UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Study of the control for the recovery of a rocket's launch system

Document:
Annexes

Author:
Angel Pan Du

Director:
Fatiha Nejjari Akhi-Elarab

Degree:
Bachelor's degree in Aerospace Technology Enginneering

Examination session:
Spring, 2021

BACHELOR FINAL THESIS

Contents

List of Figures **ii**

A Simulink model **1**

A.1 Problem model 1

A.2 Control model 16

B Matlab codes **22**

B.1 Inputs code 22

B.2 Transfer function code 23

B.3 Plots code 24

List of Figures

1 Simulink model 2
2 Instantaneous mass calculation 3
3 Centre of mass calculation 3
4 Inertia calculation 4
5 6-DOF with custom variable mass and Euler Angles equations solver block 4
6 Forces and moments block subsystem 6
7 Wind-frame to body-frame transformation matrix 7
8 Gravitational forces in the body reference frame 7
9 Gravity calculation 8
10 Thrust forces and moments in the body reference frame 8
11 Aerodynamic forces and moments in the body reference frame . . . 9
12 Lift and drag coefficients computation 9
13 Lift and drag calculation 10
14 Cold gas thrusters forces and moments in the body reference frame 10
15 Hypersonic grid fins forces and moments in the body reference frame 11
16 Dynamic viscosity of the fluid calculation 12
17 Normal force coefficient calculation for hypersonic velocities 12
18 Normal force coefficient calculation for subsonic velocities 13
19 Axial force coefficient calculation 14
20 Skin friction drag coefficient calculation for laminar flow 15
21 Skin friction drag coefficient calculation for turbulent flow 15
22 Normal and axial forces of the grid fins calculation 15
23 Simulink controlled model 17
24 Moments PID controller 18
25 Moments MPC controller 18
26 Forces and moments block subsystem of the controlled model 20

27 Forces controller 21

A Simulink model

In this section, the different Simulink models used to accomplish the results displayed in the report are detailed.

Moreover, the constant inputs are determined with the Matlab script from section

A.1 Problem model

Firstly, the overall problem model, developed in the section 4 of the report, takes the following configuration, with its corresponding subsystems.

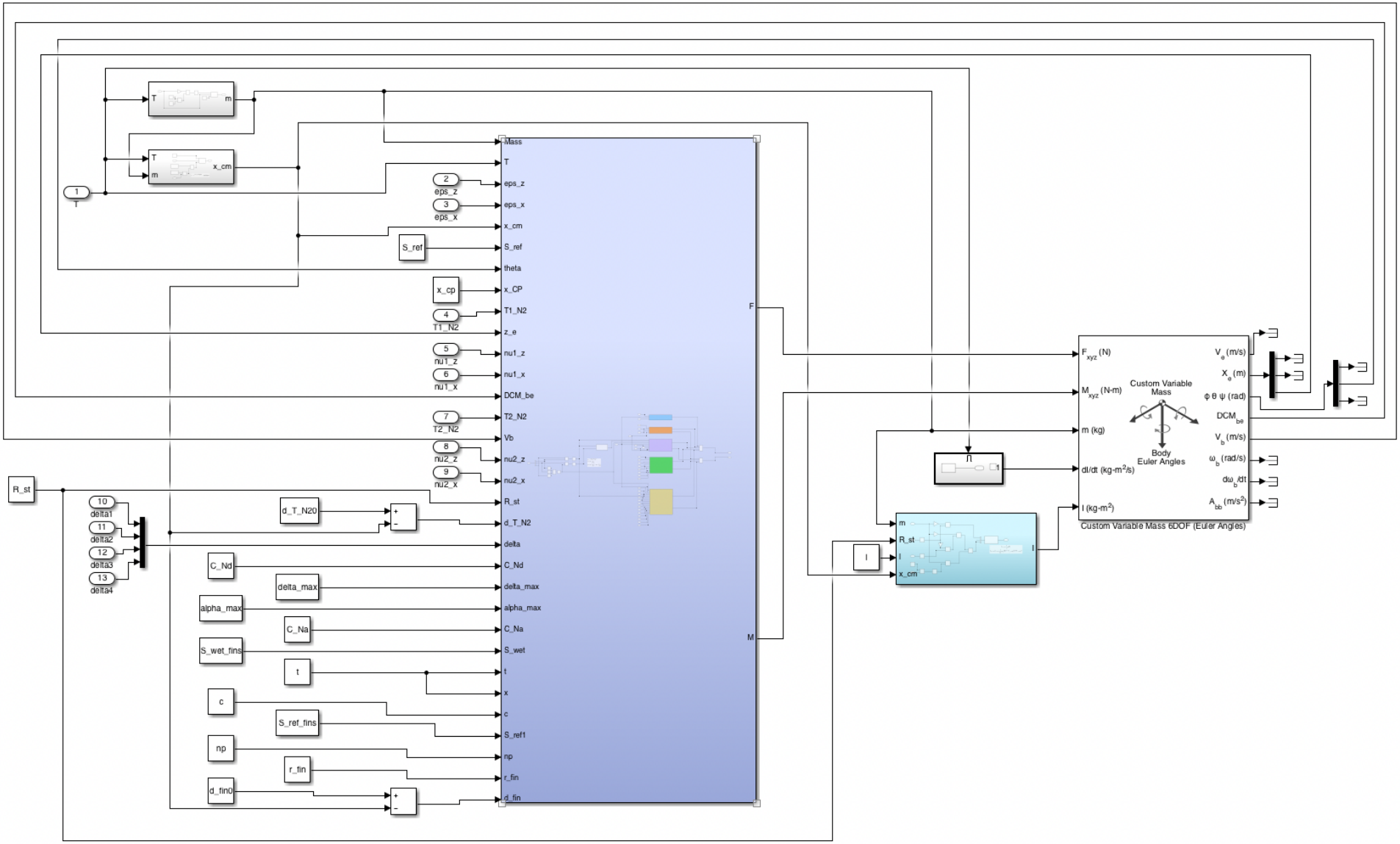

Figure 1: Simulink model

The upper left block, which calculates the mass whenever the main engines thrust are triggered (activated with a switch), is composed with the following equations:

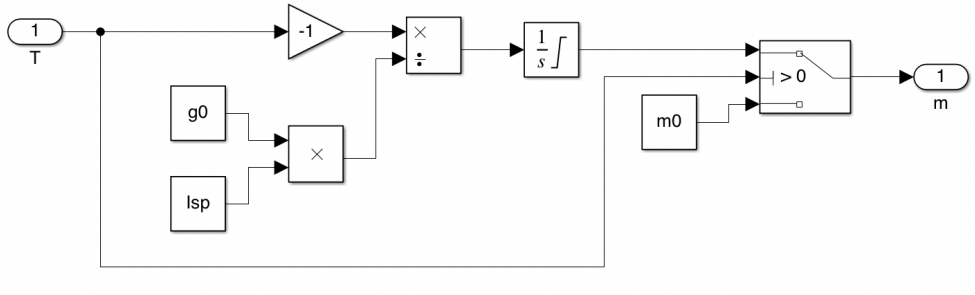

Figure 2: Instantaneous mass calculation

Its block below, also taking into account the main thrusters reactivation, computes the centre of mass of the stage as described in section 5 from the report:

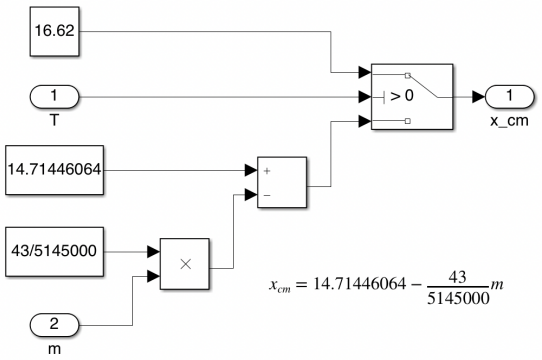

Figure 3: Centre of mass calculation

Also with this thrust parameter an enabled system is employed, which is switched on along with these engines and arranges the inertia temporal variation matrix ($\frac{dI}{dt}$), obtained by deriving its values.

The smaller light blue block, which gives the inertia matrix, has the following sketch, shaping its diagonal terms into a matrix, since it is considered that the non-diagonal ones are null, with the DSP System Toolbox.

