

Target group: M2SAAS - Universite Evry Paris Saclay

Flight Modeling Lab Final Report

Prepared by: Melkamu Amare ==> ID: 20245847

Abel Yehuala ==> ID: 20245853

M2SAAS - 2024

Submitted to: Mr. Amayas Benoudiba

Universite Evry Paris-Saclay

Table of Contents

Objectives.....	1
Introduction theory about the plant.....	2
Questions.....	4
a. Analyze the provided Simulink model	4
b. Compute the equilibrium points of the system and show that the origin is an equilibrium point.....	5
1. Linear longitudinal and lateral control.....	6
1. Compute the linearized models around the origin of each model ?	7
2. For each motion, compute a linear controller by state feedback control and pole placement technique.....	7
3. Apply the proposed control techniques directly on the nonlinear model. Discuss the results.....	10
4. Modify the control input in order to track :	13
(a) a constant trajectory	13
(b) a time varying trajectory.....	15
State feedback with LQR method.....	17
2. Nonlinear longitudinal and lateral control.....	24
1. For each motion, compute a nonlinear controller by state feedback control with exact linearization approach.....	24
2. Apply the proposed control techniques directly on the nonlinear model. Discuss the results	27
3. Modify the control input in order to track:	29
4. Compare the results with the results obtained by the linear controller.....	33
3. Nonlinear Vertical control.....	33
4. Autonomous control of the complete system	38
Summary.....	41
Appendix.....	42

Objectives

Objectives of this lab:

- Familiarization with modeling and simulation with Matlab/Simulink — Linear control systems : linearization and stabilization
- Design an autopilot for a drone using the "exact linearization" approach and other nonlinear approaches.
- Stabilization and trajectory tracking control. Consider a drone (rocket) having airships at angles in two perpendicular directions.

Introduction theory about the plant

The nonlinear model of the drone is expressed as a state representation form with 12 variables that correspond to positions, velocities, angular positions and angular velocities, which will be controlled via 4 control inputs. The 4 controls are respectively : the thrust T, the 2 angles of rotation α and β , the roll control input $M^*\psi$.

By choosing the following state variables :

$$x = \begin{pmatrix} x \\ u \\ y \\ v \\ z \\ \omega \\ \phi \\ p \\ \theta \\ q \\ \psi \\ r \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{pmatrix} \Rightarrow \dot{x} = \begin{pmatrix} \dot{x} \\ \dot{u} \\ \dot{y} \\ \dot{v} \\ \dot{z} \\ \dot{\omega} \\ \dot{\phi} \\ \dot{p} \\ \dot{\theta} \\ \dot{q} \\ \dot{\psi} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{pmatrix} = \begin{pmatrix} x_2 \\ \dot{x}_2 \\ x_4 \\ \dot{x}_4 \\ x_6 \\ \dot{x}_6 \\ x_8 \\ \dot{x}_8 \\ x_{10} \\ \dot{x}_{10} \\ x_{12} \\ \dot{x}_{12} \end{pmatrix}$$

Then we obtain the state representation of our system, we replace these state variables in the equations of motion we get :

$$\left\{ \begin{array}{l} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_{12}x_4 - x_{10}x_6 - g \sin x_9 - \frac{T}{m} \cos \beta \sin \alpha \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = x_8x_6 - x_{12}x_2 + g \sin x_7 \cos x_9 + \frac{T}{m} \sin \beta \\ \dot{x}_5 = x_6 \\ \dot{x}_6 = x_{10}x_2 - x_8x_4 + g \cos x_7 \cos x_9 - \frac{T}{m} \cos \beta \cos \alpha \\ \dot{x}_7 = x_8 \\ \dot{x}_8 = \frac{I_{yy}-I_{zz}}{I_{xx}}x_{12}x_{10} + \frac{1}{I_{xx}}(-T \sin \beta(AB \cos \alpha + GA)) \\ \dot{x}_9 = x_{10} \\ \dot{x}_{10} = \frac{I_{zz}-I_{xx}}{I_{yy}}x_{12}x_8 + \frac{1}{I_{yy}}(-GA.T \cos \beta \sin \alpha) \\ \dot{x}_{11} = x_{12} \\ \dot{x}_{12} = \frac{I_{xx}-I_{yy}}{I_{zz}}x_8x_{10} + \frac{1}{I_{zz}}(AB.T \sin \beta \sin \alpha) + \frac{1}{I_{zz}}M_\psi \end{array} \right.$$

From the equations(1), we define the control input vector \mathbf{u} by :

$$\left\{ \begin{array}{l} u_1 = -\frac{T}{m} \cos \beta \sin \alpha \\ u_2 = \frac{T}{m} \sin \beta \\ u_3 = -\frac{T}{m} \cos \beta \cos \alpha \\ u_4 = \frac{1}{I_{zz}}(AB.T \sin \beta \sin \alpha) \\ u_5 = \frac{1}{I_{zz}}M_\psi \end{array} \right. \quad (2)$$

For small rotation angles :

$$-10^\circ \leq \alpha \leq 10^\circ \quad (3)$$

$$-10^\circ \leq \beta \leq 10^\circ \quad (4)$$

It is possible to use the approximations :

$$\begin{aligned} \cos \alpha &= \cos \beta = 1 \\ \sin \alpha &= \alpha \\ \sin \beta &= \beta \end{aligned}$$

The, the control inputs u_i , $i = 1, \dots, 5$ are simplified as follows :

$$\left\{ \begin{array}{l} u_1 = -\frac{T}{m}\alpha \\ u_2 = \frac{T}{m}\beta \\ u_3 = -\frac{T}{m} \\ u_4 = \frac{T \cdot AB}{I_{zz}}\alpha\beta \\ u_5 = \frac{1}{I_{zz}}M_\psi \end{array} \right. \quad (5)$$

which gives the following state space representation :

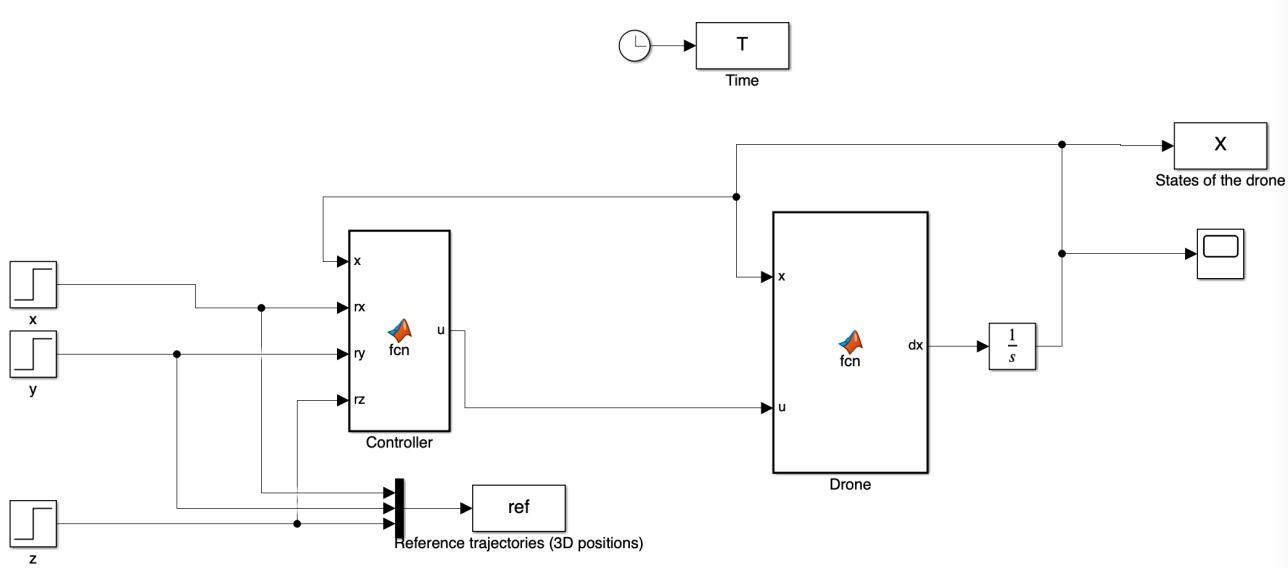
$$\left\{ \begin{array}{l} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_{12}x_4 - x_{10}x_6 - g \sin x_9 + u_1 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = x_8x_6 - x_{12}x_2 + g \sin x_7 \cos x_9 + u_2 \\ \dot{x}_5 = x_6 \\ \dot{x}_6 = x_{10}x_2 - x_8x_4 + g \cos x_7 \cos x_9 + u_3 \\ \dot{x}_7 = x_8 \\ \dot{x}_8 = a_1x_{12}x_{10} + b_1u_2 \\ \dot{x}_9 = x_{10} \\ \dot{x}_{10} = a_2x_{12}x_8 + b_2u_1 \\ \dot{x}_{11} = x_{12} \\ \dot{x}_{12} = a_3x_8x_{10} + u_4 + u_5 \end{array} \right. \quad (6)$$

with

$$\left\{ \begin{array}{l} a_1 = \frac{I_{yy}-I_{zz}}{I_{xx}} \\ a_2 = \frac{I_{zz}-I_{xx}}{I_{yy}} \quad et \quad \left\{ \begin{array}{l} b_1 = \frac{m}{I_{xx}}(AB+GA) \\ b_2 = -\frac{m}{I_{yy}}GA \end{array} \right. \\ a_3 = \frac{I_{xx}-I_{yy}}{I_{zz}} \end{array} \right. \quad (7)$$

Questions

a. Analyze the provided Simulink model



The given Simulink model:

Components of the Simulink model: reference input, controller, plant(Drone),

The provided drone model is a 6-degree-of-freedom (6DoF) model featuring 12 states, where each pair of states represents a two-block derivative of a single degree of freedom. The state space vector, denoted as X , captures the essential states of the Unmanned Aerial Vehicle (UAV), and it can be further articulated in terms of linear and angular accelerations.

$$X = [x, u, y, v, z, w, \dot{\phi}, \dot{p}, \dot{\theta}, \dot{q}, \dot{\psi}, \dot{r}]^T$$

The control input of the quadcopter model, represented by u , consists of four components which are u_1 = total thrust input, u_2 = roll control input, u_3 = pitch control input, and u_4 = yaw control input.

The model also exhibits strong non-linearity due to the interdependence among states, involving functions such as "sin" and "cos." The nonlinear UAV model can be expressed as a set of differential equations that govern the dynamics of the system.

b. Compute the equilibrium points of the system and show that the origin is an equilibrium point.

to get the equilibrium point lets start from

$$\dot{x} = 0$$

$$\begin{aligned}
0 &= x_2 \\
0 &= x_{12}x_4 - x_{10}x_6 - g \sin x_9 + u_1 \\
0 &= x_4 \\
0 &= x_8x_6 - x_{12}x_2 + g \sin x_7 \cos x_9 + u_2 & u_1 &= g \sin x_9, \\
0 &= x_6 & u_2 &= -g \sin x_7 \cos x_9, \\
0 &= x_{10}x_2 - x_8x_4 + g \cos x_7 \cos x_9 + u_3 & u_3 &= -g \cos x_7 \cos x_9, \\
0 &= x_8 & \implies & b_1u_2 = 0, \\
0 &= a_1x_{12}x_{10} + b_1u_2 & b_2u_1 = 0, \\
0 &= x_{10} & u_4 + u_5 = 0. \\
0 &= a_2x_{12}x_8 + b_2u_1 \\
0 &= x_{12} \\
0 &= a_3x_8x_{10} + u_4 + u_5
\end{aligned}$$

Since g , b_1 and b_2 are different from zero, u_1 and u_2 are zero at equilibrium point.

$$\begin{aligned}
g \sin x_9 &= 0, & \sin x_9 &= 0, \text{ iff, } x_9 = k\pi, \text{ where, } k = 0, 1, 2, \dots \\
\text{Therefore, } g \sin x_7 \cos x_9 &= 0, & \sin x_7 \cos x_9 &= 0, \text{ iff, } x_9 = (2k+1)\pi/2 (\text{or}) x_7 = k\pi, \text{ where, } k = 0, 1, 2, \dots \\
g \cos x_7 \cos x_9 &= 0, & \cos x_7 \cos x_9 &= 0, \text{ iff, } x_7 = x_9 = (2k+1)\pi/2, \text{ where, } k = 0, 1, 2, \dots
\end{aligned}
\implies$$

From this condition, we can say that u_3 must be different from zero and $u_4 = -u_5$ to make our plant stable at the origin. For u_3 , if x_7 and x_9 are taken to be zero, there **cos** will be 1.

So, if the above condition is satisfied, the equilibrium point will be

$$[x, u] = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, u_3, u_4, -u_5]$$

1. Linear longitudinal and lateral control

After decoupling (under some conditions which will be explained), the decoupled, lateral and longitudinal motions of the drone are given by the dynamical models:

$$Long \left\{ \begin{array}{l} \dot{x}_1 = x_2 \\ \dot{x}_2 = -g \sin(x_9) + u_1 \\ \dot{x}_9 = x_{10} \\ \dot{x}_{10} = b_2 u_1 \end{array} \right.$$

where x_1 et x_2 represent the position and the velocity with respect to the x-axis and x_9 and x_{10} the roll angle and its rate. u_1 is the thrust generated by the wings with respect to x-axis.

$$Lat \left\{ \begin{array}{l} \dot{x}_3 = x_4 \\ \dot{x}_4 = g \sin(x_7) + u_2 \\ \dot{x}_7 = x_8 \\ \dot{x}_8 = b_1 u_2 \end{array} \right.$$

where x_3 and x_4 represent the position and the velocity with respect to the y-axis and x_7 and x_8 the pitch angle and its rate. u_2 is the thrust generated by the wings with respect to y-axis.

1. Compute the linearized models around the origin of each model ?

Longitudinal linearization at the origin

$$\begin{aligned} \frac{\partial \dot{x}_1}{\partial x_1} &= 0, & \frac{\partial \dot{x}_1}{\partial x_2} &= 1, & \frac{\partial \dot{x}_1}{\partial x_9} &= 0, & \frac{\partial \dot{x}_1}{\partial x_{10}} &= 0, \\ \frac{\partial \dot{x}_2}{\partial x_1} &= 0, & \frac{\partial \dot{x}_2}{\partial x_2} &= 0, & \frac{\partial \dot{x}_2}{\partial x_9} \Big|_{x_9=0} &= -g, & \frac{\partial \dot{x}_2}{\partial x_{10}} &= 0, & A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}; B = \begin{bmatrix} 0 \\ 1 \\ 0 \\ b_2 \end{bmatrix} \\ \frac{\partial \dot{x}_9}{\partial x_1} &= 0, & \frac{\partial \dot{x}_9}{\partial x_2} &= 0, & \frac{\partial \dot{x}_9}{\partial x_9} &= 0, & \frac{\partial \dot{x}_9}{\partial x_{10}} &= 1 \\ \frac{\partial \dot{x}_{10}}{\partial x_1} &= 0, & \frac{\partial \dot{x}_{10}}{\partial x_2} &= 0, & \frac{\partial \dot{x}_{10}}{\partial x_9} &= 0, & \frac{\partial \dot{x}_{10}}{\partial x_{10}} &= 0 \end{aligned}$$

Lateral linearization at the origin

$$\begin{aligned} \frac{\partial \dot{x}_3}{\partial x_3} &= 0, & \frac{\partial \dot{x}_3}{\partial x_4} &= 1, & \frac{\partial \dot{x}_3}{\partial x_7} &= 0, & \frac{\partial \dot{x}_3}{\partial x_8} &= 0, \\ \frac{\partial \dot{x}_4}{\partial x_3} &= 0, & \frac{\partial \dot{x}_4}{\partial x_4} &= 0, & \frac{\partial \dot{x}_4}{\partial x_9} \Big|_{x_7=0} &= g, & \frac{\partial \dot{x}_4}{\partial x_8} &= 0, & A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}; B = \begin{bmatrix} 0 \\ 1 \\ 0 \\ b_1 \end{bmatrix} \\ \frac{\partial \dot{x}_7}{\partial x_3} &= 0, & \frac{\partial \dot{x}_7}{\partial x_4} &= 0, & \frac{\partial \dot{x}_7}{\partial x_7} &= 0, & \frac{\partial \dot{x}_7}{\partial x_8} &= 1 \\ \frac{\partial \dot{x}_8}{\partial x_3} &= 0, & \frac{\partial \dot{x}_8}{\partial x_4} &= 0, & \frac{\partial \dot{x}_8}{\partial x_7} &= 0, & \frac{\partial \dot{x}_8}{\partial x_8} &= 0 \end{aligned}$$

2. For each motion, compute a linear controller by state feedback control and pole placement technique.

State feedback - pole placement is a control design technique used to place the poles of a closed-loop system at desired locations in the complex plane by designing a suitable state feedback controller. The goal is to achieve desired system dynamics, such as stability, performance, and responsiveness, by modifying the eigenvalues of the closed-loop system matrix.

