Control System Complex Engineering Problem

Missile Longitudinal Autopilots: Guidance, Navigation and Control

Submitted By:

Hassan Iqbal [21-ENC-49]

Muhammad Mutahhar Baig [21-ENC-58]

Muhammad Huzaifa Saeed [21-ENC-63]

Submitted To:

Dr. Hammad Zaki

Department of Electronics Engineering

University of Engineering and Technology, Taxila.

Table of Content

CLOs:

CLO1	Mathematical Modeling
CLO2	Analysis
CLO3	Design controller

Missile Longitudinal Autopilots: Guidance, Navigation and Control

Problem Statement:

Design a Missile Longitudinal Autopilots for Guidance, Navigation and Control. The goal of this study is to evaluate the robustness of various three loop autopilot topologies for missile longitudinal control. Specifically, it aims to determine which topology offers the best robustness properties while maintaining identical closed-loop performance. The challenge is to systematically compare these topologies, which include different combinations of acceleration, angular rate, and fin deflection feedback, and identify the most robust configuration for both stable and unstable missile dynamics.

Abstract

In this research, several three loop autopilot topologies for missile longitudinal control are thoroughly analyzed. As a point of comparison, Raytheon Missile Systems commonly uses the traditional three loop autopilot. We investigate the resilience properties of ten different topologies that all reach the same performance level. Each topology is assessed under comparable closed-loop settings using optimal control theory, enabling a targeted comparison of their open-loop resilience characteristics. The differences in robustness between the topologies are illustrated by numerical simulations using both stable and unstable missile models, underscoring the benefits of the traditional arrangement.

Introduction

For more than 50 years, missile longitudinal autopilots have been a crucial component of tactical missile systems. The traditional three-loop autopilot, known as the "Raytheon autopilot," has been a common design. Any autopilot system's primary goal is to use sensor data to provide a steady and precise reaction to command inputs. This paper revisits the traditional three loop autopilot and investigates other topologies that could provide improved robustness without sacrificing efficiency.

Previous studies connected the dots between optimal control and classical approaches, showing that different weightings of control metrics and acceleration error can result in different autopilot structures, such as two, three, and four loop configurations. We study all conceivable topologies generated from combinations of acceleration, angular rate, and first-order lead feedback with a particular focus on the three-loop autopilot in this research.

In order to guarantee that every topology attains the same performance levels, we utilize an optimal control framework, which separates robustness as the primary metric for comparison. We assess the robustness of each topology by analyzing both stable and unstable missile models using numerical simulations, and we finally determine which design has the best robustness performance. For the design and selection of autopilot systems in contemporary missile applications, this analysis offers crucial insights.

Block Diagrams with Explanations

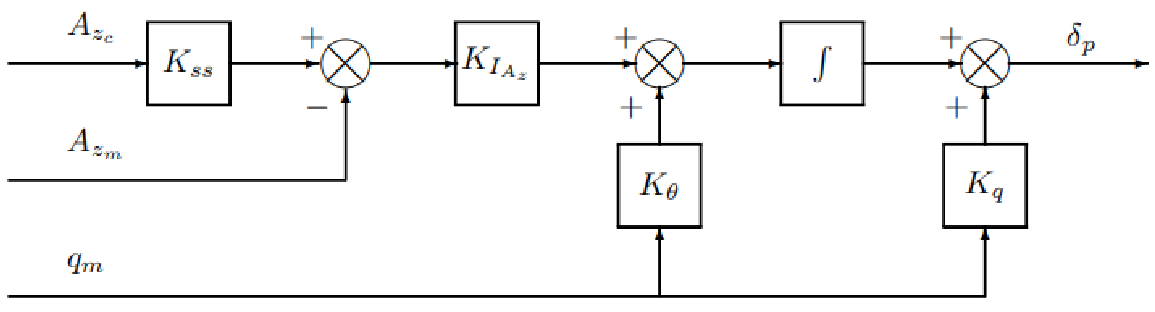


Figure 1:Topology 1 — Classic Three Loop Topology

The block diagram provided represents a three-loop autopilot topology for missile longitudinal control. Here is an explanation of the components and flow of the diagram:

1. Inputs:

- Azc : Commanded acceleration (the desired acceleration).
- Azm: Measured acceleration (the actual acceleration measured by the sensors).
- qm: Measured pitch rate (the rate of change of the missile's pitch angle).

2. Feedback Loop:

- The commanded acceleration Azc is fed into a gain block Kss .

- The output of Kss is then compared with the measured acceleration Azm by subtracting Azm from Azc. This forms the acceleration error signal.

- This error signal is then amplified by KI_Az (integral gain for acceleration error), generating a signal that is integrated to account for cumulative error over time.

3. Pitch Rate Feedback:

- The measured pitch rate qm is fed into a gain block Kq.

- The output of Kq provides feedback based on the rate of change of pitch, contributing to damping and stability.

4. Pitch Angle Feedback:

- The integrated signal is further modified by Kθ which represents the gain associated with the pitch angle.

- This signal is summed with the output from the pitch rate feedback Kq.

5. Integrator:

- The output of KI Az goes through an integrator block, which integrates the acceleration error over time, contributing to reducing steady-state error.

6. Summation and Output:

- The outputs from the integrator, pitch angle feedback Kθ, and pitch rate feedback K_q is summed together.

- The final output, δp, is the control surface deflection command (such as the deflection of missile fins or control surfaces), which is used to adjust the missile's trajectory to match the commanded acceleration.

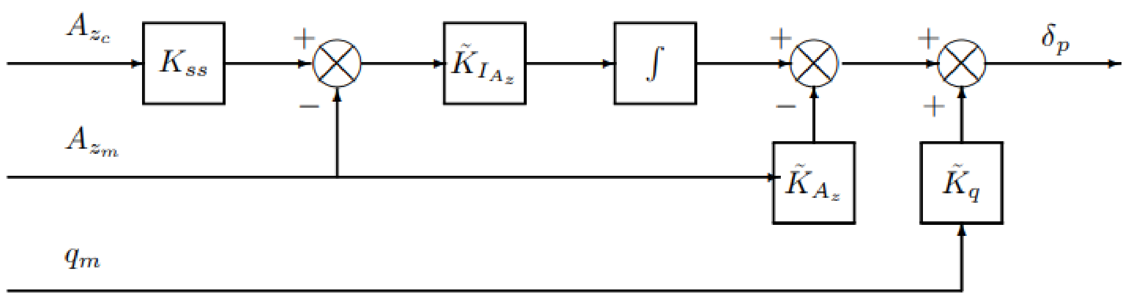


Figure 2: Topology 4 — Block Diagram

1. Inputs:

- Azc: This is a command or reference input for acceleration or another controlled variable.
- Azm: This represents the measured value of the same variable (acceleration or another).

2. Error Signal Generation:

- The command input Azc is multiplied by a constant Kss (possibly a steady-state gain) and then compared with the measured value Azm. The difference (error signal) is computed by a summing junction with Azc . Kss on the positive input and Azm on the negative input.

3. Error Signal Integration:

- The error signal is then fed into a block with a gain ḰI Az and an integrator. This part of the system integrates the error over time to provide an accumulated error signal. This helps in reducing steady-state error in the system.

4. Combining Integrated Error with Additional Feedback:

- The output of the integrator is combined with another feedback path where the measured value Azm is multiplied by a gain Ḱ Az and subtracted from the integrator output.

5. Additional State Feedback:

- There is a state qm (possibly another measured state like angular velocity or another derivative state) that is multiplied by a gain Ḱq and then added to the previously computed signal.

6. Output:

- The resulting signal after all summations and multiplications is the final control output δp, which is likely a control action (like actuator position or force) that will be applied to the system to achieve the desired Azc.