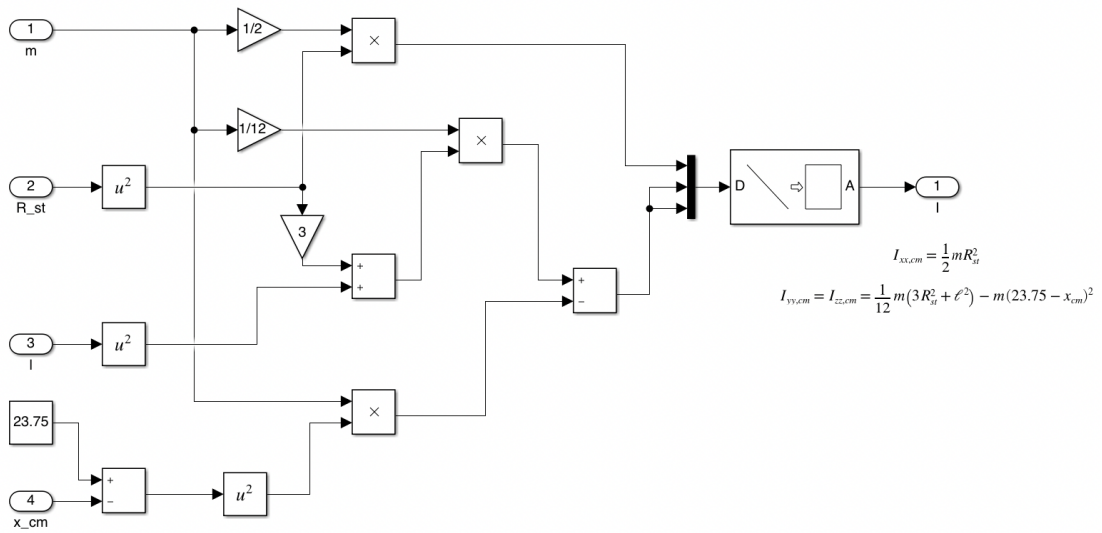

Figure 4: Inertia calculation

The white right block is extracted from the Aerospace Blockset add-on and is responsible for integrating and solving the 6-DOF (Degree Of Freedom) motion equations with a custom variable mass and through the Euler Angles.

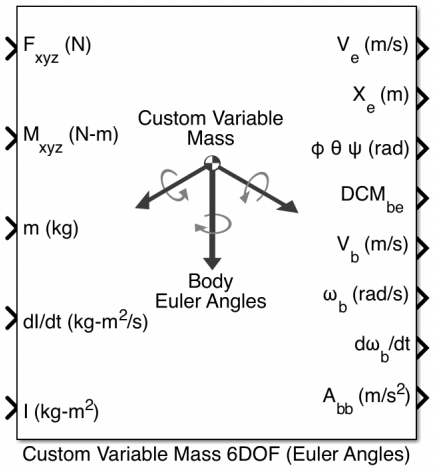

Figure 5: 6-DOF with custom variable mass and Euler Angles equations solver block

whose inputs are the blocks detailed, added to the one of the forces and moments in the body reference frame that is further defined, and the outputs are the linear velocity in the inertial frame (V_e), the position in the inertial frame (X_e),

the euler angles (ψ θ ϕ), the matrix transformation from the Earth-frame to the body-frame (DCM_{be}), the linear and angular velocity in the body reference frame (V_b and ω_b), the angular acceleration in body-fixed axes ($\mathrm{d}\omega_b/\mathrm{dt}$) and the linear acceleration with respect to inertial frame (A_{bb}), respectively in the figure order.

And the larger blue block computes the forces and moments in the body reference frame.

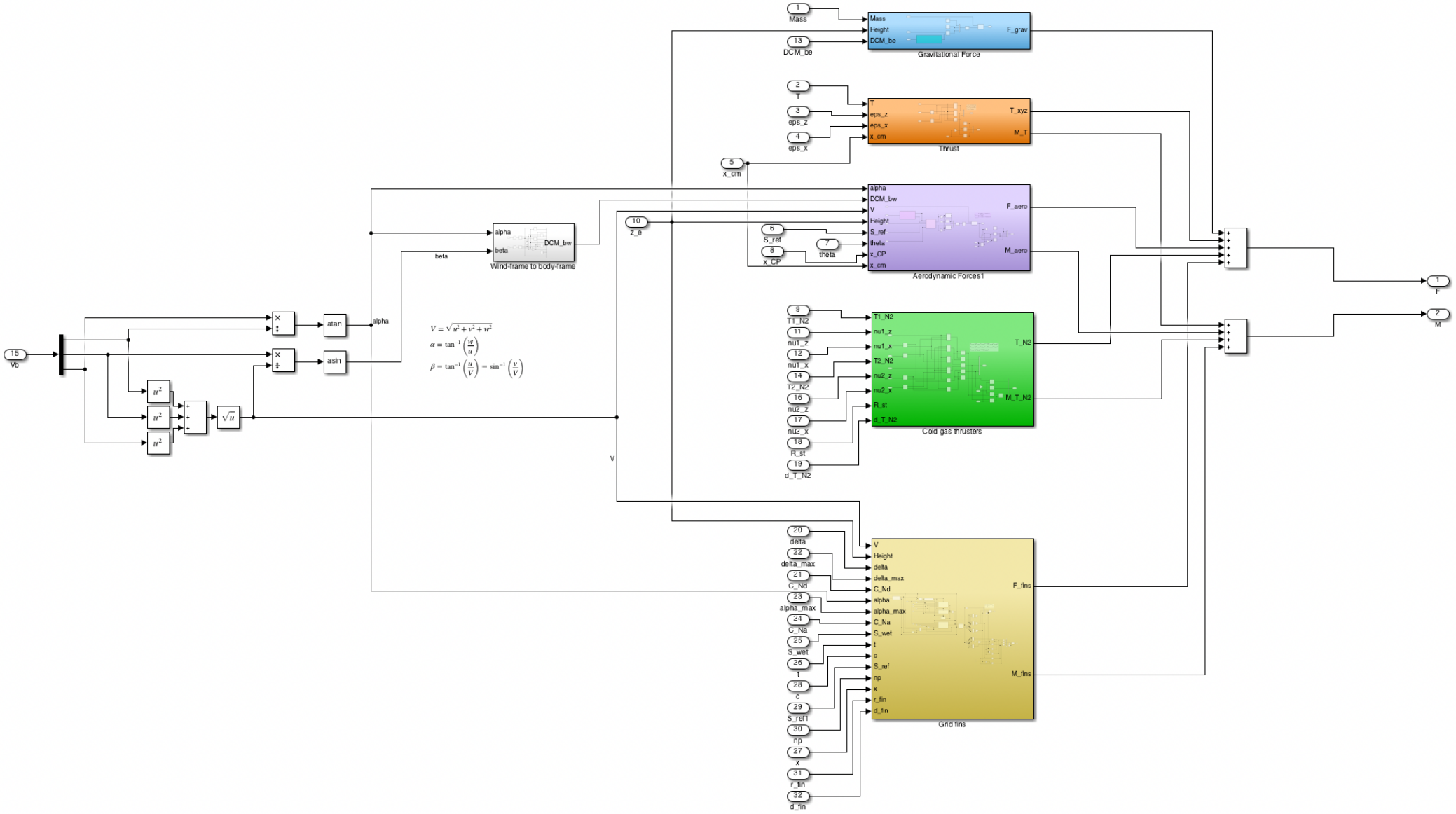

Figure 6: Forces and moments block subsystem

Within this last subsystem, starting out with the transformation matrix arrangement block, which is gray coloured, this would be computed by using the reshape Simulink block, presenting the scheme below.

