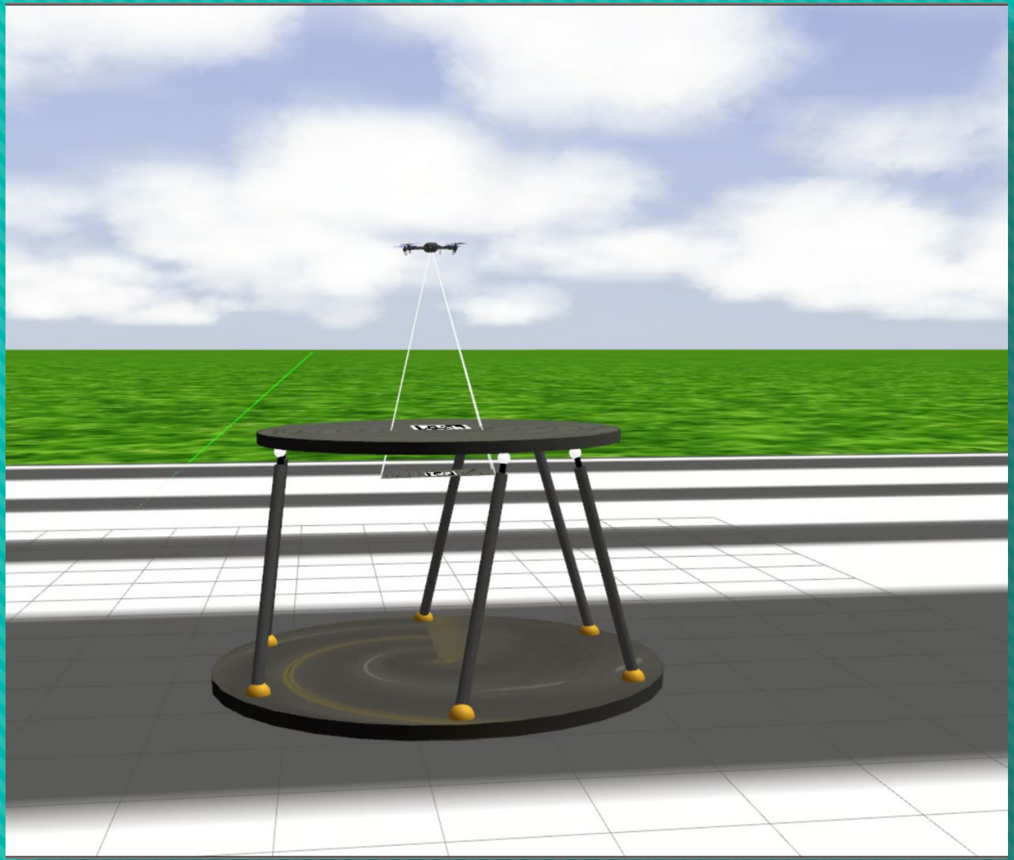


Linear tracking of drone on a stochastically moving platform

EE650 MINI PROJECT

GROUP 20

Collaborators

RITWIK(210866) AMANDEEP (210114) KIRTI (210522) ADRIJA(210071)

INDEX

Drone Dynamics

The drone's motion follows Newton's laws and rotational dynamics.

Horizontal Acceleration in X and Y

$$\ddot{x} = -g\sin(\theta)$$

$$\ddot{y} = -g\sin(\phi)$$

Here, $-g\sin(\theta)$ and $-g\sin(\phi)$ are horizontal gravity components based on roll and pitch.

Rotational Equations for Roll (ϕ) and Pitch (θ)

$$I_x\ddot{\phi} = \text{torque}_x$$

$$I_y\ddot{\theta} = \text{torque}_y$$

where I_x and I_y are the moments of inertia.

Reference citation : State Space System Modelling of a Quad Copter UAV

State Space Model

States of the System

- x: Position along the X-axis
- $\dot{x}$: Velocity along the X-axis
- θ: Angle of pitch
- $\dot{\theta}$: Rate of pitch
- y: Position along the Y-axis
- $\dot{y}$: Velocity along the Y-axis
- ϕ: Angle of roll
- $\dot{\phi}$: Rate of roll

State-space Equations

$$\dot{X} = AX + BU$$

$$Y = CX + DU$$

State Vector X

$$X = \begin{bmatrix} x & \dot{x} & \theta & \dot{\theta} & y & \dot{y} & \phi & \dot{\phi} \end{bmatrix}^T$$

