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

Study and Evaluation of the Performance of a Multistage Solid-Propellant Vehicle, Including Atmospheric Ascent and Orbital Insertion

Document:
Annexes

Author:
Roger Vergés Eiras

Director:
Josep Oriol Lizandra Dalmases

Degree:
Bachelor's degree in Aerospace Technology Engineering

Examination session:
Spring, 2021

BACHELOR FINAL THESIS

Contents

Appendix A

Algorithm of resolution

Algorithm 1: Numerical solving of the differential equations

Result: Calculation of the trajectory of the rocket
initialization;
for *i=2:length(Time vector)* **do**
  if *first stage* **then**
    Calculation of velocity;
    if *gravity turn has not started* **then**
      Flight path angle = $\pi/2$;
    else
      Calculation of flight path angle;
    end
    Calculation of height;
    Calculation of mass;
    Calculation of range;
    Interpolation of pressure, density, gravity and speed of sound at a given height;
    if *there is flow detachment* **then**
      Recalculation of parameters and thrust coefficient reckoning;
    else
      Calculation of thrust coefficient;
    end
    Calculation of rocket's Mach number;
    Calculation of drag coefficient and drag;
    Calculation of thrust and specific impulse;
    Calculation of acceleration;
    Calculation of dynamic pressure;
  else
    Calculations of the second stage follow the same procedure as those of the first;
  end
end

