AERO3600 — Embedded Control Systems

Lab 01 - Modelling and Simulation[1]

Learning Outcomes

This lab will assess your ability to:

1. Formulate state space models and linear approximations of a physical systems.
2. Build simulators for different models of physical systems.
3. Analyses and evaluate the state trajectories obtained via numerical simulations.

1 Physical systems

In this course, we will use two benchmark systems to apply a structured, systematic approach for embedded control system design. These two case studies are the rotary pendulum and 2-DOF aero systems shown in Figure 1.

Figure 1: Rotary pendulum and aero systems.

[1]Updated: 20 Feb 2024.

2 Rotary pendulum system

In this section, we describe the dynamics of the rotary pendulum system. Note that the modelling assumptions are implicit in the idealised model.

2.1 Nonlinear model

The idealised model of the rotary pendulum is shown in Figure 2, and its dynamics can be written in the Euler-Lagrange form as follow

$$\begin{aligned}
\left(J_r + m_p L_r^2 + \frac{1}{4} m_p L_p^2 - \frac{1}{4} m_p L_p^2 \cos^2(q_2)\right) \ddot{q}_1 + \left(\frac{1}{2} m_p L_p L_r \cos(q_2)\right) \ddot{q}_2 & & \\
+ \left(\frac{1}{2} m_p L_p^2 \sin(q_2)\cos(q_2)\right) \dot{q}_1 \dot{q}_2 - \left(\frac{1}{2} m_p L_p L_r \sin(q_2)\right) \dot{q}_2^2 &= \tau, & (1) \\
\frac{1}{2} m_p L_p L_r \cos(q_2) \ddot{q}_1 + \left(J_p + \frac{1}{4} m_p L_p^2\right) \ddot{q}_2 - \frac{1}{4} m_p L_p^2 \cos(q_2)\sin(q_2) \dot{q}_1^2 & & \\
+ \frac{1}{2} m_p L_p g \sin(q_2) &= 0, & (2)
\end{aligned}$$

where q_1 is the angle of the arm and q_2 is the angle of the pendulum.

The model (1),(2) can also be written the following matrix form

$$M(q) \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + C(q, \dot{q}) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{2} m_p L_p g \sin(q_2) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tau, \tag{3}$$

where

$$M(q) = \begin{bmatrix} J_r + m_p L_r^2 + \frac{1}{4} m_p L_p^2 - \frac{1}{4} m_p L_p^2 \cos^2(q_2) & \frac{1}{2} m_p L_p L_r \cos(q_2) \\ \frac{1}{2} m_p L_p L_r \cos(q_2) & J_p + \frac{1}{4} m_p L_p^2 \end{bmatrix} \tag{4}$$

and

$$C(q, \dot{q}) = \begin{bmatrix} \frac{1}{2} m_p L_p^2 \sin(q_2)\cos(q_2) \dot{q}_2 & -\frac{1}{2} m_p L_p L_r \sin(q_2) \dot{q}_2 \\ -\frac{1}{4} m_p L_p^2 \cos(q_2)\sin(q_2) \dot{q}_1 & 0 \end{bmatrix} \tag{5}$$

are the mass and Coriolis matrices respectively.

The torque applied to the rotary arm is generated by the motor and can be computed as follows

$$\tau = \frac{k_m}{R_m} V_m - \frac{k_m^2}{R_m} \dot{q}_1 \tag{6}$$

where V_m is the motor voltage, that is the control input. The parameters of the model are indicated in Figure 2 and given in Table 1.

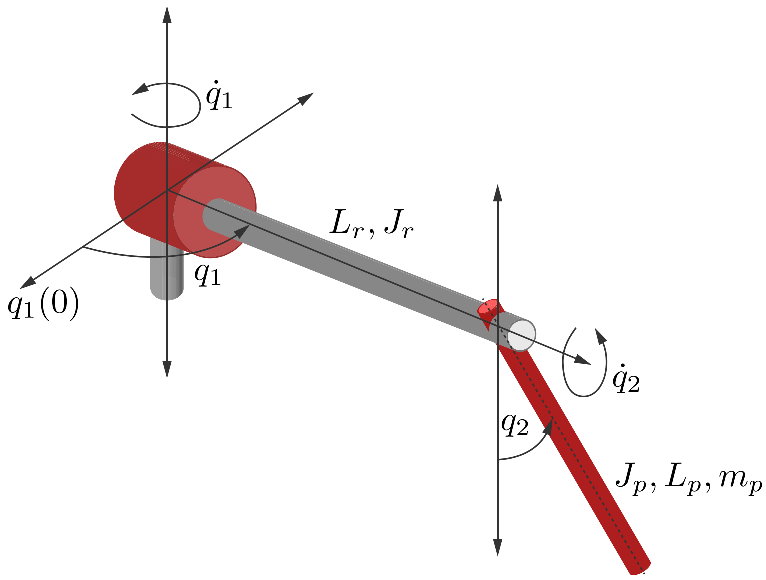


Figure 2: Idealised model of the rotary pendulum system.

Table 1: Model parameters.

	Description	Parameter	Value
Rotary arm			
	Mass	m_r	0.095 kg
	Length	L_r	0.085 m
	Moment of inertia	J_r	$\frac{1}{12}m_rL_r^2$ kg·m^2
Pendulum link			
	Mass	m_p	0.024 kg
	Length	L_p	0.129 m
	Moment of inertia	J_p	$\frac{1}{12}m_pL_p^2$ kg·m^2
	Gravity constant	g	9.81 m/s^2
Motor			
	Resistance	R_m	8.4 Ω
	Current-torque and ω-emf constant	k_m	0.042 v·s/rad