Control Input Vector U

$$U = \begin{bmatrix} \mathrm{torque}_x & \mathrm{torque}_y \end{bmatrix}^T$$

Output Vector Y

$$Y = \begin{bmatrix} x & y \end{bmatrix}^T$$

State Space Model

Constants

- $g = 9.8$: Gravitational acceleration
- $I_y = 0.87$: Moment of inertia around the y-axis
- $I_x = 0.87$: Moment of inertia around the x-axis (assuming symmetry)

State Matrix A

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Input Matrix B

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{I_x} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{I_y} \end{bmatrix}$$

Output Matrix C

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Feedforward Matrix D

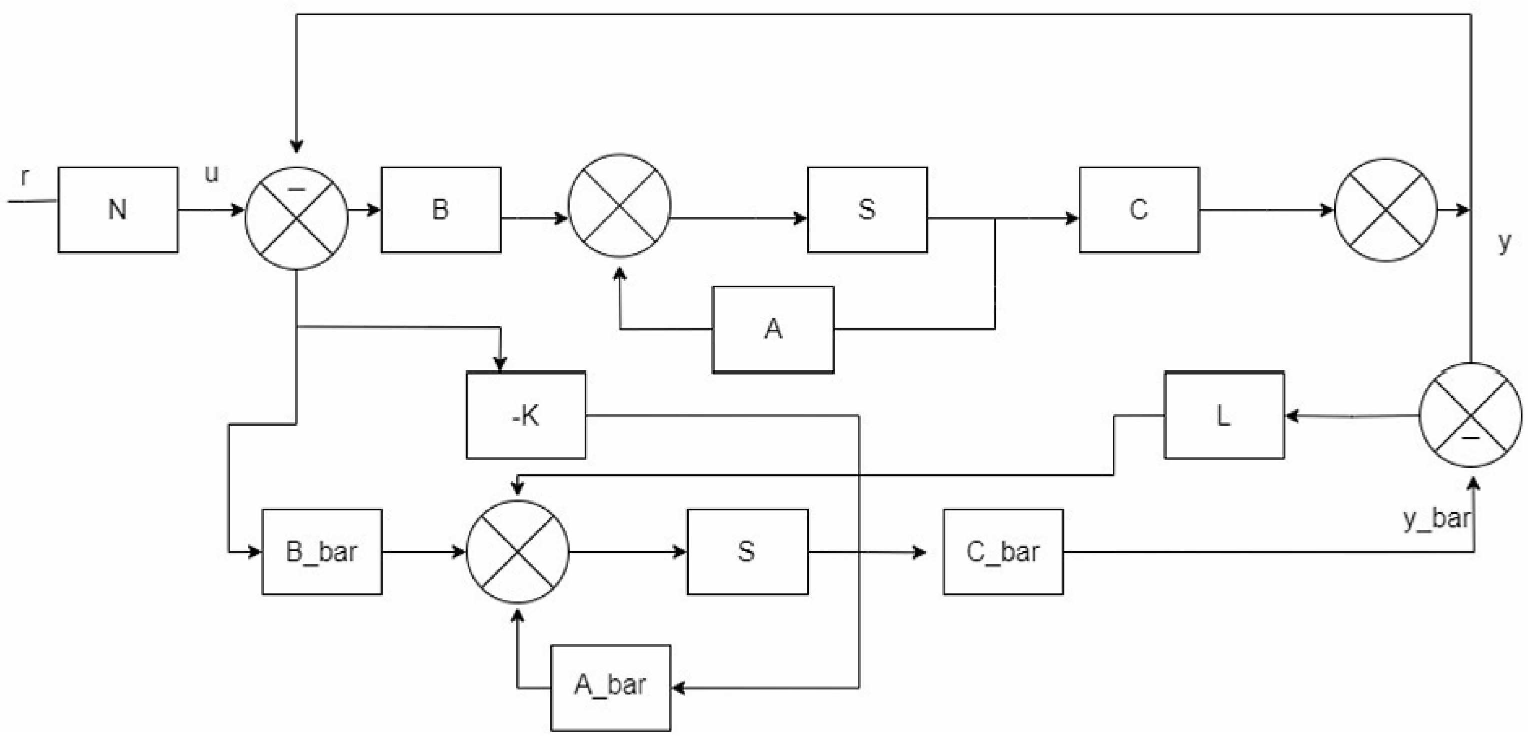
$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Objective of Controller

- We get external setpoints of X and Y position which are sent to our controller to track
- Setpoints could be generated via Vision, Lidar or MoCap systems

Poles of Original and Controlled system

Block Diagram of Controller



Design of the state-feedback controller

In state-space form, the control input U is designed as:

$$\mathbf{U}(t) = -\mathbf{K} \cdot \mathbf{X}(t) + \mathbf{N} \cdot \mathbf{U}_{\text{ref}}$$

where K is the state feedback gain matrix and U_{ref} is a reference input that helps in tracking a desired trajectory. $\mathbf{A}_{\text{cl}} = \mathbf{A} - \mathbf{B} \cdot \mathbf{K}$ is the closed-loop system matrix.