Appendix B

MATLAB code

```
clear all; clc
%% Information
%Author: Roger Verges Eiras

%Director: Josep Oriol Lizandra i Dalmases

%Bachelor's Final Thesis MATLAB CODE

%Project name: Study and Evaluation of the Performance of a
    Multistage
%Solid-Propellant Rocket Vehicle, Including Atmospheric Ascent
%and Orbital Insertion

%Course 2020-2021

%% US STANDARD ATMOSPHERE (1976 MODEL) CHARACTERIZATION

h = linspace(0,230000,75001);                % height vector [m]

disp('Characterizing the atmosphere...')

%US standard atmosphere function
[T, P, rho, g, c] = US_Standard(h);

% Pressure in function of height
figure (1)
plot (h/1000, P)
grid minor;
title ('Pressure in function of height','interpreter','latex');
xlabel ('Height [km]','interpreter','latex');
ylabel ('Pressure [atm]','interpreter','latex');
xlim ([0 220]);
```

```
32
33 % Gravity in function of height
34 figure (2)
35 plot (h/1000, g)
36 grid minor;
37 title ('Gravity in function of height','interpreter','latex');
38 xlabel ('Height [km]','interpreter','latex');
39 ylabel ('Gravity [$\frac{m}{s^2}$]','interpreter','latex');
40 xlim ([0 220]);
41
42 % Density in function of height
43 figure (3)
44 plot (h/1000, rho)
45 grid minor;
46 title ('Density in function of height','interpreter','latex');
47 xlabel ('Height [km]','interpreter','latex');
48 ylabel ('Density [$\frac{kg}{m^3}$]','interpreter','latex');
49 xlim ([0 220]);
50
51 % Temperature in function of height
52 figure (4)
53 plot (h/1000, T)
54 grid minor;
55 title ('Temperature in function of height','interpreter','latex');
56 xlabel ('Height [km]','interpreter','latex');
57 ylabel ('Temperature [K]','interpreter','latex');
58 xlim ([0 220]);
59
60 disp('Atmosphere characterized')
61
62 %% CHAMBER CHARACTERISTICS AND DATA
63
64 Ru = 8.314;             % Ideal gas constant
65 pc = 100;               % Combustion chamber pressure [atm]
66
67 %% ANALYSIS OF NUMBER OF STAGES
68
69 disp('Analysing effect of number of stages...')
70 n=[1 2 3 4 5 1e12];
71 budget_characteristic=0:0.001:7;
72 structural_ratio=0.075;
73 for i=1:length(budget_characteristic)
74     for j=1:length(n)
75         if (budget_characteristic(i)>5 && n(j)==2)
76         lambda_total(i,j)=0;
77         else
78 lambda_total(i,j)=((exp(-budget_characteristic(i)/(n(j)))-...
```

```
    structural_ratio)/(1-structural_ratio))^n(j);
       end
   end
end

% Plot of the results
figure
semilogy (budget_characteristic,lambda_total)
ylim ([0.001 1.1]);
yticklabels ({  '0.001','0.01', '0.1','1'});
legend ('1 Stage', '2 Stages','3 Stages','4 Stages','5 Stages',...
   '$\infty$ Stages','interpreter','latex');
xlabel ('$\Delta v$ / $u_e$','interpreter','latex');
ylabel ('$\lambda_*$','interpreter','latex');
grid on; grid minor;
title ('Final budget depending on total payload ratio and stages, $\
   epsilon=0.075$','interpreter','latex')

disp('Analysis of number of stages completed')

%% CEARUN ANALYSIS

disp('Analysizing the choice of propellant...')

%Case 1. Oxidizer NH4CLO4. Data extracted from NASA - CEARUN analysis
    .
fuel_weight_CLO4=[1 5 10 15 20 25 30 35 40];
gamma_gas_propellant_CLO4=[1.1682 1.1457 1.1337 1.1313 1.1348...
    1.144 1.1562 1.1671 1.1977];
MW_propellant_CLO4=[27.360 27.279 27.016 26.354 25.472...
    24.403 23.223 22.044 21.026];
Temperature_CLO4=[2916.61 3278.69 3557.99 3700.48 3731.77...
    3655.96 3490.65 3256.38 2851.43];

%Case 2. Oxidizer NH4NO3. Data extracted from NASA - CEARUN analysis.
fuel_weight_NO3=[1 5 10 15 20 25 30 35 40];
gamma_gas_propellant_NO3=[1.1474 1.1368 1.1333 1.1383 ...
    1.1501 1.1658 1.1835 1.2021 1.2126];
MW_propellant_NO3=[22.384 22.395 22.142 21.667 21.039 ...
    20.331 19.600 18.881 18.214];
Temperature_NO3=[3233.54 3432.66 3563.16 3585.79 3520.34 ...
    3389.59 3215.59 3012.11 2781.93];

for i=1:length(fuel_weight_CLO4)
% Specific gas constant NH4CLO4 case
R_specific_propellant_CLO4(i) = Ru/(MW_propellant_CLO4(i)/1000);
% Characteristic velocity NH4CLO4 case
```

```
124 c_characteristic_propellant_CLO4(i) = sqrt(R_specific_propellant_CLO4
    (i)...
125     *Temperature_CLO4(i))/(sqrt(gamma_gas_propellant_CLO4(i))...
126     /(1+(gamma_gas_propellant_CLO4(i)-1)/2)^((
            gamma_gas_propellant_CLO4(i)...
127     +1)/(2*(gamma_gas_propellant_CLO4(i)-1))));
128 % Specific gas constant NH4NO3 case
129 R_specific_propellant_NO3(i) = Ru/(MW_propellant_NO3(i)/1000);
130 % Characteristic velocity NH4CLO4 case
131 c_characteristic_propellant_NO3(i) = sqrt(R_specific_propellant_NO3(i
    )...
132     *Temperature_NO3(i))/(sqrt(gamma_gas_propellant_NO3(i))...
133     /(1+(gamma_gas_propellant_NO3(i)-1)/2)^((gamma_gas_propellant_NO3
            (i)...
134     +1)/(2*(gamma_gas_propellant_NO3(i)-1))));
135 end
136
137 % Plot of the results
138 figure
139 sgtitle('CEARUN - NASA results')
140
141 subplot(4,1,1)
142 plot(fuel_weight_CLO4, gamma_gas_propellant_CLO4, fuel_weight_NO3,...
143     gamma_gas_propellant_NO3)
144 ylim([1.1 1.25]);xlim([1 40]);
145 grid on; grid minor;
146 title('Coefficient of adiabatic expansion of the gas vs \% fuel
    weight','interpreter','latex');
147 legend('Using $NH_4ClO_4$','Using $NH_4NO_3$','interpreter','latex',
    'location','best')
148 ylabel('$\gamma$','interpreter','latex')
149
150 subplot(4,1,2)
151 plot(fuel_weight_CLO4, MW_propellant_CLO4, fuel_weight_NO3,...
152     MW_propellant_NO3)
153 ylim([15 30]);xlim([1 40]);
154 title('Molecular Weight vs \% fuel weight','interpreter','latex')
155 ylabel('$MW \; [g/mol]$','interpreter','latex')
156 grid on; grid minor;
157
158 subplot(4,1,3)
159 plot(fuel_weight_CLO4, Temperature_CLO4, fuel_weight_NO3,...
160     Temperature_NO3)
161 ylim([2500 4000]);xlim([1 40]);
162 title('Chamber temperature vs \% fuel weight','interpreter','latex')
163 grid on; grid minor;
164 xlabel('\% Fuel (expressed in total weight)','interpreter','latex')
```

```
ylabel ('$T_c \;[K]$','interpreter','latex')

subplot (4,1,4)
plot (fuel_weight_CLO4,c_characteristic_propellant_CLO4,
   fuel_weight_NO3,c_characteristic_propellant_NO3)