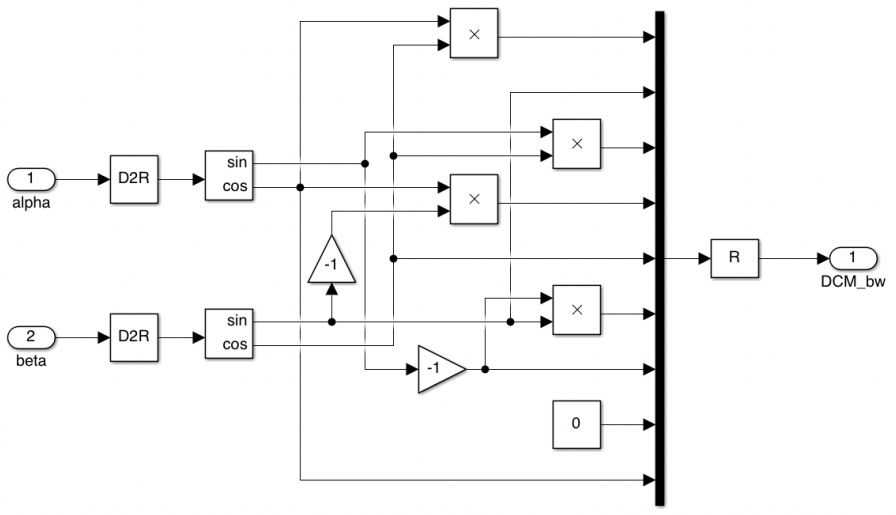

Figure 7: Wind-frame to body-frame transformation matrix

Secondly, the blue block calculates the gravitational forces through obtaining the gravity.

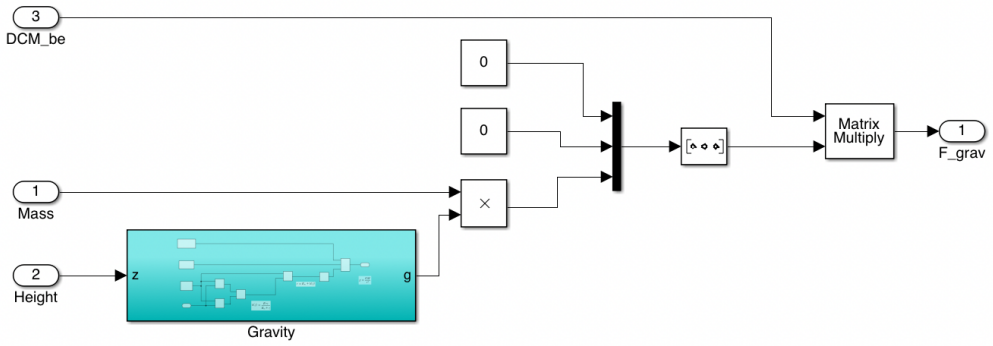

Figure 8: Gravitational forces in the body reference frame

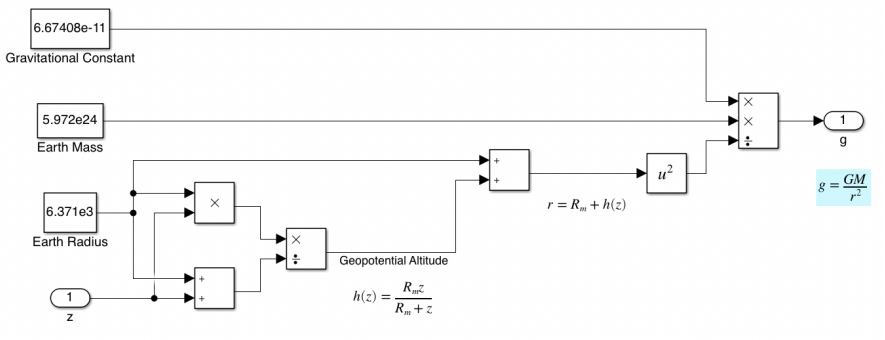

Figure 9: Gravity calculation

Next in order, the orange block computes the respective forces and moments derived from the main thrusters.

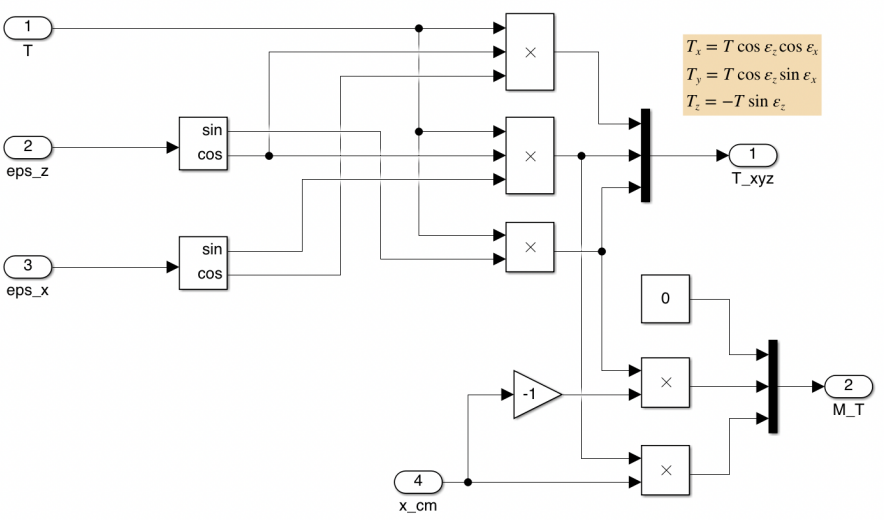

Figure 10: Thrust forces and moments in the body reference frame

The purple block is responsible for the aerodynamic forces and moments, where the lift and drag are determined through their correspondent coefficients and taking into account the pitch angle in order to invert the direction of the forces, since up to $\theta > 90°$ they possess opposite directions with respect the computed values due to the flip manoeuvre.

