TMME50 Assignment V

David Wiman
20000120-8495

December 5, 2023

Instructions 2023 for reporting the computer assignments

The computer assignments are reported in writing, and submitted *printed on paper*. The assignments are performed individually. The use of ChatGPT or any similar system is not allowed. It is permissible to discuss the assignments and to show parts of solutions in that context, but *copying of Matlab code or sections of reports is not allowed.* Further, it is not allowed to possess copies of other students reports or Matlab code, either electronically or on paper, or to supply this to another student; this also means that you hand in and pick up your assignments yourself, not with the help of a friend. The reports shall contain:

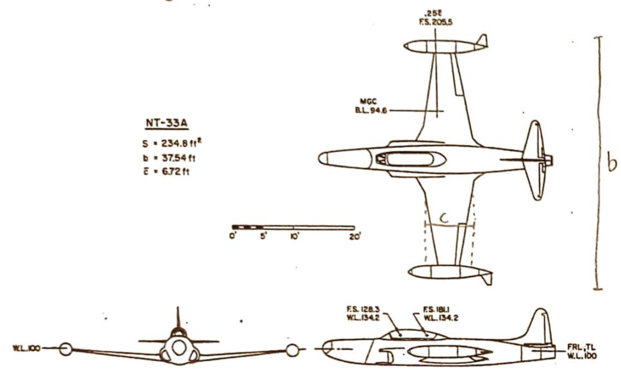
- *A copy of this page of instructions.*
- Name and complete (10 digits) civic registration number of the student (sometimes called p-number among exchange students).
- *Which aeroplane and which reference condition* that has been used. Specify the number of the column on the data sheet that has been used.
- Answers to all the questions appearing under the headings "Assignment I:a" etc. and all requested plots.
- A complete set of Matlab files for each computer assignment. Choose the most complete set, such as the one for part I:c in assignment I. In assignment II, also also include root loci and a graphic representation of your Simulink model for the final version of your model with all numerical values shown explicitly.
- The ODE system implemented in assignments I, III, IV and V must be given in the report in the order actually implemented and written in a *single* frame containing *all* the equations of the ODE and *nothing* else.

Further, note:

- With the exeption of flying qualities tables, illustrations from the lab-PM defining the problem and this page of instructions, no copying of text, figures, equations or code from another document is allowed (unless it is a document you have created yourself).
- It must be clear what data has been used in what way. Data is converted from American to SI units, and this should be done in a way that can be followed in detail either in the text of the report or in the Matlab files, so that misstakes can be found at a glance without making any calculations.
- Nothing written in Matlab syntax or pseudocode is allowed in the main text of the reports: your Matlab code is appended at the end of the report.
- The report must contain sufficient text and illustrations such that it is possible to understand without ever having seen the lab-PM.
- Use the simulation time given in the assignments. For a small number of datasets it is necessary to use a longer simulation time than 100 s in order to to see a full phugoid period, but the time should never be shorter than the time given and never longer than 400 s.

T-33 Shooting Star

David Wiman



h
V, u_0
m
I_x
I_y
I_z
I_{xz}
Q

surface, span, chord

$S = 234{,}8\ ft^2$, $b = 37{,}54\ ft$, $\bar{c} = 6{,}72\ ft$

	1	2	3	4	5	6	7	8
h ft	S/L	S/L	S/L	S/L	20000	20000	20000	40000
v(u_0) ft/s	228	270	447	782	311	570	778	629
m slug	367	426	426	426	426	426	426	426
I_x slug·ft²	12700	23801	23801	23801	23801	23801	23801	23801
I_y slug·ft²	20700	21101	21101	21101	21101	21101	21101	21101
I_z slug·ft²	32001	43802	43802	43802	43802	43802	43802	43802
I_{xz} slug·ft²	480	480	480	480	480	480	480	480
Q lb/ft²	61,7	86,7	247	726	61,3	206	383	117

	1	2	3	4	5	6	7	8
X_u s^{-1}	−0,0391	−0,00484	−0,0104	−0,0415	0,00477	−0,00735	−0,0511	−0,0355
X_α ft/s²	18,58	35,37	25,12	−16,50	20,43	22,29	7,67	24,59
Z_u s^{-1}	−0,248	−0,153	−0,128	−0,162	−0,114	−0,107	−0,0703	−0,0766
Z_α ft/s²	−213,41	−267,57	−773,31	−2807	−140,26	−712,5	−1400	−437,8
M_u $ft^{-1}s^{-1}$	0,000318	0,000603	0,000283	−0,00076	0,000114	−0,000183	−0,00151	−0,000183
M_α s^{-2}	−1,89	−1,81	−9,21	−33,7	−0,23	−9,95	−18,59	−5,42
$M_{\dot\alpha}$ s^{-1}	−0,35	−0,40	−0,63	0	−0,24	−0,31	−0,16	−0,06
M_q s^{-1}	−0,694	−0,806	−1,37	−2,80	−0,50	−0,981	−1,56	−0,535
X_{δ_e} ft/s²	0,516	1,47	0,62	−2,65	1,88	0,50	−0,432	0,996
Z_{δ_e} ft/s²	−13,4	−16,2	−44,4	−152	−11,3	−40,9	−82,4	−23,8
M_{δ_e} s^{-2}	−4,19	−5,83	−16,0	−52,7	−4,13	−14,2	−28,7	−8,28

