AERO3600 — Embedded Control Systems

Lab 03 - Output-feedback Control Design[1]

Learning Outcomes

This lab will assess your ability to:

1. Use Matlab to design output-feedback controllers using state observers.
2. Build Simulink models of the closed-loop system and run numerical simulations.
3. Analyses and evaluate the state trajectories of the closed loop obtained via numerical simulations.

1 Physical systems

In this lab we continue the process used to design Embedded Control Systems. We follow our design on the rotary pendulum and 2-DOF aero systems shown in Figure 1.

Figure 1: Rotary pendulum and aero systems.

[1] Updated: 11 Mar 2023.

2 Rotary pendulum system

In this section, we describe the output control design task for the rotary pendulum system.

2.1 Problem formulation

We consider the nonlinear model of the rotary pendulum presented in the document *Lab 01 - Modelling and Simulation*, and the equilibrium points $\bar{x}_a$ and $\bar{x}_b$ defined as

$$\bar{x}_a = \begin{bmatrix} \bar{x}_{1a} \\ \bar{x}_{2a} \\ \bar{x}_{3a} \\ \bar{x}_{4a} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \; ; \qquad \bar{x}_b = \begin{bmatrix} \bar{x}_{1b} \\ \bar{x}_{2b} \\ \bar{x}_{3b} \\ \bar{x}_{4b} \end{bmatrix} = \begin{bmatrix} 0 \\ \pi \\ 0 \\ 0 \end{bmatrix}, \tag{1}$$

with input $\bar{V}_m = 0$ for both equilibria.

We also consider the linearised models about the equilibria. The linearised model about the equilibrium point $\bar{x}_a$ is

$$\dot{x}_a = A_a \, x_a + B_a \, u_a \tag{2}$$

where $x_a \triangleq x - \bar{x}_a$, $u_a \triangleq u - \bar{u}_a$, $u = V_m$, and the matrices A_a and B_a correspond to Jacobian matrices respect to states and input evaluated at the equilibrium $\bar{x}_a$ with $\bar{u}_a = \bar{V}_m = 0$.

The linearised model about the equilibrium point $\bar{x}_b$ is

$$\dot{x}_b = A_b \, x_b + B_b \, u_b \tag{3}$$

where $x_b \triangleq x - \bar{x}_b$, $u_b \triangleq u - \bar{u}_b$, $u = V_m$, and the matrices A_b and B_b correspond to Jacobian matrices respect to states and input evaluated at the equilibrium $\bar{x}_b$ with $\bar{u}_b = \bar{V}_m = 0$.

In the lab 02, we design controllers to stabilise the equilibrium points under the assumption that the states are measured. In this lab, we drop that assumption and **we consider that the only measured variables are the angle of the arm and the angle of the pendulum**. That is, the output is

$$y = C \, x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x. \tag{4}$$

The first task is to design an output feedback controller using the linearised model to stabilise the system about the equilibrium points $\bar{x}_a$. To achieve this objective, use the controller designed in the lab 02 and design an observer to estimate the states and use the estimation in the feedback control.
The second problem is to design another output feedback controller using the linearised model to stabilise the system about the equilibrium point $\bar{x}_b$. Use the controller designed in the lab 02 and design an observer to estimate the states and use the estimation in the feedback control.
Notice that all the equations and matrices that characterise the nonlinear and linearised models are given in the document Lab 01 - Modelling and Simulation.

2.2 Output-feedback control system design using linearised models.

In this section, you have to build and simulate the control system of the rotary pendulum system in closed loop with an observer-based output-feedback controller that stabilised the desired equilibrium points $\bar{x}_a$ and $\bar{x}_b$. We design the observer and the controller and compare the time histories of the states and the state estimation of the control system for both the nonlinear and its linear approximation. The task is to write a MATLAB script that performs the following actions:

1. Call the function `rp_parameters.m` that defines the parameters and matrices needed for the nonlinear and linearised models, as well as the equilibrium point $\bar{x}_a$ and the input $\bar{V}_m$. Store all these values in the structure `rp_p`.