Mathematical Modelling:[CLO-I]

A=$\begin{bmatrix} -1.064 & 1 \\ -290.26 & 0 \end{bmatrix}$ B= $\begin{bmatrix} -0.25 \\ -331.39 \end{bmatrix}$

C=$\begin{bmatrix} --101.71 & 0 \\ 0 & 1 \end{bmatrix}$ D=$\begin{bmatrix} -13.51 \\ 0 \end{bmatrix}$

Compute |sI-A|

sI-A=$\begin{bmatrix} s+1.064 & -1 \\ -290.6 & s \end{bmatrix}$

Compute the determinant of (Si-A) for the inverse:

det(sI-A) = (s+1.064)s+290.26 = s^2+1.064s+290.26

Now find inverse

$(sI\text{-}A)^{-1} = \frac{1}{\det(sI-A)}\begin{bmatrix} s+1.064 & -1 \\ -290.6 & s \end{bmatrix}$

$= \frac{1}{s^2+1.064s+290.26}\begin{bmatrix} s+1.064 & -1 \\ -290.6 & s \end{bmatrix}$

Multiplying B

$(sI\text{-}A)^{-1}.\ B = \frac{1}{s^2+1.064s+290.26}\begin{bmatrix} s+1.064 & -1 \\ -290.6 & s \end{bmatrix}.\begin{bmatrix} -0.25 \\ -331.39 \end{bmatrix}$

$= \frac{1}{s^2+1.064s+290.26}\begin{bmatrix} -0.25s+331.39 \\ 72.65-331.39s-352.376 \end{bmatrix}$

$= \frac{1}{s^2+1.064s+290.26}\begin{bmatrix} -0.25s+331.39 \\ -331.39s-279.72 \end{bmatrix}$

Extract the second row to find the transfer function for the second output (q):

C_2 = [0.00 1.00]

Multiply by the result from above:

$C_2(sI - A)^{-1}B =$ (-331.40s 424.8146) $\frac{1}{1\ s2+1.064s+290.26}$

Add D_2:

Since $D_2 = 0$:

$G_{21}(s) = \frac{-331.40s\ 424.8146}{s2+1.064s+290.26}$

Thus, the transfer function in mathematical form with numerical values is:

$G_{21}(s) = \frac{-13.51s\text{^}2 + 10.91s + 29780}{s\text{^}2 + 1.064s - 290.26}$……………………………………………..(i)

We obtain open loop transfer functions as

$\frac{Azm}{\delta p} = \frac{-13.51s2 + 10.91s + 29780}{s2 + 1.064s + 290.26} = \frac{-13.51(s+46.55)(s-47.35)}{(s+0.53 \pm 17.03j)}$………………………………… (ii)

$\frac{qm}{\delta p} = \frac{-331.4s - 280.3}{s2\ + 1.064s + 290.26}$

$= \frac{-13.51(s+0.846)}{(s+0.53 \pm 17.03j)}$………………………………………………………(iii)

Ruth Howitz stability Criteria:

s^2	1	-715.1
s^1	-330.31	0
s^0	-715.1	0

The closed loop transfer function that we obtained is:

$$\frac{-331.40s - 424.846}{s^2 - 330.34s - 715.1}$$

By Using Characteristics equation, we can create the Routh table as shown below:

From the table this is unstable system we have to make it stable.

Relative Stability

It includes the Steady-State Errors.

Steady-State Errors

The closed loop transfer function of our system is a type 0 system with two poles. Step response of the closed loop transfer function of the system is.

The steady-state error is given by:

$$e(\infty) = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) = \lim_{s\to 0} \frac{sR(s)}{1+G(s)}$$

Here G(s) is given as,

$$G(s) = \frac{-331.4s - 424.7}{s^2 + 1.064s - 290.28}$$

Step input

The static position error constant Kp is defined by:

$$Kp = \lim_{s\to 0} G(s)$$

$$Kp = \frac{-424.7}{-290.28}$$

$$Kp = 1.46$$

$$e(\infty) = \frac{1}{1+Kp}$$

$$\mathrm{e}(\infty) = \frac{1}{1 + 1.46}$$

$$\mathrm{e}(\infty) = 0.406$$

The static position error constant Kp is defined by:

Ramp input

The static velocity error constant Kv is defined by:

$$\mathrm{Kv} = \lim_{s\to 0} s\,\mathrm{G(s)}$$

$$\mathrm{Kv} = 0$$

$$\mathrm{e}(\infty) = \lim_{s\to 0} \frac{1}{\mathrm{Kv}}$$

$$\mathrm{e}(\infty) = \text{infinity}$$

Parabolic input

The static acceleration error constant Ka is defined by:

$$Ka = lim_{s\to 0}\, s^2 G(s)$$

$$Ka = 0$$

$$e(\infty) = \frac{1}{Ka}$$

$$e(\infty) = \frac{1}{0}$$

$$e(\infty) = \infty$$

Table summarizes the steady-state errors for type 0 they are subjected to various inputs.

Input	**Formula**	**Static error constant**	**Error**
Step	$\frac{1}{1+Kp}$	Kp=1.46	$\frac{1}{1+Kp}$
Ramp	$\frac{1}{Kv}$	Kv=0	∞
Parabola	$\frac{1}{Ka}$	$Ka = 0$	∞

Table 1:Table 1:Steady-State Error in Terms of Gain K

Analysis: [CLO-II]

Root Locus

Construction of Root Locus for First equation:

$$G(s) = \frac{-331.4s - 424.7}{s^2 + 1.064s - 290.28}$$

Step:1

Locating the poles and Zero

There are the two poles s=-17.58, 16.51.

There is only 1 zero at s=-1.28

Step:2

Finding angles of Asymptotes and Intersection point

AOA=$\frac{\pm 180(2q+1)}{n-m}$

Centroid=$\frac{\sum poles - \sum zeros}{n-m}$

At q=0,1 angles of Asymptotes=±180

Centroid=0.21

Step:3

Finding the characteristic equation:

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)}$$

$$\frac{C(s)}{R(s)} = \frac{\frac{-331.4s - 424.7}{s^2 + 1.064s - 290.28}}{1 + \frac{-331.4s - 424.7}{s^2 + 1.064s - 290.28}H(s)}$$

$$\frac{C(s)}{R(s)} = \frac{-331.4s - 424.7}{s^2 - 330.36s - 714.98}$$

Step:4

Finding crossing point on the Imaginary axis

Put s=jw in characteristic equation:

Characteristic equation =$s^2 - 330.36s - 714.98$

Characteristic equation =$jw^2 - 330.36jw - 714.98$

By comparing the real and imaginary parts.

W=$\pm$26.73j

Step:5

Finding value of gain K

1+ KG(s)H(s)=0

$$K = \frac{1}{\mathrm{G(s)H(s)}}$$

Put s=0

$$K = \frac{1}{\frac{-424.7}{-290.28}}$$

$$K = 0.683$$

Construction of Root Locus for second equation

$$G(s) = \frac{-331.4s - 280.3}{s^2 + 1.064s + 290.28}$$

Step:1

Locating the poles and Zero

There are the two poles S=-0.53$\pm 17.03j$

There is only 1 zero at s=-0.84

Step:2

Finding the Root Locus on real axis

Step:3

Finding angles of Asymptotes and Intersection point

AOA $\frac{\pm 180(2q+1)}{n-m}$

Centroid=$\frac{\sum poles - \sum zeros}{n-m}$

At q=0,1   angles of Asymptotes=± 180

Centroid=-0.22

Step:4

Finding the characteristic equation:

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)}$$

$$\frac{C(s)}{R(s)} = \frac{\frac{-331.4s - 280.3}{s^2 + 1.064s + 290.28}}{1 + \frac{-331.4s - 280.3}{s^2 + 1.064s + 290.28}H(s)}$$

$$\frac{C(s)}{R(s)} = \frac{-331.4s - 280.3}{s^2 - 330.36s + 9.96}$$

Step:5

Finding crossing point on the Imaginary axis

Put s=jw in characteristic equation:

Characteristic equation =$s^2 - 330.36s + 9.96$

Characteristic equation =$jw^2 - 330.36jw + 9.96$

By comparing the real and imaginary parts.

