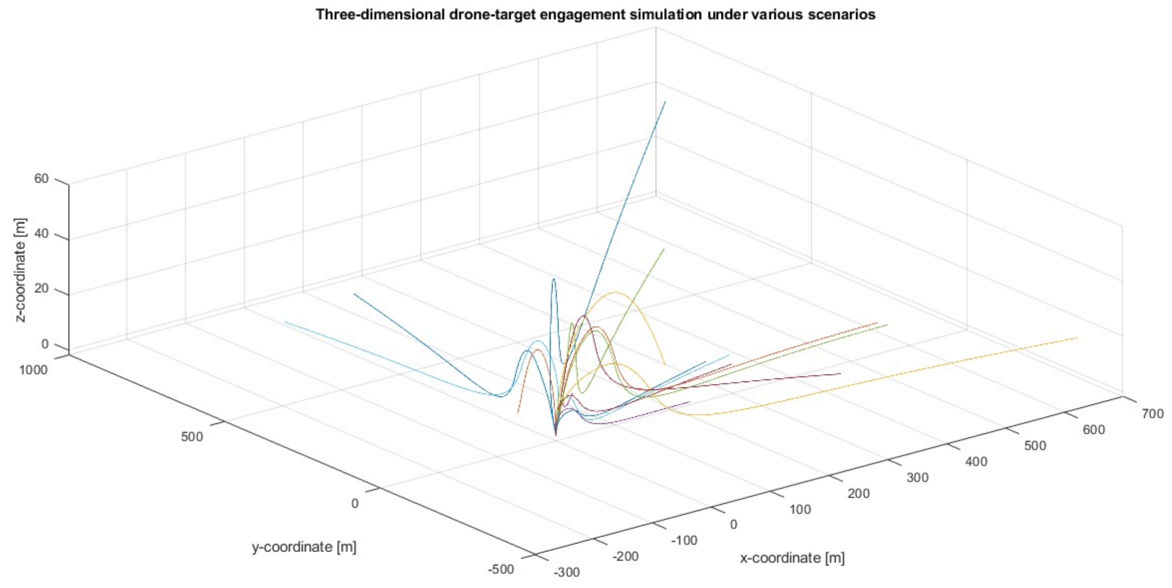


Figure 5.16: Plot illustrating the behavior of the interceptor drone equipped with the APN controller in various randomly generated interception scenarios.