	1	2	3	4	5	6	7	8
Y_β ft/s²	−28,4	−30,1	−81,0	−264	−21,6	−72,7	−144	−42,4
L_β s^{-2}	−5,49	−4,72	−8,02	−18,0	−4,06	−7,42	−9,89	−5,08
L_p s^{-1}	−2,03	−1,32	−2,15	−4,51	−0,82	−1,56	−2,23	−0,877
L_r s^{-1}	0,641	0,305	0,320	0,495	0,214	0,256	0,328	0,179
N_β s^{-2}	0,667	0,94	2,71	10,6	0,54	2,60	6,24	1,68
N_p s^{-1}	−0,116	−0,112	−0,0512	0,0118	−0,103	−0,0393	−0,0141	−0,0428
N_r s^{-1}	−0,207	−0,173	−0,291	−0,561	−0,104	−0,204	−0,318	−0,110
Y_{δ_r} ft/s²	0,0295	0,0301	0,0503	0,102	0,0185	0,0363	0,0571	0,0195
L_{δ_r} s^{-2}	−0,0125	0,443	1,57	5,89	0,287	1,39	3,20	0,408
N_{δ_r} s^{-2}	−1,24	−1,25	−3,50	−12,6	−0,883	−3,21	−6,99	−1,92
L_{δ_a} s^{-2}	6,01	4,53	12,6	47,0	3,14	11,7	24,0	7,13
N_{δ_a} s^{-1}	0,0286	0,134	0,165	0,260	0,164	0,121	0,195	0,118

Background

In this assignment, the full equations of motion for a rigid body moving in three dimensions are to be implemented, as well as the gravitational forces and the linear model for aerodynamic forces.

The force model is extended from assignment I with lateral forces. The reference state is given by all phase variables being equal to zero except $u_0 = V$ and $z_f^0 = -H$. Reference is indicated with a sub- or super-script 0.

The data sheet for the tasks at hand were given in imperial units and must therefore be converted to SI units before use. Two of the values given were:

$$\begin{aligned} V &= 228 \text{ ft/s} \cdot 0.3048 = 69.4944 \text{ m/s}, \\ H &= 100 \text{ m}. \end{aligned} \tag{1}$$

I use data from a T-33 Shooting Star at sea level flight (column one on the data sheet).

The model for the aerodynamical forces is:

$$\begin{aligned} X/m &= g\sin(\theta_0) + X_u(u-u_0) + X_w(w-w_0) \\ Y/m &= -\cos(\theta_0)\sin(\varphi_0) + Y_v(v-v_0) \\ Z/m &= -g\cos(\theta_0)\cos(\varphi_0) + Z_u(u-u_0) + Z_w(w-w_0) \\ L/I_{xx} &= L_v(v-v_0) + L_p(p-p_0) + L_r(r-r_0) \\ M/I_{yy} &= M_w(w-w_0) + M_q(q-q_0) \\ N/I_{zz} &= N_v(v-v_0) + N_p(p-p_0) + N_r(r-r_0) \end{aligned} \tag{2}$$

V:a

We begin by writing down the twelve ordinary differential equations governing the problem.

$$\begin{aligned} \dot{u} &= -qw + rv - g\sin(\theta) + X/m \\ \dot{v} &= -ru + pw + g\sin(\varphi)\cos(\theta) + Y/m \\ \dot{w} &= qu - pv + g\cos(\varphi)\cos(\theta) + Z/m \end{aligned} \tag{3}$$

$$\begin{aligned} \dot{p} &= \frac{1}{1-\frac{I_{xx}I_{zz}}{I_{xz}^2}}\left(qrI_{zz}\frac{I_{zz}-I_{yy}}{I_{xz}^2} - pq\frac{I_{zz}}{I_{xz}} - L\frac{I_{zz}}{I_{xz}^2} + pq\frac{I_{yy}-I_{xx}}{I_{xz}} + qr - N/I_{xz}\right) \\ \dot{q} &= -rp\frac{I_{xx}-I_{zz}}{I_{yy}} - (p^2-r^2)\frac{I_{xz}}{I_{yy}} + M/I_{yy} \\ \dot{r} &= \dot{p}\frac{I_{xx}}{I_{xz}} + qr\frac{I_{zz-I_{yy}}}{I_{xz}} - pq - L/I_{xz} \end{aligned} \tag{4}$$

$$\begin{aligned} \dot{\varphi} &= p + q\tan(\theta)\sin(\varphi) + r\tan(\theta)\cos(\varphi) \\ \dot{\theta} &= q\cos(\varphi) - r\sin(\varphi) \\ \dot{\psi} &= q\frac{\sin(\varphi)}{\cos(\theta)} + r\frac{\cos(\varphi)}{\cos(\theta)} \end{aligned} \tag{5}$$

$$\begin{aligned} \dot{x}_f &= u\cos(\theta)\cos(\psi) - v(-\cos(\varphi)\sin(\psi) + \sin(\varphi)\sin(\theta)\cos(\psi)) + w(\sin(\varphi)\sin(\psi) + \cos(\varphi)\sin(\theta)\cos(\psi)) \\ \dot{y}_f &= u\cos(\theta)\sin(\psi) + v(\cos(\varphi)\cos(\psi) + \sin(\varphi)\sin(\theta)\sin(\psi)) + w(-\sin(\varphi)\cos(\psi) + \cos(\varphi)\sin(\theta)\sin(\psi)) \\ \dot{z}_f &= -u\sin(\theta) + v\sin(\varphi)\cos(\theta) + w\cos(\varphi)\cos(\theta) \end{aligned} \tag{6}$$