Objective of pole placement:

By choosing K, the designer can move the eigenvalues of (A–BK) (the closed-loop poles) to specific locations that correspond to desired dynamics, such as:

- Faster response (poles with larger negative real parts in the left-half plane).
- Oscillatory behavior (complex conjugate poles with controlled damping and frequency).
- Reduced overshoot and settling time.

From state feedback, we have $u = -Kx(t) + r(t)$

$$\begin{aligned}\dot{x} &= Ax + Bu; \text{ where } u = -Kx + v \\ \dot{x} &= (A - BK)x + Bv\end{aligned}$$

We can insure the referencer tracking

$$\begin{aligned}\dot{x} &= Ax + Bu, \quad y = Cx \\ \dot{x} &= (A - BK)x + Bv. \quad \dot{x} = 0 \implies \text{At steady state :} \\ 0 &= (A - BK)x + Bv, \text{ which leads to : } x = -(A - BK)^{-1}Bv.\end{aligned}$$

From the output equation:

$$y = Cx, \text{ substitute for } x \Rightarrow y = C[-(A - BK)^{-1}B]v.$$

To ensure reference tracking ($y = r$):

$$r = [-C(A - BK)^{-1}B]v.$$

Solving for v:

$$v = -[C(A - BK)^{-1}B]^{-1}r.$$

If the matrix is not invertible, we use the pseudo-inverse:

$$v = -\text{pinv}(C(A - BK)^{-1}B)r.$$

Now, we can find the value of K using Ackerman's function in MATLAB by setting a desired pole p in the LHS region for stability

```
clc;
% longitudinal state feedback
g = 9.81; b1 = 1; b2 = 1;
p = [-6, -8, -5, -4]; % desired poles
A = [0 1 0 0; 0 0 -g 0; 0 0 0 1; 0 0 0 0];
B = [0; 1; 0; b2];
K1 = acker(A, B, p)
```

```
K1 = 1×4
-97.8593 -72.5790 291.8593 95.5790
```

```
size(K1)
```

```
ans = 1×2
1 4
```

```
C = eye(4);
A1 = inv(A - (B*K1));
```

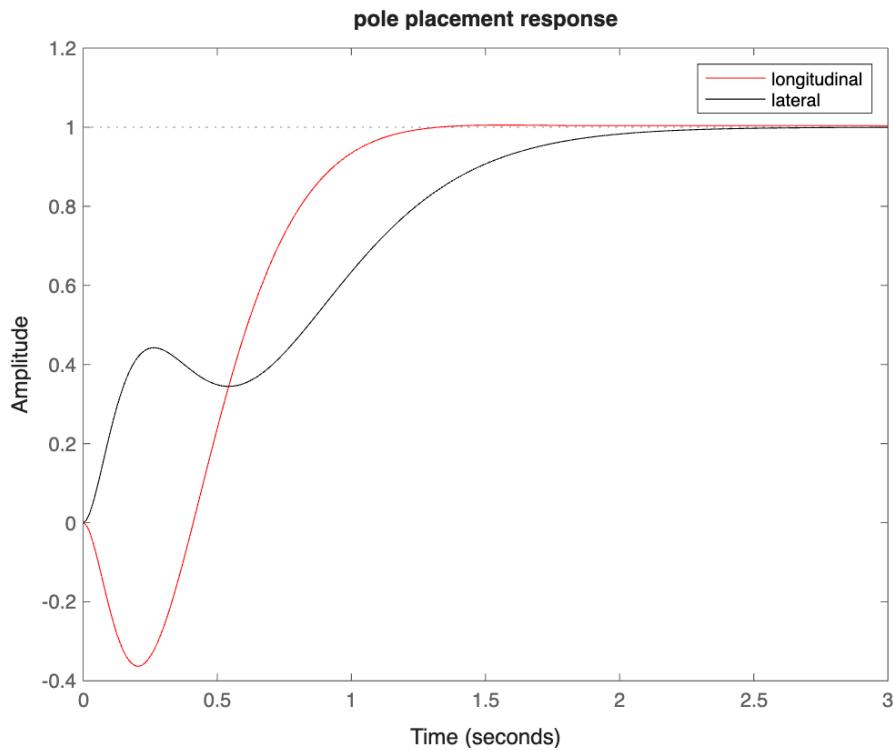
```
V_mat1 = -pinv(C*A1*B);
```

```
% lateral state feedback  
A2 = [0 1 0 0; 0 0 g 0; 0 0 0 1; 0 0 0 0];  
B2 =[0; 1; 0; b1];  
K2 = acker(A2, B2, p)
```

```
K2 = 1x4  
97.8593 72.5790 96.1407 -49.5790
```

```
C2 = eye(4);  
D = 0;  
A21 = inv(A2 - (B2*K2));  
V_mat2 = -pinv(C2*A21*B2);
```

```
% State-Space for Longitudinal with pole placement  
lon_pole = ss(A - B*K1, B, C, D);  
  
% Step Responses for Longitudinal Models figure;  
step(lon_pole(1)/dcgain(lon_pole(1)), 'r-');  
  
hold on;  
  
% State-Space for Longitudinal with pole placement  
lat_pole = ss(A2 - B2*K2, B2, C2, D);  
  
% Step Responses for Longitudinal Models figure;  
step(lat_pole(1)/dcgain(lat_pole(1)), 'k-');  
hold off;  
  
legend('longitudinal', 'lateral');  
title('pole placement response');
```



As we can see the result, the step response of the pole placement method is settled in 1 second for longitudinal and 2 seconds for lateral motion. This is a short period of time for non-critical systems to be stable, so the controller is efficient for constant inputs in the linear state space model. Lets apply it to the nonlinear plant.

3. Apply the proposed control techniques directly on the nonlinear model. Discuss the results.

Since the longitudinal and lateral motions are controlled by u_1 and u_2 respectively, we only play with the two inputs to control the motion.

To apply the linearized controller in to the non-linear plant, lets add the following code snippet into the original controller code (given with the model).

```
xlong = [x(1); x(2); x(9); x(10)];  
u(1) = -K*xlong+V_mat1(1)*rx; % To be completed;  
  
xlat = [x(3); x(4); x(7); x(8)];  
u(2) = -K2*xlat + V_mat2(1)*ry; % To be completed;
```

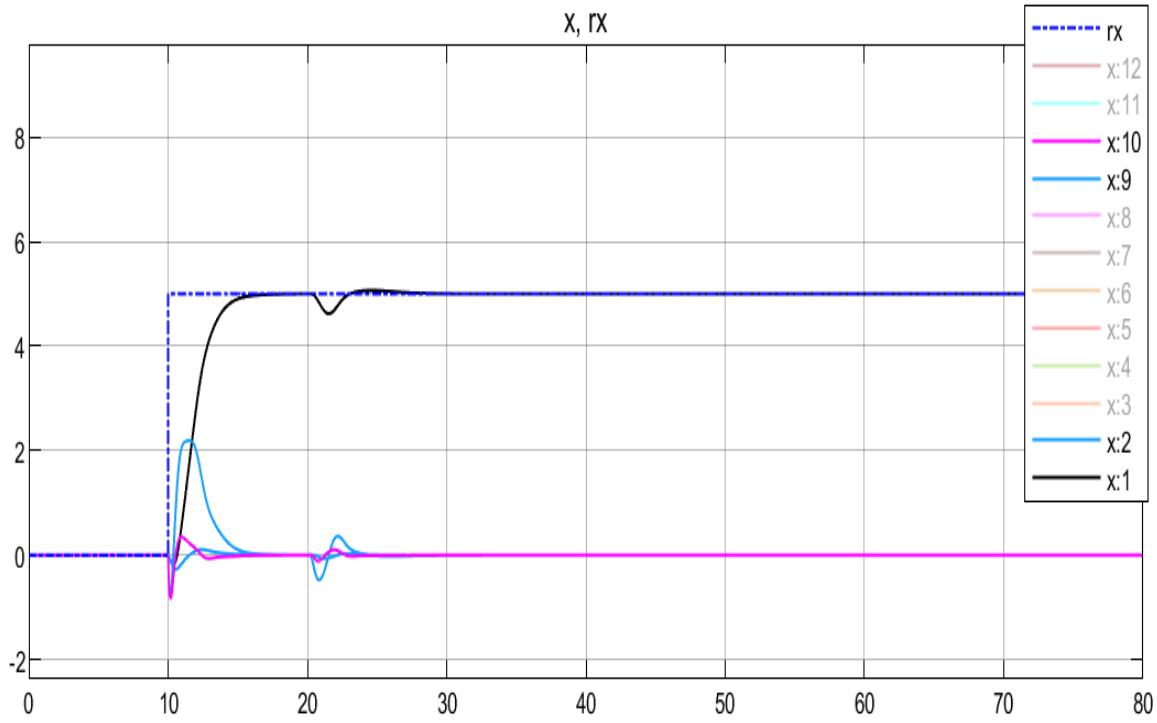


Fig a: Results from the longitudinal(x_1, x_2, x_9, x_{10}) control at constant reference

The dashed line was the reference, and the line tracking the reference was the output of the drone model in the longitudinal direction. From the result, it was observed that the output of the plant tracked the reference input in less than five seconds. The result showed that the controller had good efficiency in controlling the longitudinal motion of the drone. The rise time, the settling time, and the steady-state error showed acceptable results. The drone exhibited stable behavior with good reference tracking at steady state. However, the transient response (overshoot and oscillations) showed some spikes due to extra input in the lateral direction at 20 seconds. It needs some improvement for robustness.

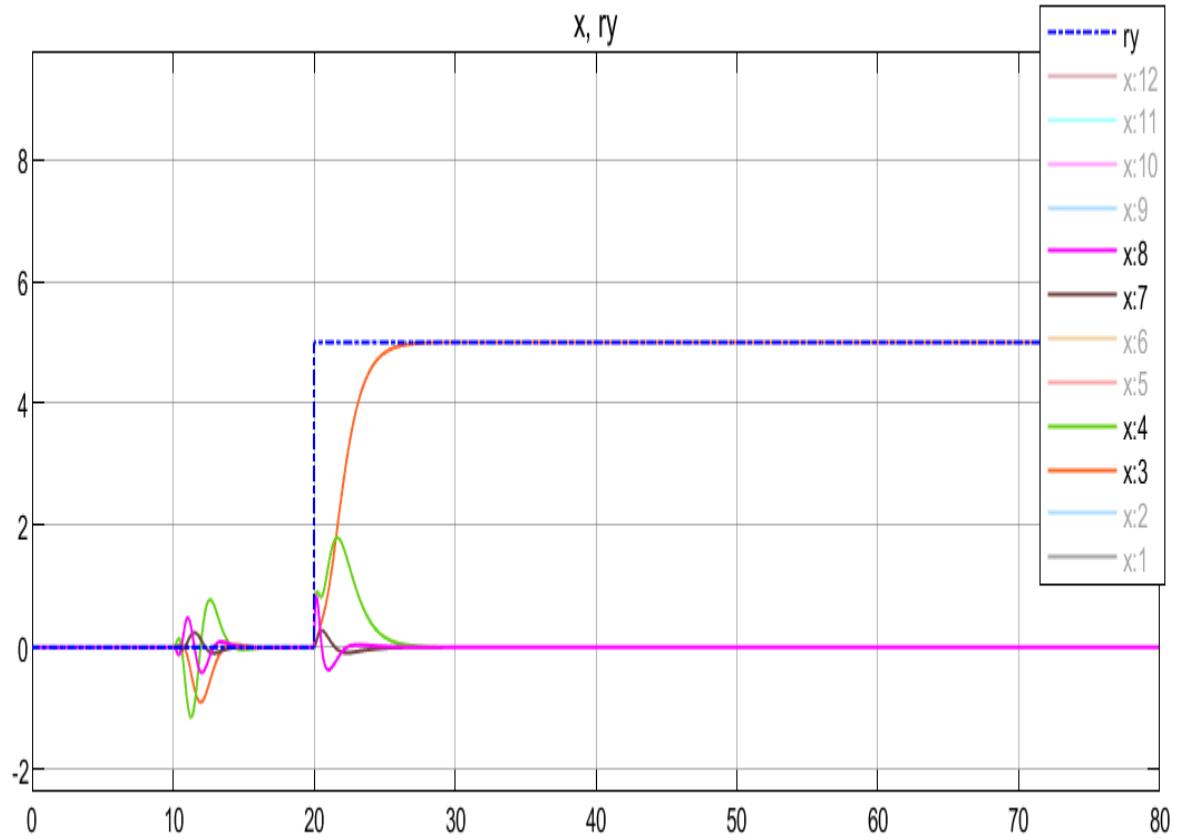


Fig b: Results from the lateral(x_3, x_4, x_7, x_8) control at constant reference

The dashed line is the reference and the line tracking the reference is the output of the drone model in the lateral direction. From the result also, we can see that the output of the plant tracked the reference input in less than five seconds. The result showed that the controller has a good efficiency to control the longitudinal motion of the drone. The rise time, the settling time and the steady state error has shown acceptable results. The drone showed stable behavior with good reference tracking in the longitudinal direction at steady state. However, the transient response (overshoot and oscillations) showed some spikes.

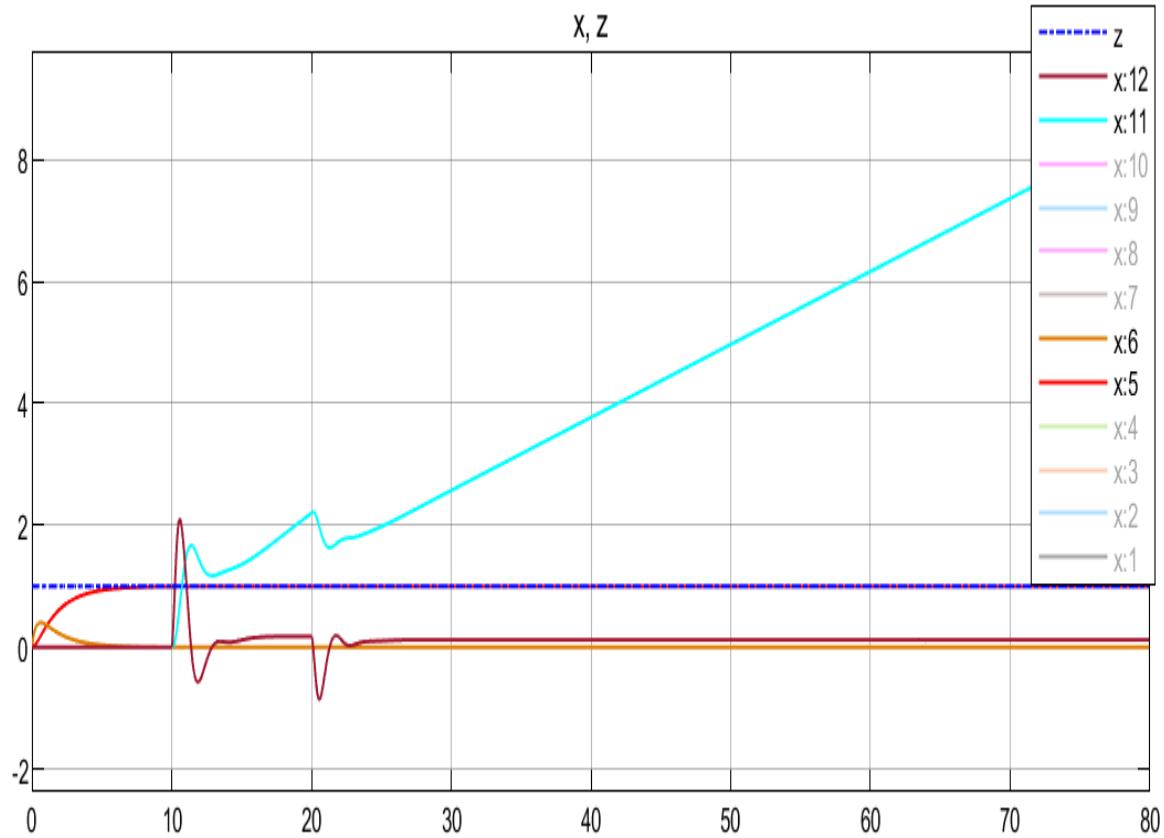


Fig c: Results of the uncontrolled vertical(x_5, x_6, x_{11}, x_{12}) motion

The vertical motion of the drone is not controlled and the rolling rate shows undesirable(unstable result). So, the vertical motion must be controlled to make the whole system stable. However, the vertical position of the drone is stable.