2. Create the function `rp_sfc_design.m` that accepts the matrices A and B of the linearised model, and the vector E_c that contains the desired eigenvalues of the closed loop, and returns

i) The variable `rp_p.COcheck=1` if the linearised model is completely controllable, otherwise `rp_p.COcheck=0`.

ii) The gain of the state-feedback controller `rp_p.K` such that the eigenvalues of the closed loop are the entries of the vector `rp_p.Ec`.

3. Define the vector of desired eigenvalues of the closed loop in the main script and use the function `rp_sfc_design.m` to compute the gain of the state-feedback controller. Use simulations to select the eigenvalues such that the input does not exceed its maximum value, that is $\max |V_m| \leq 15$v, and simultaneously maximise the *region of attraction* with zero initial velocities, that is $x_3(0) = 0$ and $x_4(0) = 0$.[2]

4. Create the function `rp_obs_design.m` that accepts the matrices A and C of the linearised model, and the vector E_o that contains the desired eigenvalues of the closed loop, and returns

   i) The variable `rp_p.OBcheck=1` if the linearised model is completely observable, otherwise `rp_p.OBcheck=0`.

   ii) The gain of the state observer `rp_p.L` such that the eigenvalues of the observer error dynamics are the entries of the vector `rp_p.Eo`.

   The function syntax should be

   ```
   [rp_p.OBcheck,rp_p.L] = rp_obc_design(rp_p.A,rp_p.C,rp_p.Eo);
   ```

5. Define the vector of desired eigenvalues of the observer error dynamics in the main script and use the function `rp_obs_design.m` to compute the gain of the state observer. Use simulations to select the observer eigenvalues such that the full closed loop has a satisfactory performance, input does not exceed its maximum value, that is $\max |V_m| \leq 15$v, and simultaneously maximise the *region of attraction* with zero initial velocities, that is $x_3(0) = 0$ and $x_4(0) = 0$.[3]

6. Create the Simulink block diagrams `rp_ofc_nl.slx` and `rp_ofc_lin.slx` for the control systems using the nonlinear model and linearised model of the rotary pendulum, respectively. Use the initial conditions for both models that correspond to the same scenario. The simulink models should be built as discussed in the lectures and use the functions

   i) `rp_nl_model` and `rp_lin_model` created in lab 1 to compute the state-space equations fo the rotary pendulum.

   ii) `rp_sfc_controller` created in lab 2 to compute the control input.bxcvmnmn

   iii) `rp_observer` that computes the state-space questions of the observer.

7. Plots the time histories of the states for both the nonlinear and linearised control system against each other. Plot the states $x_1(t)$ and its linear approximation $x_{1a}(t) + \bar{x}_{1a}$ together with their estimations an on top of each other in one graph, then plot $x_2(t)$, $x_{2a}(t) + \bar{x}_{2a}$ and their estimations together on another graph, and so on. Sample results for the initial conditions $x_1(0) = 60$ deg, $x_2(0) = 60$ deg, $x_3(0) = 0$ deg/s and $x_4(0) = 0$ deg/s are shown in Figure 2, where we selected the following eigenvalues: $\{-5, -6, -7, -8\}$ for the control design, and $\{-25, -26, -27, -28\}$ for the observer. Note that the initial condition of the observer is set to zero.

8. Save your script as `rp_mainfile_ofc_comparison_a.m`.

9. Repeat this process for initial conditions about the equilibrium point $\bar{x}_b$. Save your script as `rp_mainfile_ofc_comparison_b.m`. Sample results for the initial conditions $x_1(0) = 10$ deg, $x_2(0) = 160$ deg, $x_3(0) = 0$ deg/s and $x_4(0) = 0$deg/s are shown in Figure 3, where we selected the following eigenvalues:$\{-5, -6, -7, -8\}$ for the control design, and $\{-30, -31, -32, -33\}$ for the observer.