To formulate the model (1),(2),(6) in state-space form

$$\dot{x} = f(x, u), \tag{7}$$

we select the states $x_1 = q_1$, $x_2 = q_2$, $x_3 = \dot{q}_1$, $x_4 = \dot{q}_2$, and we obtain the state-space model as follows

$$\begin{aligned}
\dot{x}_1 &= x_3 && (8)\\
\dot{x}_2 &= x_4 && (9)\\
\dot{x}_3 &= \frac{L_pm_p}{\Delta}\sin(x_2)\Big[-2x_3^2L_p^2L_rm_p\cos^2(x_2) - 2x_3x_4L_p\cos(x_2)\left(4J_p + L_p^2m_p\right)\\
&\quad +2x_4^2L_r\left(4J_p + L_p^2m_p\right) + 4gL_pL_rm_p\cos(x_2)\Big]\\
&\quad -\frac{4k_m^2}{R_m\Delta}(4J_p + L_p^2m_p)\,x_3 + \frac{4k_m}{R_m\Delta}(4J_p + L_p^2m_p)\,V_m && (10)\\
\dot{x}_4 &= \frac{L_pm_p}{\Delta}\sin(x_2)\Big\{L_p\cos(x_2)\Big[x_3^2\left(4J_r + L_p^2m_p + 4L_r^2m_p\right) - 4x_4^2L_r^2m_p\Big]\\
&\quad -x_3^2L_p^3m_p\cos^3(x_2) + 2L_p^2m_p\cos^2(x_2)(2x_3x_4L_r + g)\\
&\quad -2g\left[4J_r + m_p\left(L_p^2 + 4L_r^2\right)\right]\Big\} + \frac{8k_m^2}{R_m\Delta}L_pL_rm_p\cos(x_2)x_3\\
&\quad -\frac{8k_m}{R_m\Delta}L_pL_rm_p\cos(x_2)\,V_m && (11)
\end{aligned}$$

where

$$\Delta = \left(4J_p + L_p^2m_p\right)\left[4J_r + m_p\left(L_p^2 + 4L_r^2\right)\right] - L_p^2m_p\cos^2(x_2)\left[4J_p + m_p\left(L_p^2 + 4L_r^2\right)\right].$$

Alternatively, one can use the matrix form and write the last two equations of the state-space model as

$$\begin{bmatrix}\dot{x}_3\\ \dot{x}_4\end{bmatrix} = M^{-1}(q)\left\{-C(q,\dot{q})\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\end{bmatrix} - \begin{bmatrix}0\\ \frac{1}{2}m_pL_pg\sin(q_2)\end{bmatrix} + \begin{bmatrix}1\\ 0\end{bmatrix}\left(\frac{k_m}{R_m}V_m - \frac{k_m^2}{R_m}\dot{q}_1\right)\right\}, \tag{12}$$

where we have replaced τ by (6). In general, the matrix form is more convenient for coding the model in Matlab/Simulink.

2.2 Linearised model

In this section, we present the linear approximation models about the desired equilibrium points to be stabilised. For the rotary pendulum system we consider two equilibrium points:

$$\bar{x}_a = \begin{bmatrix}\bar{x}_{1a}\\ \bar{x}_{2a}\\ \bar{x}_{3a}\\ \bar{x}_{4a}\end{bmatrix} = \begin{bmatrix}0\\0\\0\\0\end{bmatrix}\;;\qquad \bar{x}_b = \begin{bmatrix}\bar{x}_{1b}\\ \bar{x}_{2b}\\ \bar{x}_{3b}\\ \bar{x}_{4b}\end{bmatrix} = \begin{bmatrix}0\\ \pi\\0\\0\end{bmatrix}, \tag{13}$$

where $\bar{x}_a$ and $\bar{x}_b$ correspond to the pendulum pointing downward and upward respectively, with the arm at the coordinate framework origin. In both cases $\bar{V}_m = 0$.

The linear model about the equilibrium point $\bar{x}_a$ is characterised by the matrices

$$A_a = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{gL_p^2 L_r m_p^2}{J_T} & -\frac{k_m^2(4J_p + L_p^2 m_p)}{J_T R_m} & 0 \\ 0 & -\frac{2gL_p m_p(J_r + L_r^2 m_p)}{J_T} & \frac{2k_m^2 L_p L_r m_p}{J_T R_m} & 0 \end{bmatrix}, \quad B_a = \begin{bmatrix} 0 \\ 0 \\ \frac{k_m(4J_p + L_p^2 m_p)}{J_T R_m} \\ -\frac{2k_m L_p L_r m_p}{J_T R_m} \end{bmatrix} \tag{14}$$

where $J_T = J_r L_p^2 m_p + 4J_p(J_r + L_r^2 m_p)$.

Similarly, the linear model about the equilibrium point $\bar{x}_b$ is characterised by the matrices

$$A_b = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{gL_p^2 L_r m_p^2}{J_T} & -\frac{k_m^2(4J_p + L_p^2 m_p)}{J_T R_m} & 0 \\ 0 & \frac{2gL_p m_p(J_r + L_r^2 m_p)}{J_T} & -\frac{2k_m^2 L_p L_r m_p}{J_T R_m} & 0 \end{bmatrix}, \quad B_b = \begin{bmatrix} 0 \\ 0 \\ \frac{k_m(4J_p + L_p^2 m_p)}{J_T R_m} \\ \frac{2k_m L_p L_r m_p}{J_T R_m} \end{bmatrix}. \tag{15}$$

Notice that the form of the matrices C_a, D_a, C_b and D_b will depend on the selected outputs. In general, we will consider all the states or only the angles as outputs.

3 Tasks

You will need to build simulators for the rotary pendulum system using the models presented in the previous section. The tasks in Section 3.1 and 3.2 are constructive to achieve the task in Section 3.3. The main objective in this lab is to build and test the correctness of the script and models used in Section 3.3.

3.1 Nonlinear Simulink model

The build a simulator for the nonlinear model of the rotary pendulum, use the following procedure:

a) Verify that the dynamics (1)-(2) can be written in state-space form (8),(9),(10),(11) or (8),(9),(12).

1. Build a Simulink model for the rotary pendulum system that returns the state vector (the angle and angular rate of the arm and the angle and angular rate of the pendulum), the input voltage and the simulation time to the workspace in the structure `sim_nl`: `sim_nl.x`, `sim_nl.vm` and `sim_nl.t`. Save the model as `rp_modelling_nl.slx`. Use the blocks `Interpreted Matlab Function`, `Mux` and `Demux`, and write the state equations in the function `rp_nl_model.m`.