4. Modify the control input in order to track :

(a) a constant trajectory

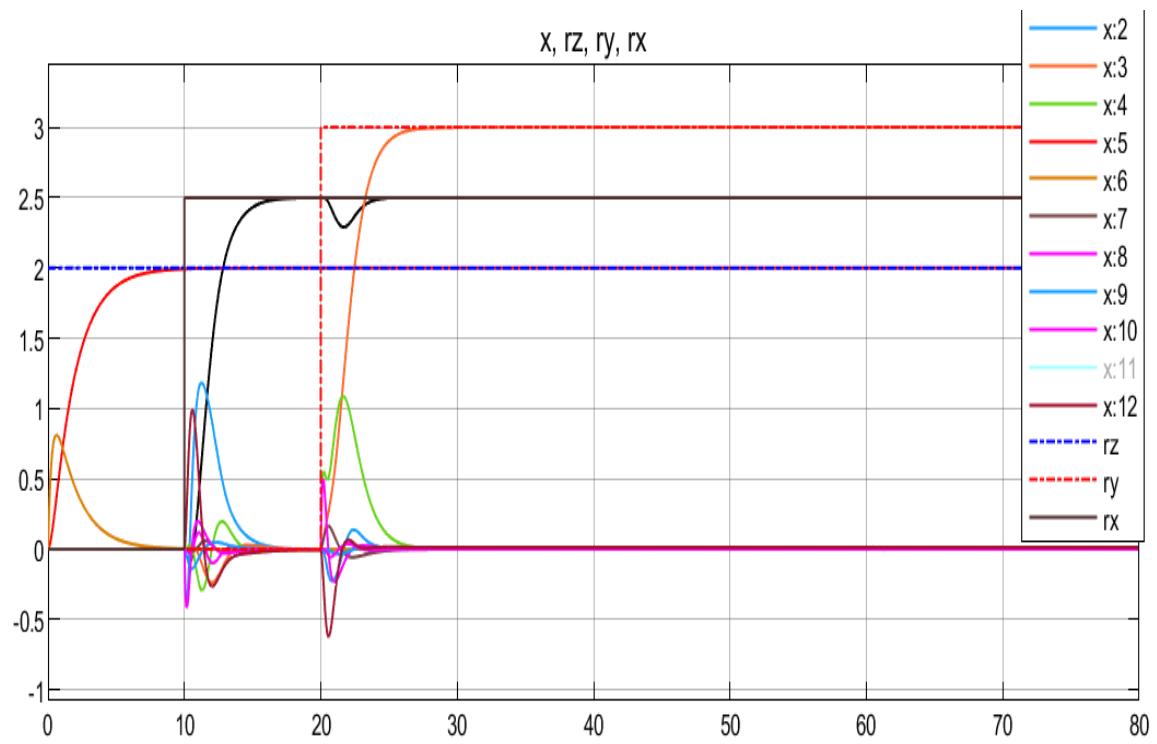


Fig d: The result of the simulation at $rx = 2.5$, $ry = 3$ & $rz = 2$

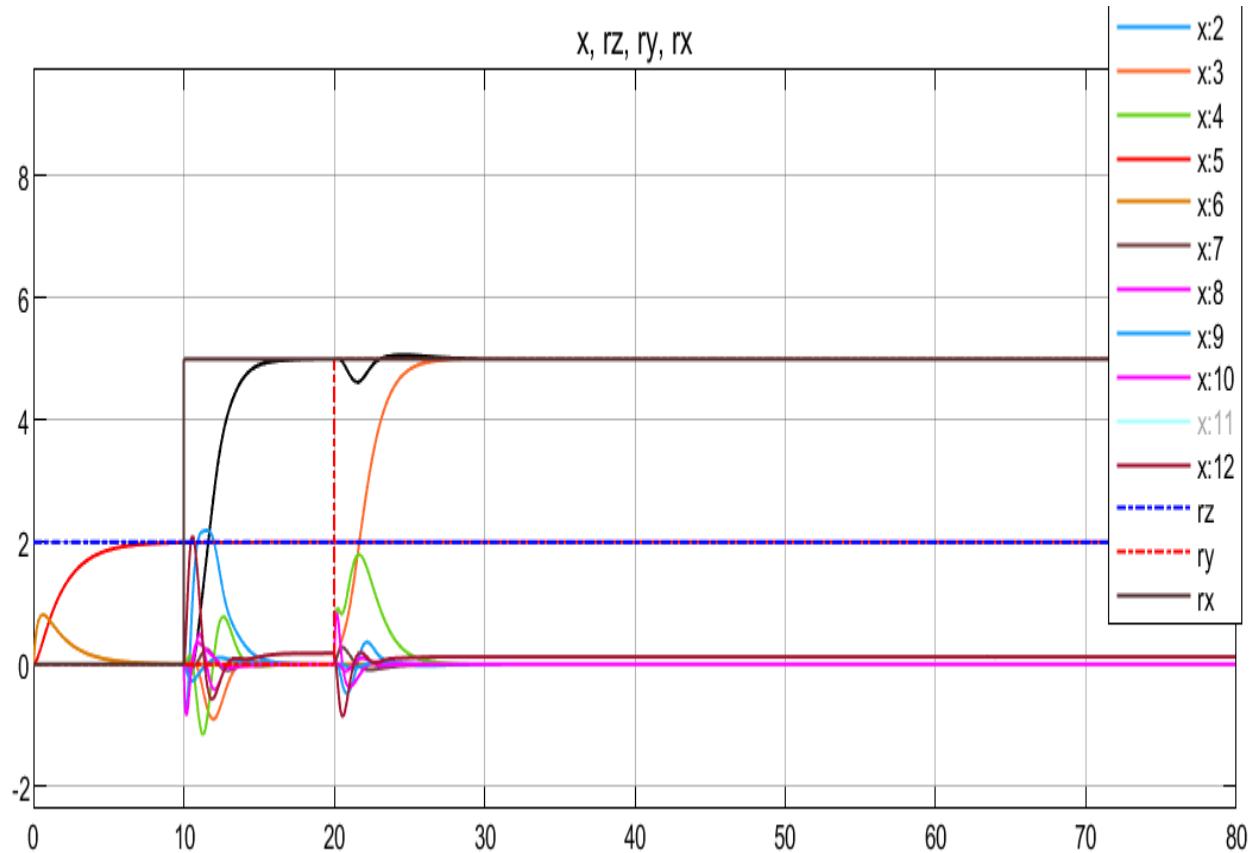


Fig e: The result of the simulation at $rx = 5$, $ry = 5$ & $rz = 2$

From the two results, it could be concluded that the controller had good efficiency at lower reference inputs. When the reference inputs had larger values, there was a higher initial overshoot, but the controller overcame the situation effectively..

(b) a time varying trajectory

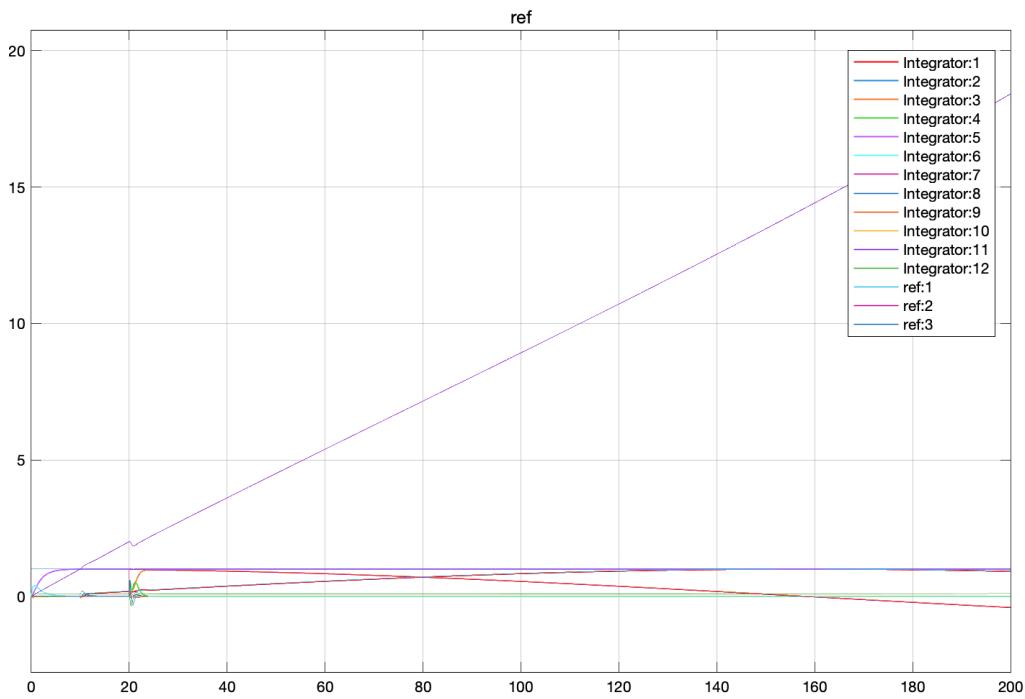


Fig f: The result of the simulation at $rx = \text{step}(10 \text{ sec}) * \sin(\omega t)$, $ry = \text{step}(20 \text{ sec}) * \sin(\omega t)$ & $rz = 0.1$; $\omega_1 = 0.01$ rad/s and $\omega_2 = 0.1$ rad/s

The result showed that the controller managed the stability when the reference inputs are time varying. But as we can see from the second figure, the error between reference and output is visibly higher. In real life, this may cause a destruction in the drone. The efficiency of controllers is measured by their response for sudden changes and the error due to this change. Since this control method has higher errors, we must try another method for the drone control.

State feedback with LQR method

Instead of finding K by pole placement, lets use the LQR method with cost function.

Quadratic Cost Function: The goal of an LQR controller is to minimize a cost function of the form:

$$J = \int_0^{\infty} (x(t)^T Q x(t) + u(t)^T R u(t)) dt$$

- Q is a positive semi-definite matrix that penalizes deviations in the state x(t).
- R is a positive definite matrix that penalizes the control effort u(t).
- The integral computes the cost over time, and the objective is to minimize this cost by choosing the optimal control input u(t).

From Riccati equation of the form:

$$PA + A^T P - PBR^{-1}B^T P + Q = 0$$

We can use the function $[K, P] = lqr(A, B, Q, R)$ to find the value of K; where **P is the solution of the Riccati equation**

Here, we can play on the Q & R values that penalize the state deviation or the control effort. Lets define a simple Q & R values, $R = I$ and $Q = 0.01*I$, where I - is identity matrix

```
clc;
% longitudinal control
Q = 360*eye(4);
R = 30*eye(1);
A = [0 1 0 0; 0 0 -9.81 0; 0 0 0 1; 0 0 0 0];
B = [0; 1; 0; 1];
C = eye(4);
[K1,S,P] = lqr(A, B, Q, R)
```

```
K1 = 1x4
-3.4641   -6.4337   41.6177   16.4490
S = 4x4
10^4 ×
    0.0669   0.0441   -0.1709   -0.0545
    0.0441   0.0645   -0.2630   -0.0838
   -0.1709   -0.2630    1.2320    0.3879
   -0.0545   -0.0838    0.3879    0.1331
P = 4x1 complex
-1.0005 + 0.0000i
-2.8452 + 1.4566i
-2.8452 - 1.4566i
-3.3245 + 0.0000i
```

```
A1 = inv(A - (B*K1));
V_mat1 = -pinv(C*A1*B);
```

```
% lateral control
A2 = [0 1 0 0; 0 0 9.81 0; 0 0 0 1; 0 0 0 0];
```

```
B2 =[0; 1; 0; 1];
C2 = eye(4);
D = 0;
[K2,S,P] = lqr(A2, B2, Q, R)
```

```
K2 = 1x4
    3.4641    5.1527   20.6030    3.3405
S = 4x4
10^3 ×
    0.5355    0.2183    0.3472   -0.1143
    0.2183    0.2893    0.6307   -0.1347
    0.3472    0.6307    3.3860   -0.0126
   -0.1143   -0.1347   -0.0126    0.2349
P = 4x1 complex
-0.9366 + 2.2733i
-0.9366 - 2.2733i
-1.0004 + 0.0000i
-5.6197 + 0.0000i
```

```
A21 = inv(A2 - (B2*K2));
V_mat2 = -pinv(C2*A21*B2);
```

```
%%%%%
% State-Space for Longitudinal with pole placement
lon_pole = ss(A - B*K1, B, C, D);

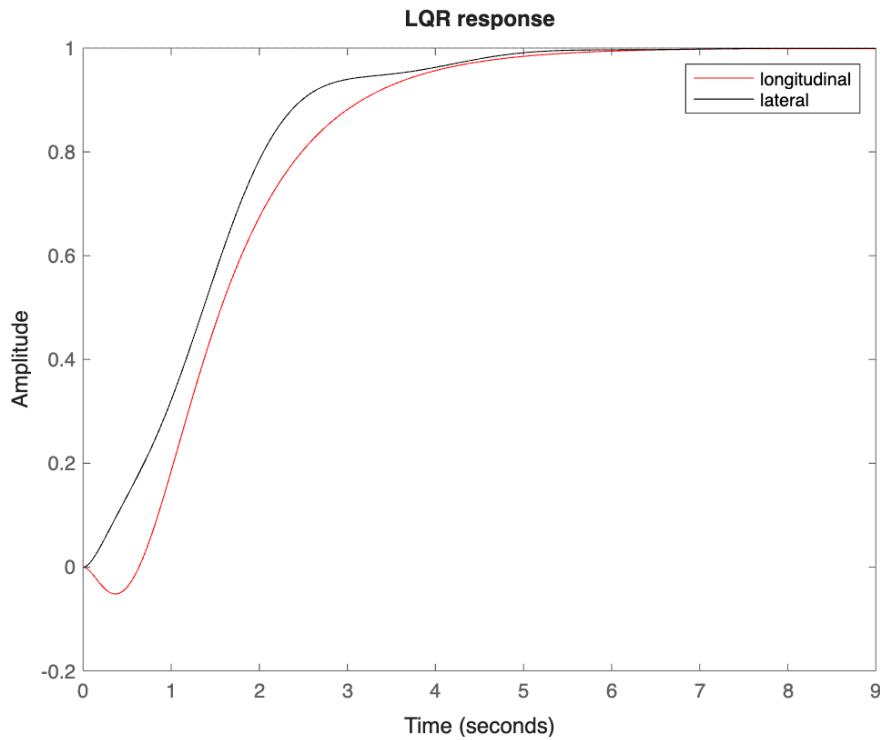
% Step Responses for Longitudinal Models figure;
step(lon_pole(1)/dcgain(lon_pole(1)), 'r-');

hold on;

% State-Space for Longitudinal with pole placement
lat_pole = ss(A2 - B2*K2, B2, C2, D);

% Step Responses for Longitudinal Models figure;
step(lat_pole(1)/dcgain(lat_pole(1)), 'k-');
hold off;

legend('longitudinal', 'lateral');
title('LQR response');
```



As we can see the result, the step response of the LQR method is settled in 5 seconds. Compared to the pole placement, **it took more time to settle**, so the controller **seems less efficient** than the pole placement method in the linear state space model. This discrepancy may be attributed due to the optimization process inherent in LQR, which prioritize certain criteria over achieving an exact match with the chosen pole locations. Lets apply it to the nonlinear plant compare results.

Now, lets try the new K value on the nonlinear model and see the result.