[2]Use the vaules in the sample solution for the demonstration.
[3]Use the vaules in the sample solution for the demonstration.

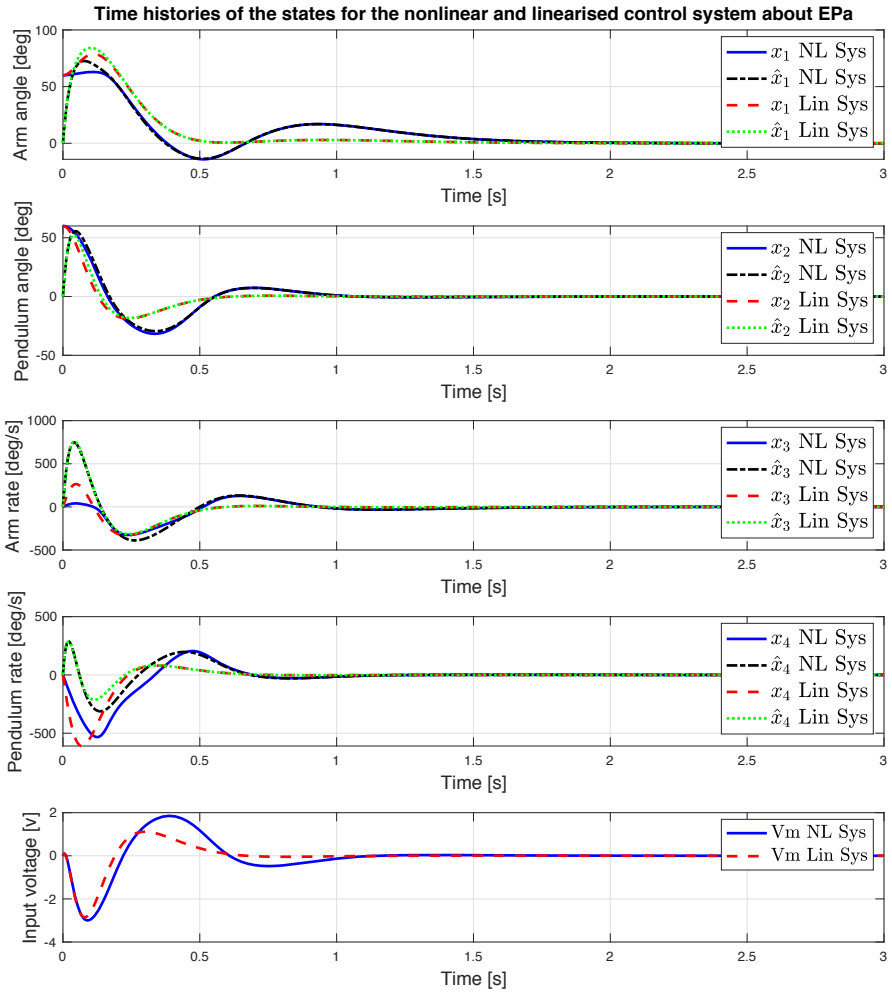


Figure 2: Time histories of the states, their estimations and input voltage.