To design the system for stabilization, we place the eigenvalues of the closed-loop matrix $\mathbf{A}_{\text{cl}}$ in the left half of the complex plane (negative real part), ensuring stability.

Matrix for Reference Signal

The matrix $\mathbf{N}$ used for the reference signal is:

$$\mathbf{N} = \mathbf{C}^{-1} \cdot (\mathbf{A} - \mathbf{B} \cdot \mathbf{K})^{-1} \cdot \mathbf{B}$$

Design of the state-feedback controller

The `place` function is used to compute the state feedback gain K such that the closed-loop poles of $A - BK$ match the desired values. The function adjusts K to achieve the desired eigenvalues of the matrix $A - BK$:

```
k_tracking = place(a, b, [-30, -35, -45, -40, -30.5, -35.5, -45.5, -40.5])
```

$$K_{\text{tracking}} = 10^5 \times \begin{bmatrix} -1.7712 & -0.1906 & 0.0748 & 0.0013 & -0.0013 & -0.0001 & 0.0000 & 0.0000 \\ -0.0011 & -0.0001 & 0.0000 & 0.0000 & -1.6779 & -0.1831 & 0.0729 & 0.0013 \end{bmatrix}$$

Justification for the performance of the controller

- **CONTROLLER GAINS**: THE CONTROLLER GAINS PLACE THE POLES AT SPECIFIC LOCATIONS, INFLUENCING THE SYSTEM'S STABILITY AND RESPONSE SPEED. WE HAVE ARBITRARILY CHOSEN OUR POLES TO BE [-30, -35, -45, -40, -30.5, -35.5, -45.5, -40.5].

- **VELOCITY DEPENDENCE**: THE MAXIMUM VELOCITY OF THE DRONE IS DEPENDENT ON THE POLE PLACEMENT OF THE CONTROLLER AS WELL AS THE INITIAL CONDITIONS.

- **PERFORMANCE METRICS**: WE GET A SETTLING TIME OF ABOUT 0.24s FOR THE SELECTED POLES.

- **ADJUSTMENTS**: DECREASING THE CONTROLLER GAINS LEADS TO FASTER SETTLING TIME BUT INCREASES STEADY STATE ERROR AND GENERATES IMPRACTICAL MOTION DEMANDS .

Design of Luenberger Observer

The observer dynamics are designed as:

$$\dot{\hat{X}} = A\hat{X} + BU + L(Y - C\hat{X})$$

where $\hat{X}$ is the estimated state vector and L is the observer gain matrix.

$$L_{\text{tracking}} = 10^8 \times \begin{bmatrix} 0.0000 & 0.0001 \\ 0.0003 & 0.0272 \\ -0.0010 & -0.2721 \\ 0.0000 & -8.8830 \\ -0.0000 & 0.0000 \\ -0.0000 & 0.0009 \\ 0.0000 & -0.0066 \\ 0.0010 & -0.1851 \end{bmatrix}$$

- Our error term is as follows:

$$\frac{de}{dt} = \frac{d(X(t) - X_e(t))}{dt} = (A - L \cdot C) \cdot e$$

To ensure that the error term decays to zero asymptotically, we have to place the eigenvalues of the matrix $(A - L \cdot C)$ in the open left half-plane.