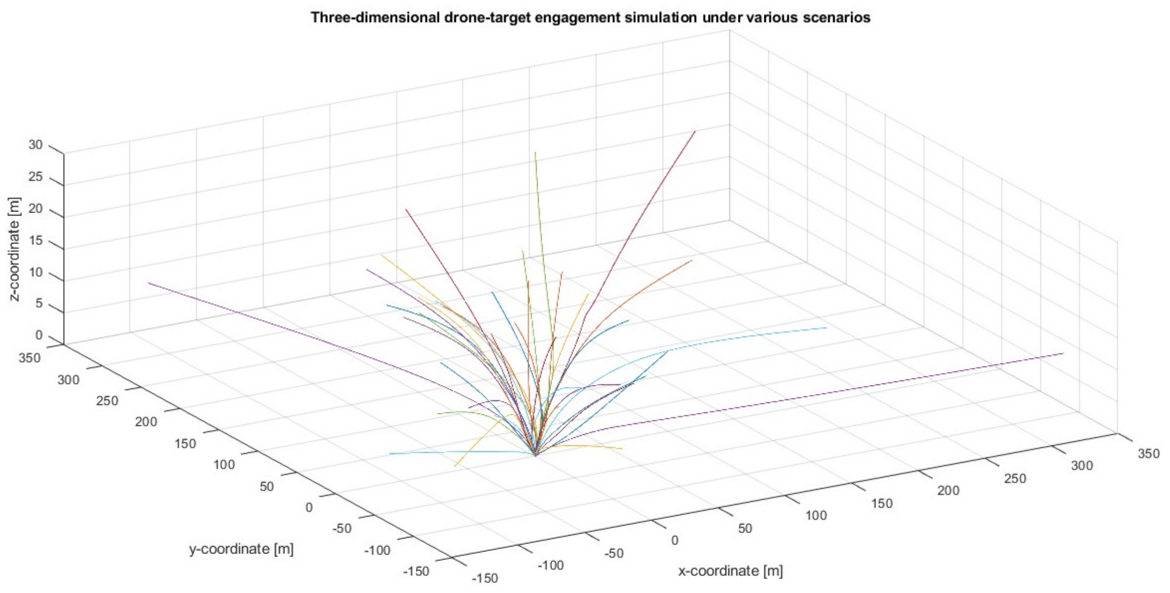


Figure 5.17: Plot illustrating the behavior of the interceptor drone equipped with the OGL controller in various randomly generated interception scenarios.

PID Guidance

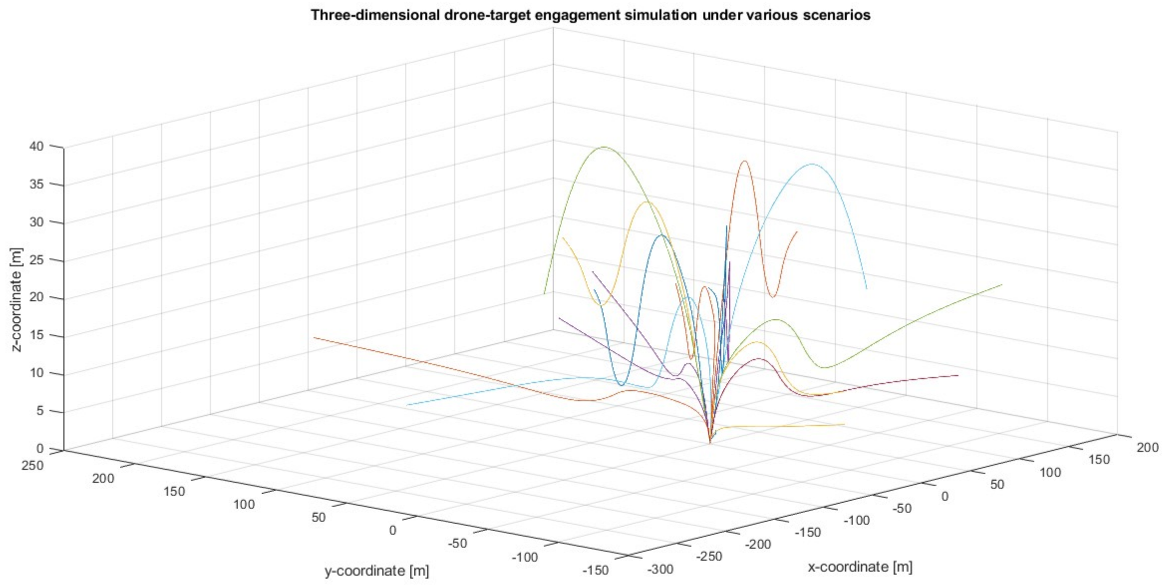


Figure 5.18: Plot illustrating the behavior of the interceptor drone equipped with the PID controller in various randomly generated interception scenarios.

Conclusions

In conclusion, this thesis aimed to enhance airspace safety by meticulously developing the dynamic model of the drone in all its facets, encompassing aspects such as rotational and translational dynamics. The focus extended to the development of a flight controller tasked with precisely instructing the dynamic system based on measurements obtained from the drone's sensors. The controller then compares these measurements with the commands received from various guidance systems. Notably, the development and evaluation of four distinct guidance subsystems—Proportional Navigation (PN), Augmented Proportional Navigation (APN), Optimal Guidance Law (OGL), and PID Guidance—have been pivotal in unraveling the intricacies of navigating dynamic target scenarios.
A notable aspect of this research lies in the deliberate reconsideration and adjustment of assumptions within the algorithmic frameworks. By deviating from conventional assumptions, the resulting guidance subsystems exhibit enhanced adaptability and effectiveness. The recalibration of assumptions serves as a testament to the meticulous refinement and evolution of the guidance algorithms presented herein. The empirical results derived from extensive simulations underscore the efficacy of each guidance subsystem, affirming their utility in diverse operational contexts. The comparative analysis between the four subsystems reveals nuanced strengths and considerations for each. Significantly, the Optimal Guidance Law emerges as the standout performer, demonstrating remarkable effectiveness in achieving interception objectives.
The optimization process and fine-tuning of parameters within the