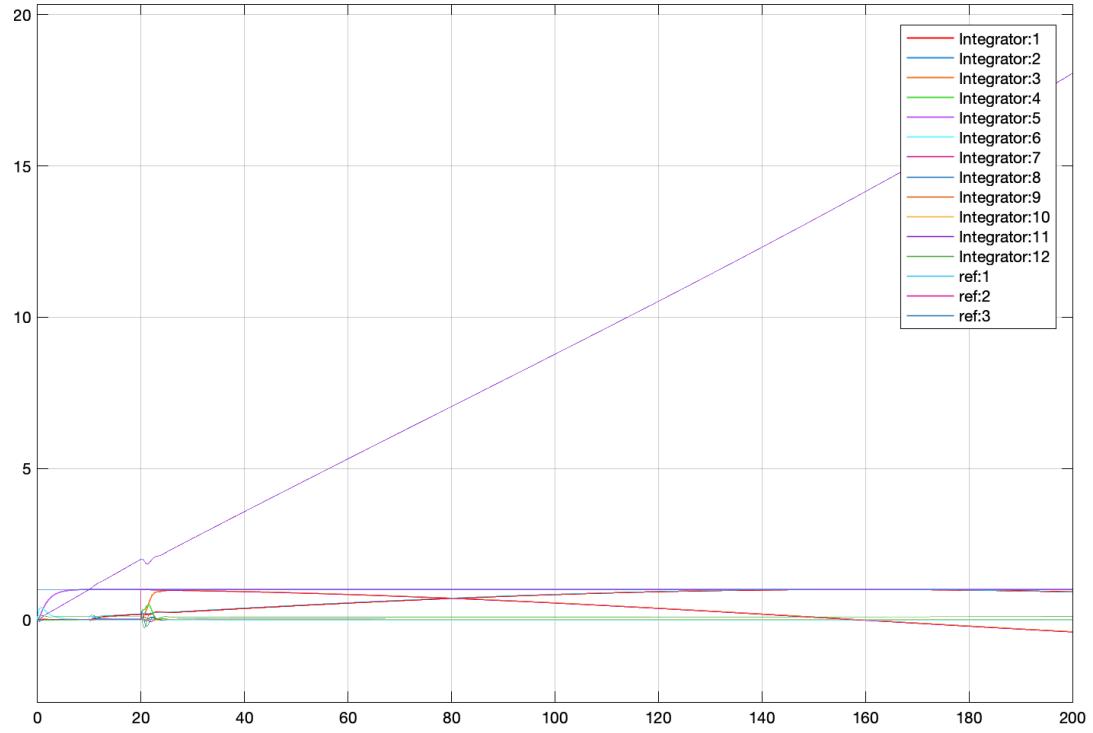
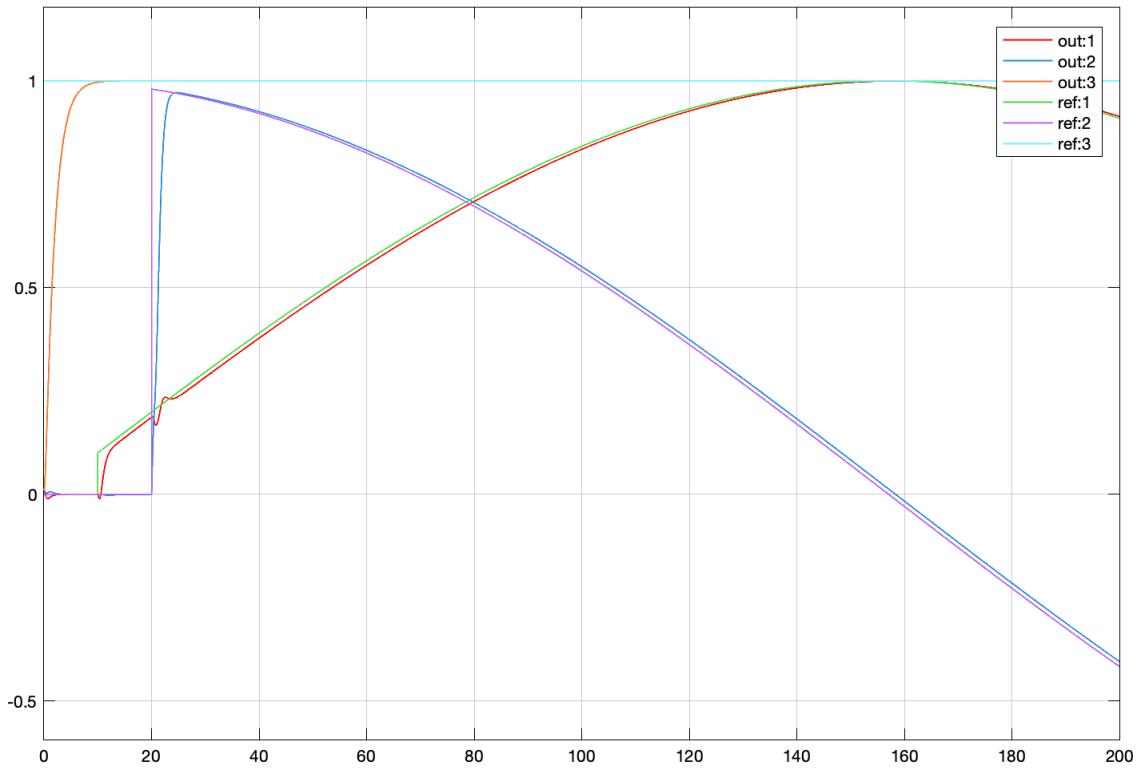


Fig g: The result of the simulation at $rx = \text{step}(10\text{sec}) * \sin(\omega t)$, $ry = \text{step}(20\text{sec}) * \sin(\omega t)$ & $rz = 0.1$; $\omega = 0.01\text{rad/s}$

As we see from the first figure, the output tracked the reference with better efficiency than pole placement. However the yaw component of the drone is unstable like the previous control method.

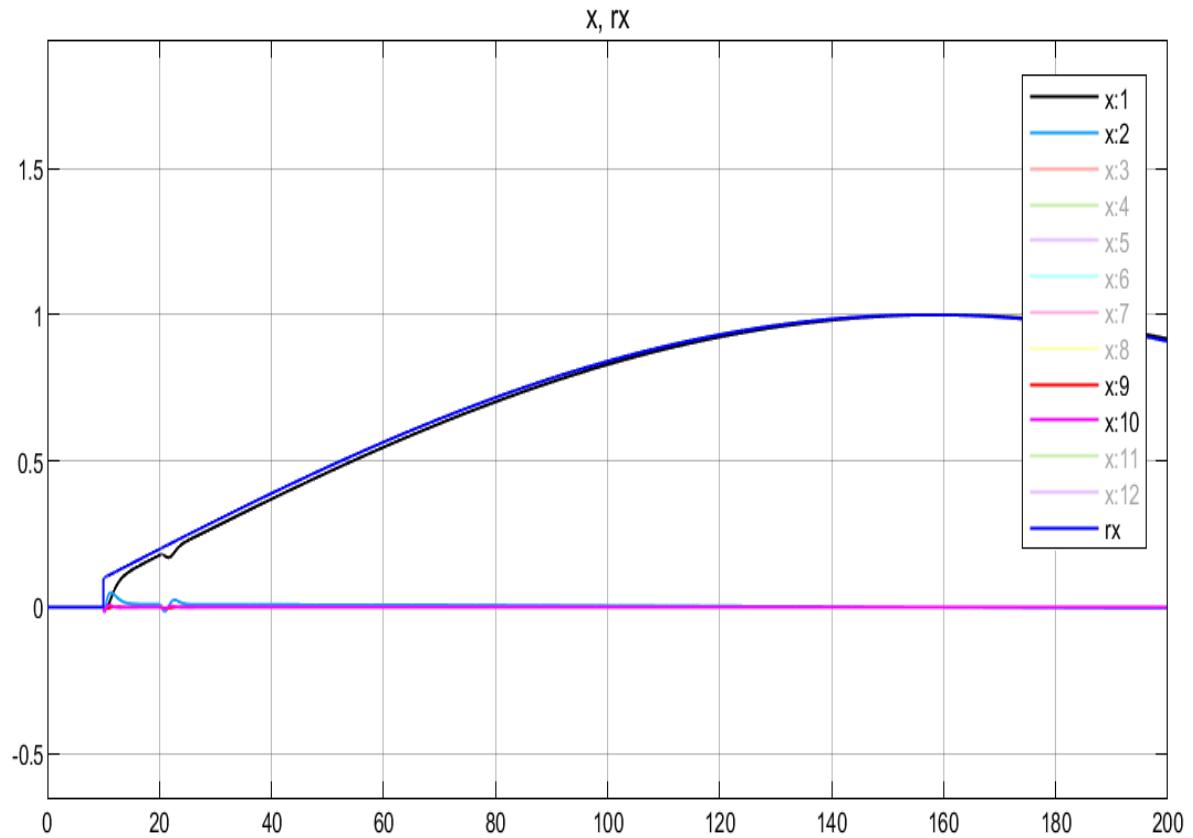


Figure h: Fig a: Results from the longitudinal(x_1, x_2, x_9, x_{10}) control at variable reference

As shown in figure h, the system demonstrates good reference tracking for the longitudinal motion under variable reference. While there is some overshoot and slight oscillations during the transient phase, the controller successfully stabilizes the system and achieves effective tracking of the reference signal.

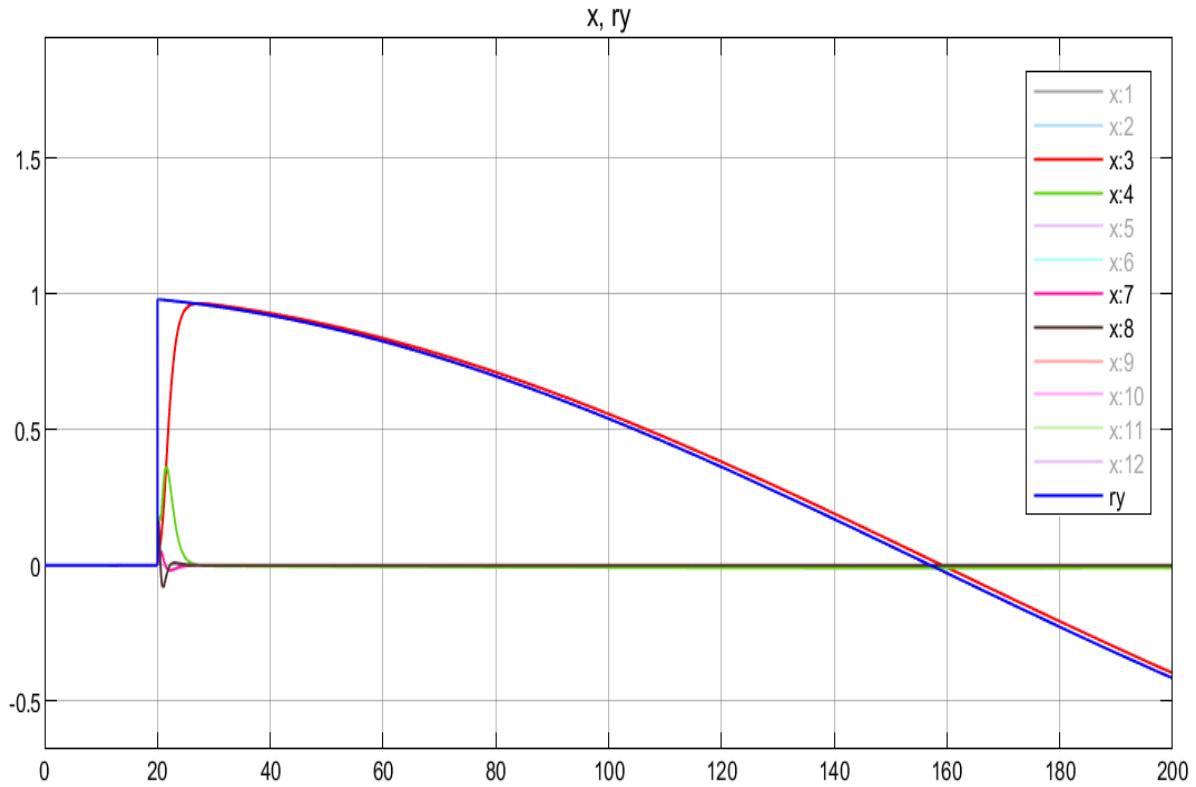


Figure i: Fig a: Results from the lateral $I(x_3, x_4, x_7, x_8)$ control at variable reference

As shown in figure i, the system demonstrates good reference tracking for the lateral motion under variable reference. While there is some overshoot and slight oscillations during the transient phase, the controller successfully stabilizes the system and achieves effective tracking of the reference signal.

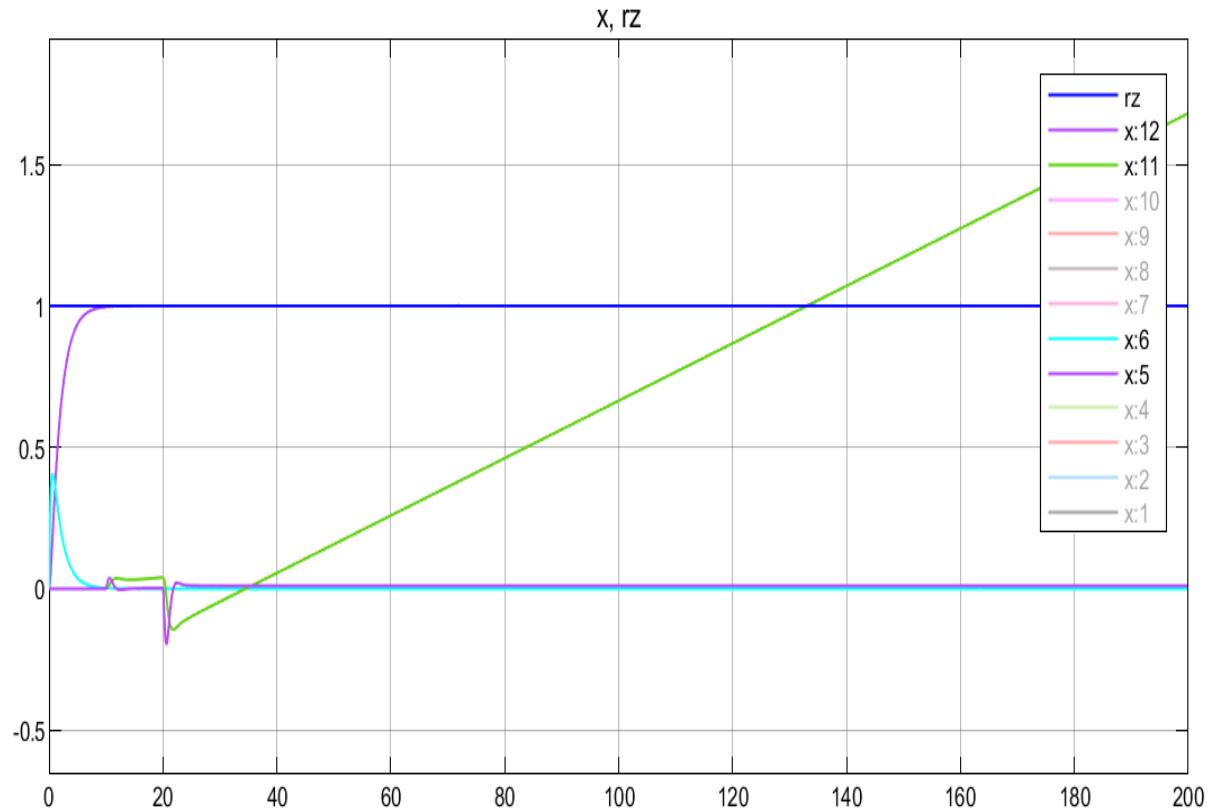


Figure j: Fig a: Results of the uncontrolled vertical(x_5, x_6, x_{11}, x_{12}) motion

As shown in figure , the z-position successfully tracks the reference signal r_z , achieving steady-state stability after an initial transient phase. However, the yaw angle (x_{11}) becomes unstable, showing a continuous increasing . This behavior suggests a lack of control robustness in the yaw direction.