ylim ([1200 2000]);xlim ([1 40]);
title ('Characteristic velocity vs \% fuel weight','interpreter','
   latex')
grid on; grid minor;
xlabel ('\% Fuel (expressed in total weight)','interpreter','latex')
ylabel ('$c_* \;[m/s]$','interpreter','latex')

disp('Propellant analysed and selected')




%% ROCKET AND NOZZLE DATA

disp('Initial calculations...')

%From CEARUN ANALYSIS
MW = 21.039/1000;          % Gas molar mass [kg/mol]
Tc = 3520.34;              % Combustion chamber temperature [K]
gamma_gas = 1.1501;        % Coefficient of adiabatic expansion of the
    gas

%Rocket data
m0_1  = 120e3;             % Initial mass (non-optimized case) [kg]
m0_2  = 61e3;              % Initial mass (optimized case)     [kg]
pl    = 3e3;               % Payload [kg]

%optimized - non optimized first stage quantity [%]
cases=[65 60];

m0_stage2_1=(m0_1-pl)*((100-cases(2))/100)+pl;      % Initial mass.
   Second stage[kg]

m0_stage2_2=(m0_2-pl)*((100-cases(1))/100)+pl;      % Initial mass.
   Second stage[kg]
m0_stage3_2=(m0_2-pl)*((100-90)/100)+pl;            % Initial mass.
   Third stage[kg]


tbs1 = 120;                     % Burning time 1 stage 1 [s]
tbs2 = 280;                     % Burning time 1 stage 2 [s]
```

```
206
207 tbs1prima = 115;                    % Burning time 2 stage 1 [s]
208 tbs2prima = 172;                    % Burning time 2 stage 2 [s]
209 tbs3prima = 113;                    % Burning time 2 stage 3 [s]
210
211 tb  = tbs1+tbs2;                    % Total burning time [s]
212
213 prop_mass1_1 = (m0_1-pl)*cases(2)/100-(m0_1-pl)*cases(2)/100*0.075; %
     Propellant mass 1 stage 1 [kg]
214 prop_mass1_2 = (m0_2-pl)*cases(1)/100-(m0_2-pl)*cases(1)/100*0.075; %
     Propellant mass 1 stage 1 [kg]
215
216
217 prop_mass2_1 = (m0_1-pl)*((100-cases(2))/100)-(m0_1-pl)*((100-cases
    (2))/100)*0.075;    % Propellant mass stage 2 [kg]
218 prop_mass2_2 = (m0_2-pl)*((90-cases(1))/100)-(m0_2-pl)*((90-cases(1))
    /100)*0.075;      % Propellant mass stage 2 [kg]
219
220 prop_mass2_3=(m0_2-pl)*((75-cases(1))/100)-(m0_2-pl)*((75-cases(1))
    /100)*0.075;  % Propellant mass stage 3 [kg]
221
222
223 mflow_1stage1 = prop_mass1_1/tbs1;      % Mass flow rate stage 1[kg/s
    ]
224 mflow_1stage2 = prop_mass2_1/tbs2;      % Mass flow rate stage 2[kg/s
    ]
225
226 mflow_2stage1 = prop_mass1_2/tbs1prima; % Mass flow rate stage 1[kg/s
    ]
227 mflow_2stage2 = prop_mass2_2/tbs2prima; % Mass flow rate stage 2[kg/s
    ]
228 mflow_2stage3 = prop_mass2_3/tbs3prima; % Mass flow rate stage 3[kg/s
    ]
229
230
231
232
233 S = 25;                                % Vehicle reference area
    [m^2]
234 theta_nozzle=[12 16 21.592];           % Half angle of the
    nozzle
235 eta_nozzle=(1+cos(deg2rad(theta_nozzle)))/2;% Efficiency of the
    nozzle
236
237 % Numerical solving parameters. Time increment
238 deltaT = 0.01;
239 t = linspace(0,tb,tb/deltaT);
```

```
240
241 %% INITIAL CALCULATIONS
242
243 % Specific gas constant
244 R_specific = Ru/MW;
245
246 % Characteristic velocity
247 c_characteristic = sqrt(R_specific*Tc)/(sqrt(gamma_gas)/...
248 (1+(gamma_gas-1)/2)^((gamma_gas+1)/(2*(gamma_gas-1))));
249
250 % Area of the throat of the nozzle At
251 At1_stage1 = mflow_1stage1*c_characteristic/(pc*101325);
252 At1_stage2 = mflow_1stage2*c_characteristic/(pc*101325);
253
254 % Area of the throat of the nozzle At
255 At2_stage1 = mflow_2stage1*c_characteristic/(pc*101325);
256 At2_stage2 = mflow_2stage2*c_characteristic/(pc*101325);
257 At2_stage3 = mflow_2stage3*c_characteristic/(pc*101325);
258
259 % Flow parameter
260 m_1 = sqrt(gamma_gas)/(1+(gamma_gas-1)/2)^((gamma_gas...
261     +1)/(2*(gamma_gas-1)));
262
263
264 %% RELATION OF AREAS STUDY
265
266 Ae_At_study=[6:1:400];
267 for i=1:length(Ae_At_study)
268 % Exit mach
269 [Mestudy(i)] = Exit_Mach_study(gamma_gas,Ae_At_study(i));
270
271 % Pressure at the exit of the nozzle
272 pestudy(i)= pc/(1+(gamma_gas-1)/2*Mestudy(i)^2)^(gamma_gas/(gamma_gas
    -1));
273
274 % Exhaust velocity
275 u_exhauststudy(i)=sqrt(((2*gamma_gas)/(gamma_gas-1))*R_specific*Tc
    *(1-...
276 (pestudy(i)/pc)^((gamma_gas-1)/gamma_gas)));
277 end
278
279 % Plot of the results
280 figure
281 plot (Ae_At_study, u_exhauststudy)
282 grid on; grid minor;
283 title ('Exhaust velocity vs relation of areas','interpreter','latex')
    ;
```

```
ylabel ('Exhaust velocity [$m/s$]','interpreter','latex');
xlabel ('Relation of areas','interpreter','latex');
xlim ([6 400]);
xticks ([6 50 100 150 200 250 300 350 400])


%% EXIT MACH AT THE END OF THE NOZZLE

Ae_At=[20 40 68];                %Exit area - throat area ratio

%Exit mach calculation
for i=1:length(Ae_At)
[Me(i)] = Exit_Mach(gamma_gas,Ae_At(i));

% Thrust coefficient - void conditions
cF_v(i) = (2/(gamma_gas+1))^((gamma_gas+1)/(2*(gamma_gas-1)))*...
(gamma_gas*Me(i)+1/Me(i))/(sqrt(1+(gamma_gas-1)/2*Me(i)^2));

% Exit pressure
pe(i)    = pc/(1+(gamma_gas-1)/2*Me(i)^2)^(gamma_gas/(gamma_gas-1));

%exhaust velocity
u_exhaust(i)=sqrt(((2*gamma_gas)/(gamma_gas-1))*R_specific*Tc*(1-...
    (pe(i)/pc)^((gamma_gas-1)/gamma_gas)));
end

disp('Initial calculations reckoned')

%% TRAJECTORY RECKONING

% Vectors definition [non-optimized case]
h_rocket1 = zeros(length(t),1); % Height of the rocket [m]
gamma1 = zeros(length(t),1);    % Flight path angle [rad]
v1 = zeros(length(t),1);        % Velocity of the rocket [m/s]
a1 = zeros(length(t),1);        % Acceleration of the rocket [m/s^2]
m1 = zeros(length(t),1);        % Mass of the rocket [kg]
F1 = zeros(length(t),1);        % Thrust of the rocket [N]
cF1 = zeros(length(t),1);       % Thrust coefficient
t_flight=zeros(length(t),1);    % Flight time [s]
q1=zeros(length(t),1);          % Dynamic pressure [Pa]
isp1=zeros(length(t),1);        % Specific impulse [s]
M_flight1 = zeros(length(t),1); % Mach number
c_d1= zeros(length(t),1);       % Drag coefficient
x1=zeros(length(t),1);          % Range [m]
pa1=zeros(length(t),1);         % Ambient pressure [Pa]
D1=zeros(length(t),1);          % Drag [N]