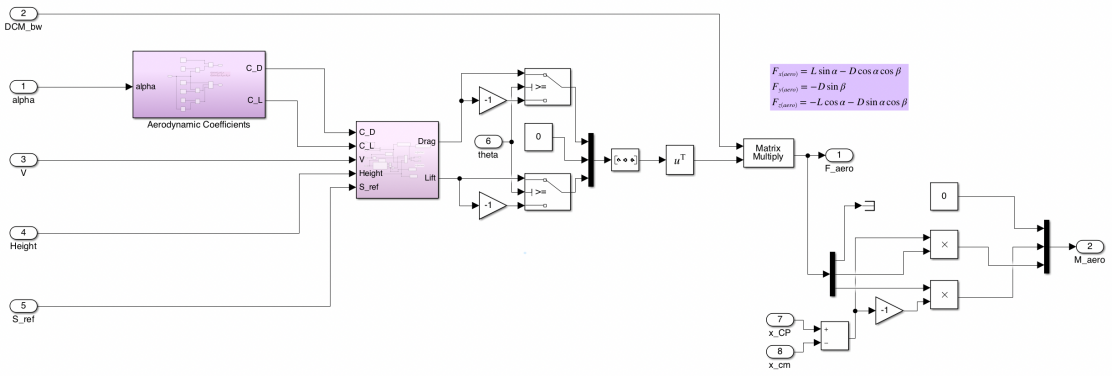

Figure 11: Aerodynamic forces and moments in the body reference frame

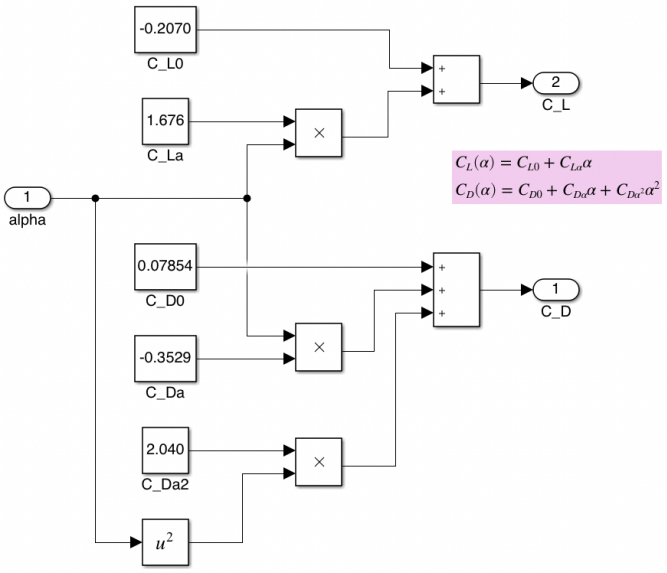

Figure 12: Lift and drag coefficients computation

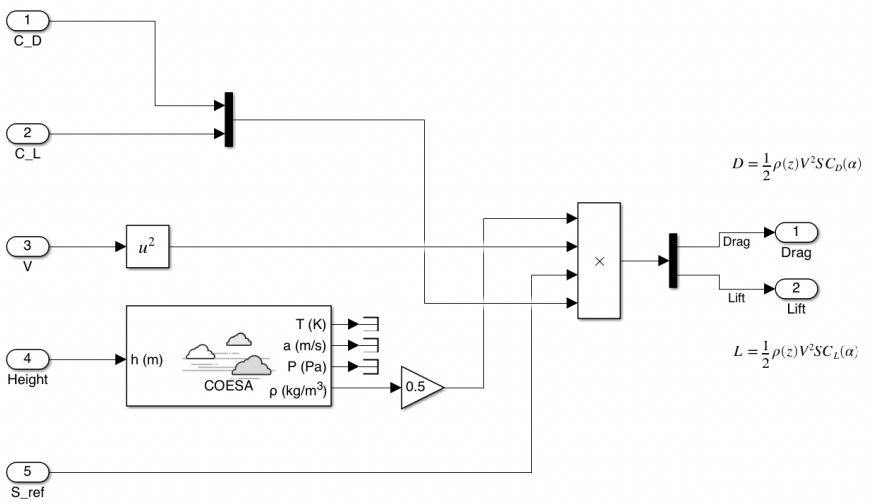

Figure 13: Lift and drag calculation

Afterwards, the green block considers the cold gas thrusters.

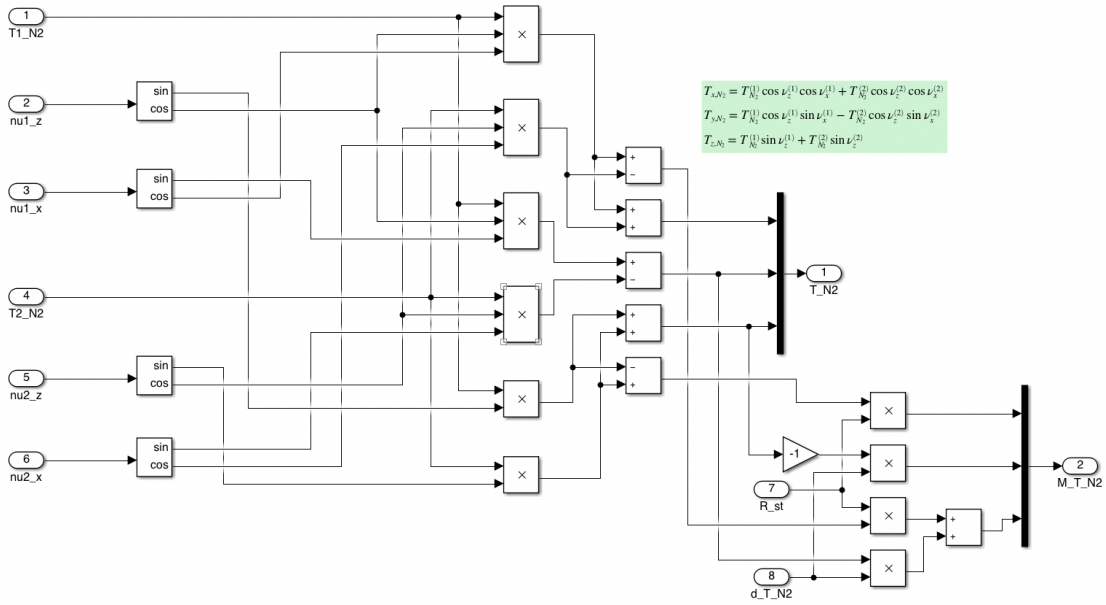

Figure 14: Cold gas thrusters forces and moments in the body reference frame

Finally, the yellow block computes the hypersonic grid fins, which has to be activated whenever they are deployed (with a switch). It is worth remarking that the input *delta* of this block is a vector of 4 values, one for each fin of the stage.

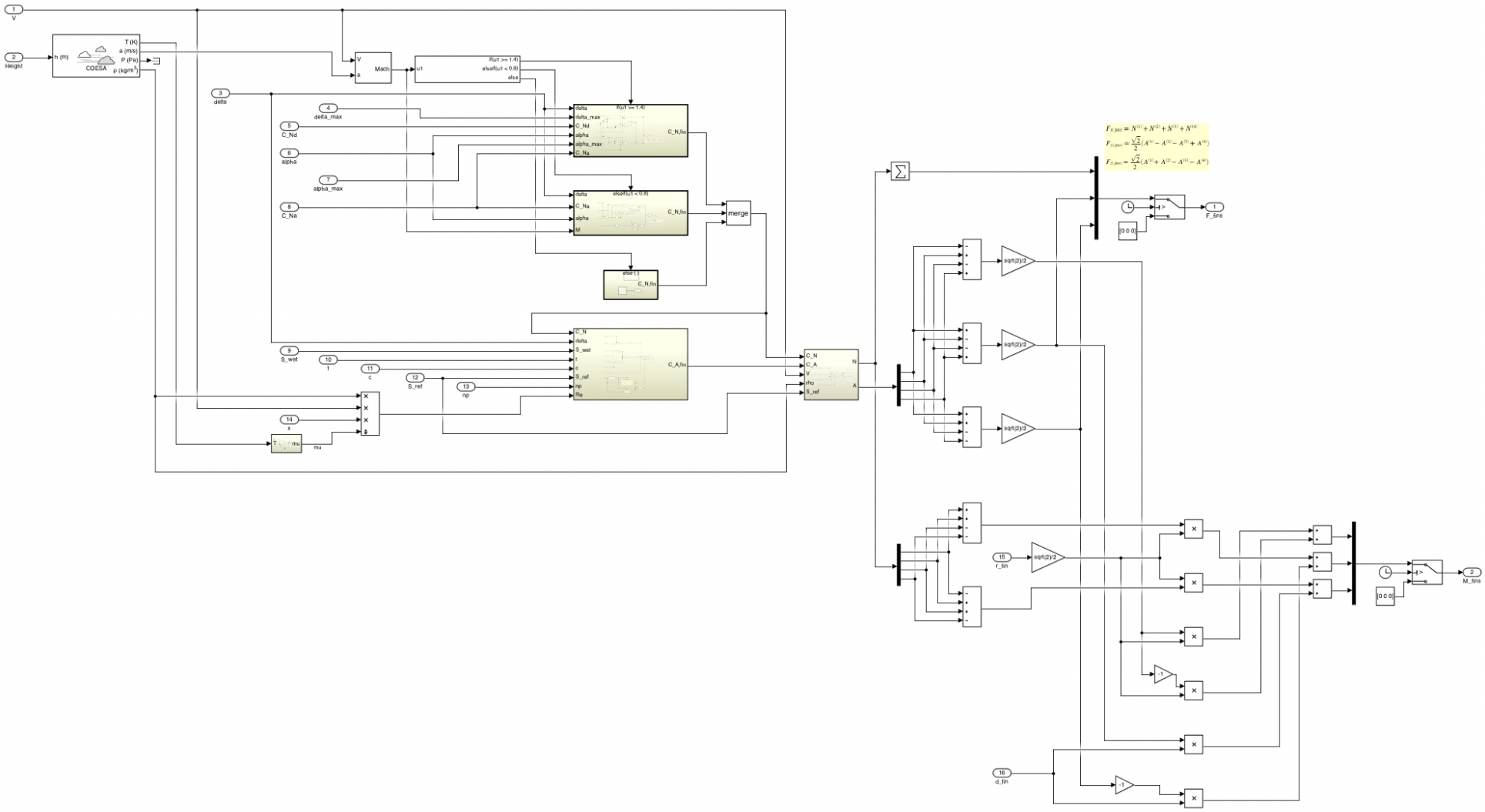

Figure 15: Hypersonic grid fins forces and moments in the body reference frame

From this last subsystem, there are more blocks within it. Firstly, the dynamic viscosity of the fluid calculation:

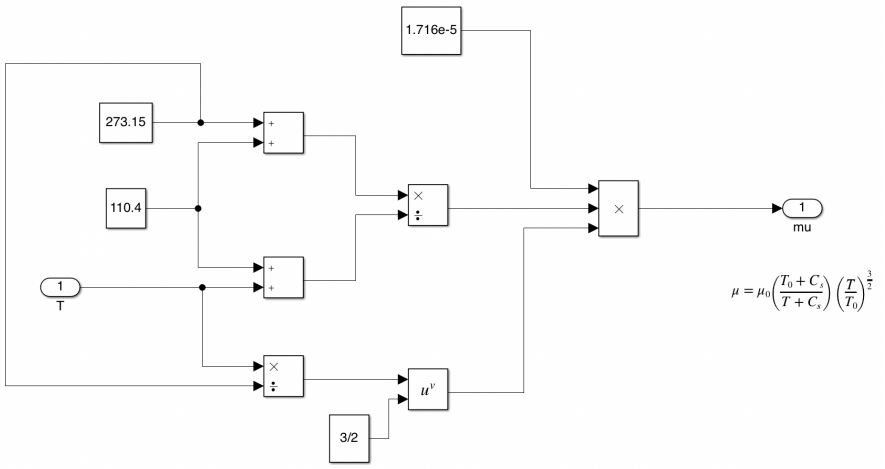

Figure 16: Dynamic viscosity of the fluid calculation

Secondly, the normal force coefficient C_N of the grid fins for hypersonic, subsonic or transonic speed, the last of which is just set to 0 due to its null effects:

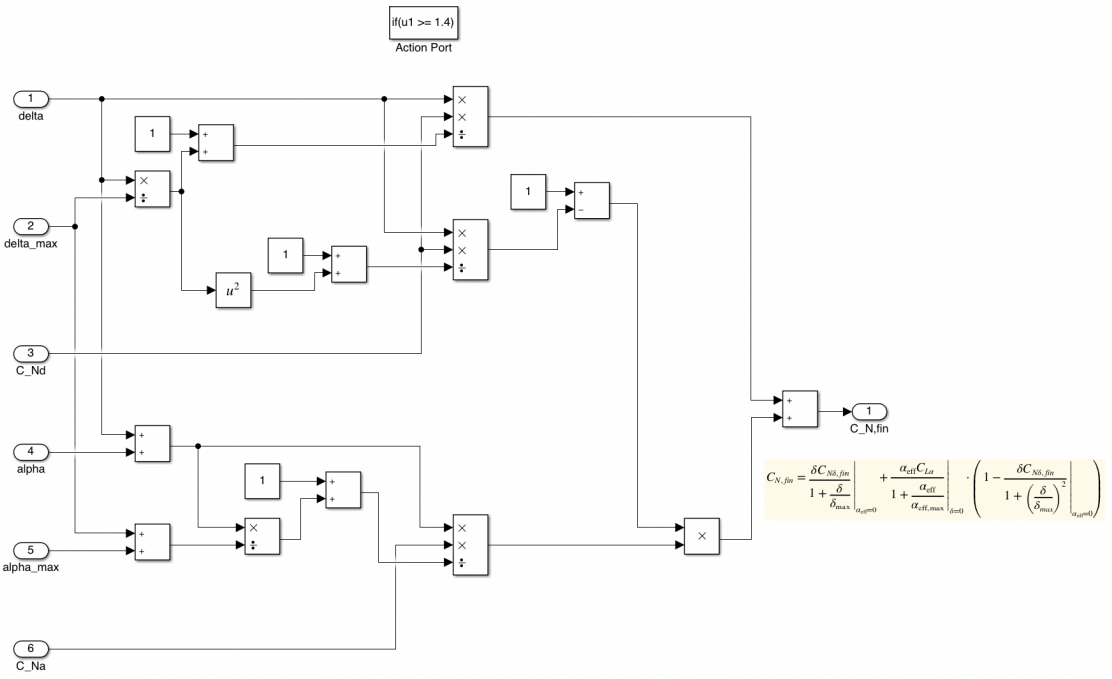

Figure 17: Normal force coefficient calculation for hypersonic velocities

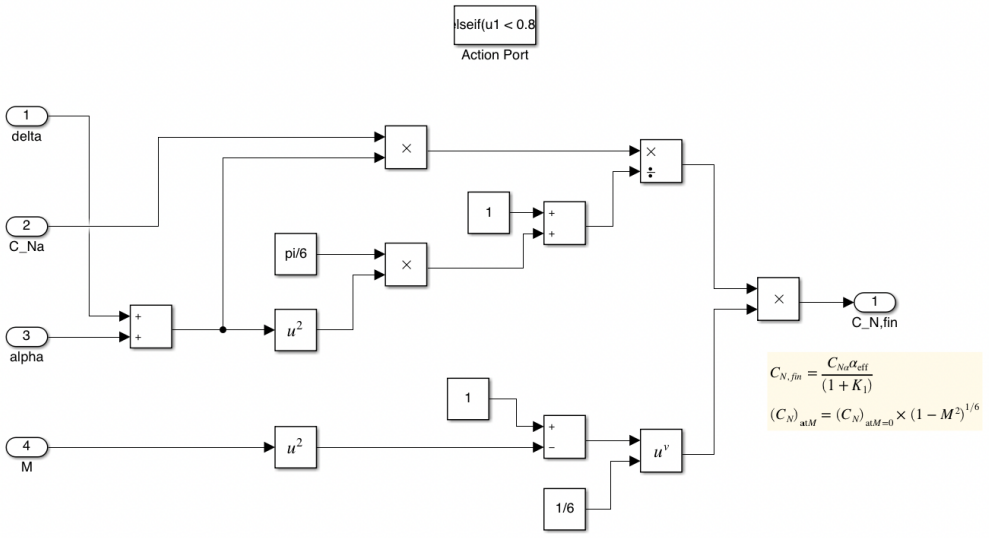

Figure 18: Normal force coefficient calculation for subsonic velocities

Next, the axial force coefficient C_A of the grid fins:

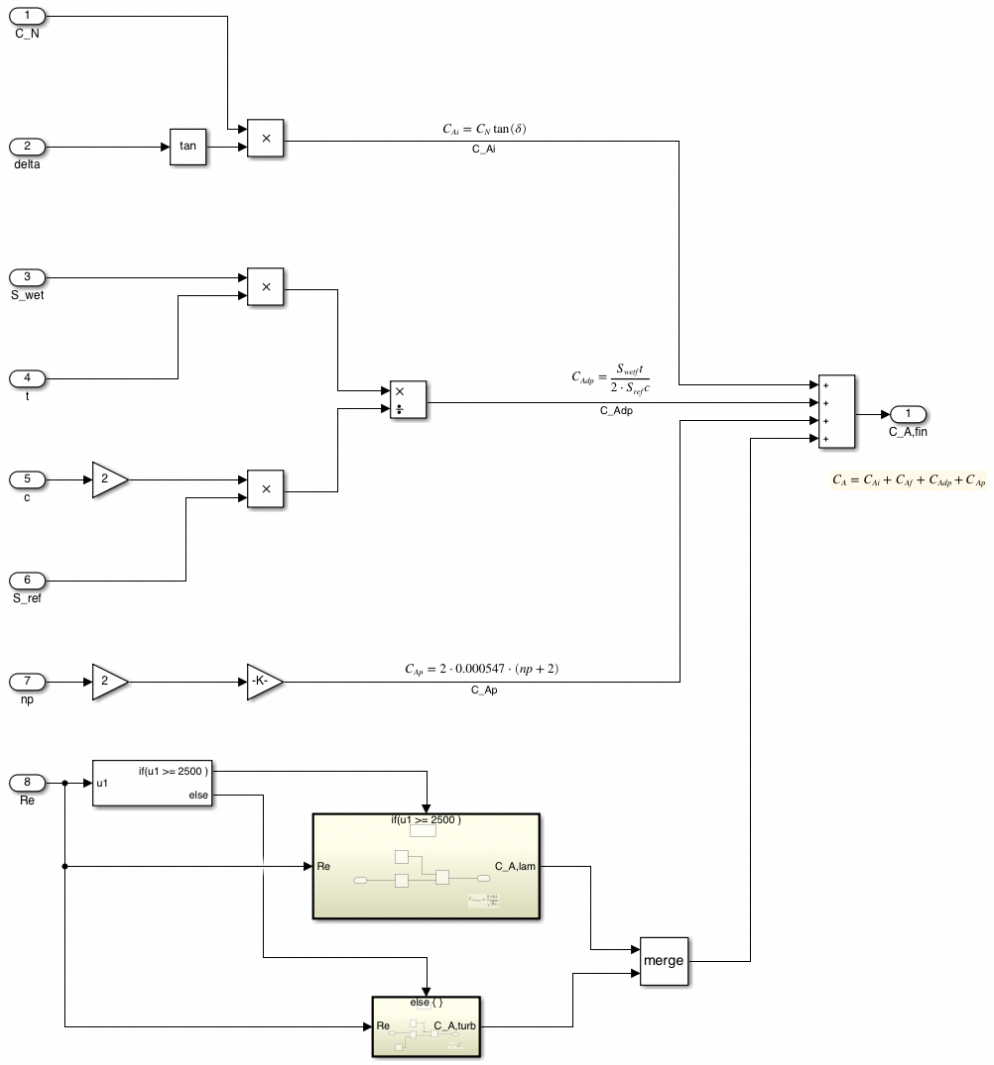

Figure 19: Axial force coefficient calculation

Within this previous block, there is the calculation of the skin friction drag coefficient as a function of whether the flow is turbulent or laminar:

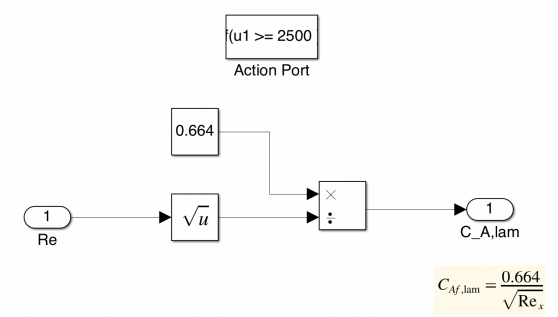

Figure 20: Skin friction drag coefficient calculation for laminar flow

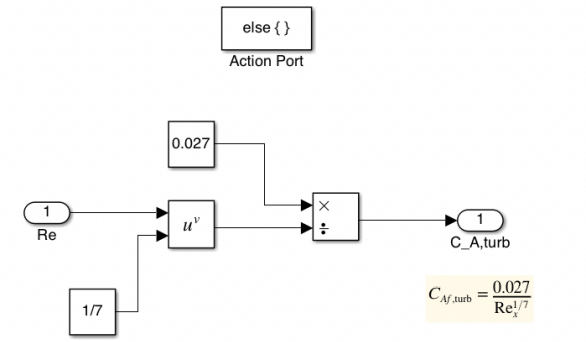

Figure 21: Skin friction drag coefficient calculation for turbulent flow

Finally, the computation of the corresponding forces for each coefficient:

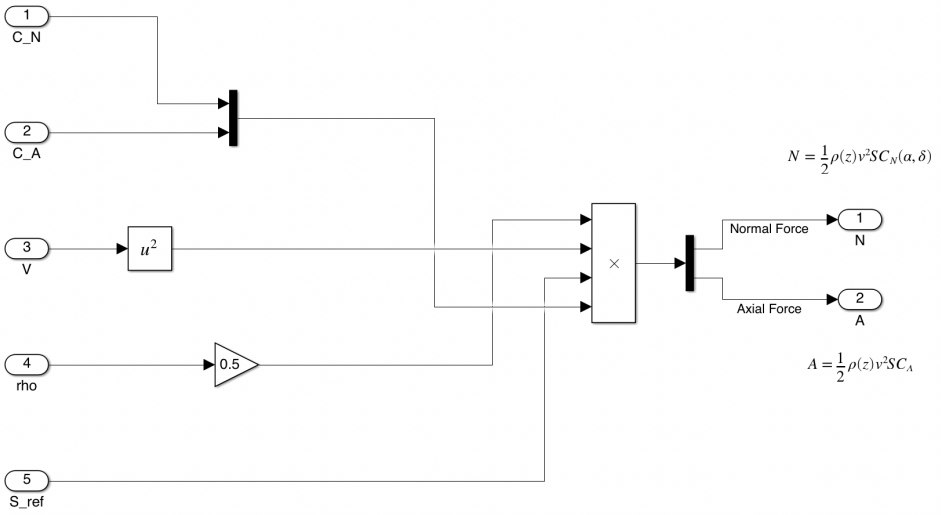

Figure 22: Normal and axial forces of the grid fins calculation

A.2 Control model

Given the previous presented complex system, it is simplified in order to make a first approach to its control. The general diagram would be the following.

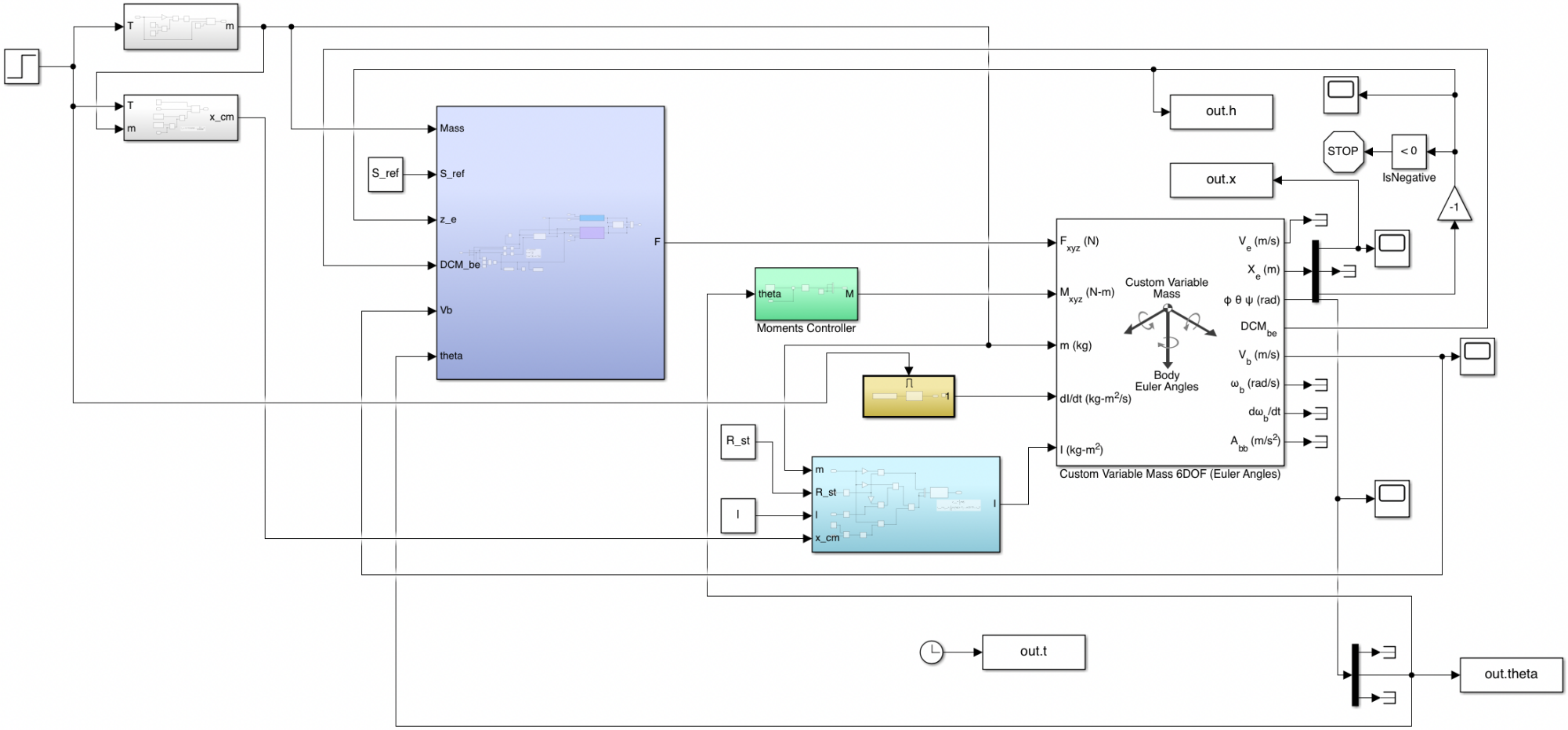

Figure 23: Simulink controlled model

In this case the moments are substituted by a controller block, whose feedback is the pitch angle (θ). Through the corresponding setpoint of the desired pitch along time, either with a predefined function or with a combination of ramps and constants, the error with this feedback and controlled with a controller, which determines the moments in the body y-axis required to accomplish this setpoint, is calculated.