Optimal Guidance Law have yielded astonishing results, surpassing the performance of the other guidance codes. This underscores the importance of a nuanced and optimized approach in algorithmic design for achieving optimal outcomes.
In summary, this thesis contributes to the field of guidance algorithms for drone interception by presenting a meticulous exploration of diverse subsystems and their comparative evaluations. The adaptation of assumptions, coupled with a rigorous examination of results, positions these algorithms as valuable assets in navigating complex and dynamic interception scenarios. The findings not only affirm the effectiveness of the developed guidance subsystems but also underscore the critical role of optimization in achieving optimal outcomes in drone interception applications.

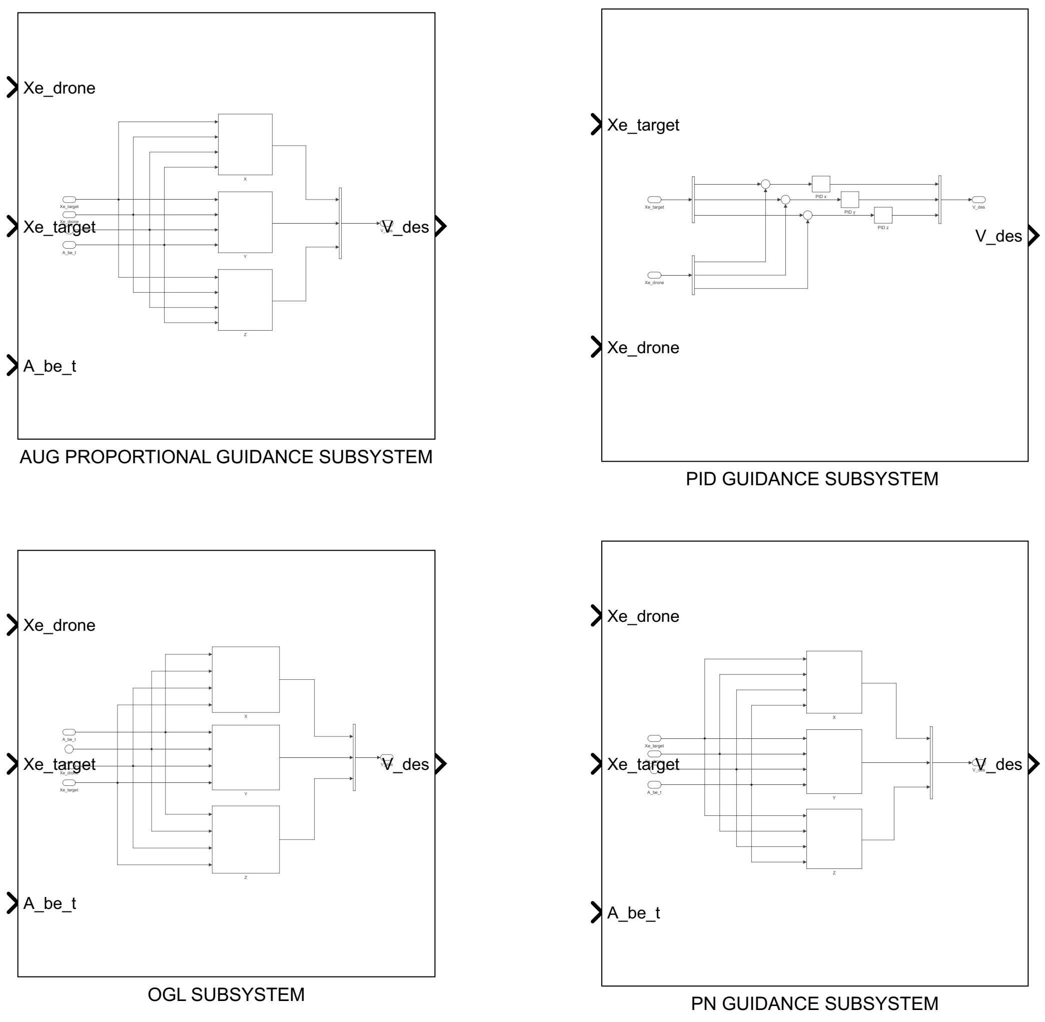


Figure 5.19: All the constructed guidance subsystems.

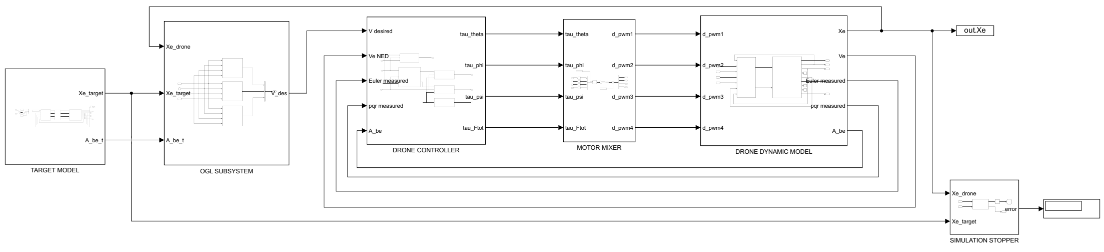


Figure 5.20: The constructed comprehensive model, incorporating a dynamic model, flight controller, guidance system, stopping condition subsystem, and equipped target subsystem.

Bibliography

[1] Theodor Benecke. *History of German guided missiles development.* Verlag E. Appelhans & Company, 1957.

[2] RR Bennett and WE Mathews. “Analytical determination of miss distances for linear homing navigation systems”. In: *Hughes Aircraft Company Technical Memorandum* 260 (1952).

[3] Roman Czyba and Grzegorz Szafrański. “Different approaches of PID control UAV type quadrotor”. In: (2011).