2. Write a script that performs the following actions:

   i) Call the function `rp_parameters.m` that returns a structure named `rp_p` and assigns the parameter values of the nonlinear model of the rotary pendulum.

   ii) Set the initial conditions and simulation parameters, and simulate the model.

   iii) Call the function `rp_plot.m` that plots the states and the input.

   Hint: You might have built similar scripts in ENGG2440.

3. Save the script as `rp_mainfile_nl.m`.

4. Simulate the model using different initial conditions and analyse its behaviour.

3.2 Linearised Simulink models

Consider now the equilibrium point of the rotary pendulum system noted as $\bar{x}_a$ and defined in (13).

1. Verify the linearised model of the rotary pendulum system about the equilibrium point $\bar{x}_a$ is characterised by the matrices in (14).

2. Build a Simulink model for the linearised model that returns the state vector, the input voltage and the simulation time to the workspace in the structure `sim_lin`:: `sim_lin.xlin`, `sim_lin.vmlin`, `sim_lin.tlin`. Save the model as `rp_modelling_lin.slx`. Hint: use the blocks `Interpreted Matlab Function`, `Mux` and `Demux`, and write the state equations in the function `rp_lin_model.m`.

3. Write a script that performs the following actions:

   i) Call the function `rp_parameters.m` that assigns the parameter values of the linearised model of the rotary pendulum (add to previous defined function).

   ii) Set the initial conditions and simulation parameters, and simulate the model.

   iii) Call the function `rp_plot.m` that plots the states and the input.

4. Save the script as `rp_mainfile_lin_a.m`.

5. Simulate the model using different initial conditions and analyse its behaviour.

6. Consider now the equilibrium point $\bar{x}_b$ defined in (13) and repeat the above process. Adapt all the instructions for the equilibrium point $\bar{x}_b$.

3.3 Comparison of the nonlinear and linearised models

In this section, we compare the state histories of the nonlinear model and the linear approximation. Write a MATLAB script that performs the following actions:

1. Defines the initial conditions for both models about the equilibrium point $\bar{x}_a$.

2. Simulates both the nonlinear model and the linear approximation.

3. Plots the time histories of the states from both the nonlinear system and linear approximation against each other. Plot the states $x_1(t)$ and its linear approximation $x_{1a}(t) + \bar{x}_{1a}$ on top of each other in one graph, then plot $x_2(t)$ and $x_{2a}(t) + \bar{x}_{2a}$ together on another graph, and so on. Sample results for the initial conditions $x_1(0) = 0$ deg, $x_2(0) = 20$ deg, $x_3(0) = 0$ deg/s and $x_4(0) = 0$ deg/s, $V_m = 0$ v are shown in Figure 3.

4. Save your script as `rp_mainfile_modelling_comparison_a.m`.

5. Repeat this process for initial conditions about the equilibrium point $\bar{x}_b$. Save your script as `rp_mainfile_modelling_comparison_b.m`. Sample results for the initial conditions $x_1(0) = 0$ deg, $x_2(0) = 160$ deg, $x_3(0) = 0$ deg/s and $x_4(0) = 0$deg/s, $V_m = 0$ v are shown in Figure 4.

Run simulations from different initial conditions and analyse the results.

3.4 Animation

We have develop an animation for you to visualise the simulation results. To use the animation, you need to download the files `rp_animation.m` and `aero3600_logo.jpg`, and place them in the same folder where you have your main file, functions and Simulink models. The animation function takes three arguments

```
rp_animation.m(sim_nl.t,sim_nl.x(:,1),sim_nl.x(:,2))
```

where `t` is the simulation time vector, `q1` is the arm angle vector, and `q2` is the pendulum angle vector (measured from downward rest position). You might need to change `t`, `q1` and `q2`, and use the name of the variable you defined in your simulations.