W=$\pm$3.15j

Step:6

Finding value of gain K

1+ KG(s)H(s)=0

$$K = \frac{1}{\mathrm{G(s)H(s)}}$$

Put s=0

$$K = 0.003213$$

Designing a lead compensator:

$$G(s) = \frac{-331.4s - 280.3}{s^2 + 1.064s + 290.28}$$

Locating the poles and Zero

There are the two poles S=-0.53$\pm$17.03j

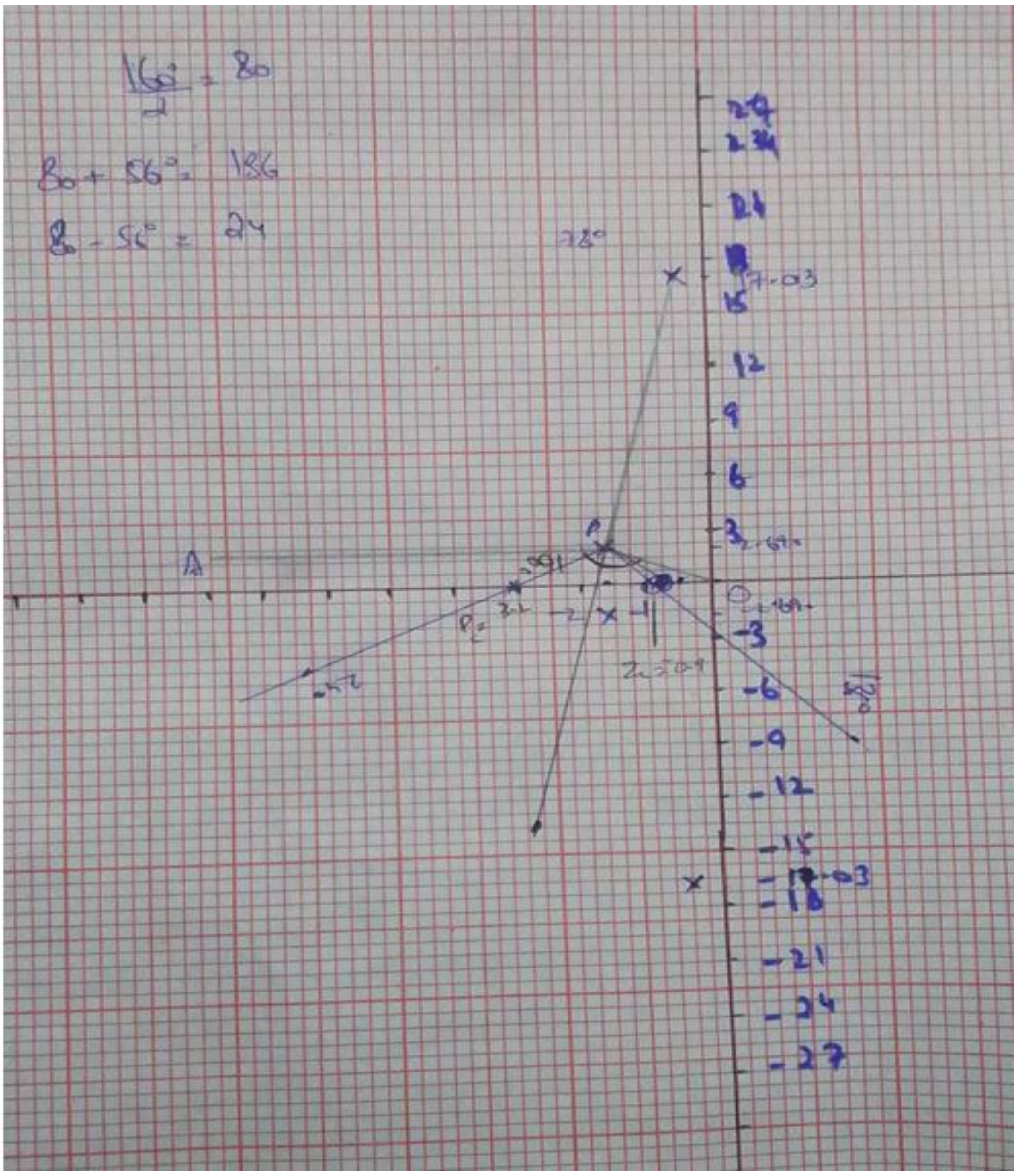
There is only 1 zero at s=-0.84

Finding dominant poles:

- **Natural Frequency (ωn)**: 3.1563.156
- **Damping Ratio (ζ)**: 52.3452.34

S=-1.651±j2.690

Draw the root locus:



Find angle contributed by lead network

Angle contributed by pole=$180 - tangentinverse(\frac{17.03}{1.12})$=78

Angle contributed by zero =$180 - tangentinverse(\frac{17.03}{0.84})$=145

Φ=78-145+180=113

As greater than 60 divide by 2

Φ=$\frac{113}{2}$=56

Finding compensated polesand zeros from graph

Zc=-0.9

Pc=-3.2

Finding transfer function of compensator

Gc(s)=$\frac{(s+Zc)^\wedge 2}{(s+pc)^\wedge 2}$

Finding open loop transfer function of compensated system

- Go(s)=$\frac{(s+Zc)^\wedge 2}{(s+pc)^\wedge 2} * \frac{-331.4s-280.3}{s^2+1.064s+290.28}$

Finding error and kp

Kp=$lim_{s\to 0} G(s)$

Kp=0.076

$$\boldsymbol{e(\infty) = 0.929}$$

Design:[CLO-III]

Controllability by poles placement:

A=$\begin{bmatrix} -1.064 & 1 \\ -290.26 & 0 \end{bmatrix}$ B= $\begin{bmatrix} -0.25 \\ -331.39 \end{bmatrix}$

Order of matrix N=2

Checking controllability:

Controllability matrix

M=[B : AB]

M=$\begin{bmatrix} -0.25 & 331.656 \\ -331.39 & 72.565 \end{bmatrix}$

RANK=2

The system is completely state controllable and arbitrary pole placement is possible.

Pole placement method:

| SI-A |= |$\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$ - $\begin{bmatrix} -1.064 & 1 \\ -290.26 & 0 \end{bmatrix}$|

| SI-A |=290.26s^2

Compare the above equation with general second order equation:

. a0s^2 +a1s +a2= 0

. a1=290.26

Required closed loop poles

S=-2.5132 S=150.1351

The desired closed loop equations are

(s +2.5132) (s +150.1351) =0

$s^2 + 152.64s + 377.31$=0

$\alpha 0 = 1$ $\alpha 1 = 152.64$ $\alpha 2 = 377.31$

K = [$(\alpha 2 - a2)$: $(\alpha 1 - a21)$]

K = **[** 373.3 : −137.61 **]**

LQR Transformation:

1. The initial state-space representation of the system is given by:

$\dot{x} = Ax + Bu$

$y = Cx + Du$

where x represents the state vector, u is the control input, y is the output, and A, B, C, and D are matrices defining the system dynamics.

2. To simplify the LQR problem, the states are augmented with the control input, and the control input is replaced with its derivative. This transforms the system to look like full state feedback

3. New State-Space Representation: The augmented system is represented as:

$\dot{x} = A_1 x_1 + B_1 u1$

$y = C_1 x_1 + D_1 u_1 - Kss\ \tau$

where x_1 is the augmented state vector, and $u_1 = \delta p$

4. Transformation Matrices: The transformation matrices are defined as:

$A_1 = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}$ $B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$C_1 = \begin{bmatrix} C & D \\ A(2,;) & B(2,;) \end{bmatrix}$ $D_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

5. The coordinate transformation is applied to get the new state vector X_2

$x2=C1x1=y_1$

$\dot{x}_2 = A_2 x_2 + B_2 u2$

$y_2 = x_2$

$A_1 = A_1 C_1 A_1^{-1}$ and $B_2 = C_1 B_1$

6. The LQR gains are found using the algebraic Riccati equation solver:

[P , eig Acl - K_{opt}] = care [A_2 B_2 Q_2 R_2]

$Q2=C_1^{-T} H_1^T Q_{11} H_1 C_1^{-T}$, $R_2 = R$

K obtained from

where Ac = A2 + B2Kopt. BcB2Kopt. Cc = H1C1^1, and Boc = B2Kopt

Simulink Model:

Step 1:

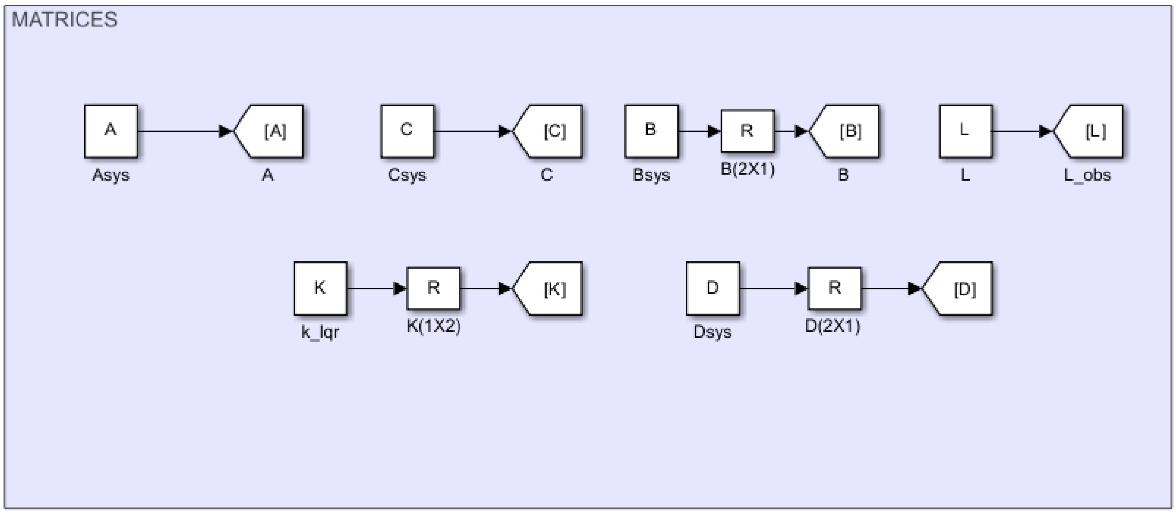


Figure 3:Matrices

illustrates the setup for a control system with state-space representations and potentially observer and controller design matrices. Here's a detailed explanation of each block and its connection:

1. A (Asys) to [A]:

- The block labeled "A" corresponds to the system's state matrix.
- It produces an output labeled "[A]", which likely represents the matrix A for further use in the model.

2. C (Csys) to [C]:

- The block labeled "C" represents the system's output matrix, C.

- The output "[C]" indicates that this matrix is also being prepared for subsequent steps in the control system design.

3. B (Bsys) to R (B(2X1)) to [B]:

- The "B" block represents the system's input matrix, B.
- The output of B, labeled as "R" with a subscript indicating its dimensions (B(2X1)), passes through an intermediary block.
- The final output is labeled "[B]", representing the processed or formatted (B) matrix.

4. L to [L]:

- The block labeled "L" is likely the observer gain matrix.
- Its output is labeled "[L]", suggesting it is prepared for use in designing the observer or feedback control.

5. K (k_lqr) to R (K(1X2)) to [K]:

- The block labeled "K" represents the LQR (Linear-Quadratic Regulator) gain matrix k _lqr
- This block outputs to "R" (K(1X2)), indicating it processes the gain matrix into a specific dimension.
- The final output "[K]" indicates the formatted or processed LQR gain matrix.

6. D (Dsys) to R (D(2X1)) to [D]:

- The block labeled "D" represents the disturbance matrix D
- Its output is labeled "R" (D(2X1)), showing the dimension of the matrix being processed.
- The final output is labeled "[D]", which suggests it is prepared for further use in the model, possibly for disturbance rejection analysis.

Step 2:

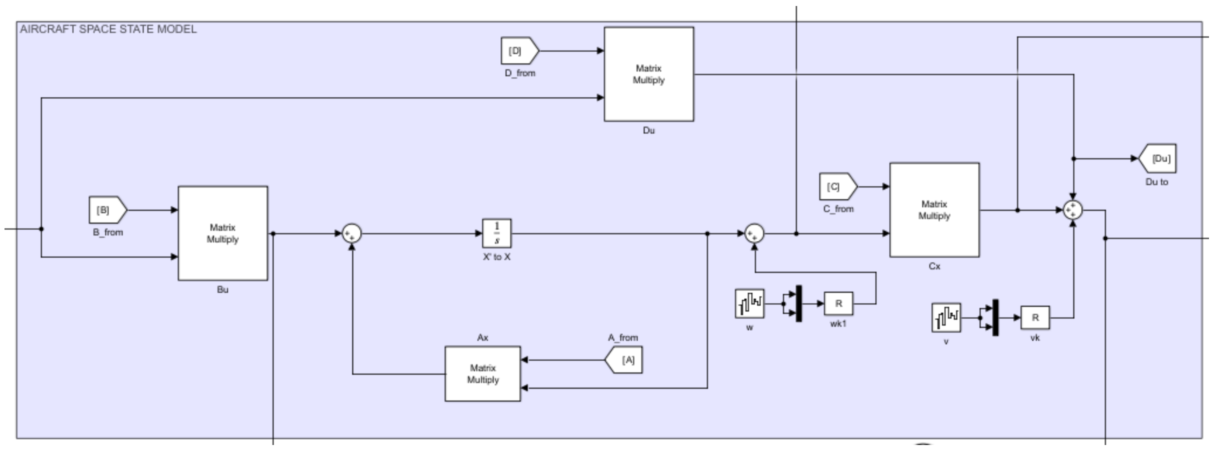


Figure 4:Aircraft Model

This Simulink block diagram represents an aircraft space state model, which is a mathematical model of a physical system represented by state variables. The model appears to follow a standard state-space representation, which is typically used in control systems to describe the behavior of dynamic systems. Here's a step-by-step explanation of the blocks and their connections:

1. State-Space Equations:
   - The general form of the state-space representation is:

$$\dot{x}= Ax+Bu+w$$

$$y= Cx+Du+v$$

where xx is the state vector, uu is the input vector, yy is the output vector, ww is the process noise, and vv is the measurement noise.

2. Matrix Inputs:
   - **[A]**: This block provides the state matrix AA, which represents the system dynamics.
   - **[B]**: This block provides the input matrix BB, which represents how the input uu affects the state xx.
   - **[C]**: This block provides the output matrix CC, which represents how the state xx affects the output yy.
   - **[D]**: This block provides the feedthrough matrix DD, which represents the direct effect of the input uu on the output yy.

3. **State Update:**
   1. Matrix Multiply (Ax): Multiplies the state xx by the state matrix AA.
   2. Matrix Multiply (Bu): Multiplies the input uu by the input matrix BB

3. **Sum Block**: Adds the results of $AxAx$ and $BuBu$ to get the state derivative x

4. State Integration:

Integrator (X' to X): Integrates the state derivative $\dot{x}$x˙ to get the state xx.

5. Output Calculation:

- **Matrix Multiply (Cx)**: Multiplies the state xx by the output matrix CC.
- **Matrix Multiply (Du)**: Multiplies the input uu by the feedthrough matrix DD.
- **Sum Block**: Adds the results of $CxCx$ and $DuDu$ to get the output yy.

6. Noise Addition:

- **w** and **v**: Represent process noise and measurement noise, respectively.
- **Sum Block**: Adds process noise ww to the state update equation.
- **Sum Block**: Adds measurement noise vv to the output equation.

7. Noise Sources:

- **wk1** and **vk**: These blocks generate the process noise ww and measurement noise vv.

8. Output:

- **Du to**: Represents the final output of the system after considering the direct feedthrough and measurement noise.

Step 3:

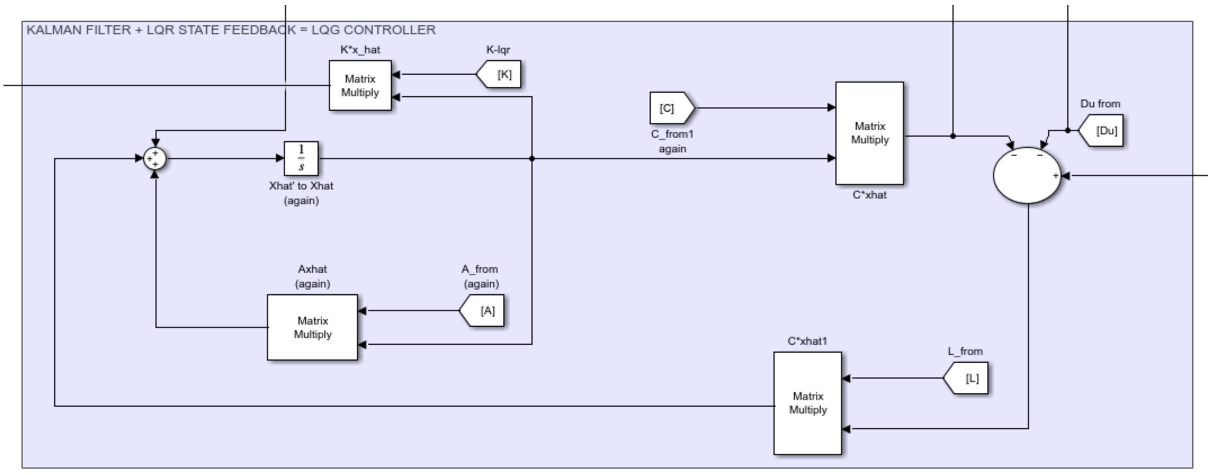


Figure 5:Kalman Filter+LQR Feedback

The LQG controller is designed to provide optimal control of a system in the presence of noise and disturbances. It consists of two main components:

1. Kalman Filter: Estimates the state of the system based on noisy measurements.

2. LQR State Feedback: Provides optimal control input based on the estimated state to minimize a cost function.

[A]: State matrix representing system dynamics.

