Design and Performance Analysis
Study of an Ion Thruster

Final Degree Project - Annex

Bachelor’s Degree in Aerospace Technology Engineering

Author: Carlos Sánchez Lara
Director: Josep Oriol Lizandra Dalmases

Terrassa, 22nd June, 2016
Contents

Annex A ........................................................................................................................................... 4
  A.1. Hohmann transfer parameters calculation ......................................................... 4
Annex B ........................................................................................................................................... 7
  B.1. Rocket equation ............................................................................................................. 7
Annex C ........................................................................................................................................... 8
  C.1. Baseline plasma ion energy cost determination. Iterative process ................. 8
  C.2. Maxwellian electron temperature determination. Iterative process ............ 10
  C.3. Exhaust velocity optimum solution. Iterative process ...................................... 12
Annex D ........................................................................................................................................... 13
  D.1. MATLAB code for calculating performance curves ........................................... 13
  D.2. MATLAB code for optimised performance parameters obtainment ............. 18
  D.3. MATLAB code for solving orbital motion differential equations .................. 20
  D.4. Mission analysis MATLAB code .............................................................................. 23
Bibliography .................................................................................................................................. 24
Annex A

A.1. Hohmann transfer parameters calculation.

Hohmann transfer orbit is an elliptical orbit used to transfer between two circular orbits of different radius in the same plane. It is the most efficient planar manoeuvre (regarding \( \Delta v \) budget), and requires two impulses, \( \Delta v_1 \) and \( \Delta v_2 \).

First one is used to move onto the ellipse from LEO.

\( \Delta v_1 \) is the difference between the velocity at the perigee of the ellipse and that of the circular orbit in the same point, as it is assumed an instantaneous impulse.

\[
\Delta v_1 = v_1 - v_{c_1}
\]  
(A.1)

Velocities are obtained from the total energy equation (Eq. A.2) of the spacecraft, being that the sum of its kinetic and potential energies.

\[
E = \frac{1}{2} m v^2 - \frac{GMm}{r} = -\frac{GMm}{2a}
\]  
(A.2)

Where \( M_E \) is the mass of the Earth, \( m \) the mass of the spacecraft and \( a \) the semimajor axis of the ellipse. Thus,

\[
v_{c_1} = \sqrt{\frac{\mu}{r_1}}
\]  
(A.3)

\[
v_1 = \frac{2\mu r_2}{\sqrt{r_1(r_1 + r_2)}}
\]  
(A.4)

Where \( r_1 \) and \( r_2 \) are respectively, the radius of the LEO and GEO, and \( \mu \) is the gravitational parameter of the Earth \( (\mu = GM_E) \). Finally,

\[
\Delta v_1 = \frac{2\mu r_2}{\sqrt{r_1(r_1 + r_2)}} - \sqrt{\frac{\mu}{r_1}} = \sqrt{\frac{\mu}{r_1}} \left( \sqrt{\frac{2r_2}{r_1 + r_2}} - 1 \right)
\]  
(A.5)
The same procedure is followed to obtain $\Delta v_2$, which is the difference between the velocity at GEO and that of the apogee.

$$\Delta v_2 = v_{c_2} - v_2$$  \hspace{1cm} (A.6)

where,

$$v_{c_2} = \sqrt{\frac{\mu}{r_2}}$$  \hspace{1cm} (A.7)

$$v_1 = \sqrt{\frac{2\mu r_1}{r_2(r_1 + r_2)}}$$  \hspace{1cm} (A.8)

And,

$$\Delta v_2 = \sqrt{\frac{\mu}{r_2}} \left( 1 - \sqrt{\frac{2r_1}{r_1 + r_2}} \right)$$  \hspace{1cm} (A.9)

Therefore, total delta-V required for performing a Hohmann transfer is

$$\Delta v_H = \Delta v_1 + \Delta v_2$$  \hspace{1cm} (A.10)

Hereafter, the conditions of our orbits will be imposed:

$$r_1 = R + h_1 = 6378 + 300 = 6678 km$$

In order to get $r_2$, we should analyse GEO features. Geostationary orbit implies that both, spacecraft and Earth (rotation), have equal angular velocities:

$$\omega_{sc} = \omega_e = \frac{2\pi}{T} = \frac{2\pi}{23.93 \cdot 3600} = 7.293 \cdot 10^5 \frac{rad}{s}$$