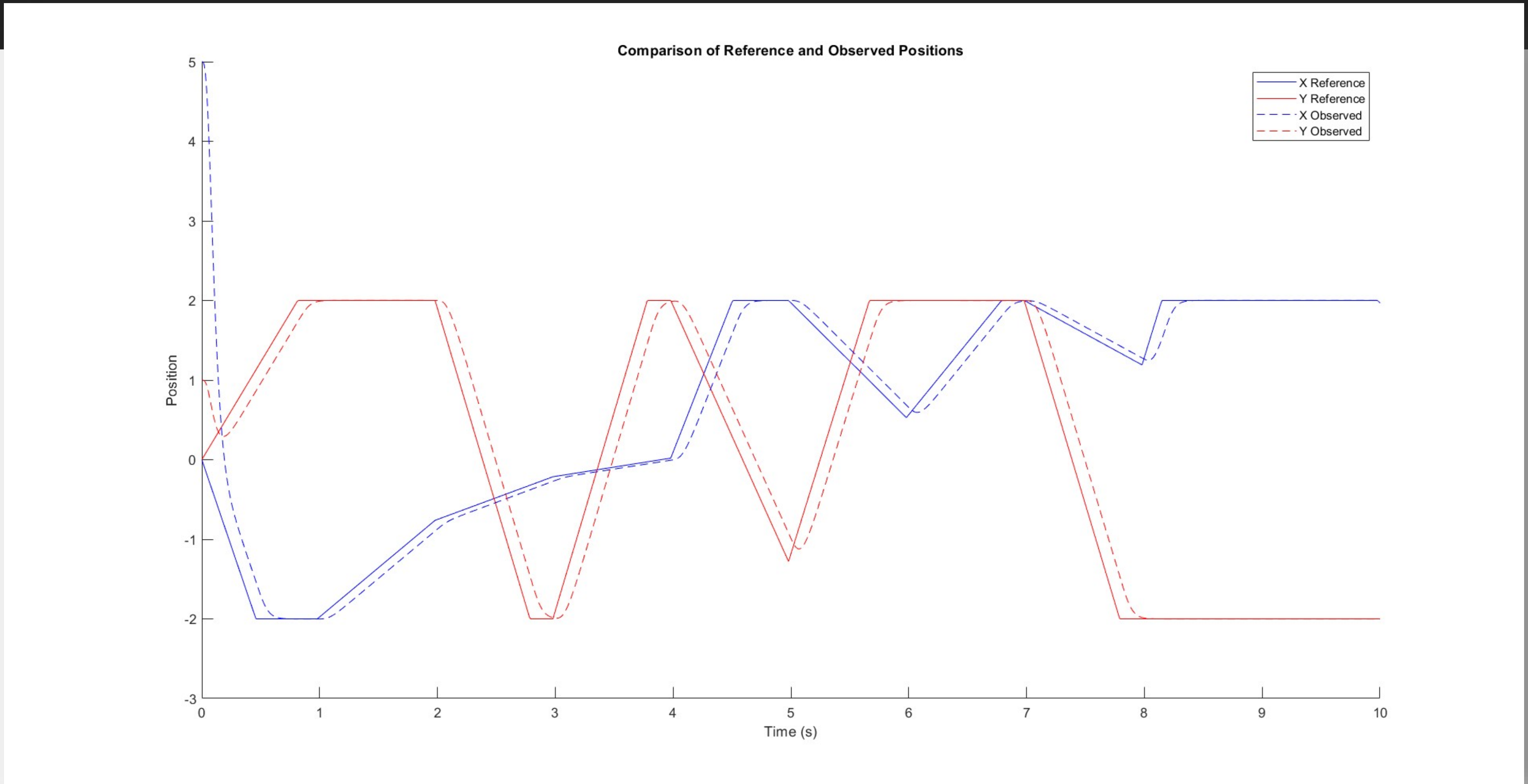
$$\mathrm{l}_{\text{tracking}} = \text{place}(A^T, C^T, [-105, -100, -90, -95, -105.5, -100.5, -90.5, -95.5])^T$$

Justification for the performance of observer

- **OBSERVER GAINS**: WE PLACED THE POLES OF L_TRACKING AT [-105, -100, -90, -95, -105.5, -100.5, -90.5, -95.5] TO ACHIEVE FAST CONVERGENCE, WHILE ALSO MAINTAINING THE ROBUSTNESS OF THE OBSERVER.
- **POLE PLACEMENT**: FASTER CONVERGENCE OF THE OBSERVER REDUCES THE TIME IT TAKES TO ESTIMATE THE ACTUAL STATE.
- **STATE ESTIMATION ACCURACY**: THE OBSERVER IS DESIGNED TO PROVIDE ACCURATE ESTIMATES OF THE UNMEASURED STATES OF THE SYSTEM.
- **ROBUSTNESS TO DISTURBANCES**: A WELL-PERFORMING OBSERVER SHOULD EXHIBIT FAST CONVERGENCE, LOW STEADY-STATE ERROR, AND ROBUSTNESS TO DISTURBANCES.
- **TUNING ADJUSTMENTS**: PLACING POLES FURTHER FROM THE ORIGIN (NEGATIVE) ENHANCES THE CONVERGENCE, OBSERVER GAINS CAN BE ADJUSTED TO BALANCE THE TRADE-OFF BETWEEN CONVERGENCE SPEED AND ROBUSTNESS