[C]: Output matrix representing how states affect the outputs.

[K]: LQR gain matrix for state feedback.

[L]: Kalman gain matrix for state estimation.

1. **State Estimation:**
    - The Kalman filter uses the state matrix AA and output matrix CC to estimate the current state $x^\wedge x^\wedge$ from the output measurements.
    - The Kalman gain LL adjusts the state estimate based on the difference between the actual output and the estimated output.
2. **State Feedback Control:**
    - The LQR controller calculates the control input uu by multiplying the estimated state $x^\wedge x^\wedge$ by the LQR gain matrix KK.
    - The control input is adjusted to minimize a cost function, typically a combination of state deviation and control effort.
3. **System Output:**
    - The estimated state $x^\wedge x^\wedge$ is used to compute the control input $K*xhatK*xhat$.
    - The final control input combines the state feedback control input with any direct feedthrough input $DuDu$.

Step 4:

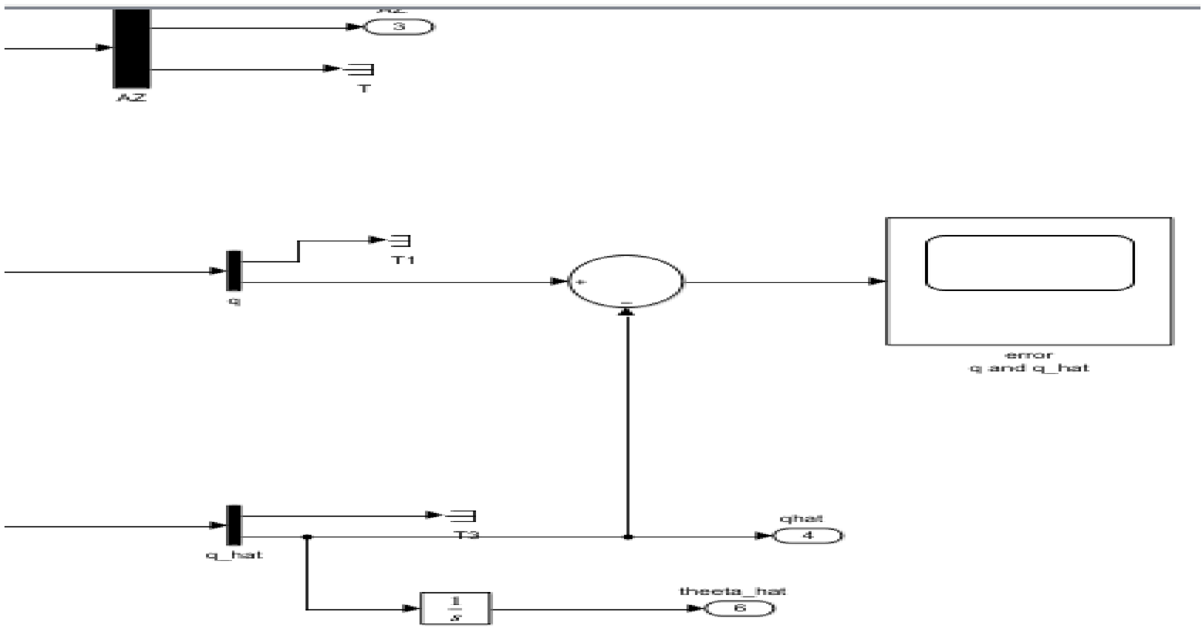


Figure 6:Output of setup 1

1. **Processing of *AZAZ*:**
   - The signal *AZAZ* is processed through a black box, producing outputs that are used elsewhere in the system. The exact nature of this processing is not specified.
2. **Calculation of Error:**
   - The actual pitch rate (*qq*) and the estimated pitch rate (*q*^*q*^) are fed into a sum block. The sum block computes the difference (*q*−*q*^*q*−*q*^), which represents the estimation error.
   - This error is then sent to a display block labeled "error q and q_hat" for monitoring.
3. **Integration for Pitch Angle:**
   - The estimated pitch rate (*q*^*q*^) is also fed into an integrator block (1/*s*1/*s*) to calculate the estimated pitch angle (*θ*^*θ*^).
   - The output of the integrator is the estimated pitch angle (*θ*^*θ*^), which is labeled as "theta_hat" and is represented as signal 6.
4. **Signal Labeling:**
   - T1, T3: These labels represent intermediate signals in the system, but their specific roles are not detailed in this part of the diagram.
   - qhat: This label indicates the output signal representing the estimated pitch rate.

Step 5:

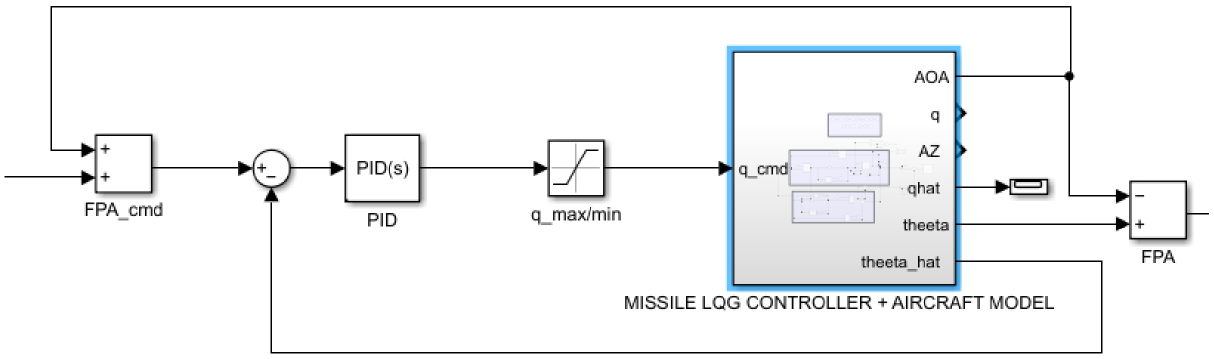


Figure 7:Missile LQG controller+Aircraft Model

The block diagram appears to be a control system for a missile, incorporating an LQG (Linear Quadratic Gaussian) controller and an aircraft model. Here's a possible interpretation:

1. The system takes in inputs such as AOA, AZ, and FPA_cmd.
2. The PID controllers (PID(s) and PID) are used to control the pitch rate (q_cmd) and flight path angle (FPA_cmd).
3. The LQG controller is used to estimate the state of the system, including the pitch rate (qhat) and angle (theeta_hat).
4. The aircraft model is used to simulate the dynamics of the missile.
5. The system outputs include the commanded pitch rate (q_cmd), estimated pitch rate (qhat), and estimated angle (theeta_hat).

Step 6:

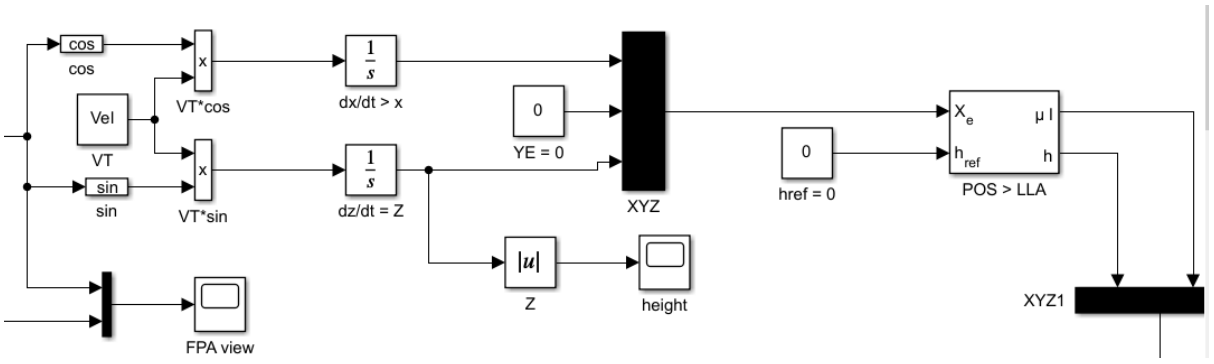


Figure 8:Design for FPA view

Processing Block:

- 1/s: These blocks represent integrators. They are used to compute the time integral of the input signals, essentially converting velocities into positions.
- VT*cos & VT*sin: These blocks likely multiply the velocity (VT) by cosine and sine functions, likely of an angle representing the direction of motion. This could be related to decomposing the velocity into its horizontal components (north/south and east/west) needed for LLA conversion.
- sin & cos: These blocks likely apply the sine and cosine functions to some angle, perhaps to calculate the direction of motion.
- 0: These blocks represent constant values of 0, likely serving as initial conditions for the integrators.
- YE = 0: This block likely represents a constant value of 0, possibly indicating an initial value for the Y-coordinate.
- |u|: This block represents the absolute value of a signal, possibly the magnitude of a velocity vector.
- FPA View: This block could be an optional output representing the Flight Path Angle (FPA) calculated based on the velocity vector.