V:b

When these equations are implemented in Matlab and a simulation is run for 100 s with the initial conditions

$$
\begin{aligned}
u_i &= V \\
\theta_i &= 0.1 \text{ rad} \\
z_f^i &= -H \\
\text{all others } &= 0
\end{aligned}
\tag{7}
$$

we get the following graph for all state variables, see Figure 1. We also get a 3D plot of the position of the aeroplane, see Figure 2.

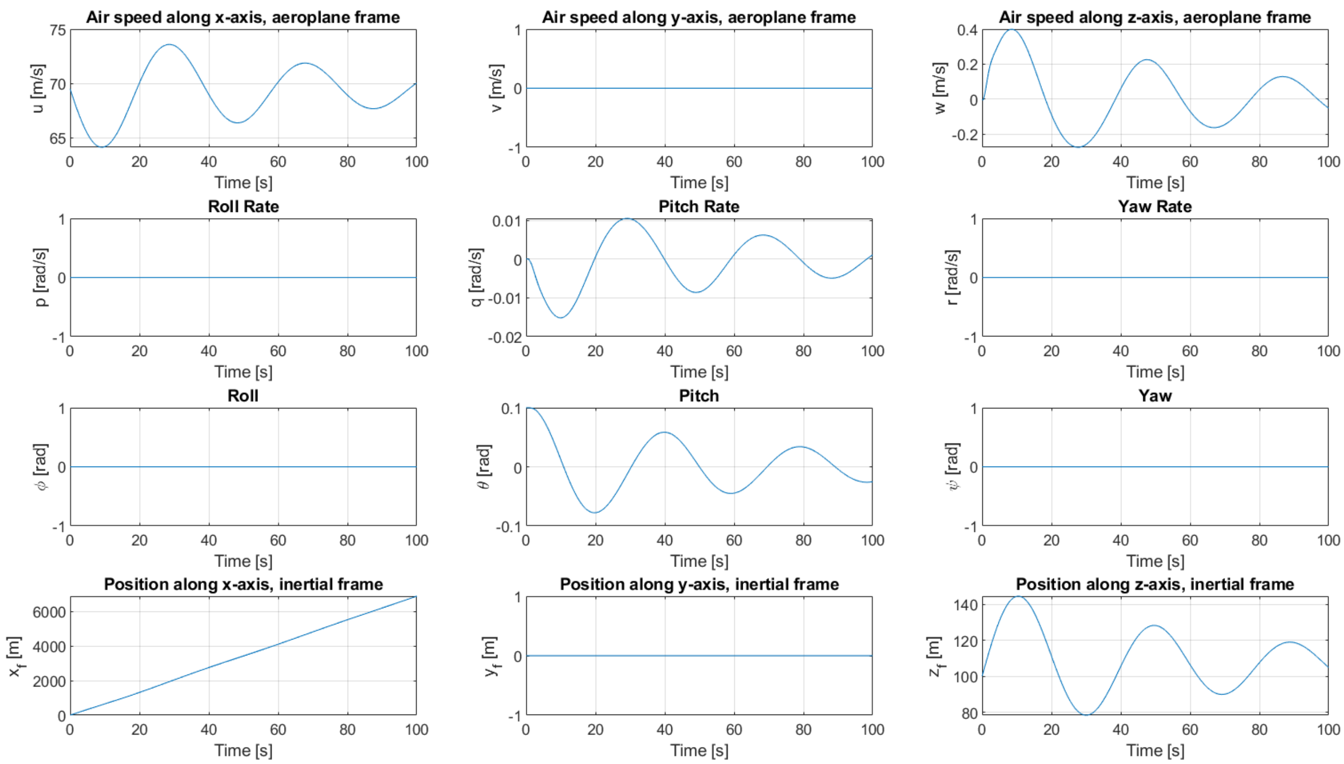


Figure 1: All phase variables.

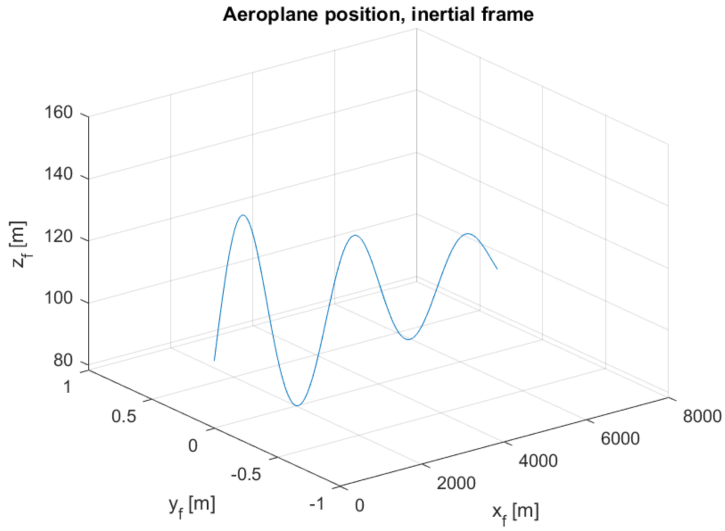


Figure 2: Position of the aeroplane.