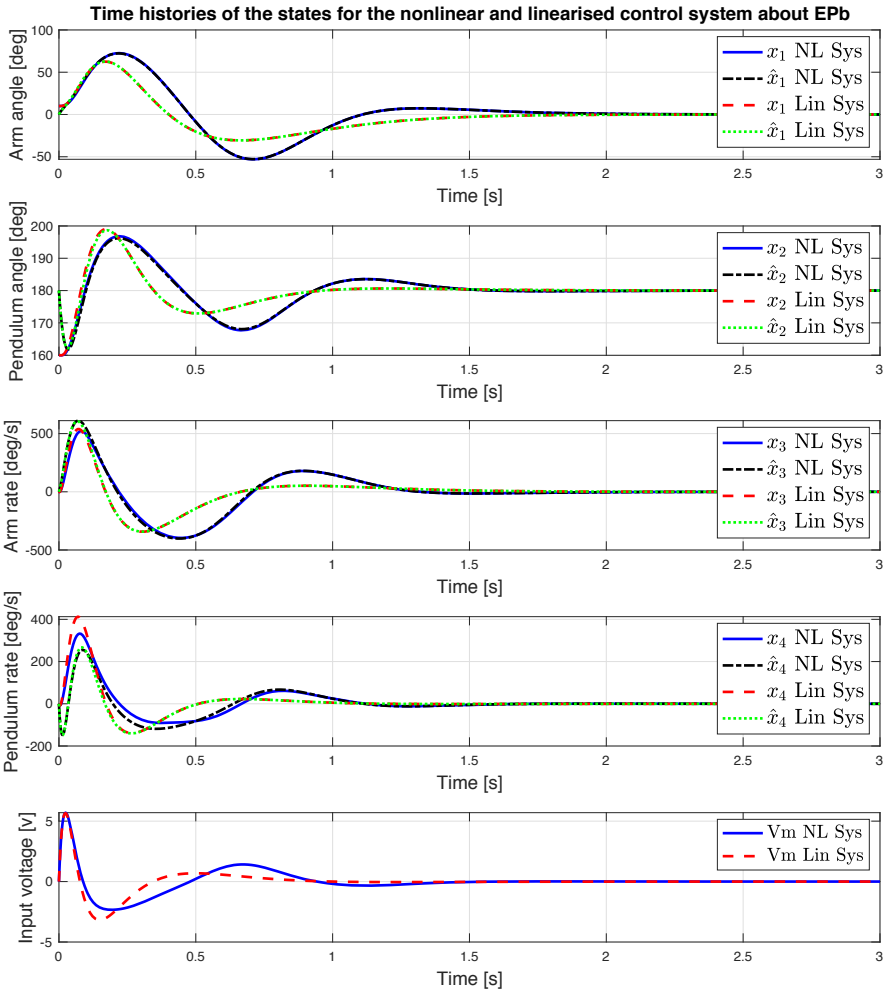


Figure 3: Time histories of the states, their estimations and input voltage.

2.3 Marking Rubric

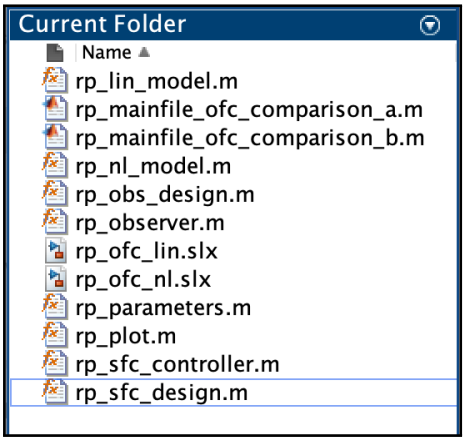
The following items will be verified in the assessment:

1. Correct output figure upon execution of the script `rp_mainfile_ofc_comparison_a.m` and `rp_mainfile_ofc_comparison_b.m`
2. Whether your functions `rp_nl_model.m`, `rp_lin_model.m`, `rp_obs_design.m`, `rp_observer.m`, `rp_parameters.m`, `rp_plot.m`, `rp_sfc_design.m`, `rp_sfc_controller.m` returns the correct expected values.
3. Whether your Simulink models `rp_ofc_nl.slx` and `rp_ofc_lin.slx` are correct and the simulation are correctly performed.

Please add your name and student number to all scripts and functions.

2.4 Code submission

We will need to submit your files for the lab assessments (see Canvas for the submission date). Please keep your files organised. Make sure that your lab folder has the following structure:



3 Aero system

In this section, we describe the control system design task for the aero system.