Output:

- X_e, h_ref, h: These blocks represent the output LLA coordinates.
  - X_e: Represents the horizontal position, likely longitude.
  - h_ref: Represents the reference height, potentially a sea level reference.
  - h: Represents the altitude above the reference height.
- XYZ1: This block likely represents the final output LLA coordinates, perhaps in a specific format.

Function:

The diagram shows a system likely designed for converting XYZ position data from an INS or similar device into the corresponding LLA coordinates (latitude, longitude, altitude). The core of the conversion process involves integration, trigonometric operations, and potentially calculations related to the Flight Path Angle.

Step 7:

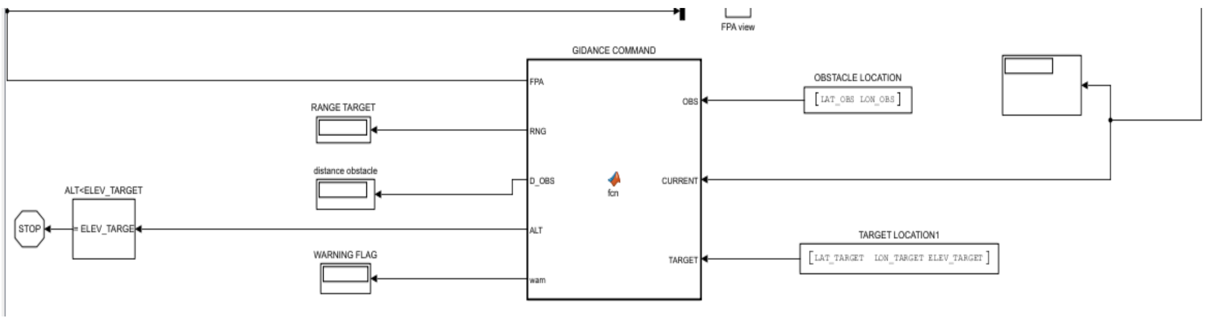


Figure 9:Overall Guidance Command

Inputs:

- ALT<ELEV_TARGET: Represents the current altitude and elevation of the vehicle relative to the target.
- RANGE TARGET: Indicates the straight-line distance between the vehicle and the target.
- OBSTACLE LOCATION: Provides the latitude and longitude coordinates of an obstacle.
- TARGET LOCATION1: Contains the latitude, longitude, and elevation of the target.

Processing Blocks:

1. **WARNING FLAG:** Logical signal that indicates the presence of a detected obstacle.
2. distance obstacle: Calculates the distance between the vehicle and the obstacle.
3. **GIDANCE COMMAND:** This appears to be the central block where all input data is combined to generate the guidance commands. Inside this block, the following computations occur

- **FPA:** Determines the flight path angle, which dictates the vehicle's ascent or descent angle.
- **RNG:** Calculates the desired range to the target, taking into account the obstacle's position.
- **OBS:** Processes the obstacle's location information.
- **CURRENT:** A placeholder for a component that might represent the current state of the vehicle (position, velocity, etc.).
- **ALT:** Calculates the required altitude adjustments based on the target's elevation and the obstacle's presence.
- **TARGET:** Combines the target's location and the computed range to determine the final desired target position.

Outputs:

FPA view: Represents the calculated flight path angle, likely used for visualization or control purposes.

Function:

The diagram suggests a guidance system that aims to achieve the following:

- Navigate towards the target: The system receives information about the target's position and calculates a path to reach it.
- Obstacle avoidance: When an obstacle is detected, the system incorporates this information to adjust the flight path and avoid collision.
- Altitude control: The system considers the target's elevation and obstacles to determine the optimal altitude for the vehicle

Step 8:

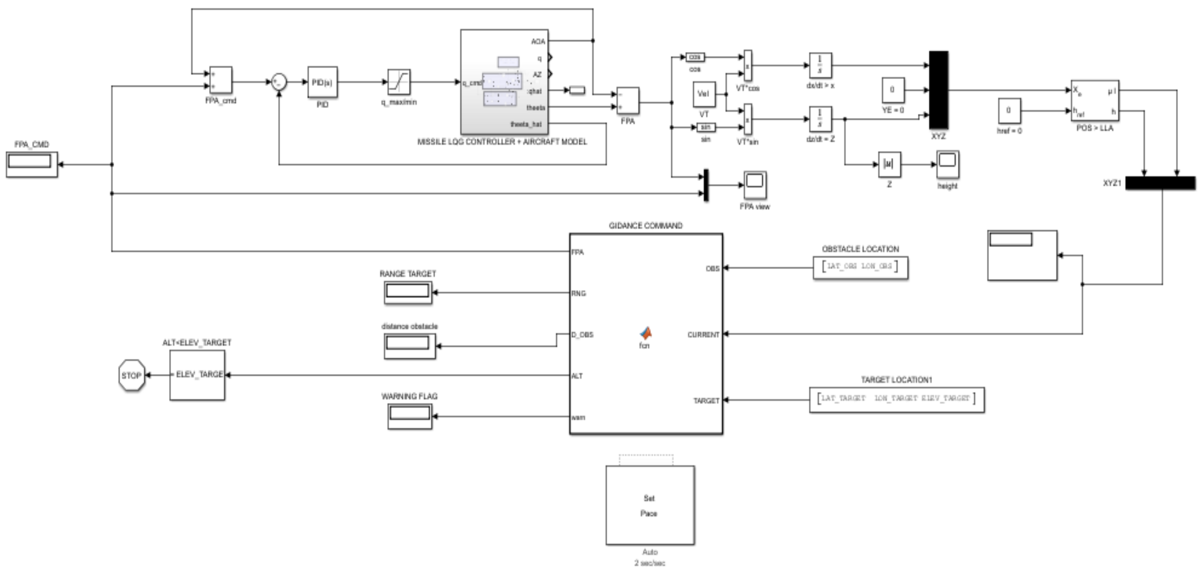


Figure 10:overall Model

Outputs:

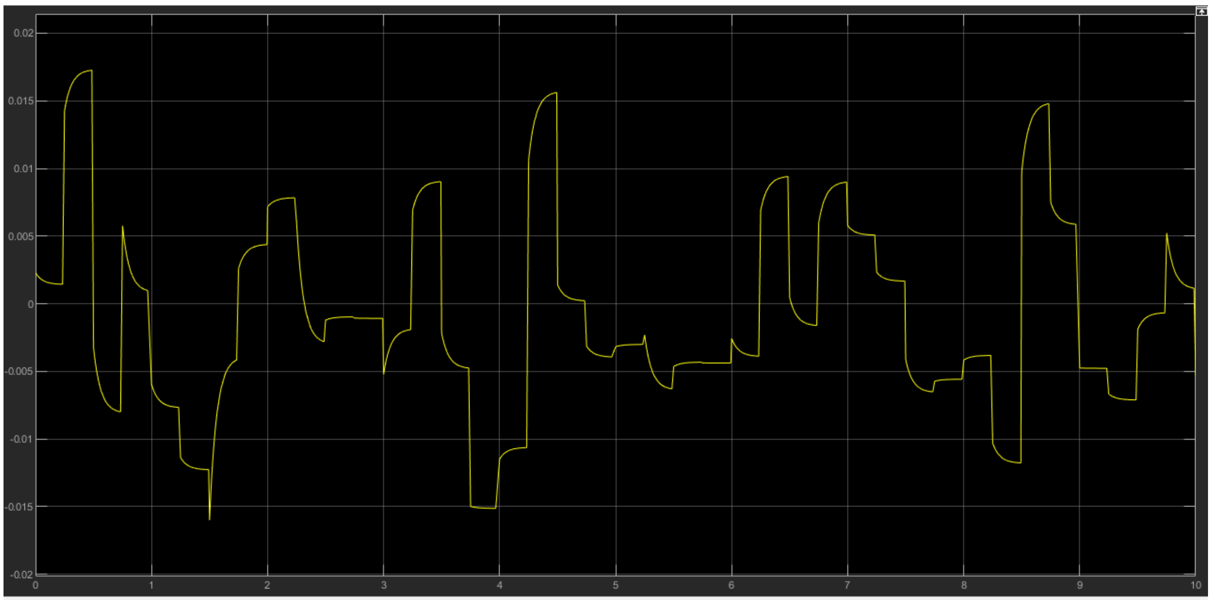


Figure 11:Error q and q_hat

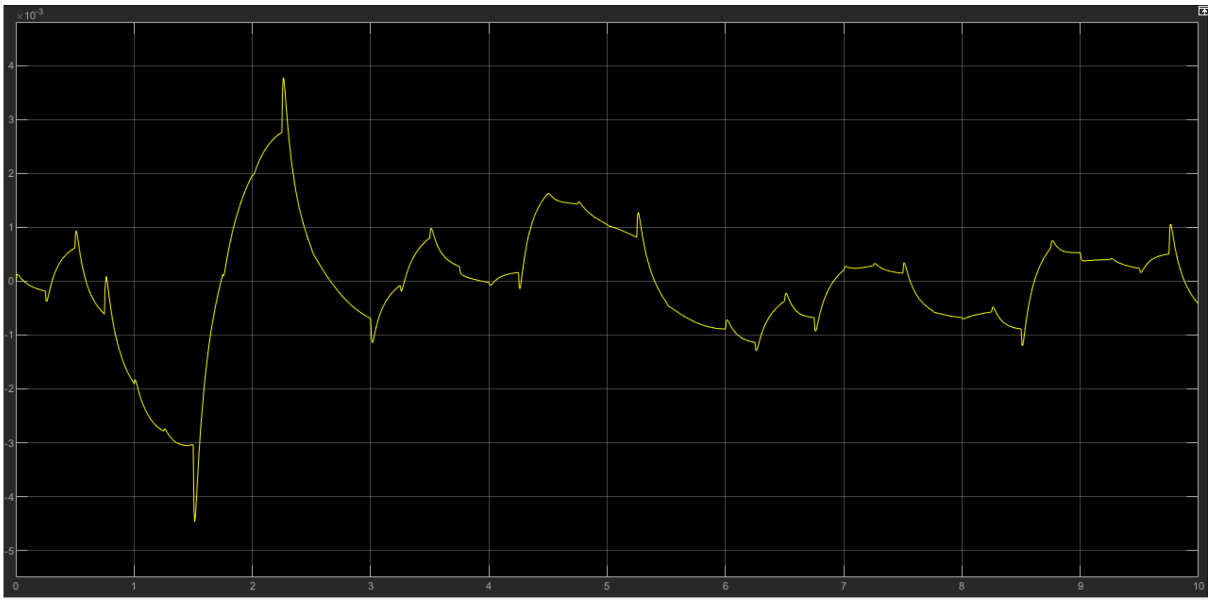


Figure 12:q_hat

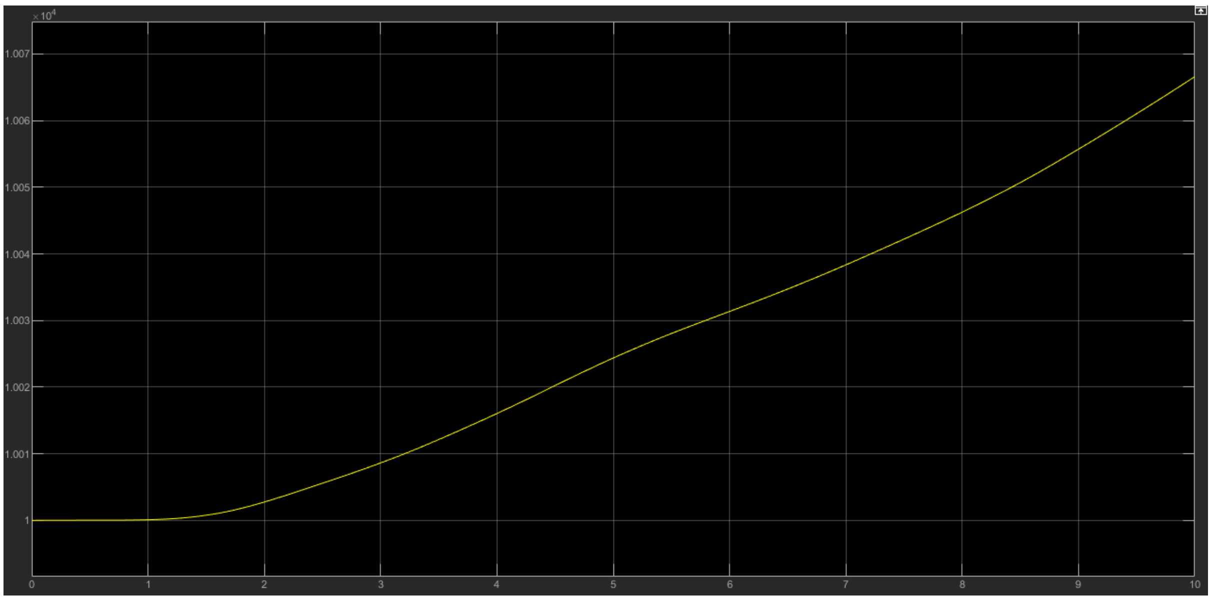


Figure 13:Height

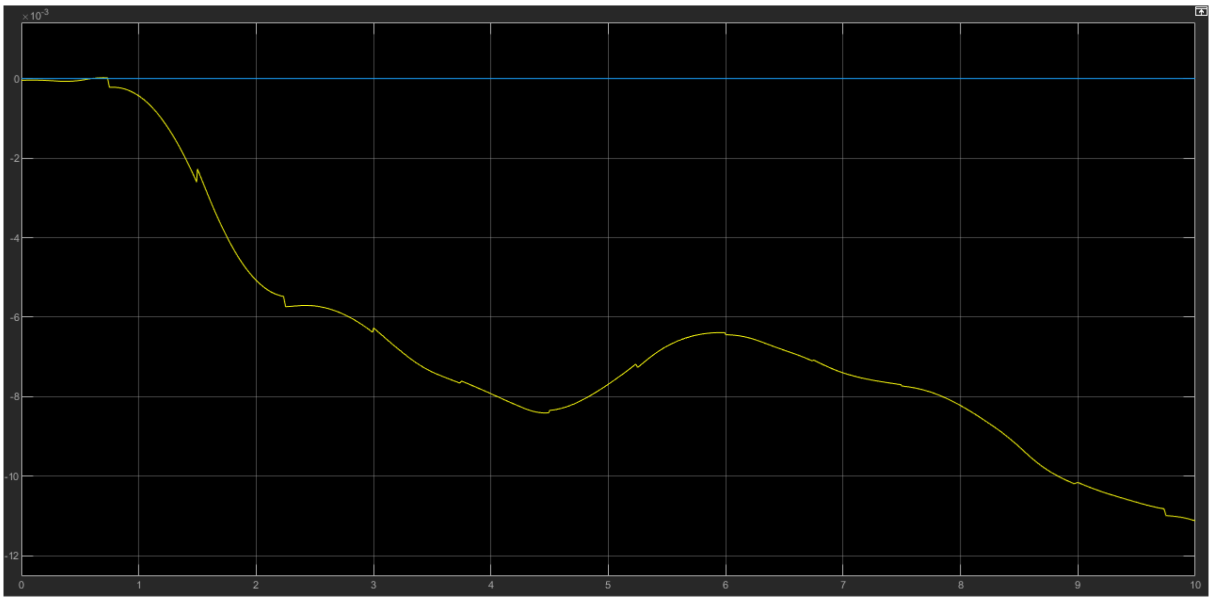


Figure 14:FPA view

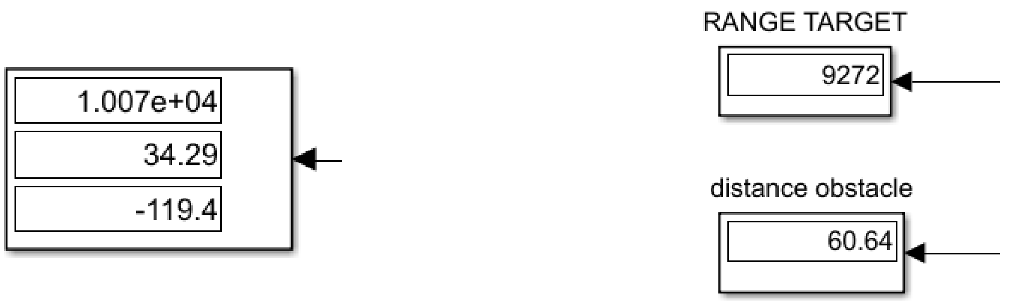


Fig 15: Target details, Target range and distance of obstacle

Applications

- Keeping the missile on the intended flight path and reacting to disruptions to keep it stable during flight.
- Trajectory tracking involves precisely hitting a target by adhering to a predefined trajectory.
- Utilizing real-time target information to dynamically modify the missile's trajectory in coordination with guidance systems.
- Enhancing the missile's performance through control law optimization for robustness, speed, and accuracy is known as performance optimization.
- The ability to withstand and adapt to various environmental disturbances such as wind, pressure changes, and other atmospheric variables.

Conclusion

When several three-loop topologies for missile longitudinal autopilots are compared, it becomes clear that each topology has unique benefits and trade-offs to meet various operational and performance criteria.

Although it may cause slower response times, cascade loop topology is used because of its ease of design and tuning and clear division of control objectives.
Due to multiple control actions, Parallel Loop Topology improves overall system response speed and robustness, but it also introduces additional complexity and potential interaction effects.
Though it presents a difficulty to ensure stability and best performance, hybrid loop topology offers design freedom by striking a balance between response time and robustness.
By utilizing both reactive and predictive control, Feedback-Feedforward Loop Topology excels in disturbance rejection and tracking performance, but it requires precise modeling.

References

1. Nise, N. S. (n.d.). *CONTROL SYSTEMS ENGINEERING Seventh edition.* California State Polytechnic University, Pomona.
2. Paul Zarchan, Tactical and Strategic Missile Guidance, Fourth Edition. AIAA Volume 199 Progress in Astronautics and Aeronautics, Reston, VA, 2002
3. https://arc.aiaa.org/doi/abs/10.2514/6.2005-6381
4. https://www.researchgate.net/publication/271374403_Missile_longitudinal_autopilot_design_using_the_state-dependent_Riccati_equation_method
5. https://www.youtube.com/watch?v=0AJ6E48Aj9U&pp=ygVBTWlzc2lsZSBMb25naXR1ZGluYWwgQXV0b3BpbG90czogR3VpZGFuY2UsIE5hdmlnYXRpb24gYW5kIENvbnRyb2w%3D
6. Missile Longitudinal Autopilots: Comparison of Multiple Three Loop Topologies
7. Missile Longitudinal Autopilots: Connections Between Optimal Control and Classical Topologies
8. Longitudinal nonlinear adaptive autopilot design for missiles with control constraint
9. Automatic Control of Aircraft and Missiles by John H. Blakelock
10. Missile Guidance and Control Systems by George M. Siouris

Appendix

MATLAB Code:

```
clc
close all
clear all
format short
A= [-1.064 1.000; 290.26 0.00];
B = [-0.25; -331.40];
C= [-123.34 0.00; 0.00 1.00];
D= [-13.51; 0];