```

```
331 %initial conditions
332 v1(1) =  0;
333 h_rocket1(1)  = 0;
334 alpha = 0;
335 F1(1)=1646000;
336 a1(1)=3.2;
337 m1(1)=m0_1;
338 gamma1(1) = pi/2;
339 x1(1)=0;
340 g(1)=9.80665;
341 R_e = 6371e3;
342 c_d1(1)=0.15;
343 t_flight(1)=0;
344 q1(1)=0;
345 isp1(1)=310;
346
347 %Latitude of Cape Canaveral, Florida, USA
348 latitude=deg2rad(28.396837);
349
350 disp('Numerical solving of the differential equations regarding the
    non-optimized rocket, please wait...')
351
352 % Trajectory characterization
353 for i=2:length(t)
354
355 %flight time
356 t_flight(i)=t(i);
357
358 if t_flight(i)<tbs1
359
360 %velocity of the rocket depending on time (with Earth rotation)
361 v1(i) = v1(i-1)+a1(i-1)*deltaT+464*cos(latitude)/length(t);
362
363 %Flight path angle depending on time
364 %Gravity turn does not start until certain time passes
365 if t_flight(i)<11.28
366 %The first time steps are devoted to vertical flight (gamma = 90 deg)
367 gamma1(i) = pi/2;
368 else
369 %After some time (arbitrary), a subtle change in the flight path
    angle is introduced
370 if gamma1(i-1)==pi/2
371 gamma1(i) = pi/2-0.025;
372 else
373 %Flight path angle depending on time
374 gamma1(i) = gamma1(i-1)+(((F1(i-1)/m1(i-1))*sin(alpha)-(g_h_1(i-1)-v1
    (i-1)^2/...
```

```
(h_rocket1(i-1)+R_e))*cos(gamma1(i-1)))/(v1(i-1)))*deltaT;
end
end

%rocket height depending on time
h_rocket1(i) = h_rocket1(i-1) + v1(i-1)*sin(gamma1(i-1))*deltaT;

%rocket mass depending on time
m1(i) = m1(i-1) - mflow_1stage1*deltaT;

%Range of the rocket depending on time
x1(i) = x1(i-1)+(R_e/(R_e+h_rocket1(i-1)))*v1(i-1)*cos(gamma1(i-1))*
    deltaT;

%Ambient pressure interpolation depending on rocket height
pa1(i) = interp1(h(:), P(:), h_rocket1(i));

%Flow detachment analysis (employing Summerfield criterion)
if pe(1) < (0.4*pa1(i))

%If detachment of the flow occurs
 Me_prima1 (i) = sqrt(2/(gamma_gas-1)*((...
pc/(0.4*pa1(i)))^((gamma_gas-1)/gamma_gas)-1));

 m_Me1 (i) = sqrt(gamma_gas)*Me_prima1(i)/(1+...
(gamma_gas-1)/2*Me_prima1(i)^2)^((gamma_gas+1)/(2*(gamma_gas-1)));

AeAt_prima1 (i) = m_1/m_Me1(i);
cF_v_prima1 (i) = (2/(gamma_gas+1))^((gamma_gas+1)/(2*(gamma_gas...
    -1)))*(gamma_gas*Me_prima1(i)+1/Me_prima1(i))/(sqrt(1+...
    (gamma_gas-1)/2*Me_prima1(i)^2));

%Thrust coefficient depending on local ambient pressure (with
    detachment)
cF1(i) = eta_nozzle(1)*(cF_v_prima1(i) - pa1(i)/pc*AeAt_prima1(i));
else
 %Thrust coefficient depending on local ambient pressure
cF1(i) = eta_nozzle(1)*(cF_v(1) - (pa1(i)/pc)*Ae_At(1));
end


%Interpolation of different atmospheric parameters depending on
    rocket height

%Density
rho_h_1(i) = interp1(h(:), rho(:), h_rocket1(i));
%Gravity
```

```
g_h_1(i) = interp1(h(:), g(:), h_rocket1(i));
%Speed of sound
c_h_1(i) = interp1(h(:), c(:), h_rocket1(i));

%Rocket Mach number depending on velocity and speed of sound
M_flight1(i) = v1(i)/c_h_1(i);

%Rocket drag coefficient depending on Mach number
if M_flight1(i)<=0.6
c_d1(i) = 0.15;
elseif (0.6<M_flight1(i) && M_flight1(i)<=1.1)
c_d1(i) = -4.32*M_flight1(i)^3+11.016*M_flight1(i)^2-8.5536*...
M_flight1(i)+2.24952;
elseif (1.1<M_flight1(i)&& M_flight1(i)<=1.3)
c_d1(i) = -M_flight1(i)^(2)+2.2*M_flight1(i)-0.79;
elseif (1.3 < M_flight1(i)&& M_flight1(i)<= 50)
c_d1(i) = 0.16769+0.17636/(sqrt(M_flight1(i)^2-1));
else
c_d1(i) = 0.16769;
end

% Computation of drag depending on density, velocity and drag
   coefficient
D1(i) = (1/2)*rho_h_1(i)*v1(i)^2*S*c_d1(i);

%Thrust value depending on thrust coefficient and stage
   characterization
F1(i) = cF1(i)*pc*101325*At1_stage1;

%Specific impulse of the rocket
isp1(i)=F1(i)/(mflow_1stage1*9.81);

%Acceleration of the rocket
a1(i) = (F1(i)*cos(alpha)-D1(i))/m1(i)-g_h_1(i)*sin(gamma1(i));

%Dynamic pressure
q1(i) = 0.5*rho_h_1(i)*v1(i)^2;


%second stage
else

 %flight time
t_flight(i)=t(i);

%velocity of the rocket depending on time (with Earth rotation)
v1(i) = v1(i-1)+a1(i-1)*deltaT;
```

```
464
465 %Flight path angle depending on time
466 gamma1(i) = gamma1(i-1)+(((F1(i-1)/m1(i-1))*sin(alpha)-(g_h_1(i-1)-v1
    (i-1)^2/...
467 (h_rocket1(i-1)+R_e))*cos(gamma1(i-1)))/(v1(i-1)))*deltaT;
468
469 %rocket height depending on time
470 h_rocket1(i) = h_rocket1(i-1) + v1(i-1)*sin(gamma1(i-1))*deltaT;
471
472 %rocket mass depending on time
473 m1(i) = m0_stage2_1 - mflow_1stage2*t(i-(length(t)*tbs1/tb-1));
474
475 %Range of the rocket depending on time
476 x1(i) = x1(i-1)+(R_e/(R_e+h_rocket1(i-1)))*v1(i-1)*cos(gamma1(i-1))*
    deltaT;
477
478 %Ambient pressure interpolation depending on rocket height
479 pa1(i) = interp1(h(:), P(:), h_rocket1(i));
480
481 %Flow detachment analysis (employing Summerfield criterion)
482 if pe(1) < (0.4*pa1(i))
483
484 %If detachment of the flow occurs
485  Me_prima1 (i) = sqrt(2/(gamma_gas-1)*((...
486 pc/(0.4*pa1(i)))^((gamma_gas-1)/gamma_gas)-1));
487
488  m_Me1 (i) = sqrt(gamma_gas)*Me_prima1(i)/(1+...
489 (gamma_gas-1)/2*Me_prima1(i)^2)^((gamma_gas+1)/(2*(gamma_gas-1)));
490
491 AeAt_prima1 (i) = m_1/m_Me1(i);
492 cF_v_prima1 (i) = (2/(gamma_gas+1))^((gamma_gas+1)/(2*(gamma_gas...
493     -1)))*(gamma_gas*Me_prima1(i)+1/Me_prima1(i))/(sqrt(1+...
494     (gamma_gas-1)/2*Me_prima1(i)^2));
495
496 %Thrust coefficient depending on local ambient pressure (with
    detachment)
497 cF1(i) = eta_nozzle(1)*(cF_v_prima1(i) - pa1(i)/pc*AeAt_prima1(i));
498 else
499  %Thrust coefficient depending on local ambient pressure
500 cF1(i) = eta_nozzle(1)*(cF_v(1) - (pa1(i)/pc)*Ae_At(1));
501 end
502
503 %Interpolation of different atmospheric parameters depending on
    rocket height
504 %Density
505 rho_h_1(i) = interp1(h(:), rho(:), h_rocket1(i));
506 %Gravity
```

```
507 g_h_1(i) = interp1(h(:), g(:), h_rocket1(i));
508 %Speed of sound
509 c_h_1(i) = interp1(h(:), c(:), h_rocket1(i));
510
511 %Rocket Mach number depending on velocity and speed of sound
512 M_flight1(i) = v1(i)/c_h_1(i);
513
514 %Rocket drag coefficient depending on Mach number
515 if M_flight1(i)<=0.6
516 c_d1(i) = 0.15;
517 elseif (0.6<M_flight1(i) && M_flight1(i)<=1.1)
518 c_d1(i) = -4.32*M_flight1(i)^3+11.016*M_flight1(i)^2-8.5536*...
519 M_flight1(i)+2.24952;
520 elseif (1.1<M_flight1(i)&& M_flight1(i)<=1.3)
521 c_d1(i) = -M_flight1(i)^(2)+2.2*M_flight1(i)-0.79;
522 elseif (1.3 < M_flight1(i)&& M_flight1(i)<= 50)
523 c_d1(i) = 0.16769+0.17636/(sqrt(M_flight1(i)^2-1));
524 else
525 c_d1(i) = 0.16769;
526 end
527