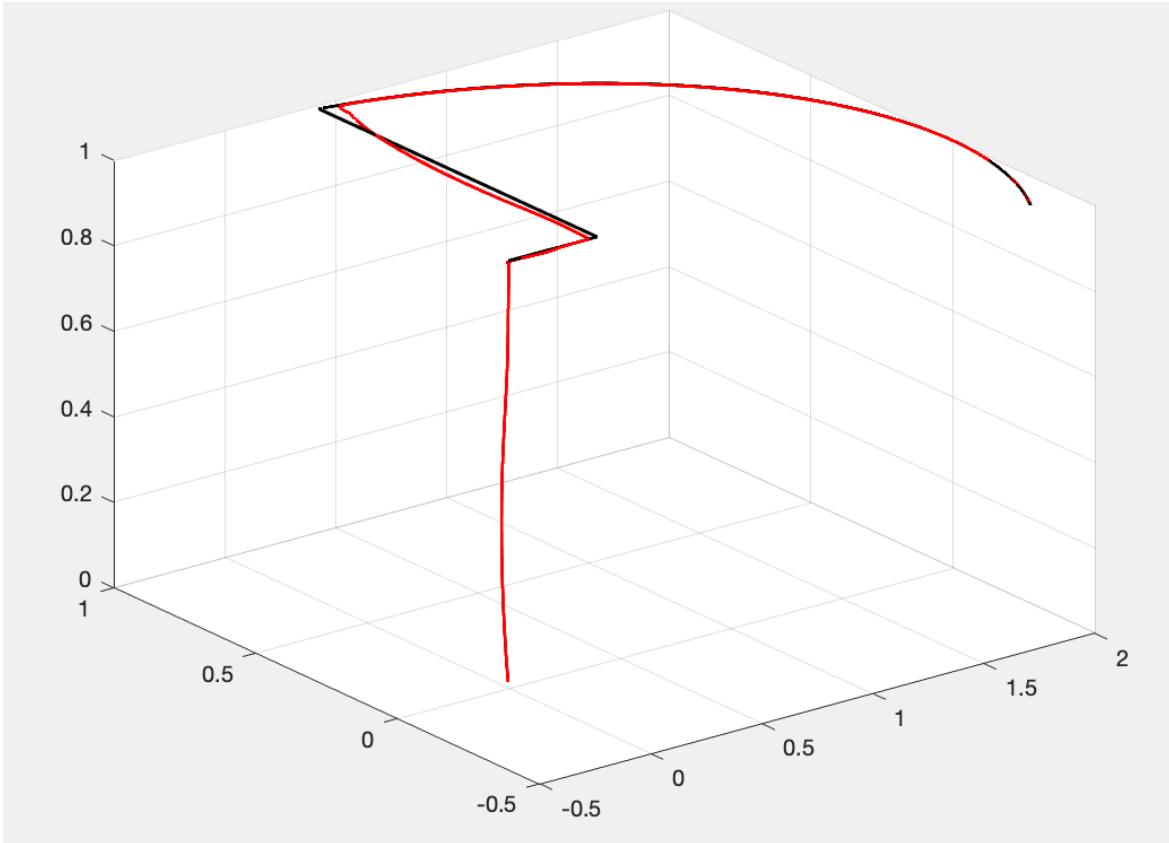


Fig h: 3D plot of the robot position at $w = 0.01\text{rad/s}$ and simulation time 200s (black - reference, red - output)

The 3D plot of the position output of the drone is not a full circle at the top due to low frequency and or short simulation time. And, it didn't track the linear component exactly, while it tracked the angular motion efficiently.

2. Nonlinear longitudinal and lateral control

1. For each motion, compute a nonlinear controller by state feedback control with exact linearization approach.

i. Define the system dynamics

longitudinal motion

lateral motion

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

$$\dot{x}_2 = x_{12}x_4 - x_{10}x_6 - g \sin x_9 + u_1$$

$$\dot{x}_9 = x_{10}$$

$$x_{10} = a_2x_{12}x_8 + b_2u_1$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = x_8x_6 - x_{12}x_2 + g \sin x_7 \cos x_9 + u_2$$

$$\dot{x}_7 = x_8$$

$$\dot{x}_8 = a_1x_{12}x_{10} + b_1u_2$$

ii. Choose an output to control

Assume that the desired output is one of the states. Let's choose $\mathbf{y} = \mathbf{x}_1$ for longitudinal and $\mathbf{y}_2 = \mathbf{x}_3$ for lateral motions, which represents the longitudinal and lateral positions of the drone respectively.

To apply exact linearization, we need to compute the relative degree of the system with respect to $\mathbf{y} = \mathbf{x}_1$ and $\mathbf{y}_2 = \mathbf{x}_3$.

Compute derivatives of \mathbf{y} until \mathbf{u}_1 appears and \mathbf{y}_2 until \mathbf{u}_2 appears

$$\begin{aligned}\dot{y} &= \dot{x}_1 = x_2 & \dot{y}_2 &= \dot{x}_3 = x_4 \\ \ddot{y} &= \dot{x}_2 = x_{12}x_4 - x_{10}x_6 - g\sin(x_9) + u_1 & ; & \ddot{y}_2 = \dot{x}_4 = x_8x_6 - x_{12}x_2 + g\sin(x_7)\cos(x_9) + u_2\end{aligned}$$

Here, u_1 and u_2 appears in the second derivative of y and y_2 . Hence, the relative degree of the system with respect to $\mathbf{y} = \mathbf{x}_1$ and $\mathbf{y}_2 = \mathbf{x}_3$ is 2.

iii. Linearize the system (Feedback Linearization)

The general form for the second-order system is: $\mathbf{ddy} = \mathbf{v}$, and $\mathbf{ddy}_2 = \mathbf{v}_2$ where \mathbf{v} and \mathbf{v}_2 are new (fictitious) control inputs.

Now lets define the control law for u_1 and u_2

We can rewrite the equations as:

$$\begin{aligned}\ddot{y} &= v, \text{ then} & \ddot{y}_2 &= v_2, \text{ then} \\ u_1 &= v - (x_{12}x_4 - x_{10}x_6 - g\sin(x_9)) & ; & u_2 = v_2 - (x_8x_6 - x_{12}x_2 + g\sin(x_7)\cos(x_9))\end{aligned}$$

This control law effectively cancels the nonlinearities in the system, transforming the system into $\mathbf{ddy} = \mathbf{v}$ and $\mathbf{ddy}_2 = \mathbf{v}_2$.

iv. Control design for v and v_2

We now design \mathbf{v} and \mathbf{v}_2 to make \mathbf{y} and \mathbf{y}_2 track a desired trajectory $\mathbf{y}_{\text{desired}}$ and $\mathbf{y}_2^{\text{desired}}$. Using a second-order error dynamics model:

$$\ddot{e} + 2\zeta\omega_n\dot{e} + \omega_n^2 e = 0$$

The control law for v_{long} and v_{lat} can be chosen as:

$$\begin{aligned}v &= \ddot{y}_{\text{desired}} + k_p(y_{\text{desired}} - y) + k_d(\dot{y}_{\text{desired}} - \dot{y}) \\ v_2 &= \ddot{y}_2^{\text{desired}} + k_{p2}(y_2^{\text{desired}} - y_2) + k_{d2}(\dot{y}_2^{\text{desired}} - \dot{y}_2)\end{aligned}$$

where the control gains are:

$$k_p = \omega_n^2, \quad k_d = 2\zeta\omega_n \quad \text{and} \quad k_{p2} = \omega_{n2}^2, \quad k_{d2} = 2\zeta\omega_{n2}$$

Here to design the gains numerically, we can use $\zeta = 1$ for perfect tracking and ω based on our requirement lets say $\omega_n = 2$ and $\omega_{n2} = 4$

then,

$$k_p = 4; k_{p2} = 16; k_d = 4; k_{d2} = 8;$$

The other method to design the gain values is using matrix method and go back to Ackerman's method in the matlab by selecting a desired poles.

z - is the transformed state

$$\begin{aligned} z &= [z_1; z_2] \\ \dot{z}_1 &= x_2 \text{ and } \dot{z}_2 = v = f(x) + g(x)u \\ \dot{z} &= Az + Bv \text{ where } v = -Kz \\ \dot{z} &= (A - BK)z \end{aligned}$$

```
A = [0 1; 0 0]; % in the transformed state longitudinal and
% lateral motions have the same A and B matrix
B = [0; 1];
C = [1 0];
D = 0;
p1 = [-6; -9] % desired poles in the transferred domain
```

```
p1 = 2x1
-6
-9
```

```
K1 = acker(A, B, p1) % for longitudinal K1 = [kp1 kd1]
```

```
K1 = 1x2
 54    15
```

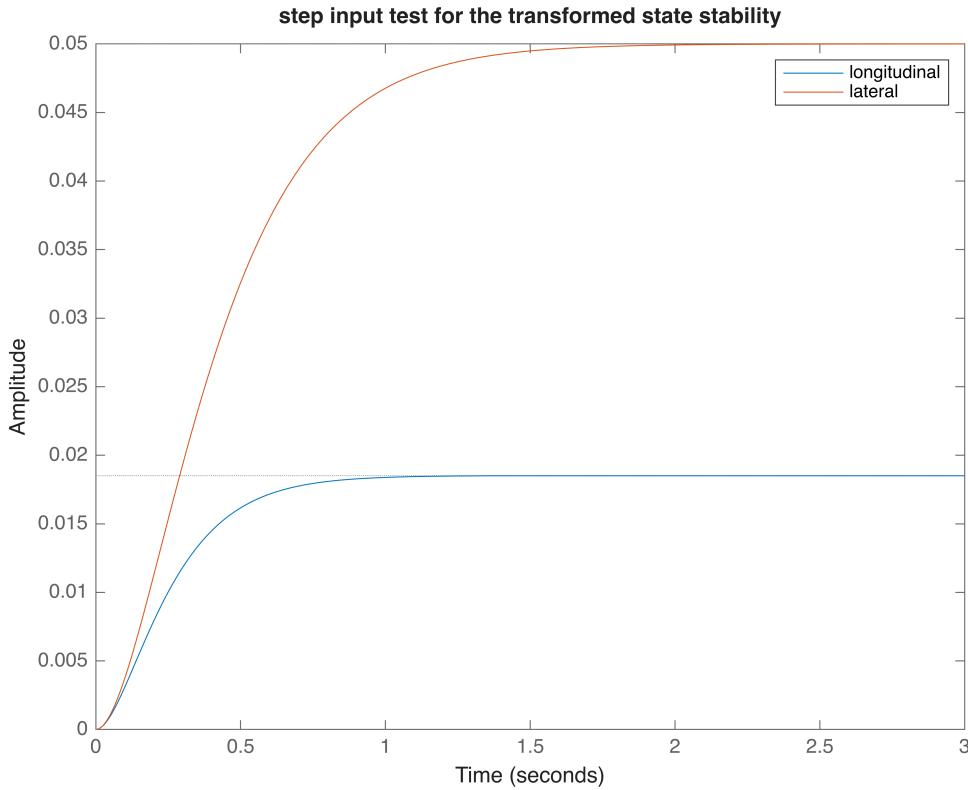
```
p2 = [-4; -5]
```

```
p2 = 2x1
-4
-5
```

```
K2 = acker(A, B, p2) % for lateral K2 = [kp2 kd2]
```

```
K2 = 1x2
 20    9
```

```
z_ss1 = ss(A-B*K1, B, C, D);
step(z_ss1);
hold on
z_ss2 = ss(A-B*K2, B, C, D);
step(z_ss2)
hold off
legend('longitudinal', 'lateral')
title('step input test for the transformed state stability')
```



As we see from the step response, the transformed state model is stable, so we can apply this into the nonlinear plant.

v. Finally,

The control input **u_1** and **u_2** can be expressed as below:

$$u_1 = \ddot{y}_{\text{desired}} + k_p(y_{\text{desired}} - y) + k_d(\dot{y}_{\text{desired}} - \dot{y}) - (x_{12}x_4 - x_{10}x_6 - g \sin(x_9))$$

$$e = y_{\text{desired}} - y$$

$$\dot{e} = \dot{y}_{\text{desired}} - \dot{y}$$

$$u_1 = \ddot{y}_{\text{desired}} + k_p e + k_d \dot{e} - (x_{12}x_4 - x_{10}x_6 - g \sin(x_9))$$

$$u_2 = \ddot{y}_2_{\text{desired}} + k_{p2}(y_2_{\text{desired}} - y_2) + k_{d2}(\dot{y}_2_{\text{desired}} - \dot{y}_2) - (x_8x_6 - x_{12}x_2 + g \sin(x_7)\cos(x_9))$$

$$e_2 = y_2_{\text{desired}} - y_2$$

$$\dot{e}_2 = \dot{y}_2_{\text{desired}} - \dot{y}_2$$

$$u_2 = \ddot{y}_2_{\text{desired}} + k_{p2}e_2 + k_{d2}\dot{e}_2 - (x_8x_6 - x_{12}x_2 + g \sin(x_7)\cos(x_9))$$

2. Apply the proposed control techniques directly on the nonlinear model. Discuss the results.

The reference input for our system in the longitudinal direction is **rx**. So, we can replace the desired input by **rx** and try the controller on the nonlinear model. In the lateral direction, the reference input is **ry**. To include the first and second derivative of the reference input into the system, we can use a differential blocks in the simulink block and include the derivatives in the matlab function as input parameter. In the constant input, the derivatives of the reference inputs are zero and we can skip. But for the time varying it can't be zero. So, to make it a generalized model, we should include the differentials in the equations. **Here, I am using a pure feedback with exact linearization, so it is also possible to remove the feedforward input (the second derivative of the reference). Below is the modified simulink model:**

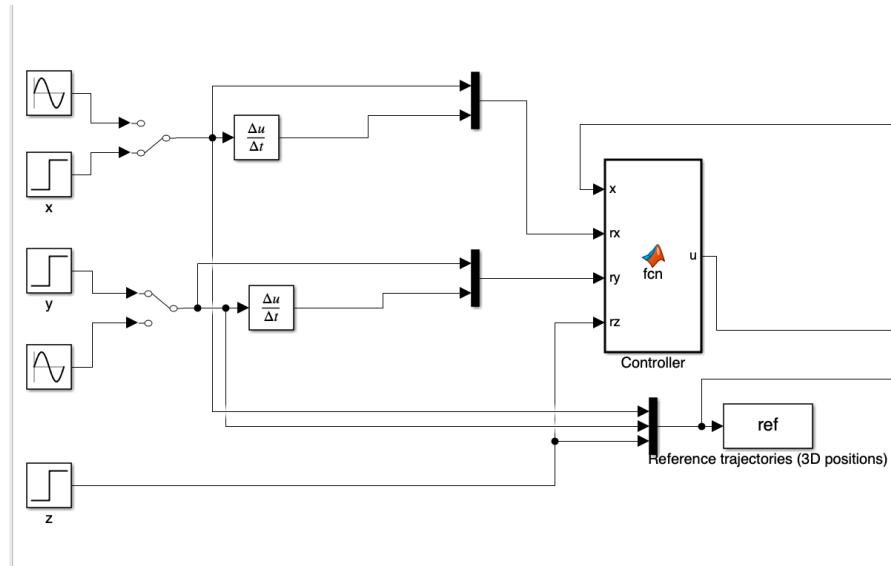


Fig i.a: modified simulink block with reference derivatives

The manual switch block is used to change references

```
% longitudinal control
u(1) = K1(1)*(rx(1)-x(1))+K1(2)*(rx(2)-x(2))-(x(12)*x(4)-x(10)*x(6)-
g*sin(x(9))); 

% lateral control

u(2) = K2(1)*(ry(1)-x(3))+K2(2)*(ry(2)-x(4))-(x(8)*x(6)-
x(12)*x(2)+g*sin(x(7))*cos(x(9)));
```



Fig i.b : zero reference response of the drone model

As we can see from the result, the whole states are stable at the origin. This justified that our system is linearized perfectly at the origin and near to the origin.