It is important to mention that the setpoint has some limitations, because the rank of pitch angles that the solver block admits is between –180° to 180°, turning into negatives values if 180° is outpaced and vice versa, offering singularities when trying to maintain the limit value (either 180° or –180°) since the response would oscillate. Thus, if the controller has some overshooting this setpoint must be slightly smaller in order not to excess this values.

Since two different methodologies have been analysed, the moments controller takes the following possible subsystems.

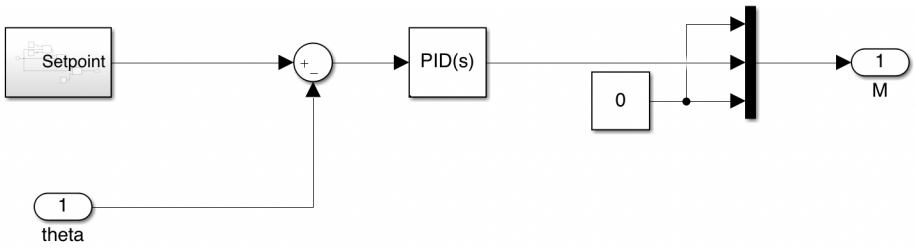

Figure 24: Moments PID controller

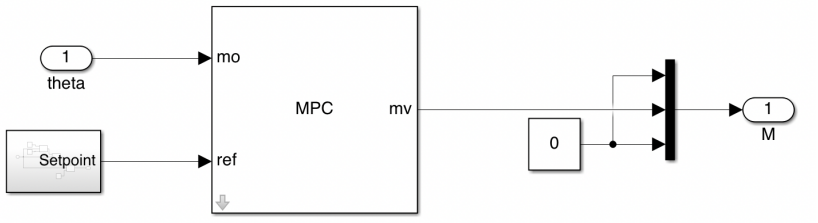

Figure 25: Moments MPC controller

The other blocks of the system would remain equal, except for the forces block, which just calculate the gravitational and the aerodynamic forces, because the overall control surfaces are calculated through a new block that computes the required forces to achieve the adequate performance.

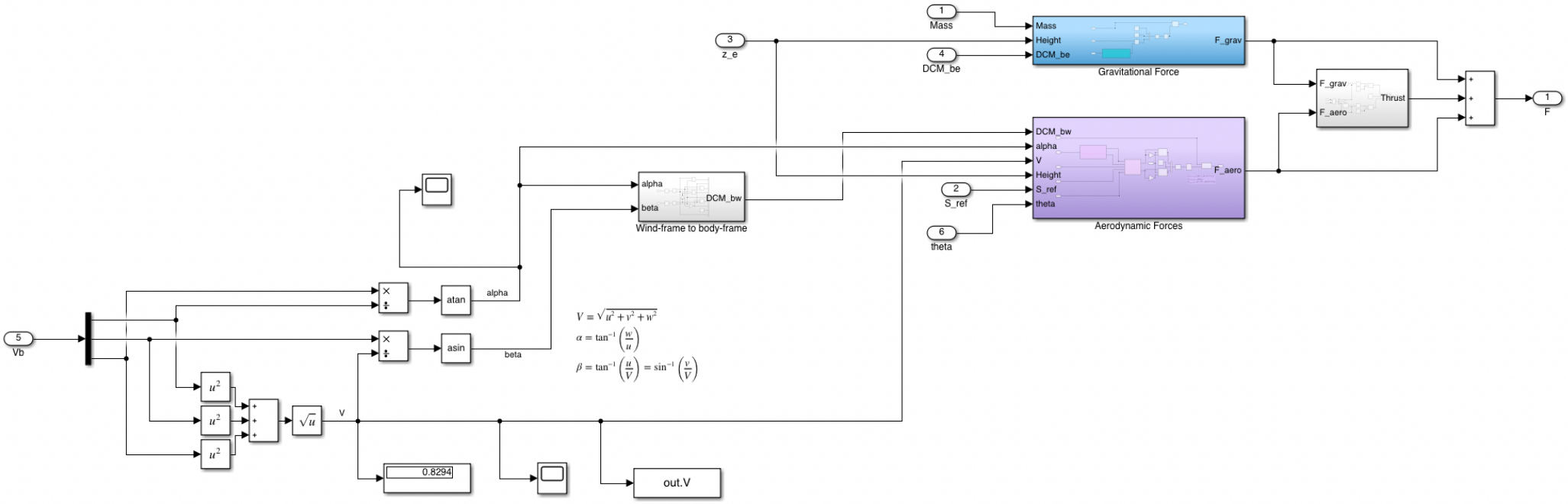

Figure 26: Forces and moments block subsystem of the controlled model

Hence, the remaining blocks are equivalent to the previous section, and this new block is just controlled with a simple feedback of the gravitational and aerodynamic forces in order to calculate the required resultant to reach the sought final speed. Consequently, the forces of the body x and z axes are separated and inserted their corresponding setpoint.

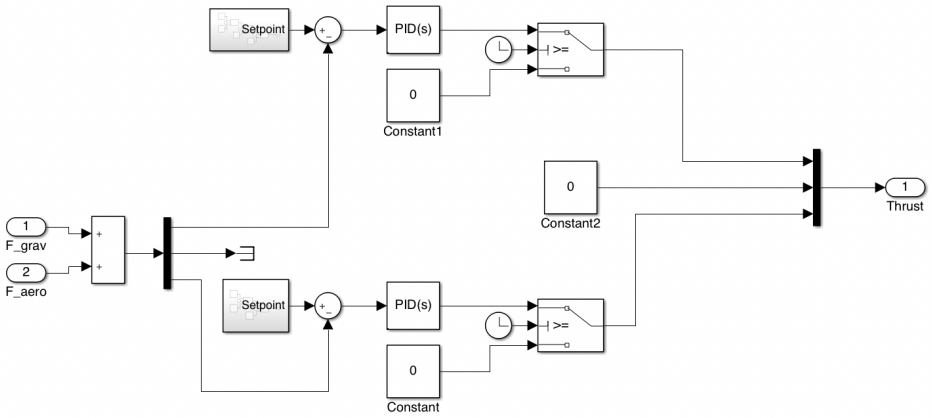

Figure 27: Forces controller

Since in this case the forces are computed under its resultant, the exact value of the thrust from main engines is not established, thus, a mean value is used to calculate the mass and the centre of mass. Moreover, the inertia variation along time is also set with its mean value.

B Matlab codes

In this section, the applied codes of Matlab in the project development are attached.

B.1 Inputs code

The declaration of inputs for the Simulink system.

```
%% Inputs

clc; close; clear;

% Dimensions

R_st = 1.83; % [m]
d_T_N20 = 42; % [m]
r_fin = 5.33; % [m]
d_fin0 = 41; % [m]
alpha_max = deg2rad(90); % [rad]
delta_max = deg2rad(25); % [rad]
S_ref = 173.85; % [m^2]
l = 47.5; % [m]


% Initial conditions

m0 = 228000; % [kg]
m_st = 22200; % [kg]
h0 = 80000; % [m]
V0 = 2833.33; % [m/s]
alpha_0 = deg2rad(45); % [rad]
% Vb0 = [V0*cos(alpha_0) 0 V0*sin(alpha_0)]; % [m/s]
Vb0 = [2e3 0 1e3]; % [m/s]
```

```
euler0 = [0 deg2rad(45) 0]; % [rad]
omega0 = [0 -0.005 0]; % [rad]
g0 = 9.80665; % [m/s^2]


% Thrust parameters

Isp = 282; % [s]


% Fins parameters

t = 1.5; % [m]
c = 7; % [m]
S_ref_fins = 21; % [m^2]
S_wet_fins = 25; % [m^2]
np = 66;
C_Na = 0.067;
C_Nd = 0.02;


% Centre of pressure

x_cp = 14;
```