528 % Computation of drag depending on density, velocity and drag
      coefficient
529 D1(i) = (1/2)*rho_h_1(i)*v1(i)^2*S*c_d1(i);
530
531 %Thrust value depending on thrust coefficient and stage
      characterization
532 F1(i) = cF1(i)*pc*101325*At1_stage2;
533
534 %Specific impulse of the rocket
535 isp1(i)=F1(i)/(mflow_1stage2*9.81);
536
537 %Acceleration of the rocket
538 a1(i) = (F1(i)*cos(alpha)-D1(i))/m1(i)-g_h_1(i)*sin(gamma1(i));
539
540 %Dynamic pressure
541 q1(i) = 0.5*rho_h_1(i)*v1(i)^2;
542 end
543 end
544
545 disp('Non-optimized rocket computed!')
546
547 % Vectors definition [Optimized case]
548 h_rocket2 = zeros(length(t),1);
549 gamma2 = zeros(length(t),1);
550 v2 = zeros(length(t),1);
551 a2 = zeros(length(t),1);
```

```
552 m2 = zeros(length(t),1);
553 F2 = zeros(length(t),1);
554 cF2 = zeros(length(t),1);
555 t_flight=zeros(length(t),1);
556 q2=zeros(length(t),1);
557 isp2=zeros(length(t),1);
558 M_flight2 = zeros(length(t),1);
559 c_d2= zeros(length(t),1);
560 x2=zeros(length(t),1);
561 pa2=zeros(length(t),1);
562 D2=zeros(length(t),1);
563
564 %initial conditions
565 v2(1) =  0;
566 h_rocket2(1)  = 0;
567 alpha = 0;
568 F2(1)=1157000;
569 a2(1)=4.42;
570 m2(1)=m0_2;
571 gamma2(1) = pi/2;
572 x2(1)=0;
573 g(1)=9.80665;
574 R_e = 6371e3;
575 c_d2(1)=0.15;
576 t_flight(1)=0;
577 q2(1)=0;
578 isp2(1)=300.1;
579 %Latitude of Cape Canaveral, Florida, USA
580 latitude=deg2rad(28.396837);
581
582 disp('Numerical solving of the differential equations regarding the
      optimized rocket, please wait...')
583
584 % Trajectory characterization
585 for i=2:length(t)
586
587 %flight time
588 t_flight(i)=t(i);
589
590 if t_flight(i)<tbs1prima
591
592 %velocity of the rocket depending on time (with Earth rotation)
593 v2(i) = v2(i-1)+a2(i-1)*deltaT+464*cos(latitude)/length(t);
594
595 %Flight path angle depending on time
596 %Gravity turn does not start until certain time passes
597 if t_flight(i)<7.79
```

```
598
599 %The first time steps are devoted to vertical flight (gamma = 90 deg)
600 gamma2(i) = pi/2;
601 else
602 %After some time (arbitrary), a subtle change in the flight path
        angle is introduced
603 if gamma2(i-1)==pi/2
604 gamma2(i) = pi/2-0.025;
605 else
606 %Flight path angle depending on time
607 gamma2(i) = gamma2(i-1)+(((F2(i-1)/m2(i-1))*sin(alpha)-(g_h_2(i-1)-v2
        (i-1)^2/...
608 (h_rocket2(i-1)+R_e))*cos(gamma2(i-1)))/(v2(i-1)))*deltaT;
609
610 end
611 end
612
613 %rocket height depending on time
614 h_rocket2(i) = h_rocket2(i-1) + v2(i-1)*sin(gamma2(i-1))*deltaT;
615
616 %rocket mass depending on time
617 m2(i) = m2(i-1) - mflow_2stage1*deltaT;
618
619 %Range of the rocket depending on time
620 x2(i) = x2(i-1)+(R_e/(R_e+h_rocket2(i-1)))*v2(i-1)*cos(gamma2(i-1))*
        deltaT;
621
622 %Ambient pressure interpolation depending on rocket height
623 pa2(i) = interp1(h(:), P(:), h_rocket2(i));
624
625 %Flow detachment analysis (employing Summerfield criterion)
626 if pe(2) < (0.4*pa2(i))
627
628 %If detachment of the flow occurs
629
630  Me_prima2 (i) = sqrt(2/(gamma_gas-1)*((...
631 pc/(0.4*pa2(i)))^((gamma_gas-1)/gamma_gas)-1));
632
633  m_Me2 (i) = sqrt(gamma_gas)*Me_prima2(i)/(1+...
634 (gamma_gas-1)/2*Me_prima2(i)^2)^((gamma_gas+1)/(2*(gamma_gas-1)));
635
636 AeAt_prima2 (i) = m_1/m_Me2(i);
637 cF_v_prima2 (i) = (2/(gamma_gas+1))^((gamma_gas+1)/(2*(gamma_gas...
638      -1)))*(gamma_gas*Me_prima2(i)+1/Me_prima2(i))/(sqrt(1+...
639      (gamma_gas-1)/2*Me_prima2(i)^2));
640
```

```
641 %Thrust coefficient depending on local ambient pressure (with
    detachment)
642 cF2(i) = eta_nozzle(2)*(cF_v_prima2(i) - pa2(i)/pc*AeAt_prima2(i));
643 else
644  %Thrust coefficient depending on local ambient pressure
645 cF2(i) = eta_nozzle(2)*(cF_v(2) - (pa2(i)/pc)*Ae_At(2));
646 end
647
648 %Interpolation of different atmospheric parameters depending on
    rocket
649 %height
650 %Density
651 rho_h_2(i) = interp1(h(:), rho(:), h_rocket2(i));
652 %Gravity
653 g_h_2(i) = interp1(h(:), g(:), h_rocket2(i));
654 %Speed of sound
655 c_h_2(i) = interp1(h(:), c(:), h_rocket2(i));
656
657 %Rocket Mach number depending on velocity and speed of sound
658 M_flight2(i) = v2(i)/c_h_2(i);
659
660 %Rocket drag coefficient depending on Mach number
661 if M_flight2(i)<=0.6
662 c_d2(i) = 0.15;
663 elseif (0.6<M_flight2(i) && M_flight2(i)<=1.1)
664 c_d2(i) = -4.32*M_flight2(i)^3+11.016*M_flight2(i)^2-8.5536*...
665 M_flight2(i)+2.24952;
666 elseif (1.1<M_flight2(i)&& M_flight2(i)<=1.3)
667 c_d2(i) = -M_flight2(i)^(2)+2.2*M_flight2(i)-0.79;
668 elseif (1.3 < M_flight2(i)&& M_flight2(i)<= 50)
669 c_d2(i) = 0.16769+0.17636/(sqrt(M_flight2(i)^2-1));
670 else
671 c_d2(i) = 0.16769;
672 end
673
674 % Computation of drag depending on density, velocity and drag
    coefficient
675 D2(i) = (1/2)*rho_h_2(i)*v2(i)^2*S*c_d2(i);
676
677 %Thrust value depending on thrust coefficient and stage
    characterization
678 F2(i) = cF2(i)*pc*101325*At2_stage1;
679
680 %Specific impulse of the rocket
681 isp2(i)=F2(i)/(mflow_2stage1*9.81);
682
683 %Acceleration of the rocket
```

```
684 a2(i) = (F2(i)*cos(alpha)-D2(i))/m2(i)-g_h_2(i)*sin(gamma2(i));
685
686 %Dynamic pressure
687 q2(i) = 0.5*rho_h_2(i)*v2(i)^2;
688
689
690 %second stage
691 else if t_flight(i)>=tbs1prima && t_flight(i)<(tbs1prima+tbs2prima)
692
693       t_flight(i)=t(i);
694
695 %velocity of the rocket depending on time (with Earth rotation)
696 v2(i) = v2(i-1)+a2(i-1)*deltaT+464*cos(latitude)/length(t);
697
698 %Flight path angle depending on time
699 gamma2(i) = gamma2(i-1)+(((F2(i-1)/m2(i-1))*sin(alpha)-(g_h_2(i-1)-v2
      (i-1)^2/...
700 (h_rocket2(i-1)+R_e))*cos(gamma2(i-1)))/(v2(i-1)))*deltaT;
701
702 %rocket height depending on time
703 h_rocket2(i) = h_rocket2(i-1) + v2(i-1)*sin(gamma2(i-1))*deltaT;
704
705 %rocket mass depending on time
706 m2(i) = m0_stage2_2 - mflow_2stage2*t(i-(length(t)*tbs1prima/tb-1));
707
708 %Range of the rocket depending on time
709 x2(i) = x2(i-1)+(R_e/(R_e+h_rocket2(i-1)))*v2(i-1)*cos(gamma2(i-1))*
      deltaT;
710
711 %Ambient pressure interpolation depending on rocket height
712 pa2(i) = interp1(h(:), P(:), h_rocket2(i));
713
714 %Flow detachment analysis (employing Summerfield criterion)
715 if pe(3) < (0.4*pa2(i))
716
717 %If detachment of the flow occurs
718
719  Me_prima2 (i) = sqrt(2/(gamma_gas-1)*((...
720 pc/(0.4*pa2(i)))^((gamma_gas-1)/gamma_gas)-1));
721
722  m_Me2 (i) = sqrt(gamma_gas)*Me_prima2(i)/(1+...