PLOTS

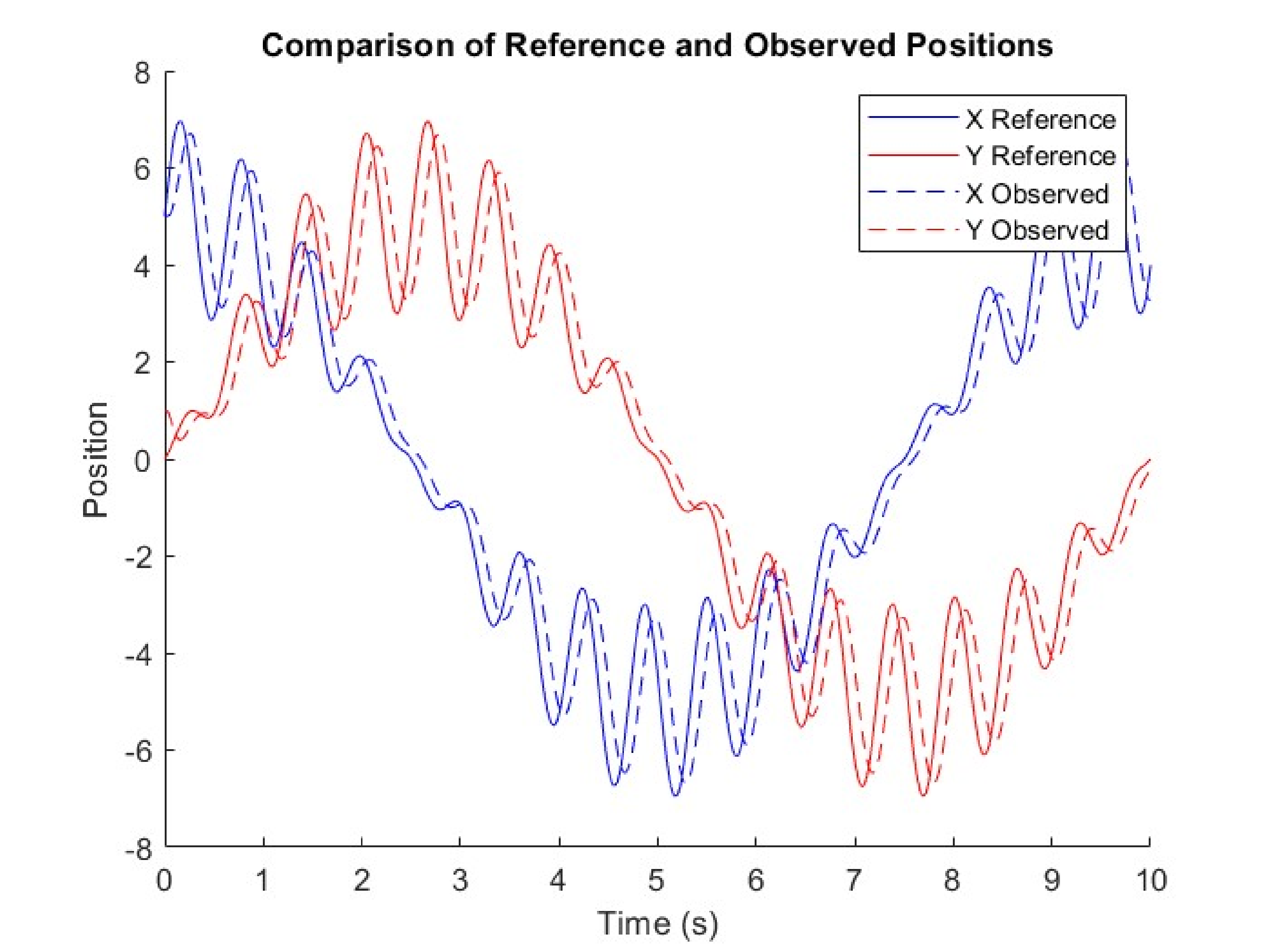
RANDOM PATH TRACKING*



*RUN CODE (PLOT NO:3) TO GENERATE LIVE SIM(AND MP4 video)

Simulation plots

Sinusoidal circle*



*RUN CODE (PLOT NO:2) TO GENERATE LIVE SIM(AND MP4 video)

PRACTICAL UTILISATIONS

- SHIP DECK LANDING OF A UAV
- WAREHOUSE ROBOT NAVIGATION
- DRONE SURVEILLANCE
- CROP SPRAYING FROM DRONES
- LOGISTICS AND DELIVERY DRONE ROUTING WITH ADAPTIVE PATHS

THANK YOU

QnA