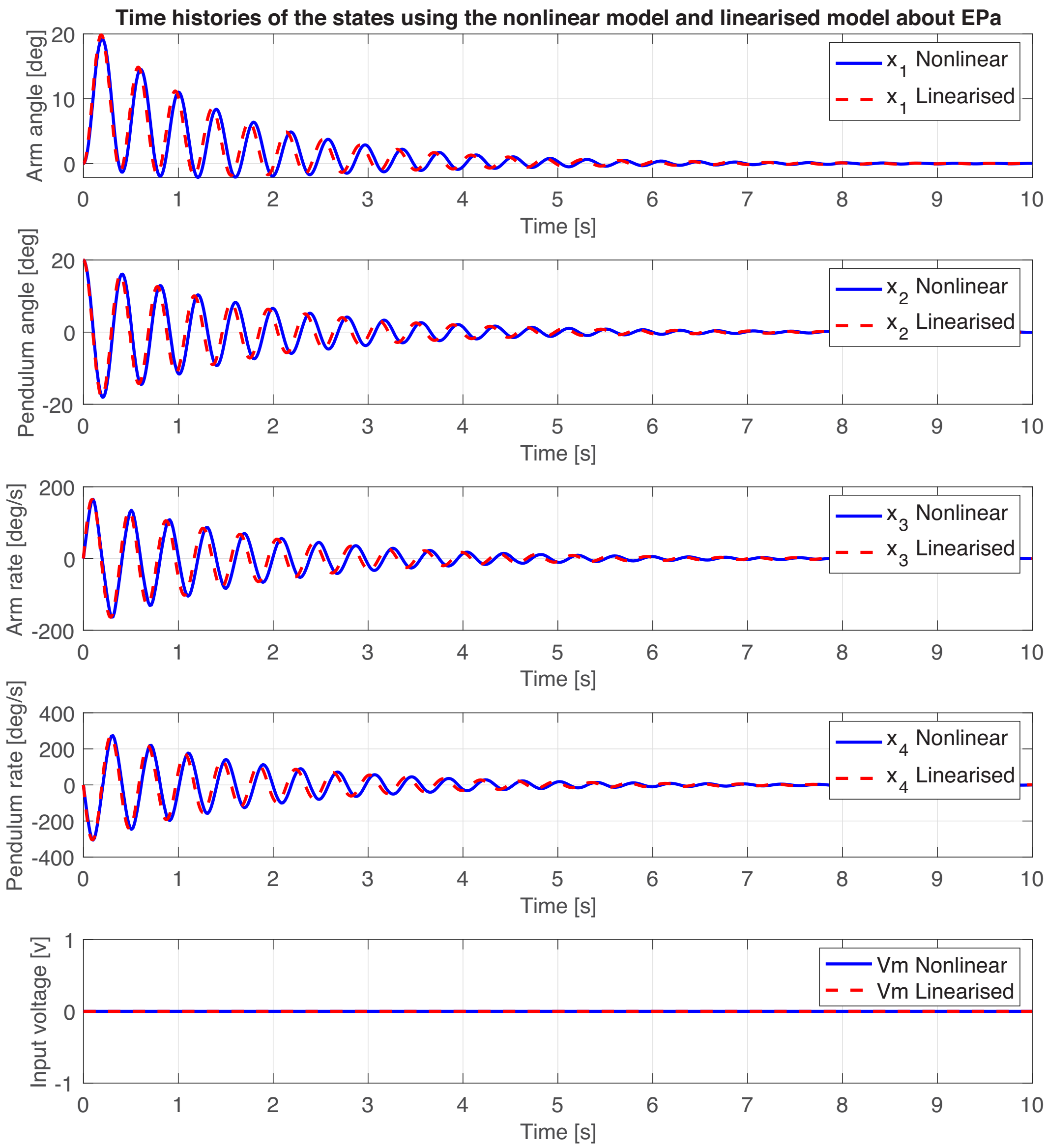


Figure 3: Time histories of the states and input voltage.

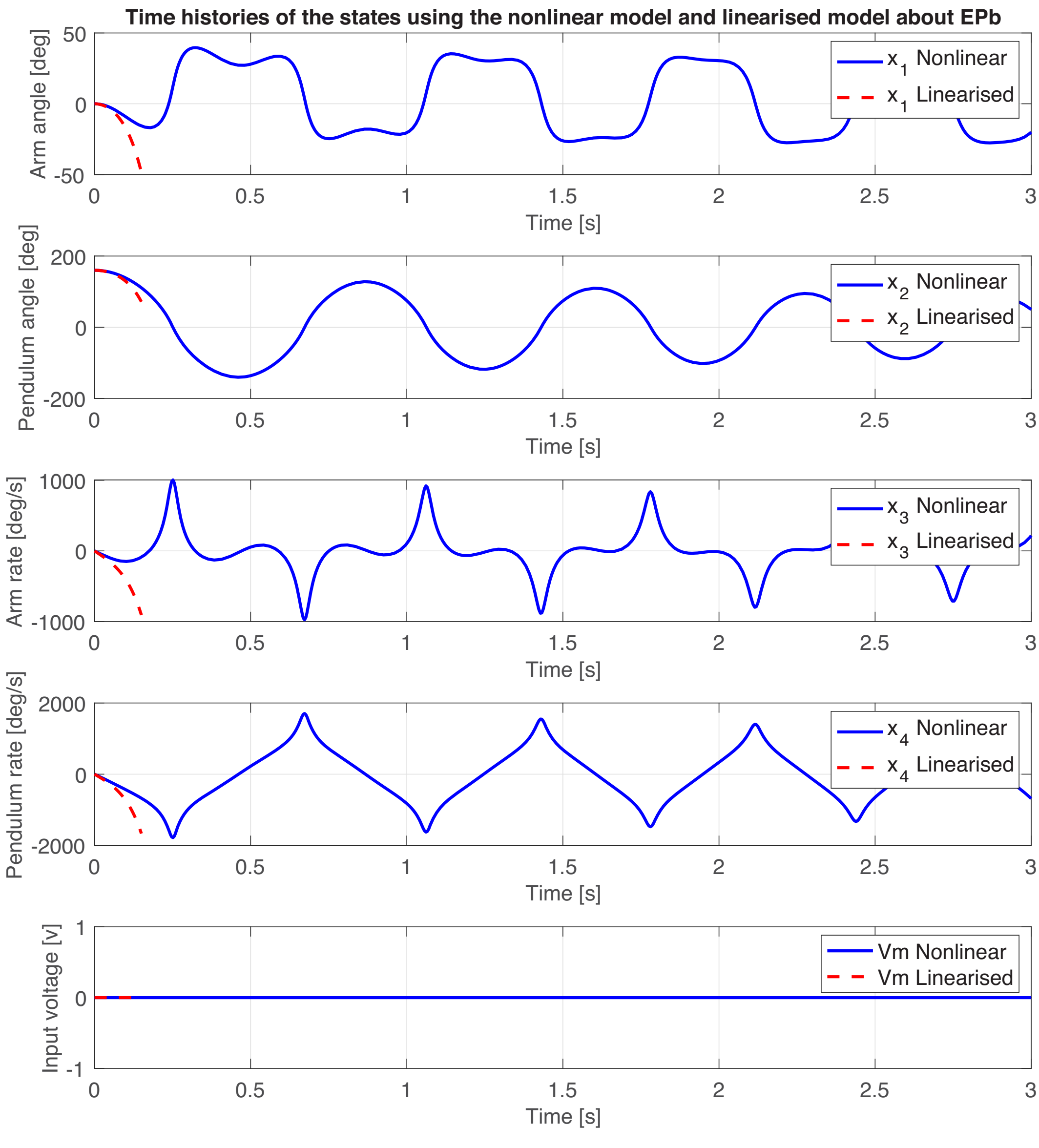


Figure 4: Time histories of the states and input voltage.

3.5 Marking Rubric

The following items will be verified in the assessment:

1. Correct output figure upon execution of the script `rp_mainfile_modelling_comparison_a.m` and `rp_mainfile_modelling_comparison_b.m`

2. Whether your functions `rp_parameters.m`, `rp_plot.m`, `rp_nl_model.m`, `rp_lin_model.m` returns the correct expected values.

3. Whether your Simulink models `rp_modelling_nl.slx` and `rp_modelling_lin.slx` are correct and the simulation are correctly performed.