723 (gamma_gas-1)/2*Me_prima2(i)^2)^((gamma_gas+1)/(2*(gamma_gas-1)));
724
725 AeAt_prima2 (i) = m_1/m_Me2(i);
726 cF_v_prima2 (i) = (2/(gamma_gas+1))^((gamma_gas+1)/(2*(gamma_gas...
727      -1)))*(gamma_gas*Me_prima2(i)+1/Me_prima2(i))/(sqrt(1+...
728      (gamma_gas-1)/2*Me_prima2(i)^2));
```

```
729
730 %Thrust coefficient depending on local ambient pressure (with
       detachment)
731 cF2(i) = eta_nozzle(3)*(cF_v_prima2(i) - pa2(i)/pc*AeAt_prima2(i));
732 else
733  %Thrust coefficient depending on local ambient pressure
734 cF2(i) = eta_nozzle(3)*(cF_v(3) - (pa2(i)/pc)*Ae_At(3));
735 end
736
737 %Interpolation of different atmospheric parameters depending on
       rocket
738 %height
739 %Density
740 rho_h_2(i) = interp1(h(:), rho(:), h_rocket2(i));
741 %Gravity
742 g_h_2(i) = interp1(h(:), g(:), h_rocket2(i));
743 %Speed of sound
744 c_h_2(i) = interp1(h(:), c(:), h_rocket2(i));
745
746 %Rocket Mach number depending on velocity and speed of sound
747 M_flight2(i) = v2(i)/c_h_2(i);
748
749 %Rocket drag coefficient depending on Mach number
750 if M_flight2(i)<=0.6
751 c_d2(i) = 0.15;
752 elseif (0.6<M_flight2(i) && M_flight2(i)<=1.1)
753 c_d2(i) = -4.32*M_flight2(i)^3+11.016*M_flight2(i)^2-8.5536*...
754 M_flight2(i)+2.24952;
755 elseif (1.1<M_flight2(i)&& M_flight2(i)<=1.3)
756 c_d2(i) = -M_flight2(i)^(2)+2.2*M_flight2(i)-0.79;
757 elseif (1.3 < M_flight2(i)&& M_flight2(i)<= 50)
758 c_d2(i) = 0.16769+0.17636/(sqrt(M_flight2(i)^2-1));
759 else
760 c_d2(i) = 0.16769;
761 end
762
763 % Computation of drag depending on density, velocity and drag
       coefficient
764 D2(i) = (1/2)*rho_h_2(i)*v2(i)^2*S*c_d2(i);
765
766 %Thrust value depending on thrust coefficient and stage
       characterization
767 F2(i) = cF2(i)*pc*101325*At2_stage2;
768
769 %Specific impulse of the rocket
770 isp2(i)=F2(i)/(mflow_2stage2*9.81);
771
```

```
772 %Acceleration of the rocket
773 a2(i) = (F2(i)*cos(alpha)-D2(i))/m2(i)-g_h_2(i)*sin(gamma2(i));
774
775 %Dynamic pressure
776 q2(i) = 0.5*rho_h_2(i)*v2(i)^2;
777
778     else
779   % Third stage
780
781  %flight time
782 t_flight(i)=t(i);
783
784 %velocity of the rocket depending on time (with Earth rotation)
785 v2(i) = v2(i-1)+a2(i-1)*deltaT+464*cos(latitude)/length(t);
786
787 %Flight path angle depending on time
788 gamma2(i) = gamma2(i-1)+(((F2(i-1)/m2(i-1))*sin(alpha)-(g_h_2(i-1)-v2
    (i-1)^2/...
789 (h_rocket2(i-1)+R_e))*cos(gamma2(i-1)))/(v2(i-1)))*deltaT;
790
791 %rocket height depending on time
792 h_rocket2(i) = h_rocket2(i-1) + v2(i-1)*sin(gamma2(i-1))*deltaT;
793
794 %rocket mass depending on time
795 m2(i) = m0_stage3_2 - mflow_2stage3*t(i-(length(t)*(tbs1prima+
    tbs2prima)/tb-1));
796
797 %Range of the rocket depending on time
798 x2(i) = x2(i-1)+(R_e/(R_e+h_rocket2(i-1)))*v2(i-1)*cos(gamma2(i-1))*
    deltaT;
799
800 %Ambient pressure interpolation depending on rocket height
801 pa2(i) = interp1(h(:), P(:), h_rocket2(i));
802
803 %Flow detachment analysis (employing Summerfield criterion)
804 if pe(3) < (0.4*pa2(i))
805
806 %If detachment of the flow occurs
807
808  Me_prima2 (i) = sqrt(2/(gamma_gas-1)*((...
809 pc/(0.4*pa2(i)))^((gamma_gas-1)/gamma_gas)-1));
810
811  m_Me2 (i) = sqrt(gamma_gas)*Me_prima2(i)/(1+...
812 (gamma_gas-1)/2*Me_prima2(i)^2)^((gamma_gas+1)/(2*(gamma_gas-1)));
813
814 AeAt_prima2 (i) = m_1/m_Me2(i);
815 cF_v_prima2 (i) = (2/(gamma_gas+1))^((gamma_gas+1)/(2*(gamma_gas...
```

```
816      -1)))*(gamma_gas*Me_prima2(i)+1/Me_prima2(i))/(sqrt(1+...
817      (gamma_gas-1)/2*Me_prima2(i)^2));
818
819 %Thrust coefficient depending on local ambient pressure (with
        detachment)
820 cF2(i) = eta_nozzle(3)*(cF_v_prima2(i) - pa2(i)/pc*AeAt_prima2(i));
821 else
822  %Thrust coefficient depending on local ambient pressure
823 cF2(i) = eta_nozzle(3)*(cF_v(3) - (pa2(i)/pc)*Ae_At(3));
824 end
825
826 %Interpolation of different atmospheric parameters depending on
        rocket height
827
828 %Density
829 rho_h_2(i) = interp1(h(:), rho(:), h_rocket2(i));
830 %Gravity
831 g_h_2(i) = interp1(h(:), g(:), h_rocket2(i));
832 %Speed of sound
833 c_h_2(i) = interp1(h(:), c(:), h_rocket2(i));
834
835 %Rocket Mach number depending on velocity and speed of sound
836 M_flight2(i) = v2(i)/c_h_2(i);
837
838 %Rocket drag coefficient depending on Mach number
839 if M_flight2(i)<=0.6
840 c_d2(i) = 0.15;
841 elseif (0.6<M_flight2(i) && M_flight2(i)<=1.1)
842 c_d2(i) = -4.32*M_flight2(i)^3+11.016*M_flight2(i)^2-8.5536*...
843 M_flight2(i)+2.24952;
844 elseif (1.1<M_flight2(i)&& M_flight2(i)<=1.3)
845 c_d2(i) = -M_flight2(i)^(2)+2.2*M_flight2(i)-0.79;
846 elseif (1.3 < M_flight2(i)&& M_flight2(i)<= 50)
847 c_d2(i) = 0.16769+0.17636/(sqrt(M_flight2(i)^2-1));
848 else
849 c_d2(i) = 0.16769;
850 end
851
852 % Computation of drag depending on density, velocity and drag
        coefficient
853 D2(i) = (1/2)*rho_h_2(i)*v2(i)^2*S*c_d2(i);
854
855 %Thrust value depending on thrust coefficient and stage
        characterization
856 F2(i) = cF2(i)*pc*101325*At2_stage3;
857
858 %Specific impulse of the rocket
```

```
isp2(i)=F2(i)/(mflow_2stage3*9.81);

%Acceleration of the rocket
a2(i) = (F2(i)*cos(alpha)-D2(i))/m2(i)-g_h_2(i)*sin(gamma2(i));

%Dynamic pressure
q2(i) = 0.5*rho_h_2(i)*v2(i)^2;
    end
    end
end

disp('Optimized rocket computed!')
disp('Calculations are finished!')

% %cd depending on Mach number
% figure
% plot (M_flight, c_d)
% grid on; grid minor;
% title ('Drag coefficient depending on Mach number','interpreter','
    latex');
% xlabel ('Mach number','interpreter','latex');
% ylabel ('Drag coefficient','interpreter','latex');
% xlim ([0 7]); ylim ([0 0.5])

%PLOTS OF THE PERFORMANCE OF THE ROCKET
figure
%velocity of the rocket depending on time
subplot (3,3,1)
plot (t_flight,v1/1000,t_flight,v2/1000)
grid on; grid minor;
title ('Velocity','interpreter','latex');
xlabel ('Time [s]','interpreter','latex');
ylabel ('Velocity [km/s]','interpreter','latex');
ylim ([0 8]);


%Mach number of the rocket depending on time
subplot (3,3,2)
plot (t_flight,M_flight1,t_flight,M_flight2)
grid on; grid minor;
title ('Mach number','interpreter','latex');
xlabel ('Time [s]','interpreter','latex');
ylabel ('Mach number','interpreter','latex');

%Flight path angle of the rocket depending on time
subplot (3,3,3)
plot (t_flight,rad2deg(gamma1),t_flight,rad2deg(gamma2))
```

```
905 grid on; grid minor;
906 title ('Flight path angle','interpreter','latex');