Equalling gravitational and centripetal forces yields,

$$m \cdot \frac{d^2 r}{dt^2} = F_g = \frac{GMm}{r^2} \rightarrow \frac{d^2 r}{dt^2} = g(r_2) = \frac{\mu}{r_2^2}$$  \hspace{1cm} (A.11)

Finally, from eq.XX 3.11 we can obtain de synchronous radius,

$$r_2 = \sqrt[3]{\frac{\mu}{\omega_{sc}^2}} = 42241 km$$

And using Eqs. A.5, A.9 and A.10,

$$\Delta v_1 = 2.431 km/s$$

$$\Delta v_2 = 1.469 km/s$$
\[ \Delta v_H = 3.900 \text{ km/s} \]

The time employed to accomplish the transference is a half period of the ellipse, which can be obtained by means of Kepler’s law,

\[ \frac{4\pi^2}{T_H^2} a^3 = \mu \]  \hspace{1cm} (A.12)

where \( a \) is the semi-major axis of the ellipse and is equal to \( \frac{r_1 + r_2}{2} \). Then, the time needed is,

\[ t_H = \frac{T}{2} = \pi \sqrt{\frac{(r_1 + r_2)^3}{8\mu}} = 18982.42 \text{ s} = 5.27 \text{ h} \]
Annex B

B.1. Rocket equation

From the momentum conservation (Eq. B.3), we can obtain the expression that gives us the delta-V provided by a rocket given its characteristic parameters. First of all, we define the momentum in an instant \( t \) and after an interval \( dt \),

\[
P(t) = mv
\]  
\[
P(t + dt) = (m - dm)(v + dv) + dm'v'
\]

Equalling the expressions above:

\[
mv = (m - dm)(v + dv) + dm'v'
\]  \( \text{(B.3)} \)

Considering that the exhaust velocity of the gases \( u \) is given by,

\[u = -v' + (v + dv)\]

we get,

\[mdv = dm_u\]

Eq. 3.15, after some modifications yields,

\[
m \frac{dv}{dt} = \frac{dm}{dt} u = -l_{sp}m_0g_0
\]

where,

\[l_{sp} = u/g_0\]

Integrating eq. 3.16, we obtain the Tsiolkovsky rocket equation:

\[
\Delta v = l_{sp}g_0 \ln \frac{m_i}{m_f}
\]

(B.7)

Since gravity loses are not taken into account, it is applicable to orbital impulsive manoeuvres, in which the propellant is discharged and delta-V applied instantaneously. Moreover, burns are applied tangentially to Earth gravity field, so the assumption is reasonably accurate.
Annex C

C.1. Baseline plasma ion energy cost determination. Iterative process

In this section, we will detail the iterative process that must be followed in order to obtain the value of the baseline plasma ion energy cost ($\varepsilon_P^*$) as a function of the Maxwellian electron temperature in the bulk plasma ($T_M$).

To that end, we will make use of Eqs. 6.29 and 7.11 from the Report [1]. The first one is retrieved here below,

$$\varepsilon_P^* = \frac{\varepsilon_0 + \varepsilon_M}{1 - \frac{V_C + \varepsilon_M}{V_D}}$$

Using $\varepsilon_0$ (from Eq. 7.2 [1]) yields to,

$$\varepsilon_P^* = \frac{U_+ + \varepsilon_M + \left(\frac{n_P}{n_M} \sigma_{ex} v_p + \langle \sigma_{ex} v_e \rangle_M \right) U_{ex}}{\frac{n_P}{n_M} \sigma_{ex} v_p + \langle \sigma_{ex} v_e \rangle_M + \frac{V_C + \varepsilon_M}{V_D}}$$  \hspace{1cm} \text{(C.1)}$$

Assuming certain operating conditions ($V_D$, $V_C$ and $T_M$), we must suppose an initial value for $\frac{n_P}{n_M}$, which we will call, “auxiliary”, denoted with a superindex, $\frac{n_P^{aux}}{n_M}$.

We can calculate now $\varepsilon_P^{*aux}$, which is $\varepsilon_P^*$ evaluated at $\frac{n_P^{aux}}{n_M}$.