If we compare this with the result from assignment I:c, marked with dashed red in Figures 3 and 4, we can see that there is no difference. This is an expected result since the longitudinal quantities do not spill over into lateral motion. Even though we have added lateral motion into our models, nothing we do triggers the aeroplane to leave the xz-plane.

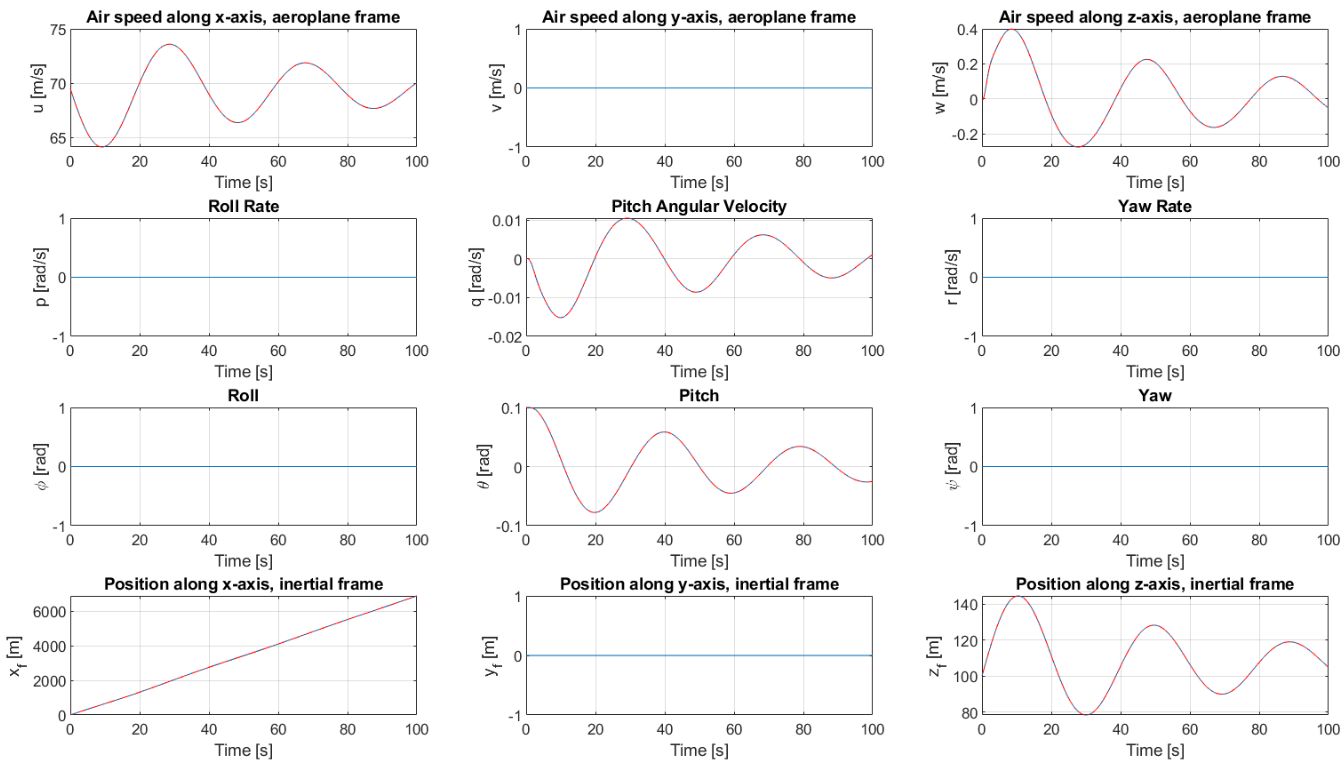


Figure 3: All phase variables with some compared to the result from assignment I:c.

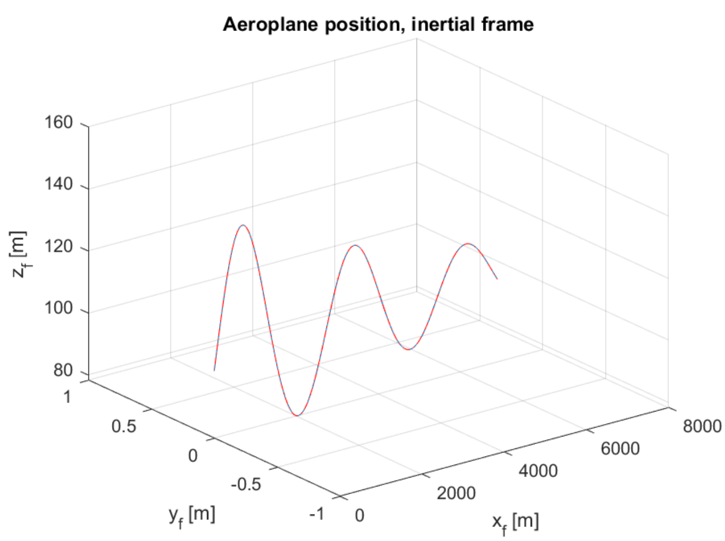


Figure 4: Position of the aeroplane.