907 xlabel ('Time [s]','interpreter','latex');
908 ylabel ('Angle path [deg]','interpreter','latex');
909
910 %height of the rocket depending on time
911 subplot (3,3,4)
912 plot (t_flight,h_rocket1/1000,t_flight,h_rocket2/1000)
913 grid on; grid minor;
914 title ('Height','interpreter','latex');
915 xlabel ('Time [s]','interpreter','latex');
916 ylabel ('Height [km]','interpreter','latex');
917 ylim ([0 220])
918
919 %Range of the rocket depending on time
920 subplot (3,3,5)
921 plot (t_flight,x1/1000,t_flight,x2/1000)
922 grid on; grid minor;
923 title ('Range','interpreter','latex');
924 xlabel ('Time [s]','interpreter','latex');
925 ylabel ('Range [km]','interpreter','latex');
926 legend ('Non-optimized','Optimized','interpreter','latex')
927
928 %Acceleration of the rocket depending on time
929 subplot (3,3,6)
930 plot (t_flight,a1/g(1),t_flight,a2/g(1))
931 grid on; grid minor;
932 title ('Gravitational force equivalent','interpreter','latex');
933 xlabel ('Time [s]','interpreter','latex');
934 ylabel ('G-force','interpreter','latex');
935
936
937 %Mass of the rocket depending on time
938 subplot (3,3,7)
939 plot (t_flight,m1/1000,t_flight,m2/1000)
940 grid on; grid minor;
941 title ('Mass','interpreter','latex');
942 xlabel ('Time [s]','interpreter','latex');
943 ylabel ('Mass [t]','interpreter','latex');
944
945
946
947 %Dynamic pressure of the rocket depending on time
948 subplot (3,3,8)
949 plot (t_flight,q1/1000,t_flight,q2/1000)
950 grid on; grid minor;
951 title ('Dynamic pressure','interpreter','latex');
```

```
952 xlabel ('Time [s]','interpreter','latex');
953 ylabel ('Dynamic pressure [$kPa$]','interpreter','latex');
954 ylim ([0 35]);
955
956 %Specific impulse of the rocket depending on time
957 subplot (3,3,9)
958 plot (t_flight,isp1,t_flight,isp2)
959 grid on; grid minor;
960 title ('Specific impulse','interpreter','latex');
961 xlabel ('Time [s]','interpreter','latex');
962 ylabel ('Specific impulse [s]','interpreter','latex');
963 ylim ([290 370])
964
965
966 %% ORBITAL INSERTION
967
968 % Earth gravitational standard parameter
969 mu_earth=3.986004418*10^14;                          %[m^3/s^2]
970
971 % Target final velocity
972 Final_Budget=sqrt(mu_earth/(R_e+200*10^3));            %[m/s]
973
974 % Semi-major axis
975 a=(2/(h_rocket2(end)+R_e)-v2(end)^2/mu_earth)^-1; %[m]
976
977 % Radius at periapsis of the final orbit
978 r_p=h_rocket2(end)+R_e;                              %[m]
979
980 % Radius at apoapsis of the final orbit
981 r_a=2*a-r_p;                                          %[m]
982
983 % Eccentricity of the final orbit
984 e=(r_a-r_p)/(r_a+r_p);
985
986 % Semi-minor axis of the final orbit
987 b=a*sqrt(1-e^2);                                      %[m]
988
989 % True anomaly [from 0 to 360 degrees]
990 theta=linspace(0,2*pi,1500);                          %[rad]
991
992 % Orbital insertion
993 xx=a*cos(theta);
994 yy=b*sin(theta);
995
996 % Ideal orbit
997 xxcirc=(200000+R_e)*cos(theta);
998 yycirc=(200000+R_e)*sin(theta);
```

```
999
1000 % Earth
1001 xxearth=cos(theta);
1002 yyearth=sin(theta);
1003
1004 % Plot of the orbit characterization
1005 figure
1006 plot (xx/R_e,yy/R_e,'b')
1007 hold on
1008 plot (xxcirc/R_e,yycirc/R_e,'r')
1009 plot (xxearth,yyearth,'k')
1010 fill (xxearth,yyearth,'c')
1011 xlabel ('Distance (Earth radius)','interpreter','latex')
1012 ylabel ('Distance (Earth radius)','interpreter','latex')
1013 title ('Final orbit characterization','interpreter','latex');
1014 xlim ([-1.05 1.05])
1015 ylim ([-1.05 1.05])
1016 axis equal
1017 legend ('Final orbit','Ideal orbit','Earth surface','Earth',...
1018 'interpreter','latex','location','best')
1019
1020
1021 %% FUNCTIONS
1022
1023 %US standard atmosphere characterization function
1024 function [T, P, rho, g, c] = US_Standard(h)
1025
1026 %The present function aims at assessing the Temperature, Pressure,
1027 %Density, Gravity force and sound velocity at a given altitude,
1028 %according to US Standard atmosphere (1976)
1029
1030 % Data
1031 gamma_air =1.4;                  % Adiabatic expansion coefficient
1032 g_0 = 9.80665;                   % Gravity [m/s^2]
1033 R_e = 6371e3;                    % Radius Earth [m]
1034 R = 287;                         % Ideal Gas constant [J/(kg K)]
1035
1036 % Temperature gradient [K/m] / constants
1037 lambda(1) = -6.5/1000;            % 0     - 10999
1038 lambda(2) = 0;                    % 11000 - 19999
1039 lambda(3) = 1/1000;               % 20000 - 31999
1040 lambda(4) = 2.8/1000;             % 32000 - 46999
1041 lambda(5) = 0;                    % 47000 - 50999
1042 lambda(6) = -2.8/1000;            % 51000 - 70999
1043 lambda(7) = -2/1000;              % 71000 - 85999
1044 lambda(8) = 0;                    % 86000 - 90999
1045 Tc = 263.1905;                   % 91000 - 110000          [K]
```

```
A = -76.3232;                       % 91000 - 110000
a = -19942.9;                       % 91000 - 110000
lambda(9) = 12/1000;                % 110000 - 120000
T_inf = 1000;                       % 120000 - 1000000          [K]
LAMBDA = 0.01875;                   % 120000 - 1000000

% Vectors
%Definition of vectors to optimize the matlab code
T =zeros(1,length(h));                  % Temperature [K]
P =zeros(1,length(h));                  % Pressure [atm]
rho =zeros(1,length(h));                % Density [kg/m^3]
g =zeros(1,length(h));                  % Gravity [m/s^2]
c =zeros(1,length(h));                  % Sound velocity [m/s]

% Calculations
for i = 1:length(h)

%Depending on the height, different properties are found
    if h(i)<11000
        T_0 = 288.15;
        T(i) = T_0 + h(i)*lambda(1);
        P_0 = 1;
        P(i) = P_0*(T(i)/T_0)^(-g_0/(R*lambda(1)));
        rho_0 = 1.225;
        rho(i) = rho_0*(T(i)/T_0)^(-1-g_0/(lambda(1)*R));

    elseif h(i)>=11000 && h(i)<20000
        T_11 = 288.15 + 11000*lambda(1);
        T(i) = T_11;
        P_11 = P_0*(T_11/T_0)^(-g_0/(R*lambda(1)));
        P(i) = P_11*exp(-g_0/(R*T_11)*(h(i)-11000));
        rho_11 = rho_0*(T_11/T_0)^(-1-g_0/(lambda(1)*R));
        rho(i) = rho_11*exp(-g_0/(R*T_11)*(h(i)-11000));

    elseif h(i)>=20000 && h(i)<32000
        T(i) = T_11 + lambda(3)*(h(i) - 20000);
        P_20 = P_11*exp(-g_0/(R*T_11)*(20000-11000));
        P(i) = P_20*(T(i)/T_11)^(-g_0/(R*lambda(3)));
        rho_20 = rho_11*exp(-g_0/(R*T_11)*(20000-11000));
        rho(i) = rho_20*(T(i)/T_11)^(-1-g_0/(lambda(3)*R));