[4] Michael Dancer, S Balakrishnan, and Ernest Ohlmeyer. “Discussion and analysis of missile igc design”. In: *AIAA guidance, navigation and control conference and exhibit.* 2008, p. 7002.

[5] Mike W Fossier. “The development of radar homing missiles”. In: *Journal of Guidance, Control, and Dynamics* 7.6 (1984), pp. 641–651.

[6] Byungjun Kim. “Proportional-integral-derivative controller in proportional navigation guidance”. PhD thesis. 2016.

[7] Inanc Moran and Turgay Altilar. “Three plane approach for 3D true proportional navigation”. In: *AIAA guidance, navigation, and control conference and exhibit.* 2005, p. 6457.

[8] F William Nesline and Paul Zarchan. “A new look at classical vs modern homing missile guidance”. In: *Journal of Guidance and Control* 4.1 (1981), pp. 78–85.

[9] Neil F Palumbo, Ross A Blauwkamp, and Justin M Lloyd. “Modern homing missile guidance theory and techniques”. In: *Johns Hopkins APL technical digest* 29.1 (2010), pp. 42–59.

[10] Julian Rothe, Michael Strohmeier, and Sergio Montenegro. “A concept for catching drones with a net carried by cooperative UAVs”. In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2019, pp. 126–132.

[11] RWH Sargent and GR Sullivan. “The development of an efficient optimal control package”. In: *Optimization Techniques: Proceedings of the 8th IFIP Conference on Optimization Techniques Würzburg, September 5–9, 1977*. Springer. 1978, pp. 158–168.

[12] Ilia Yefimovich. “iron dome”. In: Getty Images.