Please add your name and student number to all scripts and functions, for example see the following script and function

rp_mainfile_modelling_comparison_a.m

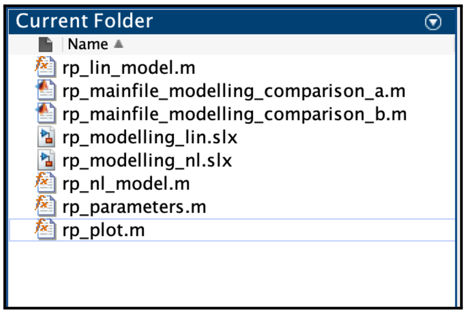
```
%%% Name: Alejandro Donaire
%%% Student Number: c1234567

close all
clear all
%clc

%% Nolinear and linearised model parameters

rp_p = rp_parameters();

%% Simulation parameters

rp_p.ic=[0;20*pi/180;0;0];
rp_p.iclin=rp_p.ic-rp_p.xbar;
rp_p.simtime=10;
rp_p.simtimelin=10;

rp_p.xbar=rp_p.xbara;
rp_p.ubar=rp_p.ubara;
rp_p.ybar=rp_p.ybara;
rp_p.A=rp_p.Aa;
rp_p.B=rp_p.Ba;
rp_p.C=rp_p.Ca;


%% Simulation nonlinear model

sim_nl=sim('rp_modelling_nl');

%% Simulation linear model

sim_lin=sim('rp_modelling_lin');

%%

rp_title='Time histories of the states using the nonlinear model and linearised model about EPa';
rp_plot(sim_nl,sim_lin,rp_title)

rp_animation(sim_nl.t,sim_nl.x(:,1),sim_nl.x(:,2))

```

rp_nl_model.m

```
%%% Name: Alejandro Donaire
%%% Student Number: c1234567

function dx=rp_nl_model(in,p)

%% Model input and state variables

vm    = in(1);
x1    = in(2);
x2    = in(3);
x3    = in(4);
x4    = in(5);


%% Dynamics equations

Delta = (4*p.Jp+p.Lp^2*p.mp)*(4*p.Jr+p.mp*(p.Lp^2+4*p.Lr^2)) ...
         - p.Lp^2*p.mp*cos(x2)^2*(4*p.Jp+p.mp*(p.Lp^2+4*p.Lr^2));

dx1 = x3;

dx2 = x4;

dx3 = (p.Lp*p.mp*sin(x2)/Delta) * ( -2*x3^2*p.Lp^2*p.Lr*p.mp*cos(x2)^2 - ...
       2*x3*x4*p.Lp*cos(x2)*(4*p.Jp+p.Lp^2*p.mp) ...
       + 2*x4^2*p.Lr*(4*p.Jp+p.Lp^2*p.mp) + 4*p.g*p.Lp*p.Lr*p.mp*cos(x2) ) ...
       - 4*p.km^2*(4*p.Jp+p.Lp^2*p.mp)*x3/(p.Rm*Delta) ...
       + 4*p.km*(4*p.Jp+p.Lp^2*p.mp)*vm/(p.Rm*Delta);

dx4 = (p.Lp*p.mp*sin(x2)/Delta) * ( p.Lp*cos(x2)*( x3^2*(4*p.Jr+p.Lp^2*p.mp+4*p.Lr^2*p.mp) ...
       - 4*x4^2*p.Lr^2*p.mp ) - x3^2*p.Lp^3*p.mp*cos(x2)^3 ...
       + 2*p.Lp^2*p.mp*cos(x2)^2*(2*x3*x4*p.Lr+p.g) ...
       - 2*p.g*(4*p.Jr+p.mp*(p.Lp^2+4*p.Lr^2)) ) ...
       + 8*p.km^2*p.Lp*p.Lr*p.mp*cos(x2)*x3/(p.Rm*Delta) ...
       - 8*p.km*p.Lp*p.Lr*p.mp*cos(x2)*vm/(p.Rm*Delta);

%% State derivative vector

dx=[dx1;dx2;dx3;dx4];
```

3.6 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:



4 2 DOF aero system

In this section we describe the dynamics of the 2 DOF aero system. Note that the modelling assumptions are implicit in the idealised model.

4.1 Nonlinear model

The idealised model of the 2 DOF aero system is shown in Figure 5, and its dynamics can be written in the Euler-Lagrange form as follow

$$J_p\ddot{q}_1 + D_p\dot{q}_1 + K_{sp}\sin(q_1) = \tau_p, \tag{16}$$
$$J_y\ddot{q}_2 + D_y\dot{q}_2 = \tau_y, \tag{17}$$

where q_1 is the pitch angle and q_2 is the yaw angle of the system. The torques in pitch and yaw generated by the rotors are

$$\tau_p = K_{pp}V_p + K_{py}V_y \tag{18}$$
$$\tau_y = K_{yp}V_p + K_{yy}V_y \tag{19}$$

where, V_p and V_y are the voltages applied to the rotors, and the gains K_{ij}, with $i, j = \{p, y\}$, represent the coupling from rotor voltges to torques. The parameters of the model are given in Table 2.

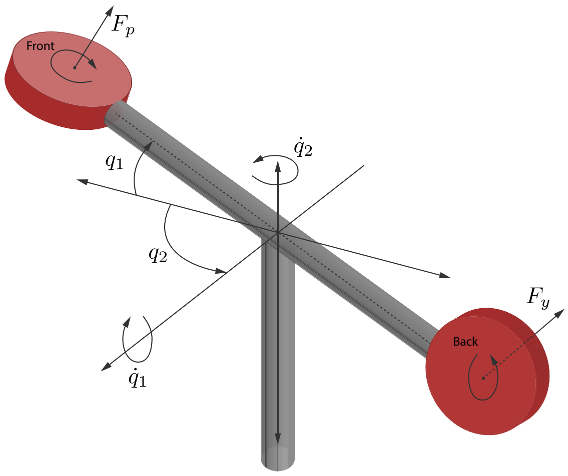


Figure 5: Idealised model of the aero system.

Table 2: Model parameters.