3. Modify the control input in order to track:

(a) a constant trajectory

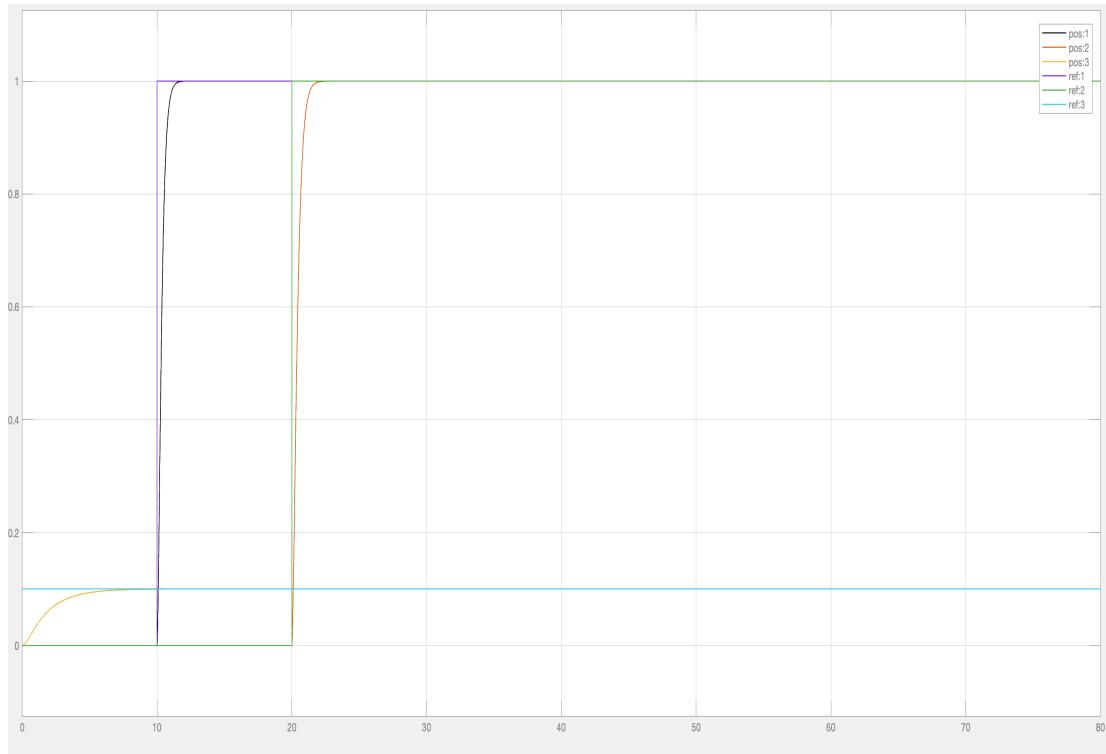


Fig j.a: constant input responses (linear positions (x_1, x_3, x_5))

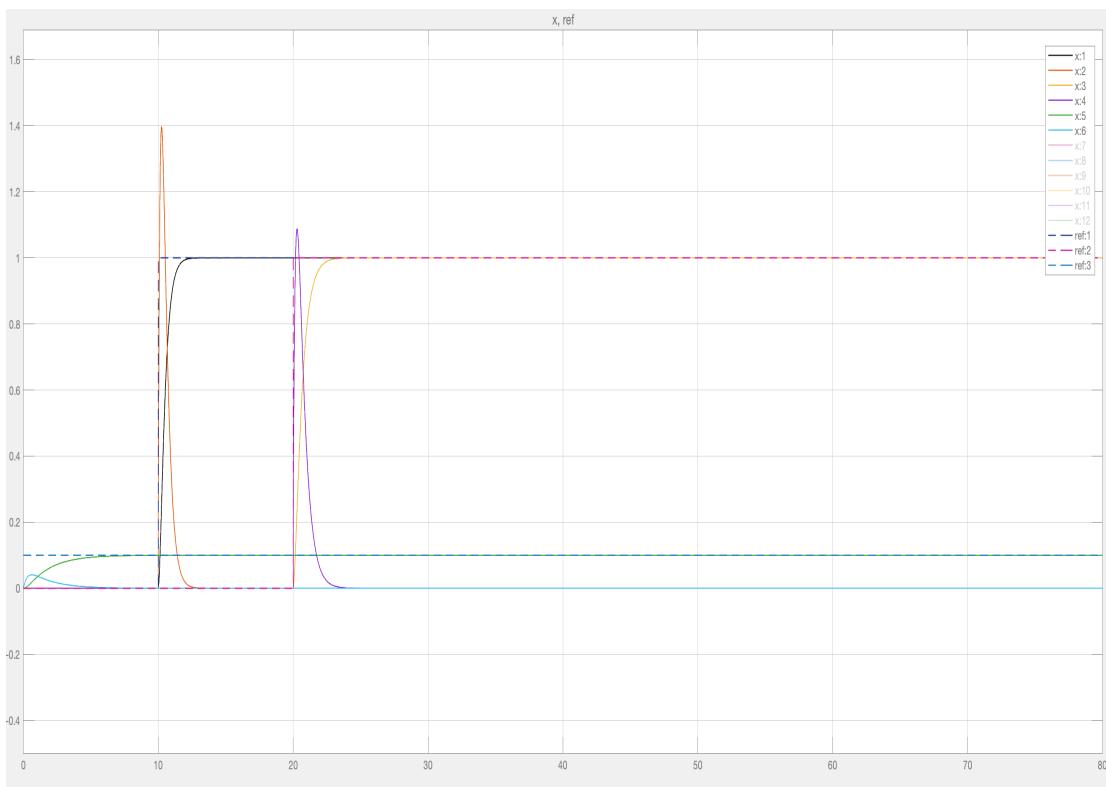


Fig j.b: constant input responses (linear positions and velocities (x_1-x_6))

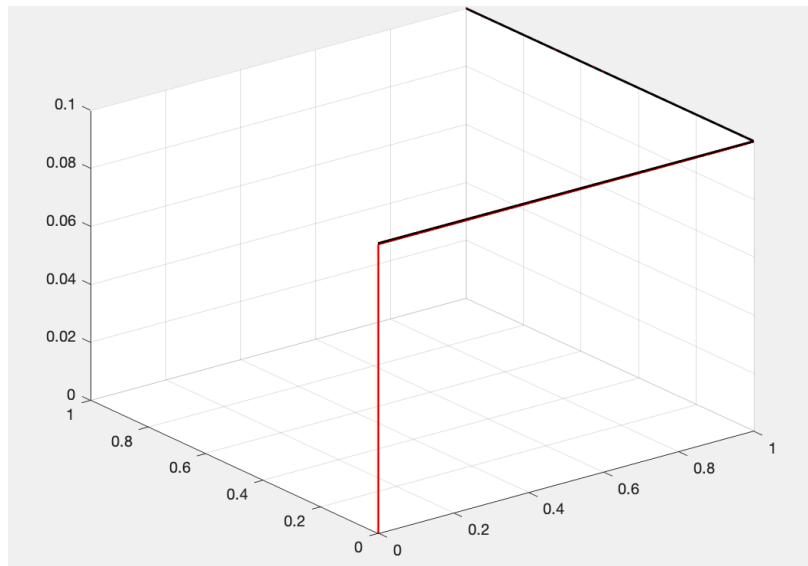


Fig j.c: plot in 3D for the constant inputs (black - reference, red - output)

In the constant input, our exact linearized plant perfectly tracked the reference input. The settling time is around 1 second after excitation and the error is zero afterward. Unlike the previous methods, now the angular positions and velocities are unstable while the linear positions and velocities are perfectly controlled. **This is due to the relative degree of the output is 2, while the dimensions of the states are 4.** In the linearization process we only considered the controllable states (the linear position and velocity). **To control the angular positions and velocities, we must choose an output with a relative degree of 4 OR take the states as internal dynamics and stabilize them at lower relative degrees.** Since they are angular values, we can also set a saturation limit in range of $[-\pi, \pi]$ for the angular positions.

(b) a time varying trajectory

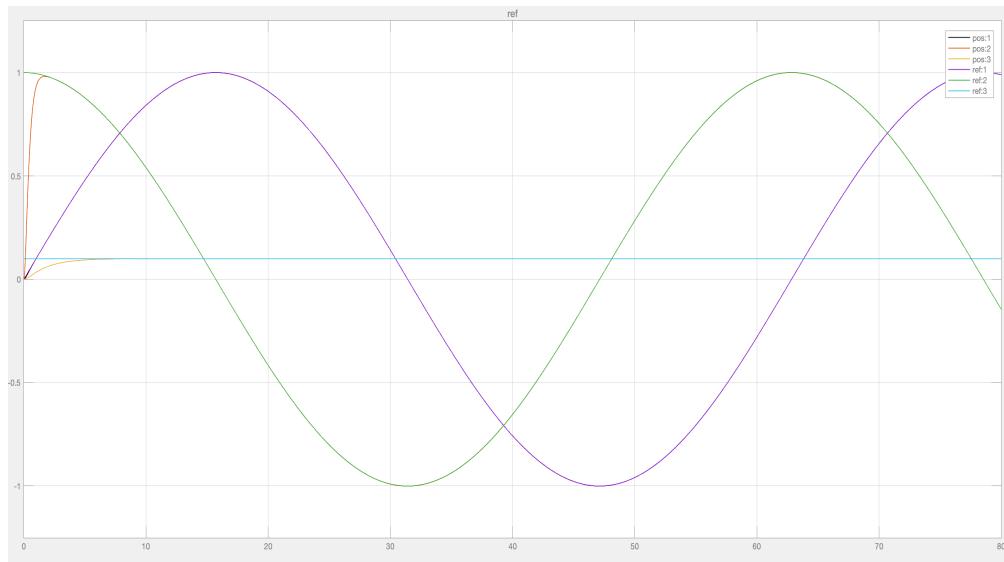


Fig k.a: pure sinusoidal with $\pi/2$ phase shift and $\omega = 0.1\text{rad/s}$ input response

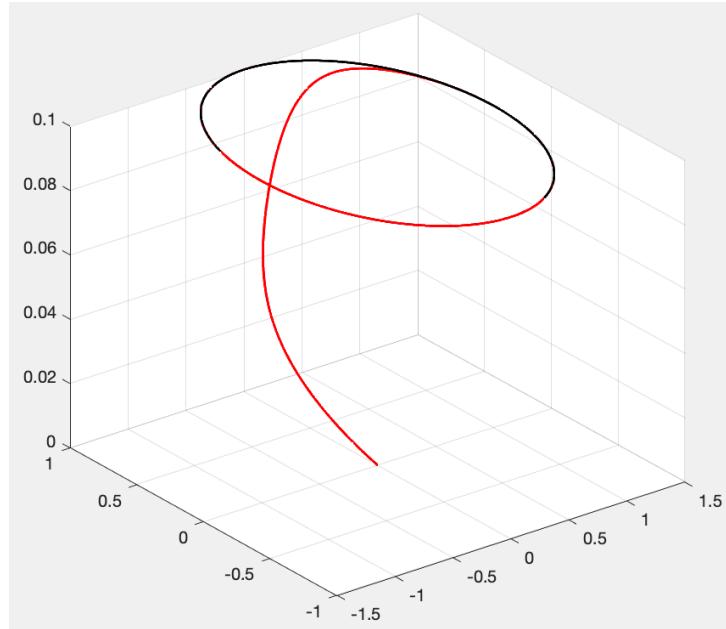


Fig k.b: plot in 3D for the pure sinusoidal input (black - reference, red - output)

As we can see from the 3D plot, even though the output tracked the reference, the plots are not in the same point for the angular motion. This is due to the case that the circular motion in the reference was started earlier while the drone is moving vertically. We can also see the effect of early angular motion tracking on the vertical motion, which is a polynomial vertical path.

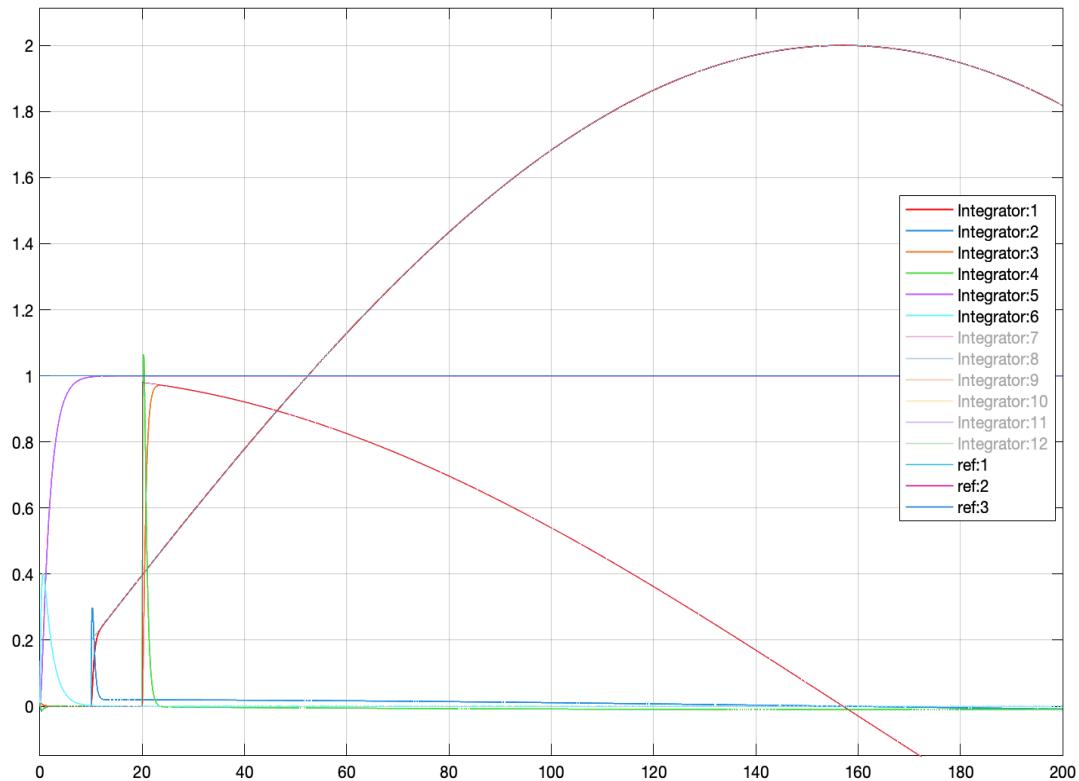


Fig k.c: variable input response at $\omega = 0.01\text{rad/s}$

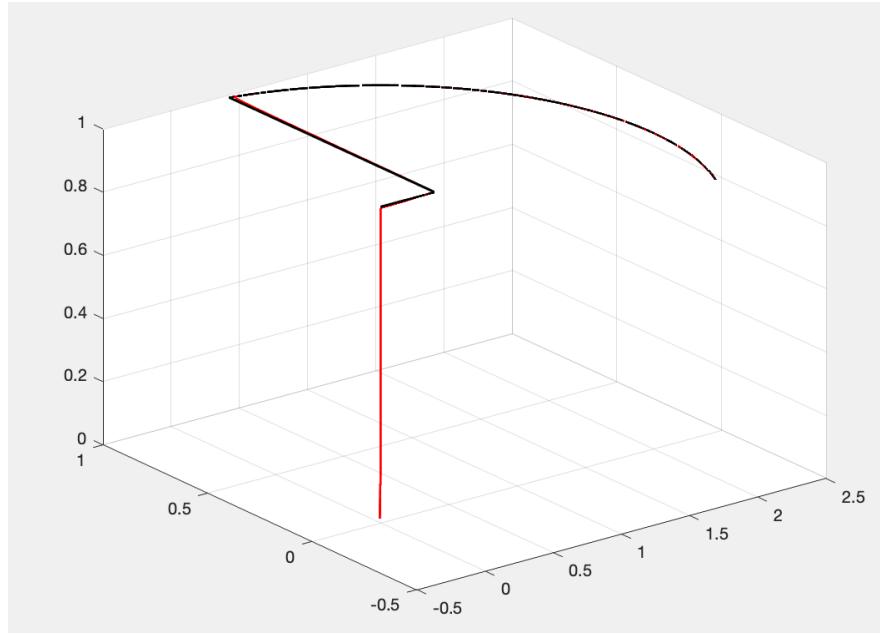


Fig k.d: plot in 3D for the pure sinusoidal input (black - reference, red - output)

The controller able to track the time varying responses efficiently and lowest error, short settling time and with no overshoot.

4. Compare the results with the results obtained by the linear controller

Based on the result, we can say that our controller handles the nonlinearity of the model and exactly stabilizes both the linear positions and velocities. This justifies that exact linearization method is more efficient than the approximation methods used in the previous control strategies. The previous methods has overshoot when they detect sudden input changes, has higher error and settling time than exact linearization method.

3. Nonlinear Vertical control

The decoupled vertical dynamics is described by:

$$\begin{cases} \dot{x}_5 = x_6 \\ \dot{x}_6 = x_2x_{10} - x_4x_8 + g \cos(x_7) \cos(x_9) + u_3 \\ \dot{x}_{11} = x_8 \\ \dot{x}_{12} = a_3x_8x_{10} + \frac{1}{I_{zz}}M_3 + u_4 \end{cases} \quad (8)$$

This dynamic can be decoupled into two sub-systems with dedicated controllers, the verticals displacement of the drone along the z-axis and its rotation around this axis, which is why the method of exact linearization is considered more than the other calculation methods.

Consider the two controllers:

$$\begin{aligned} u_3 &= -x_2x_{10} + x_4x_8 - g \cos(x_7) \cos(x_9) - k_1x_5 - k_2x_6 \\ u_4 &= -a_3x_8x_{10} - \frac{1}{I_{zz}}M_3 - k_3x_{11} - k_4x_{12} \end{aligned} \quad (9)$$

1. Prove that the following control inputs allows to obtain a linear closed-loop system and compute the values of k_1, k_2, k_3 and k_4 allowing to stabilize asymptotically the system.

i. Define the system dynamics

vertical motion

$$\begin{aligned} \dot{x}_5 &= x_6 \\ \dot{x}_6 &= x_{10}x_2 - x_4x_8 + g \cos(x_7) \cos(x_9) + u_3 \\ \dot{x}_{11} &= x_{12} \\ \dot{x}_{12} &= a_3x_{10}x_8 + \frac{1}{I_{zz}}M_3 + u_4 \end{aligned}$$

ii. Choose an output to control

Assume that the desired output is one of the states. Let's choose $y = x_5$ for vertical and $y_2 = x_8$ for angular motions, which represents the the motions in z-axis.