3.1 Problem formulation

We consider the nonlinear model of the aero system presented in the document *Lab 01 - Modelling and Simulation*, and the equilibrium points $\bar{x}_a$ and $\bar{x}_b$ defined as

$$\bar{x}_a = \begin{bmatrix} \bar{x}_{1a} \\ \bar{x}_{2a} \\ \bar{x}_{3a} \\ \bar{x}_{4a} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \; ; \qquad \bar{V}_{pa} = 0 \; ; \qquad \bar{V}_{ya} = 0, \tag{5}$$

and

$$\bar{x}_b = \begin{bmatrix} \bar{x}_{1b} \\ \bar{x}_{2b} \\ \bar{x}_{3b} \\ \bar{x}_{4b} \end{bmatrix} = \begin{bmatrix} \frac{\pi}{9} \\ 0 \\ 0 \\ 0 \end{bmatrix} \; ; \qquad \bar{V}_{pb} = \frac{K_{yy}}{K_T}\, \bar{\tau}_{pb} \; ; \qquad \bar{V}_{yb} = -\frac{K_{yp}}{K_T}\, \bar{\tau}_{pb}, \tag{6}$$

with $K_T = K_{pp}K_{yy} - K_{py}K_{yp}$ and $\bar{\tau}_{pb} = K_{sp}\sin(\bar{x}_{1b})$.

We also consider the linearised models about the equilibria. The linearised model about the equilibrium point $\bar{x}_a$ is

$$\dot{x}_a = A_a\, x_a + B_a\, u_a \tag{7}$$

where $x_a \triangleq x - \bar{x}_a$, $u_a \triangleq u - \bar{u}_a$, $u = \begin{bmatrix} V_p & V_y \end{bmatrix}^\top$, and the matrices A_a and B_a correspond to Jacobian matrices respect to x and u evaluated at the equilibrium $\bar{x}_a$ with $\bar{u}_a = \begin{bmatrix} \bar{V}_{pa} & \bar{V}_{ya} \end{bmatrix}^\top$.

The linearised model about the equilibrium point $\bar{x}_b$ is

$$\dot{x}_b = A_b\, x_b + B_b\, u_b \tag{8}$$

where $x_b \triangleq x - \bar{x}_b$, $u_b \triangleq u - \bar{u}_b$, $u = \begin{bmatrix} V_p & V_y \end{bmatrix}^\top$, and the matrices A_b and B_b correspond to Jacobian matrices respect to states and inputs evaluated at the equilibrium $\bar{x}_b$ with $\bar{u}_b = \begin{bmatrix} \bar{V}_{pb} & \bar{V}_{yb} \end{bmatrix}^\top$.

In the previous lab, we design controllers to stabilise the equilibrium points under the assumption that the states are measured. In this lab, we drop that assumption and **we consider that the only measured variables are the pitch and yaw angles**. That is the output is

$$y = C\, x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x. \tag{9}$$

The first task is to design an output feedback controller using the linearised model to stabilise the system about the equilibrium points $\bar{x}_a$. To achieve this objective, use the controller designed in the lab 02 and design an observer to estimate the states and use the estimation in the feedback control.
The second problem is to design another output feedback controller using the linearised model to stabilise the system about the equilibrium point $\bar{x}_b$. Use the controller designed in the lab 02 and design an observer to estimate the states and use the estimation in the feedback control.
Notice that all the equations and matrices that characterise the nonlinear and linearised models are given in the document Lab 01 - Modelling and Simulation.

3.2 Output-feedback control system design using linearised models.

In this section, you have to build and simulate the control system of the aero system in closed loop with an observer-based output-feedback controller that stabilised the desired equilibrium points $\bar{x}_a$ and $\bar{x}_b$. We design the observer and the controller and compare the time histories of the states and the state estimation of the control system for both the nonlinear and its linear approximation. The task is to write a MATLAB script that performs the following actions:

1. Call the function `aero_parameters.m` that defines the parameters and matrices needed for the nonlinear and linearised models, as well as the equilibrium point $\bar{x}_a$ and the inputs $\bar{V}_{pa}$ and $\bar{V}_{ya}$. Store all these values in the structure `aero_p`.