states = {'AOA','q'};
inputs = {'\delta_c'};
outputs = {'Az','q'};
sys = ss(A,B,C,D, 'statename', states,...
    'inputname', inputs,...
    'outputname', outputs);
%TF
TFs = tf(sys);
TF = TFs (2,1);
disp(pole (TF));
%LQR weight matrices
Q = [0.1 0; 0 0.1];
R = 0.5;
%LQR gain
[K,S,e] = lqr (A,B,Q,R);
fprintf('eigenvalues of A-BK\n');
disp(eig(A-B*K));
fprintf('Feedback gain K');
disp(K)
%closed loop system
Acl = A-B*K;
Bcl = B;
syscl = ss (Acl, Bcl, C, D, 'statename', states,...
'inputname', inputs,...
'outputname', outputs);
%our TF closed loop
TF = tf(syscl);
TFC = TF (2,1);
%LQG Kalman filter design
G = eye(2);
H = 0*eye (2);
%Kalman Q.R noise matrices
Qbar = diag (0.00015*ones (1,2));
Rbar = diag (0.55*ones (1,2));
%define noisy system
sys_n = ss (A, [B G],C, [D H]);
[kest, L,P] = kalman (sys_n, Qbar, Rbar, 0);
%kalman gain observer closed loop
Aob=A-L*C;
[illegible]play [illegible]rver eigenvalues
[illegible] eigenvalues\n');
[illegible]));
%[illegible]tants (you choose)
d[illegible] 0.75;
dT2 = 0.25;
```

```
%missile parameters
R = 6371e3; %earth radius
Vel = 1021.08; %speed (m/s)
m2f = 3.2811; %meters to feet
%target location
LAT_TARGET = 34.6588;
LON_TARGET = -118.769745;
ELEV_TARGET = 795; %m MSL
%initial location
LAT_INIT = 34.2329;
LON_INIT = -119.4573;
ELEV_INIT = 10000; %m p- MSL
%obstacle location
LAT_OBS = 34.61916;
LON_OBS = -118.8429;
d2r = pi/180; %degrees to radians

%convert to radians
l1 = LAT_INIT*d2r;
u1 = LON_INIT *d2r;
l2 = LAT_TARGET*d2r;
u2 =  LON_TARGET*d2r;
dl = l2-l1;
du = u2-u1;
%haversine formula
a = sin(dl/2)^2 + cos(l1)*cos (l2)*sin(du/2)^2;
c = 2*atan2(sqrt(a), sqrt(1-a));
d = R*c; %horizontal distance (in m)
%initial range (pythagorean theorem)
r = sqrt(d^2+(ELEV_TARGET-ELEV_INIT)^2);
%initial azimuth
yaw_init = azimuth(LAT_INIT, LON_INIT, LAT_TARGET, LON_TARGET);
yaw = yaw_init*d2r;
% initial flight path angle
dh = abs(ELEV_TARGET-ELEV_INIT);
FPA_INIT = atan(dh/d); %rad
```

Simulink Model Code:

```
function [FPA,RNG,D_OBS,ALT,warn] = fcn(OBS, CURRENT, TARGET)


R = 6.371e3;
d2r = pi/180;
%OBSTACLE LOCATION
LAT_OBS=OBS(1);
LON_OBS=OBS(2);
%OBSTACLE THRESHOLD (ZONE)
thres=2000;
%TARGET LOCATION
LAT_TARGET=TARGET(1);
LON_TARGET=TARGET(2);
ELEV_TARGET=TARGET(3)
%CURRENT LOCATION
ELEV_CUR=CURRENT(1);
LAT_CUR=CURRENT(2);
```

```
LON_CUR=CURRENT(3);
% % distance to target
l1= LAT_CUR*d2r;
u1 =LON_CUR*d2r;
l2 =LAT_TARGET*d2r;
u2 =LON_TARGET*d2r;
dh = abs(ELEV_TARGET-ELEV_CUR);

dl =l2-l1;
du =u2-u1;

a = sin(dl/2)^2 + cos(l1)*cos(l2)*sin(du/2)^2;
c = 2*atan2(sqrt(a),sqrt(1-a));
d = R*c; %horizontal distance (in m)

% % distance from obstacle
l3= LAT_OBS*d2r;
u3 = LON_OBS*d2r;
dl = l3-l1;
du=u3-u1;
a = sin(dl/2)^2 + cos(l1)*cos(l2)*sin(du/2)^2;
c = 2*atan2(sqrt(a), sqrt(1-a));
d_obs = R*c; %horizontal distance (in m)

% %  current range (from target)  range> distance
range=sqrt(d^2+dh^2);
% % calculate command flight path set point based on d.obs range
if abs(d_obs)>=thres
    w=0;
    FPA_CMD=atan(dh/d);
else
     w=1;
    FPA_CMD=0;
end

% % output variables
FPA = FPA_CMD;
RNG = range;
D_OBS = d_obs;
warn = w;
ALT = ELEV_CUR;
```