To apply exact linearization, we need to compute the relative degree of the system with respect to $y=x_5$ and $y_2=x_8$.

Compute derivatives of y until u_3 appears and y_2 until u_4 appears

$$\begin{aligned} \dot{y} &= \dot{x}_5 = x_6 \\ \ddot{y} &= \dot{x}_6 = x_{10}x_2 - x_4x_8 + g \cos(x_9) \cos(x_7) + u_3 \\ &\quad \text{and} \\ \dot{y}_2 &= \dot{x}_{12} = x_{12} \\ \ddot{y}_2 &= \dot{x}_{12} = a_3x_{10}x_8 + \frac{1}{I_{zz}}M_3 + u_4 \end{aligned}$$

Here, u_3 and u_4 appears in the second derivative of y and y_2 . Hence, the relative degree of the system with respect to $y = x_5$ and $y_2 = x_8$ is 2.

iii. Linearize the system (Feedback Linearization)

The general form for the second-order system is: $\ddot{y} = v$, and $\ddot{y}_2 = v_2$ where v and v_2 are new (fictitious) control inputs.

Now lets define the control law for u_3 and u_4

We can rewrite the equations as:

$$\ddot{y} = v, \text{ then} \\ u_3 = v - (x_{10}x_2 - x_4x_8 + g \cos(x_9) \cos(x_7)) ; \quad \ddot{y}_2 = v_2, \text{ then} \\ u_4 = v_2 - (a_3x_{10}x_8 + \frac{1}{I_{zz}}M_3)$$

This control law effectively cancels the nonlinearities in the system, transforming the system into **ddy = v** and **ddy = v_2**.

iv. Control design for v and v_2

We now design **v** and **v_2** to make **y** and **y_2** track a desired trajectory **y_desired** and **y_2_desired**.

Lets substitute the control inputs in the original equations:

$$\begin{aligned} \dot{y} &= \dot{x}_5 = x_6 \\ \ddot{y} &= \dot{x}_6 = x_{10}x_2 - x_4x_8 + g \cos(x_9) \cos(x_7) + (v - (x_{10}x_2 - x_4x_8 + g \cos(x_9) \cos(x_7))) \\ &\quad \text{and} \\ \dot{y}_2 &= \dot{x}_{11} = x_{12} \\ \ddot{y}_2 &= \dot{x}_{12} = a_3x_{10}x_8 + \frac{1}{I_{zz}}M_3 + (v_2 - (a_3x_{10}x_8 + \frac{1}{I_{zz}}M_3)) \end{aligned}$$

Then, from feedback rule $v = -Kx$, and the reduced version will be:

$$\begin{array}{ll} \dot{x}_5 = x_6 & \dot{x}_5 = x_6 \\ \dot{x}_6 = v \quad \dots \rightarrow v = -Kx & \dot{x}_6 = -k_1x_5 - k_2x_6; \text{ where } K = [k_1 \ k_2] \\ \text{and} & \text{and} \\ \dot{x}_{11} = x_{12} & \dot{x}_{11} = x_{12} \\ \dot{x}_{12} = v_2 \quad \dots \rightarrow v_2 = -K_2x & \dot{x}_{12} = -k_3x_{11} - k_4x_{12}; \text{ where } K_2 = [k_3 \ k_4] \end{array}$$

In matrix form, we can write it like below:

$$\dot{x} = Ax \quad \Rightarrow \quad \begin{bmatrix} \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -k_1 & -k_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -k_3 & -k_4 \end{bmatrix} \begin{bmatrix} x_5 \\ x_6 \\ x_{11} \\ x_{12} \end{bmatrix}$$

We can also rewrite A into sub-matrices

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \Rightarrow A_1 = \begin{bmatrix} 0 & 1 \\ -k_1 & -k_2 \end{bmatrix} \text{ and } A_2 = \begin{bmatrix} 0 & 1 \\ -k_3 & -k_4 \end{bmatrix}$$

From the representation its observed that we can compute $K = [k_1 \ k_2 \ k_3 \ k_4]$, by using characteristic

equation of eigenvector $\det(\lambda I - A) = 0$, where λ is the eigenvalue of A , and we can write $A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}$ so,

the characteristic equation will be $\det(\lambda I - A) * \det(\lambda I - A) = 0$. Now, we can solve for K values by selecting the desired eigen vectors that make the system stable; by default poles in LHS region.

```

syms k1 k2 k3 k4
A1 = [0 1; -k1 -k2];
A2 = [0 1; -k3 -k4];
lam1 = [-4, -3];
lam2 = [-3, -5];
eqn = det(lam1*eye(2)-A1)*det(lam2*eye(2)-A2)

```

$$eqn = (4k_2 - 4k_1 + 4)(3k_4 - 6k_3 + 3)$$

Here, we can see that

$$k_2 = k_1 - 1 \quad \text{and} \quad k_4 = 2*k_3 - 1$$

$$\text{if we choose } k_1 = 4 \text{ and } k_3 = 6, \text{ then } k_2 = 1 \text{ and } k_4 = 7$$

Then,

$$\begin{aligned} u_3 &= -k_1x_5 - k_2x_6 - (x_{10}x_2 - x_4x_8 + g \cos(x_9) \cos(x_7)) \\ u_4 &= -k_3x_{11} - k_4x_{12} - (a_3x_{10}x_8 + \frac{1}{I_{zz}}M_3); \quad \frac{1}{I_{zz}}M_3 = u_5 \end{aligned}$$

numerically,

$$\begin{aligned} u_3 &= -4x_5 - x_6 - (x_{10}x_2 - x_4x_8 + g \cos(x_9) \cos(x_7)) \\ u_4 &= -6x_{11} - 7x_{12} - (a_3x_{10}x_8 + \frac{1}{I_{zz}}M_3); \end{aligned}$$

2. Simulate the vertical model with a 0-reference z position and a non-zero constant and positive position (Hovering flight).

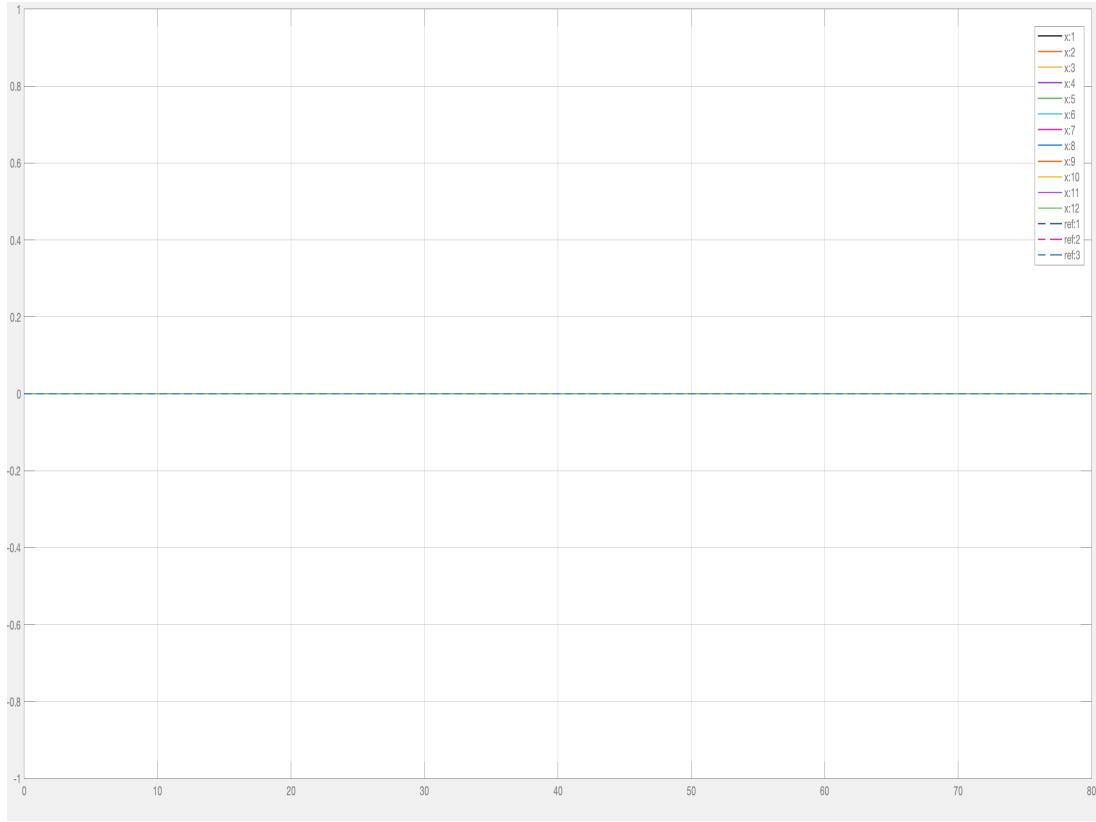


Fig I.a: response at zero reference in $rx=ry=rz=0$;

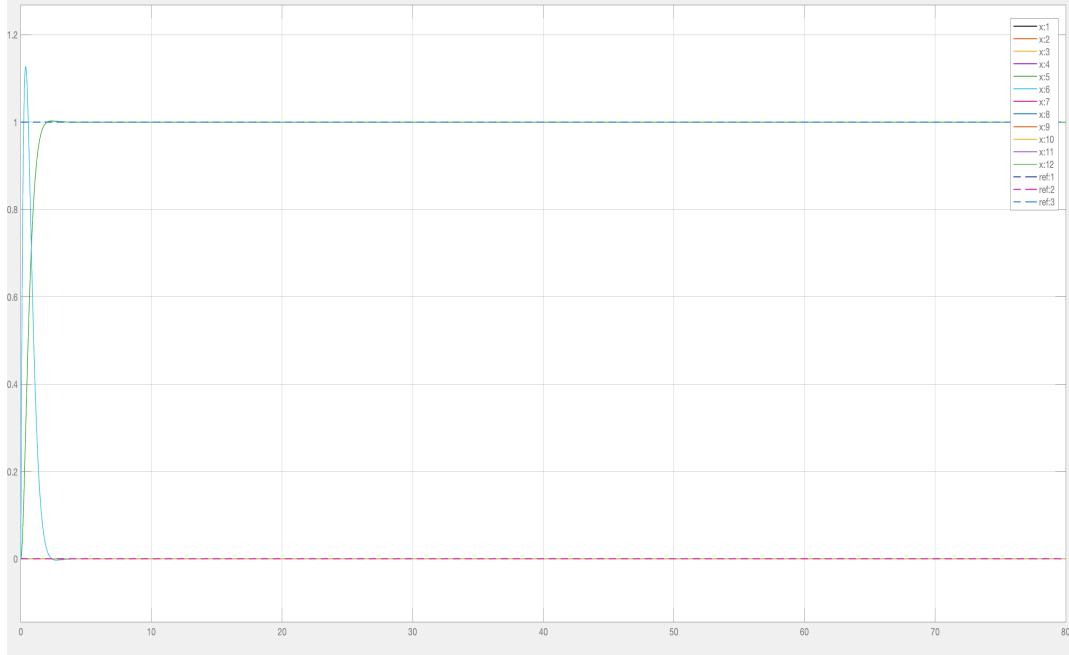


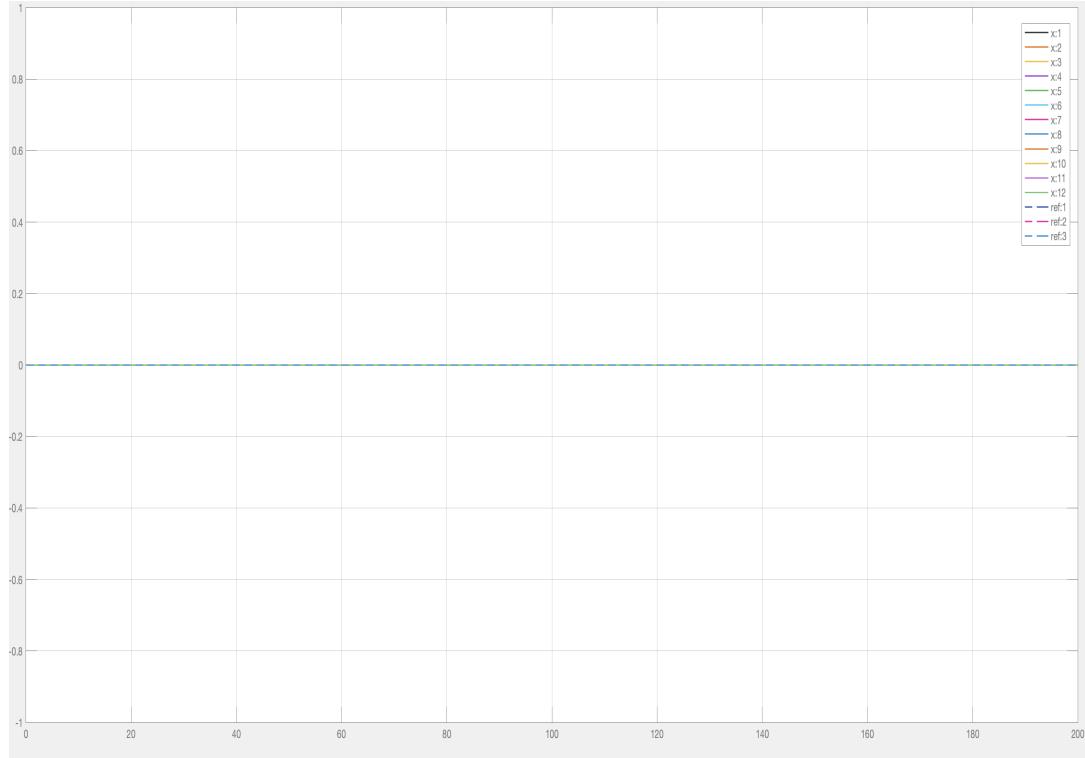
Fig I.b: response at non-zero constant reference in $rx=ry=0$; $rz=1$;

Based on the result, the designed controller perfectly controlled the vertical linear and angular motions for zero and constant reference inputs. The controller designed using exact linearization gave us a promising result. At constant reference of $rz=1$, the motion in z-axis (vertical) and all other responses stayed at zero. The settling time is also very small and there is no overshoot for the vertical position.

4. Autonomous control of the complete system

In this section, the previous controllers will be applied to the complete model.

1. Simulate the closed loop system with a setpoint in zero position.



From the zero reference input response, we observed that the system is stable at the origin and nearby points. While the exact linearization is expected to track at any points, but it has some spikes at the transient state.

2. Simulate the closed loop system with a reference trajectory:

- (a) Vertical and hovering flights at 1m
- (b) At 10s longitudinal flight 1m
- (c) At 20s lateral flight 1m
- (d) At 40s landing.