2. Create the function `aero_sfc_design.m` that accepts the matrices A and B of the linearised model, and the vector E_c that contains the desired eigenvalues of the closed loop, and returns

   i) The variable `aero_p.COcheck=1` if the linearised model is completely controllable, otherwise `aero_p.COcheck=0`.

   ii) The gain of the state-feedback controller `aero_p.K` such that the eigenvalues of the closed loop are the entries of the vector `aero_p.Ec`.

3. Define the vector of desired eigenvalues of the closed loop in the main script and use the function `rp_sfc_design.m` to compute the gain of the state-feedback controller. Use simulations to select the eigenvalues such that the inputs does not exceed their maximum values, that is $\max|V_p| \leq 15$v and $\max|V_y| \leq 15$v, and simultaneously maximise the *region of attraction* with zero initial velocities, that is $x_3(0) = 0$ and $x_4(0) = 0$[4].

4. Create the function `aero_obs_design.m` that accepts the matrices A and C of the linearised model, and the vector E_o that contains the desired eigenvalues of the closed loop, and returns

   i) The variable `aero_p.OBcheck=1` if the linearised model is completely observable, otherwise `aero_p.OBcheck=0`.

   ii) The gain of the state observer `aero_p.L` such that the eigenvalues of the observer error dynamics are the entries of the vector `aero_p.Eo`.

   The function syntax should be

   ```
   [aero_p.OBcheck,aero_p.L] = rp_obc_design(aero_p.A,aero_p.C,aero_p.Eo);
   ```

5. Define the vector of desired eigenvalues of the observer error dynamics in the main script and use the function `rp_obs_design.m` to compute the gain of the state observer. Use simulations to select the observer eigenvalues such that the full closed loop has a satisfactory performance, input does not exceed its maximum value, that is $\max|V_p| \leq 15$v and $\max|V_y| \leq 15$v, and simultaneously maximise the *region of attraction* with zero initial velocities, that is $x_3(0) = 0$ and $x_4(0) = 0$[5].

6. Create the Simulink block diagrams `aero_ofc_nl.slx` and `aero_ofc_lin.slx` for the control systems using the nonlinear model and linearised model of the rotary pendulum, respectively. Use the initial conditions for both models that correspond to the same scenario. The simulink models should be built as discussed in the lectures and use the functions

   i) `aero_nl_model` and `aero_lin_model` created in lab 1 to compute the state-space equations fo the rotary pendulum.

   ii) `aero_sfc_controller` created in lab 2 to compute the control input.bxcvmnmn

   iii) `aero_observer` that computes the state-space questions of the observer.

7. Plots the time histories of the states and their estimations for both the nonlinear and linearised control system against each other. Plot the states $x_1(t)$ and its linear approximation $x_{1a}(t) + \bar{x}_{1a}$ together with their estimations on top of each other in one graph, then plot $x_2(t)$ and $x_{2a}(t) + \bar{x}_{2a}$ and their estimations together on another graph, and so on. Sample results for the initial conditions $x_1(0) = 40$ deg, $x_2(0) = 90$ deg, $x_3(0) = 0$ deg/s and $x_4(0) = 0$ deg/s are shown in Figure 4, where we selected the following eigenvalues: $\{-1, -1.1, -1.2, -1.3\}$ or the control design, and $\{-10, -11, -12, -13\}$ for the observer. Note that the initial condition of the observer is set to zero.

8. Save your script as `aero_mainfile_ofc_comparison_a.m`.

[4] Use the vaules in the sample solution for the demonstration.
[5] Use the vaules in the sample solution for the demonstration.