[13] Luke Chia-Liu Yuan. “Homing and navigational courses of automatic target-seeking devices”. In: *Journal of Applied Physics* 19.12 (1948), pp. 1122–1128.

[14] Paul Zarchan. *Tactical and strategic missile guidance*. American Institute of Aeronautics and Astronautics, Inc., 2012.

Appendix A

Target modeling

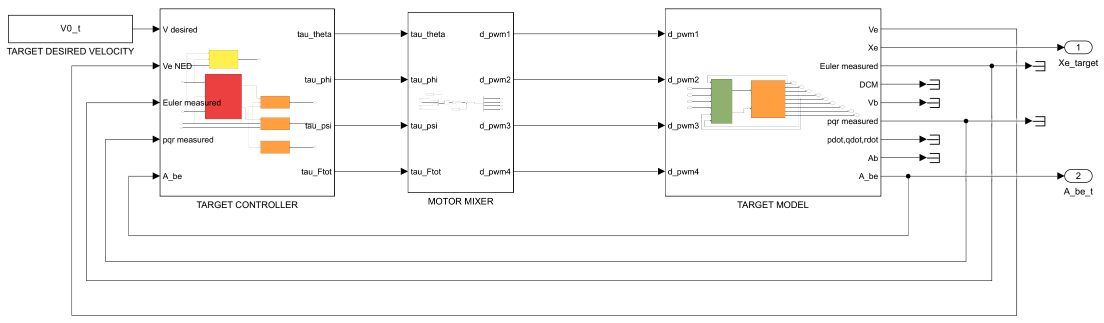


Figure A.1: Simulink architecture of the developed target model

In this appendix, an in-depth examination is conducted into the modeling of the target (Fig: A.1) to be intercepted. As outlined in the introduction, the ultimate goal pursued is the interception of a high-performance drone potentially utilized by a criminal group, employing an equally high-performance interceptor drone. Typically, target identification is delegated to a system continuously tracking its speed and position, such as a radar. Alternatively, the interceptor drone, supported by a radar, may utilize a camera for visual target identification. It is crucial to note that this aspect remains beyond the scope of this thesis and is not under consideration.

To obtain data related to the target's position, velocity, and acceleration, a deliberate choice was made to adopt a comprehensive target

modeling approach. Given the conceptual and dynamic similarity between the target and the interceptor, the decision was made to model the target precisely like the interceptor drone. This choice is rooted in the thesis's primary focus on developing the guidance controller, flight controller, and dynamic model of the interceptor drone, diverting attention from the modeling intricacies of the target.
To execute this approach, a MATLAB script was developed, encapsulating the dynamic parameters and block structure of the target, following analogous procedures employed for the interceptor drone. By aligning the modeling framework between the target and the interceptor, a cohesive and consistent development process is ensured, allowing seamless integration into the overall interception system.
Therefore, considering that the developed target model incorporates only the dynamic model and flight controller, it is evident that the target is commanded in velocity. This velocity is maintained at a constant value arbitrarily set by the programmer, who inputs this velocity expressed in the NED frame directly into the flight controller. For the sake of convenience and simplicity, this velocity is aligned with the initial velocity, denoted as V_{0t}, streamlining the setup process for various simulations to be conducted.
The deliberate choice to command the target's velocity allows for controlled and systematic simulations, enabling a thorough evaluation of the interceptor drone's guidance and control systems under different scenarios such as static target and moving target. The arbitrary yet consistent velocity setup, aligned with the initial velocity V_{0t}, ensures a standardized and efficient approach across simulations, allowing for a nuanced analysis of the interceptor drone's performance in intercepting a target with varying dynamic characteristics.
This deliberate modeling strategy ensures a concentrated effort on the key aspects of the thesis while maintaining a clear and purposeful separation between the interceptor drone and the target in terms of modeling intricacies.