	Description	Parameter	Value
Pitch			
	Moment of inertia	J_p	0.0219 kg·m^2
	Damping coefficient	D_p	0.00711 N·m·s/rad
	Restoring constant	K_{sp}	0.0375 N·m/rad
Yaw			
	Moment of inertia	J_y	0.022 kg·m^2
	Damping coefficient	D_y	0.022 N·m·s/rad
Rotor			
	Pitch thrust gain	K_{pp}	0.0011 N·m/v
	Yaw thrust gain	K_{yy}	0.0022 N·m/v
	Cross-thrust gain	K_{py}	0.0021 N·m/v
	Cross-thrust gain	K_{yp}	-0.0027 N·m/v

We select the states $x_1 = q_1$, $x_2 = q_2$, $x_3 = \dot{q}_1$, $x_4 = \dot{q}_2$ and we formulate the model

(16),(17),(18),(19) in state-space form

$$\begin{aligned}
\dot{x}_1 &= x_3 && (20)\\
\dot{x}_2 &= x_4 && (21)\\
\dot{x}_3 &= -\frac{K_{sp}}{J_p}\sin(x_1) - \frac{D_p}{J_p}x_3 + \frac{K_{pp}}{J_p}V_p + \frac{K_{py}}{J_p}V_y && (22)\\
\dot{x}_4 &= -\frac{D_y}{J_y}x_4 + \frac{K_{yp}}{J_y}V_p + \frac{K_{yy}}{J_y}V_y && (23)
\end{aligned}$$

4.2 Linearised model

In this section, we present linear approximation models about the desired equilibrium points to be stabilised. For the 2 DOF aero system we consider two equilibrium points

$$\bar{x}_a = \begin{bmatrix} \bar{x}_{1a} \\ \bar{x}_{2a} \\ \bar{x}_{3a} \\ \bar{x}_{4a} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad ; \qquad \bar{V}_{pa} = 0\ ; \qquad \bar{V}_{ya} = 0, \tag{24}$$

and

$$\bar{x}_b = \begin{bmatrix} \bar{x}_{1b} \\ \bar{x}_{2b} \\ \bar{x}_{3b} \\ \bar{x}_{4b} \end{bmatrix} = \begin{bmatrix} \frac{\pi}{9} \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad ; \qquad \bar{V}_{pb} = \frac{K_{yy}}{K_T}\,\bar{\tau}_{pb}\ ; \qquad \bar{V}_{yb} = -\frac{K_{yp}}{K_T}\,\bar{\tau}_{pb}, \tag{25}$$

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

The linear model about the equilibrium point $\bar{x}_a$ is characterised by the matrices

$$A_a = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{K_{sp}}{J_p} & 0 & -\frac{D_p}{J_p} & 0 \\ 0 & 0 & 0 & -\frac{D_y}{J_y} \end{bmatrix}, \qquad B_a = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{K_{pp}}{J_p} & \frac{K_{py}}{J_p} \\ \frac{K_{yp}}{J_y} & \frac{K_{yy}}{J_y} \end{bmatrix}. \tag{26}$$

Similarly, the linear model about the equilibrium point $\bar{x}_b$ is characterised by the matrices

$$A_b = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{K_{sp}}{J_p}\cos(\bar{x}_{1b}) & 0 & -\frac{D_p}{J_p} & 0 \\ 0 & 0 & 0 & -\frac{D_y}{J_y} \end{bmatrix}, \qquad B_b = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{K_{pp}}{J_p} & \frac{K_{py}}{J_p} \\ \frac{K_{yp}}{J_y} & \frac{K_{yy}}{J_y} \end{bmatrix}. \tag{27}$$

Notice that the form of the matrices C_a, D_a, C_b and D_b will depend on the selected outputs. In general, we will consider all the states or only the angles as outputs.

5 Tasks

You will need to build simulators for the aero system using the models presented in the previous section. The tasks in Section 5.1 and 5.2 are constructive to achieve the task in Section 5.3. The main objective in this lab is to build and test the correctness of the script and models used in Section 5.3.

5.1 Nonlinear Simulink model

The build a simulator for the nonlinear model of the rotary pendulum, use the following procedure:

a) Verify that the dynamics (16),(17),(18),(19) can be written in state-space form given in (20),(21),(22),(23).

1. Build a Simulink model for the aero system that returns the state vector (the angle and angular rate in pitch, the angle and angular rate in yaw), the input voltages and the simulation time to the workspace in the structure `sim_nl`: `sim_nl.x`, `sim_nl.vp`, `sim_nl.vy` and `sim_nl.t`. Save the model as `aero_modelling_nl.slx`. Hint: you may use the blocks `Interpreted Matlab Function`, `Mux` and `Demux`, and write the state equations in the function `aero_nl_model.m`.

2. Write a script that performs the following actions:

   i) Call the function `aero_parameters.m` that returns a structure named `aero_p` and assigns the parameter values of the nonlinear model of the aero system.

   ii) Set the initial conditions and simulation parameters, and simulate the model.

   iii) Call the function `aero_plot.m` that plots the states and the inputs.

   Hint: You have built a similar script in ENGG2440.

3. Save the script as `aero_mainfile_nl.m`.

4. Simulate the model using different initial conditions and analyse its behaviour.

5.2 Linearised Simulink models

Consider now the equilibrium point of the aero system noted as $\bar{x}_a$ and defined in (24).

1. Verify the linearised model of the aero system about the equilibrium point $\bar{x}_a$ is characterised by the matrices (26).

2. Build a Simulink model for the linearised model that returns the state vector, the input voltages and the simulation time to the workspace in the structure `sim_lin`: `sim_lin.xlin`, `sim_lin.vplin`, `sim_lin.vylin`, `sim_lin.tlin`. Save the model as `aero_modelling_lin.slx`. Hint: you may use the blocks `Interpreted Matlab Function`, `Mux` and `Demux`, and write the state equations in the function `aero_lin_model.m`.

3. Write a script that performs the following actions:

   i) Call the function `aero_parameters.m` that assigns the parameter values of the linearised model of the rotary pendulum (add to previous defined function).

   ii) Set the initial conditions and simulation parameters, and simulate the model.

   iii) Call the function `aero_plot.m` that plots the states and the inputs.

4. Save the script as `aero_mainfile_lin_a.m`.

5. Simulate the model using different initial conditions and analyse its behaviour.

6. Consider now the equilibrium point $\bar{x}_b$ defined in (25) and repeat the above process. Adapt all the instructions for the equilibrium point $\bar{x}_b$.