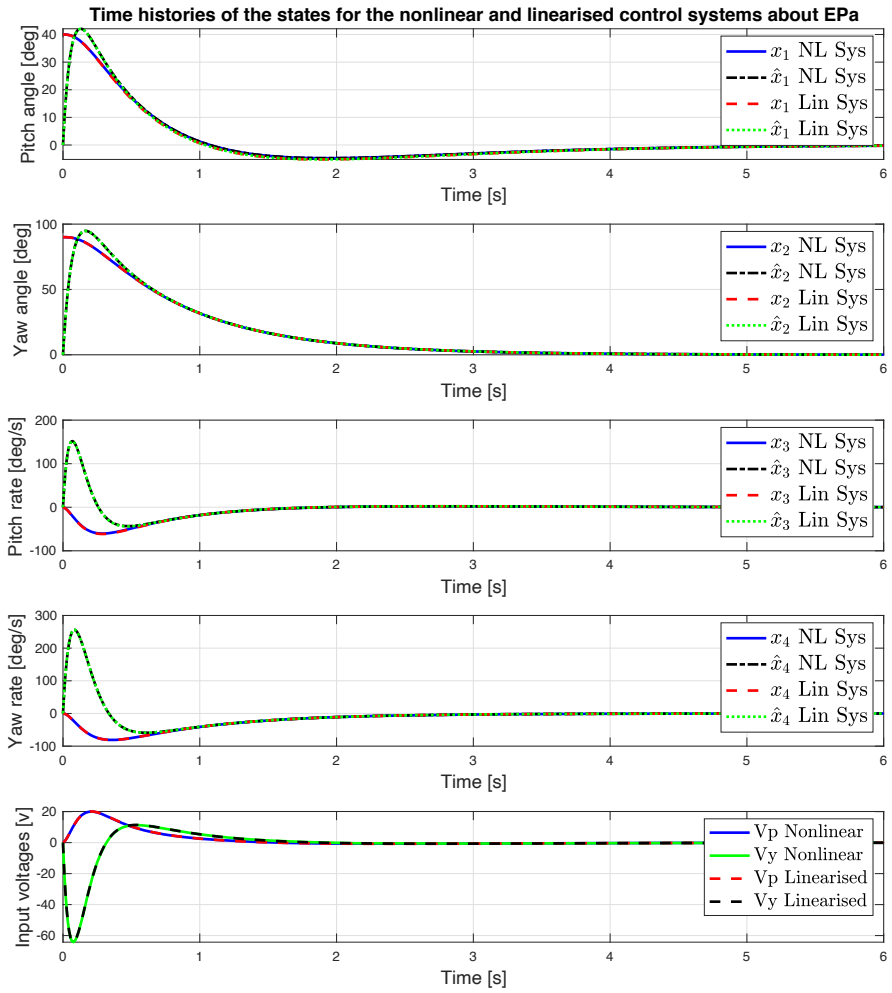


Figure 4: Time histories of the states, their estimations and input voltage.

9. Repeat this process for initial conditions about the equilibrium point $\bar{x}_b$. Save your script as `aero_mainfile_ofc_comparison_b.m`. Sample results for the initial conditions $x_1(0) = 0$ deg, $x_2(0) = 90$ deg, $x_3(0) = 0$ deg/s and $x_4(0) = 0$deg/s are shown in Figure 5, where we selected the following eigenvalues: $\{-1, -1.1, -1.2, -1.3\}$ or the control design, and $\{-10, -11, -12, -13\}$ for the observer.