Acknowledgments

Expressing these thanks is not at all easy for me. The last period has been challenging, especially for my parents. Being powerless in the face of illnesses affecting the people closest to you has been heart-wrenching. However, I vividly remember the day I visited my mother and father in the hospital, finding them there together, hand in hand, supporting each other through the darkest moments. I remember this event well because from that moment, I promised myself that I would make it through everything, for them. And for this, the milestone I've achieved is dedicated mainly to you, who are my strength, and I am yours. In addition to them, I want to thank my girlfriend, Irene, whose love has brightened all my days, every single one. Her parents, Giuseppe and Antonella, who have been a great help during my darkest times, welcoming me with open arms into their home always. I want to thank my grandmother, who, with her gravioli, has always filled my belly and heart every time I returned home. I want to thank a friend, a brother, Alessio, who has helped me immensely over these years. Thanks to you, studying and living in Tempio has been beautiful and memorable. I thank the Templars, who turned the apartment on Via Giuseppe Renzi into more than just a place to sleep, a place that I could call home for over 5 years. My friends from the AGGIAPONG group, who, through jokes and parties, improved my evenings for several years. Vanessa, a person I can confidently call a true friend. I would also like to extend a small dedication to my late uncles, Donato and Livia, who unfortunately passed away recently.
Finally, to thank all my relatives and friends that I haven't mentioned, I want to use a quote from Arnold Schwarzenegger: "I always tell peo-

ple that they can call me whatever they want, but don't ever ever call me a self-made man. The concept of a self-made man is a myth. I would never made it in my life without someone's help." Well, I can say it too; I dedicate this achievement to all of you, hoping to always help and do good towards others.

A hug,
your dear Andrea.

Esprimere questi ringraziamenti non è affatto facile per me. L'ultimo periodo è stato molto impegnativo, soprattutto per i miei genitori. Essere impotente di fronte ai malanni che colpiscono le persone a te più care è stato straziante. Tuttavia, ricordo perfettamente il giorno in cui ho visitato mia madre e mio padre in ospedale, trovandoli lì insieme, mano nella mano, che si sostenevano nei momenti più bui. Ricordo bene questo evento perché da quel momento mi sono promesso che ce l'avrei fatta nonostante tutto, per loro. E per questo, il traguardo che ho raggiunto è dedicato principalmente a voi, che siete la mia forza, ed io la vostra. Oltre a loro, voglio ringraziare la mia ragazza, Irene, il cui amore ha illuminato tutte le mie giornate, proprio tutte. I suoi genitori, Giuseppe e Antonella, che sono stati di grande aiuto nei momenti più bui, accogliendomi sempre a braccia aperte nella loro casa. Voglio ringraziare mia nonna, che con i suoi gravioli, ha sempre riempito la pancia e il cuore ogni volta che tornavo a casa. Voglio ringraziare un amico, un fratello, Alessio, che mi ha aiutato immensamente in questi anni. Grazie a te, studiare e vivere nel Tempio è stato memorabile. Ringrazio i Templari, che hanno trasformato l'appartamento in Via Giuseppe Renzi in qualcosa di più di un semplice luogo dove dormire, un luogo che ho potuto chiamare casa per oltre 5 anni. I miei amici del gruppo AGGIAPONG, che, tra scherzi e feste, hanno migliorato le mie serate per diversi anni. Vanessa, una persona che posso chiamare con sicurezza una vera amica. Vorrei anche dedicare un piccolo omaggio ai miei zii Donato e Livia, purtroppo scomparsi recentemente.
Infine, per ringraziare tutti i miei parenti e amici che non ho menzionato, voglio utilizzare una citazione di Arnold Schwarzenegger: "Dico sempre alla gente che possono chiamarmi come vogliono, ma non chiamatemi mai self-made man. Il concetto di uomo fatto da sé è un mito. Non ce l'avrei mai fatta nella mia vita senza l'aiuto di qualcuno." Bene, posso dirlo anch'io; dedico questo traguardo a tutti voi, sperando di aiutare sempre e fare del bene verso gli altri.

Un abbraccio,
il vostro caro Andrea.