5.3 Comparison of the nonlinear and linearised models

In this section, we compare the state histories of the nonlinear model and the linear approximation. Write a MATLAB script that performs the following actions:

1. Defines the initial conditions of both models about the equilibrium point $\bar{x}_a$.

2. Simulates both the nonlinear model and the linear approximation.

3. Plots the time histories of the states from both the nonlinear system and linear approximation against each other. Plot the states $x_1(t)$ and its linear approximation $x_{1a}(t) + \bar{x}_{1a}$ on top of each other in one graph, then plot $x_2(t)$ and $x_{2a}(t) + \bar{x}_{2a}$ 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, $V_p = 0$ v and $V_y = 0$ are shown in Figure 6.

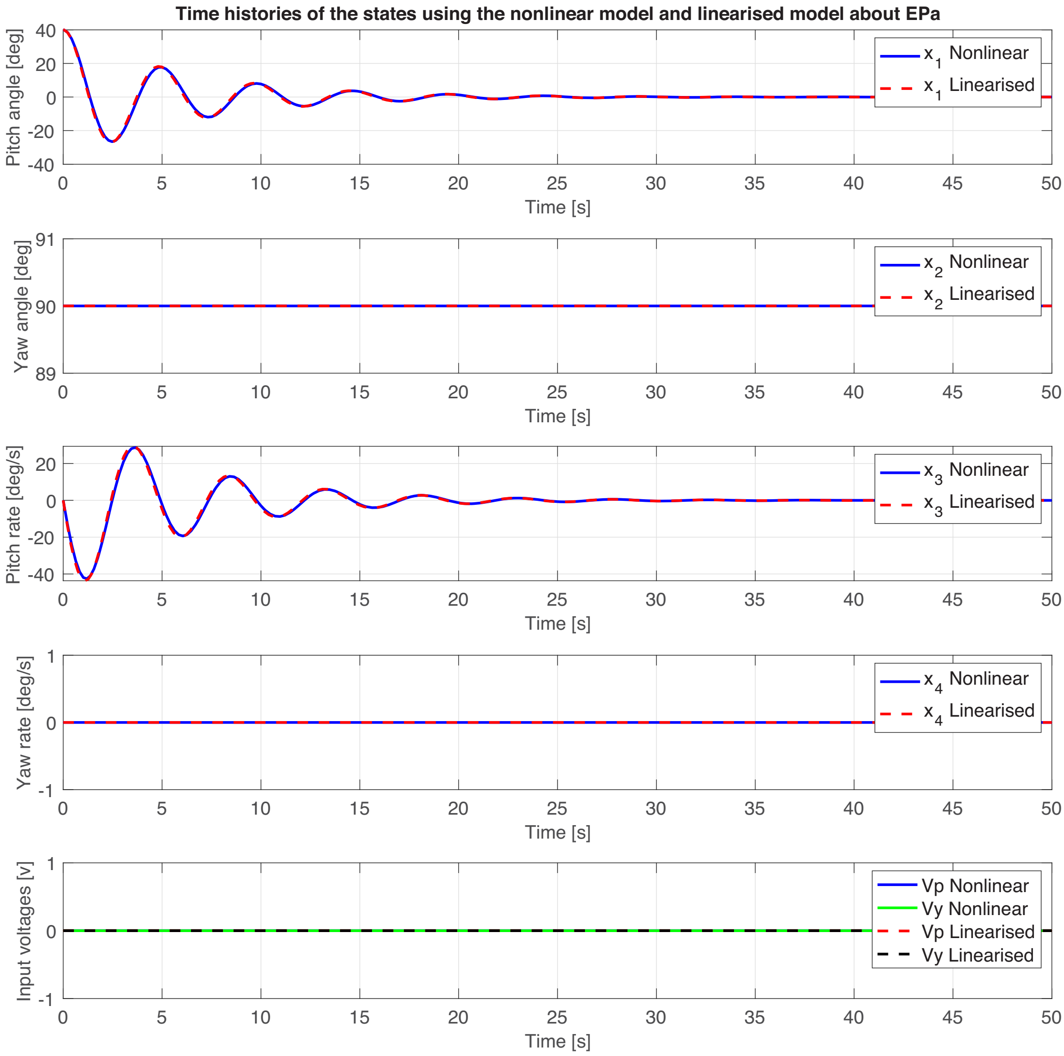


Figure 6: Time histories of the states and input voltages.

4. Save your script as `aero_mainfile_modelling_comparison_a.m`.

5. Repeat this process for initial conditions about the equilibrium point $\bar{x}_b$. Save your script as `aero_mainfile_modelling_comparison_b.m`. Sample results for the initial conditions $x_1(0) = 0$ deg, $x_2(0) = 40$ deg, $x_3(0) = 0$ deg/s and $x_4(0) = 0$ deg/s, $V_p = \bar{V}_{pb}$ and $Vy = \bar{V}_{yb}$ are shown in Figure 7.

Run simulations from different initial conditions and analyse the results.