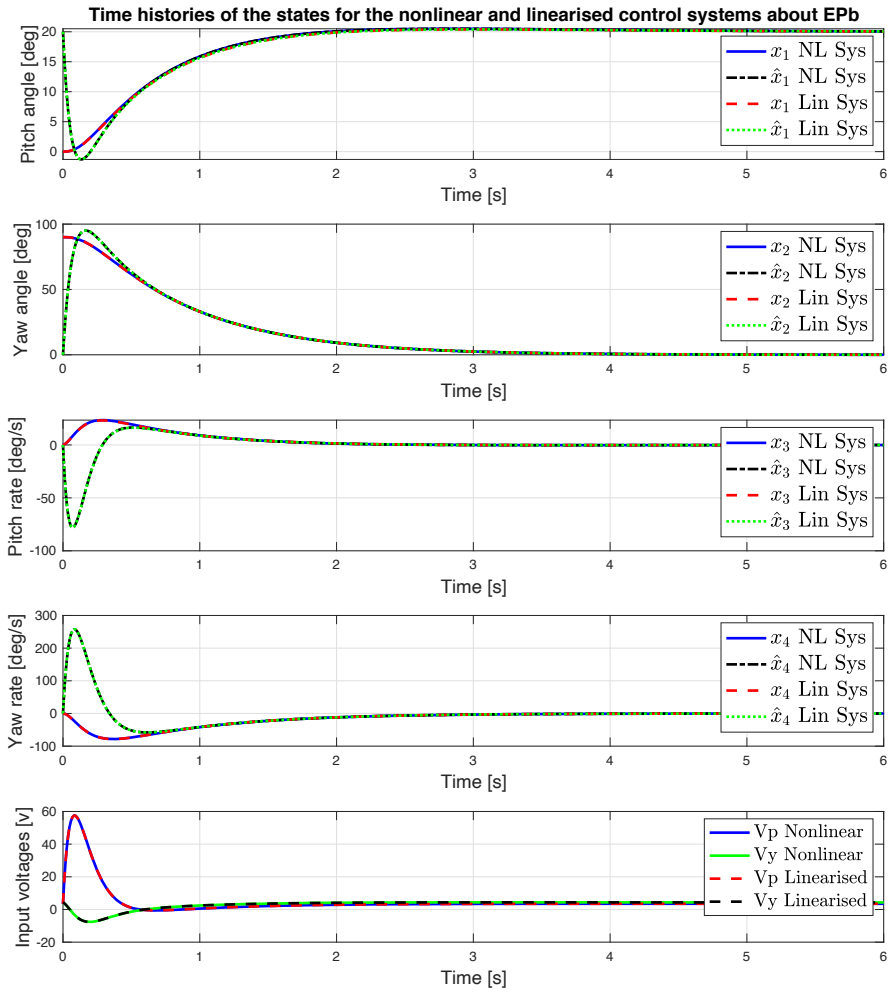


Figure 5: Time histories of the states, their estimations and input voltage.

3.3 Marking Rubric

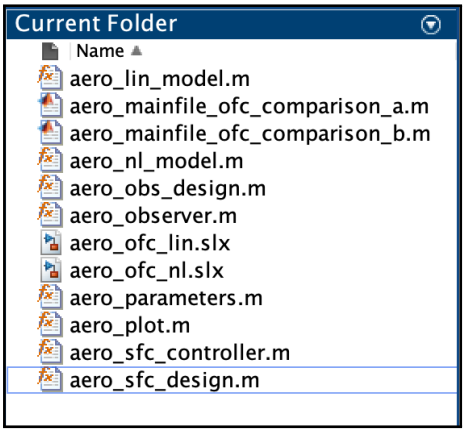
The following items will be verified in the assessment:

1. Correct output figure upon execution of the script `aero_mainfile_ofc_comparison_a.m` and `aero_mainfile_ofc_comparison_b.m`
2. Whether your functions `aero_nl_model.m`, `aero_lin_model.m`, `aero_obs_design.m`, `aero_observer.m`, `aero_parameters.m`, `aero_plot.m`, `aero_sfc_design.m`, `aero_sfc_controller.m` returns the correct expected values.
3. Whether your Simulink models `aero_ofc_nl.slx` and `aero_ofc_lin.slx` are correct and the simulation are correctly performed.

Please add your name and student number to all scripts and functions.

3.4 Code submission

We will need to submit your files for the lab assessments (see Canvas for the submission date). Please keep your files organised. Make sure that your lab folder has the following structure:


Current Folder
Name
aero_lin_model.m
aero_mainfile_ofc_comparison_a.m
aero_mainfile_ofc_comparison_b.m
aero_nl_model.m
aero_obs_design.m
aero_observer.m
aero_ofc_lin.slx
aero_ofc_nl.slx
aero_parameters.m
aero_plot.m
aero_sfc_controller.m
aero_sfc_design.m