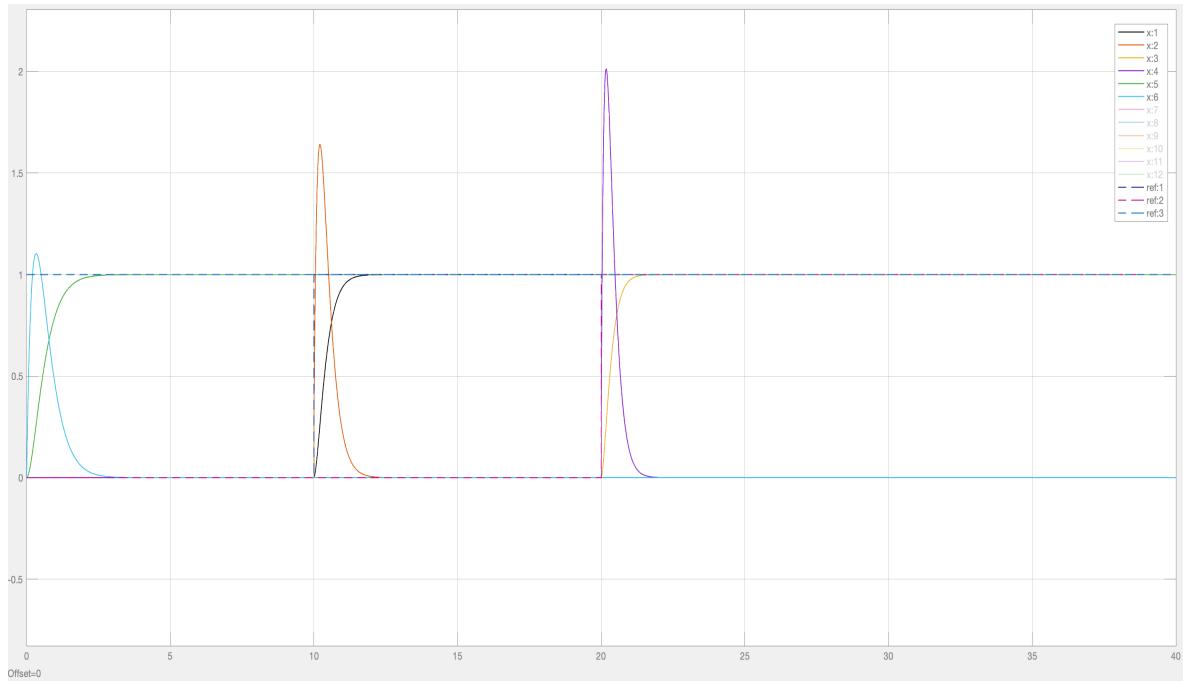


Fig I.c: combined response for constant reference inputs given above

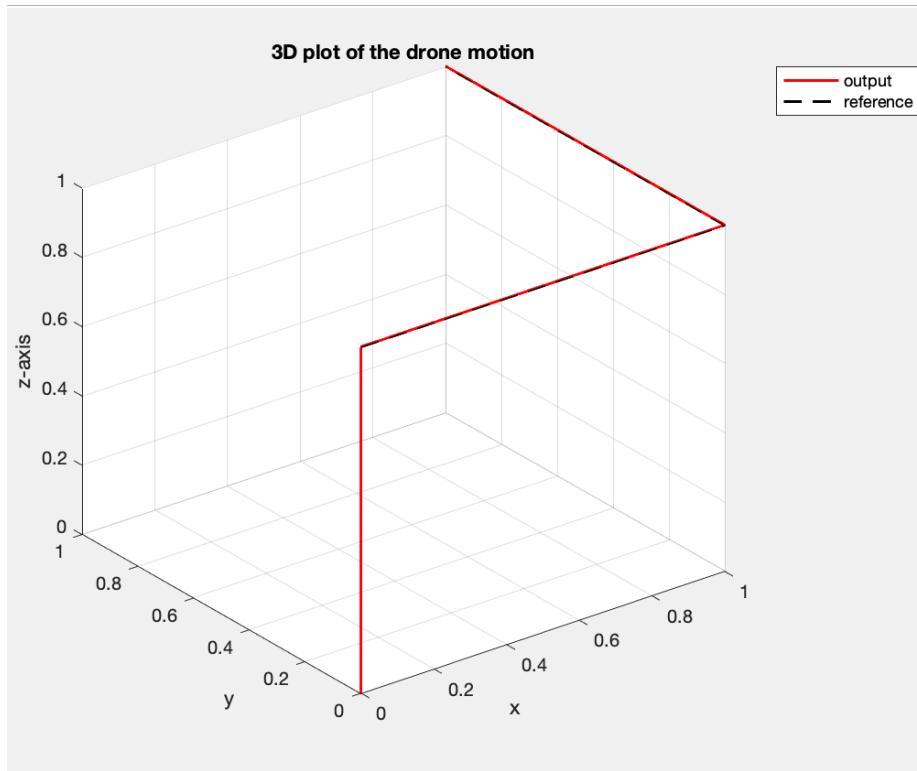


Fig I.d: 3D plot of the drone motion at constant trajectory

In a constant trajectory, the drone moved in each direction independently. First the drone moved upward, then horizontally in the x-axis and laterally in the y-axis. after moving in these directions, the drone will stay in the hovering state till a change in input is applied. In constant trajectory tracking, the positions of the drone tracked the reference in a short period of time.

3. Simulate the closed loop system with :

(a) Vertical and hovering flights at 1m

(b) Circle flight in the x – y plan

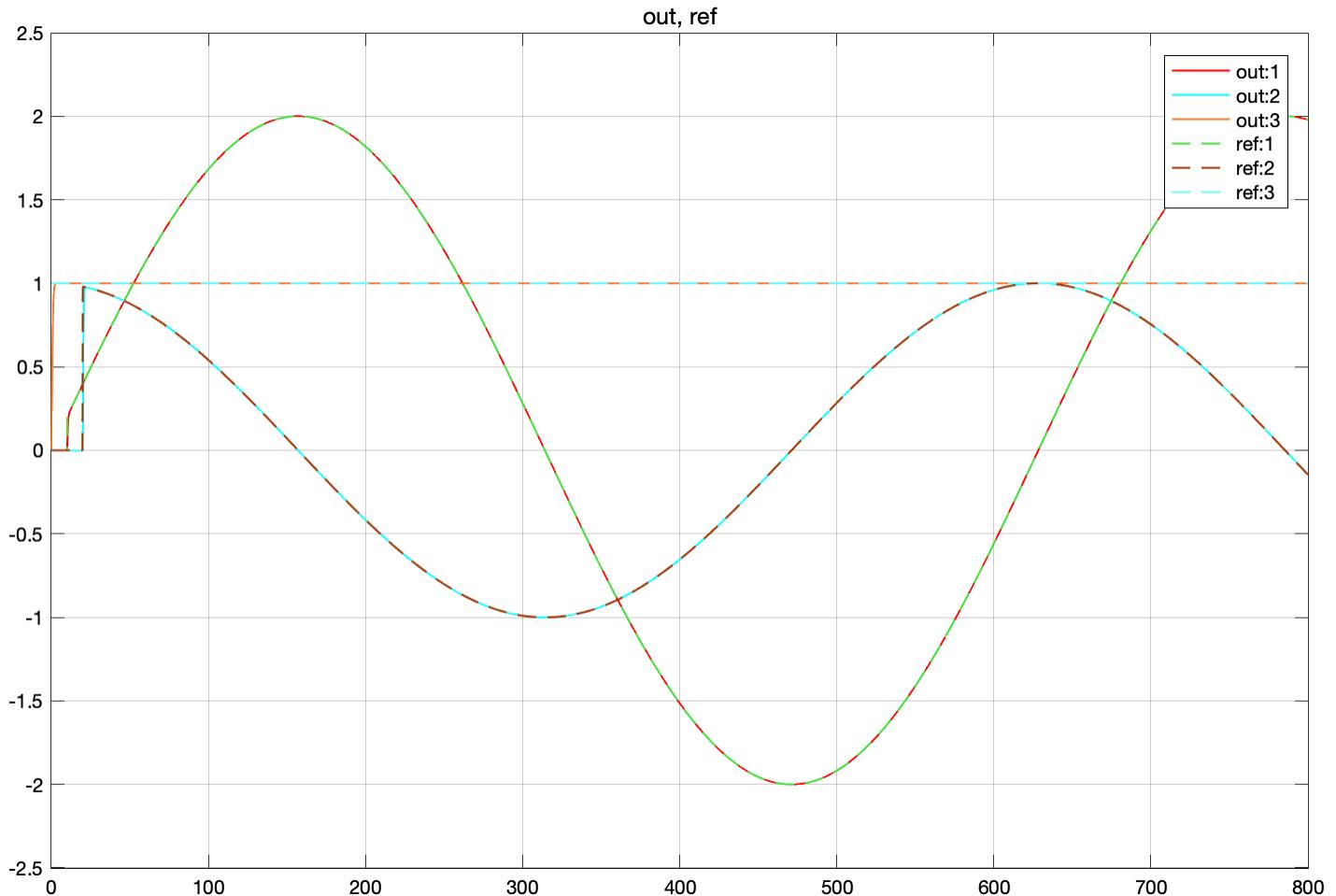


Fig m.a: time varying reference tracking

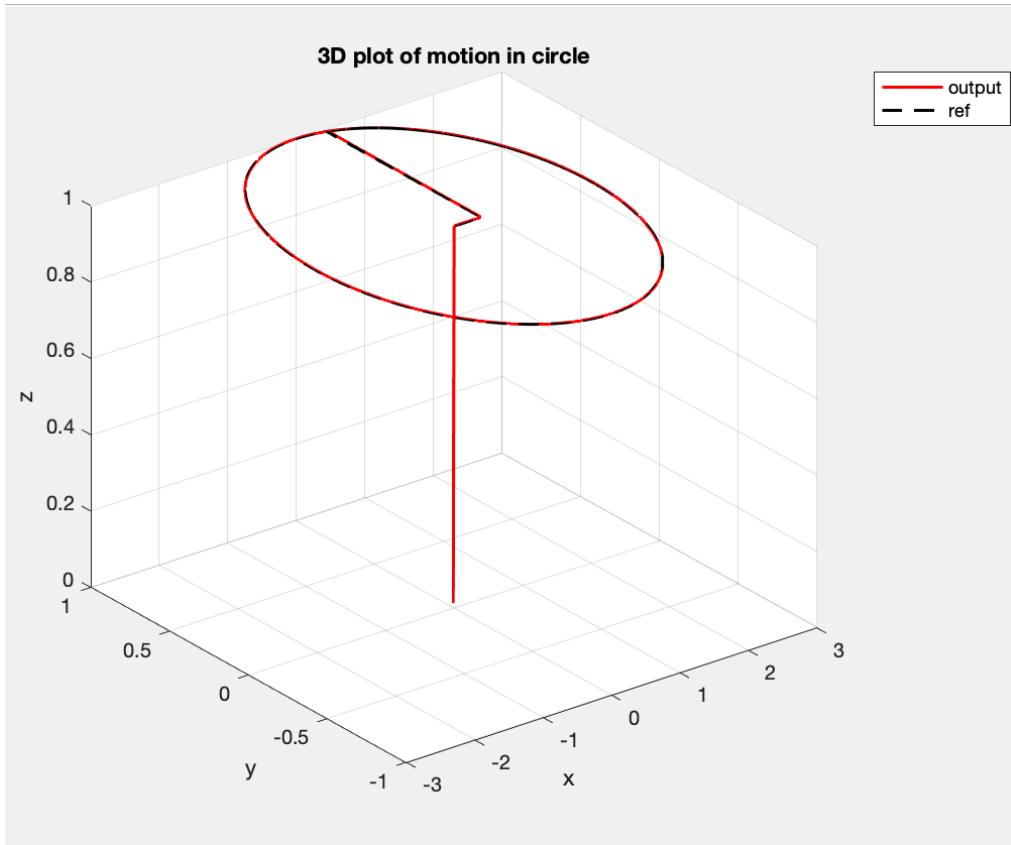


Fig m.b: 3D plot of the circular motion

The controller designed using exact linearization was able to track the circle trajectory perfectly. This justified that whatever the reference trajectory, the controller has a capability to control the drone motion.

Summary

This lab report presents the analysis, design, and implementation of flight modeling and control strategies using Pole Placement, Linear Quadratic Regulator (LQR), and Exact Linearization techniques. The objective was to develop control systems capable of achieving stability, robustness, and optimal performance for a given flight dynamics model. The Pole Placement method was employed to position the system's poles in desired locations within the complex plane, ensuring specific dynamic response characteristics. The LQR approach was utilized to achieve optimal control by minimizing a quadratic cost function, balancing control effort with system performance. Lastly, the Exact Linearization technique was applied to transform the nonlinear flight dynamics into a linear form, facilitating the design of effective control laws. Each method's performance was evaluated through simulation, with comparisons made in terms of response time, stability margins, and control effort. The results highlight the strengths and limitations of each approach, providing insights into their applicability for modern flight control systems.

Challenges of Each Control Method

Pole Placement

- Sensitivity to Model Uncertainty: Pole placement relies heavily on an accurate model of the system. Small modeling errors can result in poor control performance or instability.
- Design Complexity: Selecting appropriate pole locations to achieve a balance between system speed, stability, and overshoot can be challenging, especially for higher-order or multivariable systems.
- Limited Optimality: Unlike LQR, pole placement does not explicitly minimize a cost function, making it less efficient in terms of control effort.

Linear Quadratic Regulator (LQR)

- Tuning the Cost Function: Determining the weight matrices QQ (state cost) and RR (control cost) is non-trivial and often requires trial and error or heuristic methods.
- Full State Feedback Requirement: LQR assumes full state observability, which may not be practical in real systems where only a subset of states is measurable, requiring state estimation or observers.
- Computational Burden: While LQR is efficient for offline design, real-time applications may face computational challenges, especially for large systems with high dimensionality.

Exact Linearization

- Nonlinearity and Complexity: Exact linearization requires precise knowledge of the nonlinear dynamics and can become extremely complex for systems with strong nonlinearities.
- Model Dependence: Since the method linearizes around a trajectory or an operating point, deviations from this point (like large disturbances) can reduce control effectiveness.
- Implementation Difficulty: The transformation to a linearised form may introduce control complexities that are difficult to implement in real-time systems.

In sum up, LQR is recommended for its balance of optimality, robustness, and ease of implementation. However, each method has its own strengths and challenges, and the final choice may depend on the specific flight control application, system constraints, and operational requirements. Such as the exact linearization method has better performance and very low error, however it is not cost effective, and the design process is too complex.

Appendix

Controller code

```
function u = fcn(x,rx,ry,rz, K2, K1) % V_mat1, V_mat2,  
%#codegen  
  
u = zeros(6,1);
```

```

g = 9.81;

% les moments inertiels
Ixx=7.5*10^-3;
Iyy=7.5*10^-3;
Izz=1.3*10^-3;

k1 = 2; k2 = 3;
k3 = 3; k4 = 5;
k5 = 9; k6 = 6;
k7 = 2; k8 = 3;
k9 = 2; k10 = 6;
k11 = 2; k12 = 3;

% les coefficients
a1=((Iyy-Izz)/Ixx);
a2=((Izz-Ixx)/Iyy);
a3=((Ixx-Iyy)/Izz);

% % LQR + pole placement method
% % longitudinal
% xlong = [x(1); x(2); x(9); x(10)];
% u(1) = -K1*xlong+V_mat1(1)*rx(1); % To be completed;
%
% % lateral
% xlat = [x(3); x(4); x(7); x(8)];
% u(2) = -K2*xlat + V_mat2(1)*ry(1); % To be completed;

%
% u(3) = -(x(10)*x(2)-x(8)*x(4)+g*cos(x(7))*cos(x(9)))-k5*(x(5)-rz)-k6*x(6);
% u(4) = -(a1*x(12)*x(10))-k7*x(7)-k8*x(8);
%

% exact linearization
% longitudinal control
u(1) = K1(1)*(rx(1)-x(1))+K1(2)*(rx(2)-x(2))-(x(12)*x(4)-x(10)*x(6)-
g*sin(x(9)));

% lateral control

u(2) = K2(1)*(ry(1)-x(3))+K2(2)*(ry(2)-x(4))-(x(8)*x(6)-
x(12)*x(2)+g*sin(x(7))*cos(x(9)));

u(3) = -k5*(x(5)-rz)-k6*x(6)-(x(10)*x(2)-x(4)*x(8)+g*cos(x(9))*cos(x(7)));
u(4) = -k7*x(11)-k8*x(12)-a3*x(10)*x(8);

u(5) = -a2*x(12)*x(8)-k9*x(9)-k10*x(10);

```

```
% u(6) = -k11*x(7)-k12*x(8)-a1*x(12)*x(10);
end
```

Model/ plant code

```
function dx = fcn(x,u)
%#codegen

g = 9.81;

% les moments inertiels
Ixx=7.5*10^-3;
Iyy=7.5*10^-3;
Izz=1.3*10^-3;

% les coeficients
a1=((Iyy-Izz)/Ixx);
a2=((Izz-Ixx)/Iyy);
a3=((Ixx-Iyy)/Izz);

dx = zeros(12,1);

dx(1) = x(2);
dx(2) = x(12)*x(4)-x(10)*x(6)-g*sin(x(9))+u(1);
dx(3) = x(4);
dx(4) = x(8)*x(6)-x(12)*x(2)+g*sin(x(7))*cos(x(9))+u(2);
dx(5) = x(6);
dx(6) = x(10)*x(2)-x(8)*x(4)+g*cos(x(7))*cos(x(9))+u(3);
dx(7) = x(8);
dx(8) = a1*x(12)*x(10)+u(2);
dx(9) = x(10);
dx(10) = a2*x(12)*x(8)+u(1);
dx(11) = x(12);
dx(12) = a3*x(8)*x(10)+u(4)+u(5);

end
```