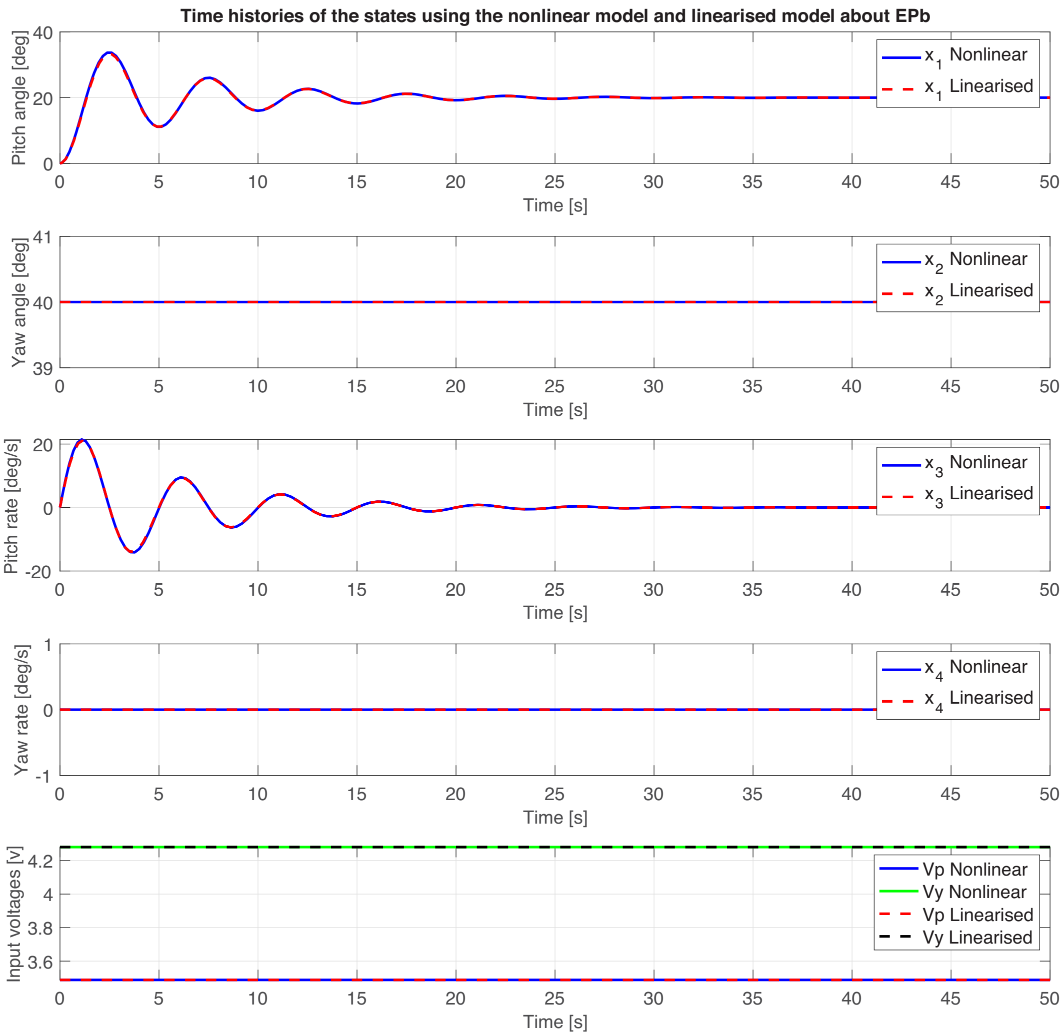


Figure 7: Time histories of the states and input voltage.

5.4 Animation

We have develop an animation for you to visualise the simulation results. To use the animation, you need to download the files `aero_animation.m` and `aero3600_logo.jpg`, and place them in the same folder where you have your main file, functions and Simulink models. The animation function takes three arguments

```
aero_animation.m(sim_nl.t,sim_nl.x(:,1),sim_nl.x(:,2))
```

where `t` is the simulation time vector, `q1` is the arm angle vector, and `q2` is the pendulum angle vector (measured from downward rest position). You might need to change `t`, `q1` and `q2`, and use the name of the variable you defined in your simulations.

5.5 Marking Rubric

The following items will be verified in the assessment:

1. Correct output figure upon execution of the script `aero_mainfile_modelling_comparison_a.m` and `aero_mainfile_modelling_comparison_b.m`
2. Whether your functions `aero_parameters.m`, `aero_plot.m`, `aero_nl_model.m`, `aero_lin_model.m` returns the correct expected values.
3. Whether your Simulink models `aero_modelling_nl.slx` and `aero_modelling_lin.slx` are correct and the simulatoin are correctly performed.

Please add your name and student number to all scripts and functions, for example see the following script aero_mainfile_modelling_comparison_a.m

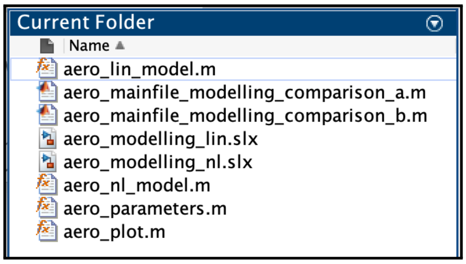
```matlab
%%% Name: Alejandro Donaire
%%% Student Number: c1234567

close all
clear all
clc

%% Nolinear and linearised model parameters

aero_p = aero_parameters();

%% Simulation parameters

aero_p.ic=[40*pi/180;90*pi/180;0;0];
aero_p.iclin=aero_p.ic-aero_p.xbar;
aero_p.simtime=50;
aero_p.simtimelin=50;

aero_p.xbar=aero_p.xbara;
aero_p.ubar(1)=aero_p.ubara(1);
aero_p.ubar(2)=aero_p.ubara(2);
aero_p.ybar=aero_p.ybara;
aero_p.A=aero_p.Aa;
aero_p.B=aero_p.Ba;
aero_p.C=aero_p.Ca;

%% Simulation nonlinear model

sim_nl=sim('aero_modelling_nl');

%% Simulation linear model

sim_lin=sim('aero_modelling_lin');

%%

aero_title='Time histories of the states using the nonlinear model and linearised model about EPa';
aero_plot(sim_nl,sim_lin,aero_title)

aero_animation(sim_nl.t,sim_nl.x(:,1),sim_nl.x(:,2))
```

aero_nl_model.m

```matlab
%%% Name: Alejandro Donaire
%%% Student Number: c1234567

function dx=aero_nl_model(input,p)

%% Model input and state variables

vp    = input(1);
vy    = input(2);
x1    = input(3);
x2    = input(4);
x3    = input(5);
x4    = input(6);

%% Dynamics equations

dx1 = x3;

dx2 = x4;

dx3 = -p.Ksp/p.Jp*sin(x1) - p.Dp*x3/p.Jp + p.Kpp*vp/p.Jp + p.Kpy*vy/p.Jp;

dx4 = -p.Dy*x4/p.Jy + p.Kyp*vp/p.Jy + p.Kyy*vy/p.Jy;

%% State derivatives

dx=[dx1;dx2;dx3;dx4];

end
```

5.6 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_modelling_comparison_a.m
aero_mainfile_modelling_comparison_b.m
aero_modelling_lin.slx
aero_modelling_nl.slx
aero_nl_model.m
aero_parameters.m
aero_plot.m