V:c

If we let $\theta_i = 0$ but instead set $p_i, q_i, r_i = 0.1$ rad one at the time and simulate for 10 s, we get the results in Figures 5 and 6.

If we compare the results in the position graph, we can see that $q_i = 0.1$ stays in the xz-plane while the others pull to the side.

As for the phase variables, q does not cause a roll or yaw compared to the other two.

We can see that longitudinal motion does not spill over into lateral motion, but lateral motion does cause longitudinal motion. This can also be seen by the mixed longitudinal and lateral quantities in the differential- and force equations.

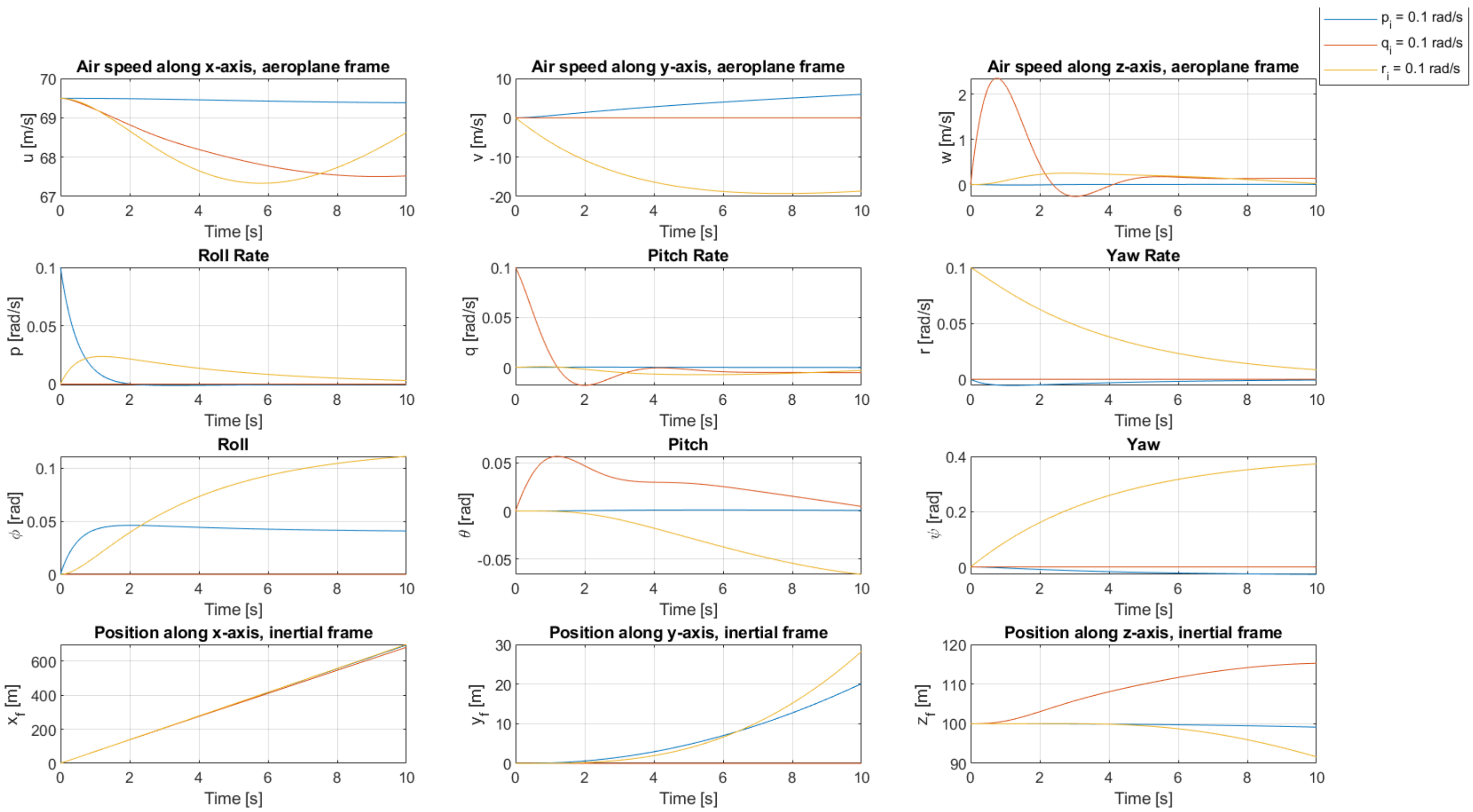


Figure 5: All phase variables compared with different initial conditions.