    elseif h(i)>=32000 && h(i)<47000
        T_32 = T_11 +lambda(3)*(32000-20000);
        T(i) = T_32 + (h(i) - 32000)*lambda(4);
        P_32 = P_20*(T_32/T_11)^(-g_0/(R*lambda(3)));
        P(i) = P_32*(T(i)/T_32)^(-g_0/(R*lambda(4)));
        rho_32 = rho_20*(T_32/T_11)^(-1-g_0/(lambda(3)*R));
```

```
        rho(i) = rho_32*(T(i)/T_32)^(-1-g_0/(lambda(4)*R));

    elseif h(i)>=47000 && h(i)<51000
        T_47 = T_32 +(47000-32000)*lambda(4);
        T(i) = T_47;
        P_47 = P_32*(T_47/T_32)^(-g_0/(R*lambda(4)));
        P(i) = P_47*exp(-g_0/(R*T_47)*(h(i)-47000));
        rho_47 = rho_32*(T_47/T_32)^(-1-g_0/(lambda(4)*R));
        rho(i) = rho_47*exp(-g_0/(R*T_47)*(h(i)-47000));

    elseif h(i)>=51000 && h(i)<71000
        T(i) = T_47 +(h(i) - 51000)*lambda(6);
        P_51 = P_47*exp(-g_0/(R*T_47)*(51000-47000));
        P(i) = P_51*(T(i)/T_47)^(-g_0/(R*lambda(6)));
        rho_51 = rho_47*exp(-g_0/(R*T_47)*(51000-47000));
        rho(i) = rho_51*(T(i)/T_47)^(-1-g_0/(lambda(6)*R));

    elseif h(i)>=71000 && h(i)<84900
        T_71 = T_47 + (71000-51000)*lambda(6);
        T(i) = T_71 + (h(i) -71000)*lambda(7);
        P_71 = P_51*(T_71/T_47)^(-g_0/(R*lambda(6)));
        P(i) = P_71*(T(i)/T_71)^(-g_0/(R*lambda(7)));
        rho_71 = rho_51*(T_71/T_47)^(-1-g_0/(lambda(6)*R));
        rho(i) = rho_71*(T(i)/T_71)^(-1-g_0/(lambda(7)*R));

    elseif h(i)>=84900 && h(i)<91000
      T_85 = T_71 + (84900-71000)*lambda(7);
      P_85 = P_71*(T_85/T_71)^(-g_0/(R*lambda(7)));
      T(i) = T_85;
      P(i) = P_85*exp(-g_0/(R*T_85)*(h(i)-84900));
      rho_85 = rho_71*(T_85/T_71)^(-1-g_0/(lambda(7)*R));
      rho(i) = rho_85*exp(-g_0/(R*T_85)*(h(i)-85000));

    elseif h(i)>=91000 && h(i)<110000
     T_91 = T_85;
     P_91 = P_85*exp(-g_0/(R*T_85)*(91000-84900));
     P(i) = P_91*exp(-g_0/(R*T_91)*(h(i)-91000));
     T(i) = Tc+A*(1-((h(i)-91000)/(a))^2)^(1/2);
     rho_91 = rho_85*exp(-g_0/(R*T_85)*(91000-85000));
     rho(i) = rho_91*exp(-g_0/(R*T_91)*(h(i)-91000));

    elseif h(i)>=110000 && h(i)<120000
     T_110 = Tc+A*(1-((110000-91000)/(a))^2)^(1/2);
     P_110 = P_91*exp(-g_0/(R*T_91)*(110000-91000));
     T(i) = T_110+lambda(9)*(h(i)-110000);
     P(i) = P_110*(T(i)/T_110)^(-g_0/(R*lambda(9)));
     rho_110 = rho_91*exp(-g_0/(R*T_91)*(110000-91000));
```

```
1140        rho(i) = rho_110*(T(i)/T_110)^(-1-g_0/(lambda(9)*R));
1141
1142
1143        elseif h(i)>=120000 && h(i)<1000000
1144         T_120=T_110+lambda(9)*(120000-110000);
1145         P_120 = P_110*(T_120/T_110)^(-g_0/(R*lambda(9)));
1146         nu=(h(i)-120000)/1000; %parameter
1147         T(i) = T_inf-(T_inf-T_120)*exp(-LAMBDA*nu);
1148         P(i) = P_120*exp(-g_0/(R*T_120)*(h(i)-120000));
1149         rho_120 = rho_110*(T_120/T_110)^(-1-g_0/(lambda(9)*R));
1150         rho(i) = rho_120*exp(-g_0/(R*T_120)*(h(i)-120000));
1151
1152        else %Higher heights than 1000 km
1153         T(i)=T_inf;
1154         P_1000=P_120*exp(-g_0/(R*T_120)*(1000000-120000));
1155         P(i) = P_1000*exp(-g_0/(R*T_inf)*(h(i)-1000000));
1156         rho_1000 = rho_120*exp(-g_0/(R*T_120)*(1000000-120000));
1157         rho(i) = rho_1000*exp(-g_0/(R*T_inf)*(h(i)-1000000));
1158
1159        end
1160        g(i) = g_0*(R_e/(R_e + h(i)))^2;
1161        c(i) = sqrt(gamma_air*R*T(i));
1162    end
1163    end
1164
1165
1166    %Exit Mach reckoning (depending on gamma and relation of areas)
1167    function [Me] = Exit_Mach(gamma_gas,Ae_At)
1168
1169    %The present function aims at assessing the Mach number at the exit
1170    %of the nozzle depending on the gamma of the gas and the relation of
        areas
1171
1172    syms  Me ;                          %Definition of Me as a variable
1173    low=0.1;                            %Lower end of the calculation
1174    up=20;                              %upper end of the calculation
1175    tol=10e-9;                          %Convergence tolerance
1176    n=ceil(log((up-low)/tol)/log(2)); %Number of iterations to achieve
        convergence
1177
1178    f=-Ae_At+ (2/(gamma_gas+1))^((gamma_gas+1)/(2*(gamma_gas-1)))*...
1179    (1/Me)*(1+(gamma_gas-1)*0.5*Me^2)^((gamma_gas+1)/(2*(gamma_gas-1)));
1180    f=inline(f);
1181    for i=1:n
1182    Me=(low+up)/2;
1183    if f(low)*f(Me)<0
1184    up=Me;
```

```
1185 else
1186 low=Me;
1187 end
1188 end
1189 end
1190
1191
1192 %Exit Mach reckoning (depending on gamma and relation of areas)
1193 function [Mestudy] = Exit_Mach_study(gamma_gas,Ae_At_study)
1194
1195 %The present function aims at assessing the Mach number at the exit
1196 %of the nozzle depending on the gamma of the gas and the relation of
     areas
1197
1198 syms  Mestudy ;                    %Definition of Me as a variable
1199 low=0.1;                           %Lower end of the calculation
1200 up=20;                             %upper end of the calculation
1201 tol=10e-9;                         %Convergence tolerance
1202 n=ceil(log((up-low)/tol)/log(2)); %Number of iterations to achieve
     convergence
1203
1204 f=-Ae_At_study+ (2/(gamma_gas+1))^((gamma_gas+1)/(2*(gamma_gas-1)))*
     ...
1205 (1/Mestudy)*(1+(gamma_gas-1)*0.5*Mestudy^2)^((gamma_gas+1)/(2*(
     gamma_gas-1)));
1206 f=inline(f);
1207 for i=1:n
1208 Mestudy=(low+up)/2;
1209 if f(low)*f(Mestudy)<0
1210 up=Mestudy;
1211 else
1212 low=Mestudy;
1213 end
1214 end
1215 end
```

Appendix C

Elaboration of budget plots

```
%% SETUP
close all; clc; clear all;

%% Information
%Author: Roger Verges Eiras

%Director: Josep Oriol Lizandra i Dalmases

%Bachelor's Final Thesis MATLAB CODE

%Project name: Study and Evaluation of the Performance of a
    Multistage Solid-Propellant Rocket Vehicle, Including Atmospheric
    Ascent and Orbital Insertion

%Course 2020-2021

%% ACADEMIC FEES
figure
x = [62 203 35];
pie (x)
legend ('Theoretical developments','Research and evaluation',...
    'Documentation','interpreter','latex')

%% LAUNCHING PROCESS

% Non-optimized case
figure
y = [6.837 12 3 1 1.2 1.25];
pie (y)
legend ('Propellant','Structure','Engines','Payload integration',...
    'Separation of stages','Operation of the mission',...
    'interpreter','latex')
```

```
31
32 % Optimized case
33 figure
34 z = [3.388 6.1 4.5 1 2.4 1.25];
35 pie (z)
36 legend ('Propellant','Structure','Engines','Payload integration',...
37     'Separation of stages','Operation of the mission',...
38     'interpreter','latex')
```