B.2 Transfer function code

The calculation of the transfer function from the output *linsys* of the Model Linearizer App of Simulink.

```
%% Obtain transfer function

A = linsys1.A;
B = linsys1.B;
```

```
C = linsys1.C;
D = linsys1.D;

[num, den] = ss2tf(A, B, C, D);
GV = minreal(tf(num, den))
```

B.3 Plots code

The creation of the plots from the report.

```
%% Plots

clc; close; clear;
out0 = load('open_model.mat');
out0 = out0.out;
out = load('control_simulation.mat');
out = out.out;

%% Open model validation

% Height

figure;
plot(out0.t,out0.h,'color',[0.4660, 0.6740, 0.1880]);
ylim([0 9e4]);
xlabel('Time [s]','interpreter','Latex','fontsize',12);
ylabel('$|z_e|$ [m]','interpreter','Latex','fontsize',12);
title('Height as a function of time','interpreter','Latex','fontsize'
    ,14);
set(gca,'TickLabelInterpreter','latex');

% Range

figure;
```

```
plot(out0.t,out0.x,'color',[0.4940, 0.1840, 0.5560]);
xlabel('Time [s]','interpreter','Latex','fontsize',12);
ylabel('$x_e$ [m]','interpreter','Latex','fontsize',12);
title('Horizontal distance as a function of time','interpreter','
    Latex','fontsize',14);
set(gca,'TickLabelInterpreter','latex');

% Velocity

figure;
plot(out0.t,out0.Ve);
xlabel('Time [s]','interpreter','Latex','fontsize',12);
ylabel('$V_e$ [m/s]','interpreter','Latex','fontsize',12);
title('Linear velocity in inertial-frame as a function of time','
    interpreter','Latex','fontsize',14);
legend('$\dot{x}_E$','$\dot{y}_E$','$\dot{z}_E$','interpreter','Latex
    ','fontsize',12);
set(gca,'TickLabelInterpreter','latex');

%% Control model

% Pitch

figure;
plot(out.t,rad2deg(out.theta),'color',[0.3010, 0.7450, 0.9330]);
ylim([40 190]);
xlabel('Time [s]','interpreter','Latex','fontsize',12);
ylabel('$\theta$ [$^\circ$]','interpreter','Latex','fontsize',12);
title('Pitch angle as a function of time','interpreter','Latex','
    fontsize',14);
set(gca,'TickLabelInterpreter','latex');

% Height

figure;
```

```
55 plot(out.t,out.h,'color',[0.4660, 0.6740, 0.1880]);
56 xlabel('Time [s]','interpreter','Latex','fontsize',12);
57 ylabel('$|z_e|$ [m]','interpreter','Latex','fontsize',12);
58 title('Height as a function of time','interpreter','Latex','fontsize'
       ,14);
59 set(gca,'TickLabelInterpreter','latex');
60
61 % Range
62
63 figure;
64 plot(out.t,out.x,'color',[0.4940, 0.1840, 0.5560]);
65 xlabel('Time [s]','interpreter','Latex','fontsize',12);
66 ylabel('$x_e$ [m]','interpreter','Latex','fontsize',12);
67 title('Horizontal distance as a function of time','interpreter','
       Latex','fontsize',14);
68 set(gca,'TickLabelInterpreter','latex');
69
70 % Velocity
71
72 figure;
73 plot(out.t,out.V);
74 xlabel('Time [s]','interpreter','Latex','fontsize',12);
75 ylabel('$V$ [m/s]','interpreter','Latex','fontsize',12);
76 title('Linear velocity module as a function of time','interpreter','
       Latex','fontsize',14);
77 set(gca,'TickLabelInterpreter','latex');
78
79 % Trajectory
80
81 figure;
82 plot(out.x,out.h,'color',[0.6350, 0.0780, 0.1840]);
83 ylim([0 12e4]);
84 xlabel('$x_e$ [m]','interpreter','Latex','fontsize',12);
85 ylabel('$|z_e|$ [m]','interpreter','Latex','fontsize',12);
86 title('Trajectory','interpreter','Latex','fontsize',14);
```

```
87 set(gca,'TickLabelInterpreter','latex');
88
89 %% Perturbed model PID
90
91 % Pitch
92
93 out1 = load('pid_perturbed.mat');
94 out1 = out1.out;
95
96 figure;
97 plot(out.t,rad2deg(out.theta),'color',[0.3010, 0.7450, 0.9330]);
98 hold on;
99 plot(out1.t_p,rad2deg(out1.theta_p),'b');
100 ylim([40 190]);
101 xlabel('Time [s]','interpreter','Latex','fontsize',12);
102 ylabel('$\theta$ [$^\circ$]','interpreter','Latex','fontsize',12);
103 title('Pitch angle as a function of time','interpreter','Latex','
    fontsize',14);
104 legend('Non-perturbed','Perturbed','interpreter','Latex','fontsize'
    ,12);
105 set(gca,'TickLabelInterpreter','latex');
106
107 fprintf('The maximum difference is %0.3f deg \n', max(abs(rad2deg(
    out1.theta_p(1:size(out.theta,1)))-rad2deg(out.theta))))
108
109 %% MPC model
110
111 % Pitch
112
113 out2 = load('mpc_simulation.mat');
114 out2 = out2.out;
115
116 figure;
117 plot(out2.t_mpc,rad2deg(out2.theta_mpc),'color',[0.2500, 0.2500,
    0.2500]);
```

```
xlim([0 300]);
ylim([40 190]);
xlabel('Time [s]','interpreter','Latex','fontsize',12);
ylabel('$\theta$ [$^\circ$]','interpreter','Latex','fontsize',12);
title('Pitch angle as a function of time','interpreter','Latex','
    fontsize',14);
set(gca,'TickLabelInterpreter','latex');

fprintf('The MPC maximum error is %0.3f deg \n', max(abs(rad2deg(out2
    .theta_mpc)-rad2deg(out2.setp_mpc))))
fprintf('The MPC mean error is %0.3f deg \n', mean(abs(rad2deg(out2.
    theta_mpc)-rad2deg(out2.setp_mpc))))

% Pitch comparison

figure;
plot(out.t,rad2deg(out.theta),'color',[0.3010, 0.7450, 0.9330]);
hold on;
plot(out2.t_mpc,rad2deg(out2.theta_mpc),'color',[0.2500, 0.2500,
    0.2500]);
xlim([0 300]);
ylim([40 190]);
xlabel('Time [s]','interpreter','Latex','fontsize',12);
ylabel('$\theta$ [$^\circ$]','interpreter','Latex','fontsize',12);
title('Pitch angle as a function of time','interpreter','Latex','
    fontsize',14);
legend('PID','MPC','interpreter','Latex','fontsize',12);
set(gca,'TickLabelInterpreter','latex');

fprintf('The PID maximum error is %0.3f deg \n', max(abs(rad2deg(out.
    theta)-rad2deg(out.setp))))
fprintf('The PID mean error is %0.3f deg \n', mean(abs(rad2deg(out.
    theta)-rad2deg(out.setp))))

%% Perturbed model MPC
```

```
146
147 % Pitch
148
149 out3 = load('mpc_perturbed.mat');
150 out3 = out3.out;
151
152 figure;
153 plot(out2.t_mpc,rad2deg(out2.theta_mpc),'color',[0.3010, 0.7450,
        0.9330]);
154 hold on;
155 plot(out3.t_mpc,rad2deg(out3.theta_mpc),'b');
156 xlim([0 300]);
157 ylim([40 190]);
158 xlabel('Time [s]','interpreter','Latex','fontsize',12);
159 ylabel('$\theta$ [$^\circ$]','interpreter','Latex','fontsize',12);
160 title('Pitch angle as a function of time','interpreter','Latex','
        fontsize',14);
161 legend('Non-perturbed','Perturbed','interpreter','Latex','fontsize'
        ,12);
162 set(gca,'TickLabelInterpreter','latex');
163
164 fprintf('The maximum difference is %0.3f deg \n', max(abs(rad2deg(
        out2.theta_mpc(1:size(out3.theta_mpc,1)))-rad2deg(out3.theta_mpc))
        ))
```