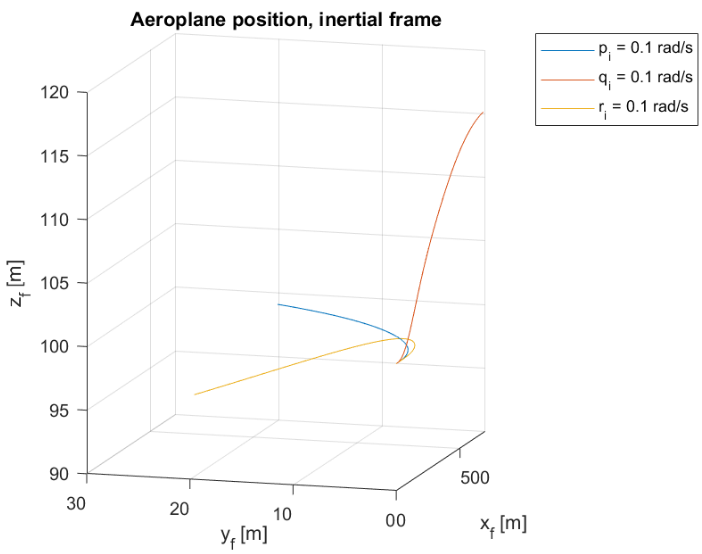


Figure 6: Position of the aeroplane with different initial conditions.

1 Matlab code

1.1 Assignment V:b

```matlab
% Set variables
run('T33_Shooting_Star_parameters.m');

% [u, v, w, p, q, r, phi, theta, psi, x_f, y_f, z_f]

% Reference state around which linearization happens
reference_state = [v, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -h];

% Set ODE options
initia_theta = 0.1;
initial_conditions = [v, 0, 0, 0, 0, 0, 0, initia_theta, 0, 0, 0, -h];
time_interval = [0, 100];

% Compute ODE
[time, state] = ode45(@(t, state) state_propagation(t, state, reference_state)
    , time_interval, initial_conditions);

% Extract all states for plotting
u = state(:,1);
v = state(:,2);
w = state(:,3);
p = state(:,4);
q = state(:,5);
r = state(:,6);
phi = state(:,7);
theta = state(:,8);
psi = state(:,9);
x_f = state(:,10);
y_f = state(:,11);
z_f = state(:,12);

%% Plot
figure(1)

% Air speeds
subplot(4,3,1)
plot(time, u);
title('Air speed along x-axis, aeroplane frame')
xlabel('Time [s]')
ylabel('u [m/s]')
grid
hold on

subplot(4,3,2)
plot(time, v);
title('Air speed along y-axis, aeroplane frame')
xlabel('Time [s]')
ylabel('v [m/s]')
grid

```

```
subplot(4,3,3)
plot(time, w);
title('Air speed along z-axis, aeroplane frame')
xlabel('Time [s]')
ylabel('w [m/s]')
grid
hold on

% Turn rates
subplot(4,3,4)
plot(time, p);
title('Roll Rate')
xlabel('Time [s]')
ylabel('p [rad/s]')
grid

subplot(4,3,5)
plot(time, q);
title('Pitch Rate')
xlabel('Time [s]')
ylabel('q [rad/s]')
grid
hold on

subplot(4,3,6)
plot(time, r);
title('Yaw Rate')
xlabel('Time [s]')
ylabel('r [rad/s]')
grid

% Euler angles
subplot(4,3,7)
plot(time, phi);
title('Roll')
xlabel('Time [s]')
ylabel('\phi [rad]')
grid

subplot(4,3,8)
plot(time, theta);
title('Pitch')
xlabel('Time [s]')
ylabel('\theta [rad]')
grid
hold on

subplot(4,3,9)
plot(time, psi);
title('Yaw')
xlabel('Time [s]')
ylabel('\psi [rad]')
grid

```

```
% Position
subplot(4,3,10)
plot(time, x_f);
title('Position along x-axis, inertial frame')
xlabel('Time [s]')
ylabel('x_f [m]')
grid
hold on

subplot(4,3,11)
plot(time, y_f);
title('Position along y-axis, inertial frame')
xlabel('Time [s]')
ylabel('y_f [m]')
grid

subplot(4,3,12)
plot(time, -z_f);
title('Position along z-axis, inertial frame')
xlabel('Time [s]')
ylabel('z_f [m]')
grid
hold on


figure(2)
plot3(x_f, y_f, -z_f)
title('Aeroplane position, inertial frame')
xlabel('x_f [m]')
ylabel('y_f [m]')
zlabel('z_f [m]')
grid
hold on
```

1.2 Assignment V:b, state propagation

```
% Function to propagate the state
function next_state = state_propagation(t, state, reference_state)

    % I don't want to pass too many variables to this function
    run('T33_Shooting_Star_parameters.m');

    % Rename state components for readability
    u = state(1);
    v = state(2);
    w = state(3);

    p = state(4);
    q = state(5);
    r = state(6);

    phi = state(7);
    theta = state(8);
```

```
    psi = state(9);

    x_f = state(10);
    y_f = state(11);
    z_f = state(12);

    % Extract reference state
    u0 = reference_state(1);
    v0 = reference_state(2);
    w0 = reference_state(3);

    p0 = reference_state(4);
    q0 = reference_state(5);
    r0 = reference_state(6);

    phi_0 = reference_state(7);
    theta_0 = reference_state(8);
    psi_0 = reference_state(9);

    x_f_0 = reference_state(10);
    y_f_0 = reference_state(11);
    z_f_0 = reference_state(12);

    % Forces
    X = m*g*sin(theta_0) + m*Xu*(u - u0) + m*Xw*(w - w0);
    Y = -m*g*cos(theta_0)*sin(phi_0) + m*Yv*(v - v0);
    Z = -m*g*cos(theta_0)*cos(phi_0) + m*Zu*(u - u0) + m*Zw*(w - w0);

    % Moments
    L = I_xx*Lv*(v - v0) + I_xx*Lp*(p - p0) + I_xx*Lr*(r - r0);
    M = I_yy*Mw*(w - w0) + I_yy*Mq*(q - q0);
    N = I_zz*Nv*(v - v0) + I_zz*Np*(p - p0) + I_zz*Nr*(r - r0);

    % State dynamics
    next_state(1) = -q*w + r*v -g*sin(theta) + X/m; % u
    next_state(2) = -r*u + p*w + g*sin(phi)*cos(theta) + Y/m; % v
    next_state(3) = q*u - p*v + g*cos(phi)*cos(theta) + Z/m; % w

    next_state(4) = (1/(1 - I_zz*I_xx/I_xz^2))*( I_zz*q*r*(I_zz - I_yy)/I_xz^2
        - I_zz*p*q/I_xz - I_zz*L/I_xz^2 + p*q*(I_yy - I_xx)/I_xz + q*r - N/
        I_xz ); % p
    next_state(5) = -r*p*(I_xx-I_zz)/I_yy - I_xz*(p^2 - r^2)/I_yy + M/I_yy; %
        q
    next_state(6) = I_xx*next_state(4)/I_xz + q*r*(I_zz - I_yy)/I_xz - p*q - L
        /I_xz; % r

    next_state(7) = p + q*tan(theta)*sin(phi) + r*tan(theta)*cos(phi); % phi
    next_state(8) = q*cos(phi) - r*sin(phi); % theta
    next_state(9) = q*sin(phi)/cos(theta) + r*cos(phi)/cos(theta); % psi

    R = [cos(theta)*cos(psi) , cos(theta)*sin(psi) , -sin(theta);
         -cos(phi)*sin(psi) + sin(phi)*sin(theta)*cos(psi) , cos(phi)*cos(psi)
             + sin(phi)*sin(theta)*sin(psi) , sin(phi)*cos(theta);
```

```
        sin(phi)*sin(psi) + cos(phi)*sin(theta)*cos(psi) , -sin(phi)*cos(psi)
            + cos(phi)*sin(theta)*sin(psi) , cos(phi)*cos(theta)];

    R_inv = inv(R);

    next_state(10) = R_inv(1,1)*u + R_inv(1,2)*v + R_inv(1,3)*w; % x_f
    next_state(11) = R_inv(2,1)*u + R_inv(2,2)*v + R_inv(2,3)*w; % y_f
    next_state(12) = R_inv(3,1)*u + R_inv(3,2)*v + R_inv(3,3)*w; % z_f

    % Transpose to get row vector
    next_state = next_state';
end
```

1.3 Assignment I:c, comparison

```
% Set variables
run('T33_Shooting_Star_parameters.m');

% Reference state around which linearization happens
reference_state = [v, 0, 0, 0, 0, h];

% Force linearization parameters, [Xu, Xw, Zu, Zw, Mw, Mq]
force_lin_param = [Xu, Xw, Zu, Zw, Mw, Mq];

% Set ODE options
initial_theta = 0.1;
initial_conditions = [v, 0, 0, initial_theta, 0, h];
time_interval = [0, 100];

% Compute ODE
[time, state] = ode45(@(t, state) state_propagation_1_c(t, state, m, I_y,
    reference_state, force_lin_param), time_interval, initial_conditions);

% Extract all states for plotting
u = state(:,1);
w = state(:,2);
q = state(:,3);
theta = state(:,4);
x_f = state(:,5);
z_f = state(:,6);

% Plot
figure(1)
subplot(4,3,1)
plot(time, u, '-.r');
title('Air speed along x-axis, aeroplane frame')
xlabel('Time [s]')
ylabel('u [m/s]')
grid on

subplot(4,3,3)
plot(time, w, '-.r');
title('Air speed along z-axis, aeroplane frame')
```

```
xlabel('Time [s]')
ylabel('w [m/s]')
grid on

subplot(4,3,8)
plot(time, theta, '-.r');
title('Pitch')
xlabel('Time [s]')
ylabel('\theta [rad]')
grid on

subplot(4,3,5)
plot(time, q, '-.r');
title('Pitch Angular Velocity')
xlabel('Time [s]')
ylabel('q [rad/s]')
grid on

subplot(4,3,10)
plot(time, x_f, '-.r');
title('Position along x-axis, inertial frame')
xlabel('Time [s]')
ylabel('x_f [m]')
grid on

subplot(4,3,12)
plot(time, z_f, '-.r');
title('Position along z-axis, inertial frame')
xlabel('Time [s]')
ylabel('z_f [m]')
grid on

figure(2)
plot3(x_f, zeros(length(x_f),1), z_f, '-.r')
title('Aeroplane position, inertial frame')
xlabel('x_f [m]')
ylabel('y_f [m]')
zlabel('z_f [m]')
grid on
```

1.4 Assignment V:c

```
% Set variables
run('T33_Shooting_Star_parameters.m');

% [u, v, w, p, q, r, phi, theta, psi, x_f, y_f, z_f]

% Reference state around which linearization happens
reference_state = [v, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -h];

% Set ODE options
initial_p = 0.0;
initial_q = 0.0;
```

```
initial_r = 0.1;
initial_conditions = [v, 0, 0, initial_p, initial_q, initial_r, 0, 0, 0, 0, 0,
    -h];
time_interval = [0, 10];

% Compute ODE
[time, state] = ode45(@(t, state) state_propagation(t, state, reference_state)
    , time_interval, initial_conditions);

% Extract all states for plotting
u = state(:,1);
v = state(:,2);
w = state(:,3);
p = state(:,4);
q = state(:,5);
r = state(:,6);
phi = state(:,7);
theta = state(:,8);
psi = state(:,9);
x_f = state(:,10);
y_f = state(:,11);
z_f = state(:,12);

%% Plot
figure(1)

% Air speeds
subplot(4,3,1)
plot(time, u);
title('Air speed along x-axis, aeroplane frame')
xlabel('Time [s]')
ylabel('u [m/s]')
grid on
hold on

subplot(4,3,2)
plot(time, v);
title('Air speed along y-axis, aeroplane frame')
xlabel('Time [s]')
ylabel('v [m/s]')
grid on
hold on

subplot(4,3,3)
plot(time, w);
title('Air speed along z-axis, aeroplane frame')
xlabel('Time [s]')
ylabel('w [m/s]')
grid on
hold on

% Turn rates
subplot(4,3,4)
plot(time, p);
```

```
title('Roll Rate')
xlabel('Time [s]')
ylabel('p [rad/s]')
grid on
hold on

subplot(4,3,5)
plot(time, q);
title('Pitch Rate')
xlabel('Time [s]')
ylabel('q [rad/s]')
grid on
hold on

subplot(4,3,6)
plot(time, r);
title('Yaw Rate')
xlabel('Time [s]')
ylabel('r [rad/s]')
grid on
hold on

% Euler angles
subplot(4,3,7)
plot(time, phi);
title('Roll')
xlabel('Time [s]')
ylabel('\phi [rad]')
grid on
hold on

subplot(4,3,8)
plot(time, theta);
title('Pitch')
xlabel('Time [s]')
ylabel('\theta [rad]')
grid on
hold on

subplot(4,3,9)
plot(time, psi);
title('Yaw')
xlabel('Time [s]')
ylabel('\psi [rad]')
grid on
hold on

% Position
subplot(4,3,10)
plot(time, x_f);
title('Position along x-axis, inertial frame')
xlabel('Time [s]')
ylabel('x_f [m]')
grid on
```

```
hold on

subplot(4,3,11)
plot(time, y_f);
title('Position along y-axis, inertial frame')
xlabel('Time [s]')
ylabel('y_f [m]')
grid on
hold on

subplot(4,3,12)
plot(time, -z_f);
title('Position along z-axis, inertial frame')
xlabel('Time [s]')
ylabel('z_f [m]')
grid on
hold on

figure(2)
plot3(x_f, y_f, -z_f)
title('Aeroplane position, inertial frame')
xlabel('x_f [m]')
ylabel('y_f [m]')
zlabel('z_f [m]')
grid on
hold on
```

1.5 Unit conversion=

```
% Constants
g = 9.81;

% Unit conversions
feet_to_meter = 0.3048;
feet2_to_m2 = 0.09290;
lb_to_kg = 0.4536;
slug_to_kg = 14.59;
slug_feet2_to_kg_m2 = 1.356;
lb_to_N = 4.448;

% Parameters in SI units
h = 100;
v = 228*feet_to_meter;
m = 367*slug_to_kg;
I_xx = 12700*slug_feet2_to_kg_m2;
I_yy = 20700*slug_feet2_to_kg_m2;
I_zz = 32001*slug_feet2_to_kg_m2;
I_xz = 480*slug_feet2_to_kg_m2;
Q = 61.7*lb_to_N/feet2_to_m2;

Xu = -0.0391;
Xalpha = 18.58*feet_to_meter;
Xw = Xalpha/v;
```

```
Zu = -0.248;
Zalpha = -213.41*feet_to_meter;
Zw = Zalpha/v;
Mu = 0.000318/feet_to_meter;
Malpha = -1.89;
Mw = Malpha/v;
Malpha_dot = -0.35;
Mw_dot = Malpha_dot/v;
Mq = -0.694;
Xdelta_e = 0.516*feet_to_meter;
Zdelta_e = -13.4*feet_to_meter;
Mdelta_e = -4.19;

Ybeta = -28.4*feet_to_meter;
Lbeta = -5.49;
Lp = -2.03;
Lr = 0.641;
Nbeta = 0.667;
Np = -0.116;
Nr = -0.207;
Ydelta_r = 0.0295*feet_to_meter;
Ldelta_r = -0.0125;
Ndelta_r = -1.24;
Ldelta_a = 6.01;
Ndelta_a = 0.0286;

S = 234.8*feet2_to_m2;
b = 37.54*feet_to_meter;
c = 6.72*feet_to_meter;

% Not found in data sheet
Yp = 0;
Yr = 0;
Yv = 0;
Lv = 0;
Nv = 0;
```