Then, we use Eq. 7.11 [1] and rearrange terms to obtain the new $\frac{n_P}{n_M}$,
\[
\frac{n_p}{n_M} = \frac{\langle \sigma_+ n_+ \rangle_M}{\sigma_+ n_p} \left( \frac{V_D \sigma_0}{\kappa_p \sigma_+} - 1 \right) \tag{C.2}
\]

We compare the result obtained with \( \frac{n_p^{\text{aux}}}{n_M} \).

\[
dif = \left| \frac{n_p^{\text{aux}}}{n_M} - \frac{n_p}{n_M} \right|
\]

and, if \( dif \) is bigger than a threshold, we go back into Eq. C.1 taking,

\[
\frac{n_p^{\text{aux}}}{n_M} = \frac{n_p}{n_M} \tag{C.3}
\]

This process must be repeated until get a value for \( dif \) smaller than a given limit.

Eq. C.3 could incorporate a relaxation factor to improve the iterative process or maybe, for making it possible, because not always one gets a convergent solution.

Iterative process

We suppose that operating conditions \((\dot{m}, V_D, \eta_u)\) and design parameters are known, then,

Supposing an initial Maxwellian electron temperature \(T_M^{aux}\), we can calculate \(\varepsilon_p^{*aux}\) following the process explained in last section.

After that, we obtain the rate factor \(Q_0^{aux}\) from Eq. 7.12 [1], retrieved here below,

\[
Q_0^{aux} = \frac{v_p^* \frac{v_p \sigma' \sigma 0'}{e_p^{aux} \sigma_+} (1)}{0.15 e_p^{aux} v_0 V_B A_B^2 \phi_0^2 \phi_i} \dot{m} (1 - \eta_u) - 1
\]  
(C.4)

Now that we have the rate factor, we can obtain the corresponding electron temperature from Fig. 7.8 [1]. Such distribution actually corresponds to experimental equations found in [2] and which are presented below,

For \(T_M < 5 \text{ eV}\)

\[
Q_0^+ = 10^{-20} \left[ (3.97 + 0.643 T_M - 0.0368 T_M^2) e^{-12.127/T_M} \right] \left( \frac{8e T_M}{\pi m_e} \right)^{1/2}
\]  
(C.5)

For \(T_M > 5 \text{ eV}\)

\[
Q_0^+ = 10^{-20} \left[ -1.031 \times 10^{-4} T_M^2 + 6.386 e^{-12.127/T_M} \right] \left( \frac{8e T_M}{\pi m_e} \right)^{1/2}
\]  
(C.6)

The Maxwellian temperature cannot be isolated from Eqs. C.4 and C.5, so a further iterative process should be applied.

Once we have calculated such temperature, \(T_M\), we can compare it with the initial one,

\[
dif = |T_M^{aux} - T_M|
\]
If $d_{if}$ is bigger than a threshold, we go back into Eq. C.4 taking,

$$T_{M}^{aux} = T_{M}$$  \hspace{1cm} (C.7)

This process must be repeated until get a value for $d_{if}$ smaller than a given limit.

Eq. C.7 could incorporate a relaxation factor to improve the iterative process or maybe, for making it possible, because not always one gets a convergent solution.
Annex C

C.3. Exhaust velocity optimum solution. Iterative process

Optimum thruster’s exhaust velocity is obtained by equalling to zero the derivative of the payload mass fraction (Eq. 9.14) [1]. Such derivative results in Eq. 9.15 [1], which is recovered down below,

\[
\frac{d \left( \frac{M_i}{M_l} \right)}{du} = \frac{ae^{-a/u}}{u^2} - 2bu \left( 1 - e^{-a/u} \right) + \frac{abe^{-a/u} (c + u^2)}{u^2} = 0
\]

Where,

\[
a = \frac{\Delta v}{\eta_u} \quad b = \frac{\alpha \eta_u}{2t} \quad c = \frac{2e\epsilon_B}{m_i}
\]

We can transform eq. 9.15 into:

\[
\sqrt{\frac{2b}{a}} \left( e^{\frac{a}{u}} - 1 \right) = u
\]

\[3 \left( \frac{a (1 + b(c + u^2))}{2b} \right) = u \quad \text{(C.8)}\]

If we consider an initial exhaust velocity \(u^{aux}\), we can calculate the new one using Eq. C.8.

Once we have calculated such velocity, \(u\), we can compare it with the initial one,

\[
dif = |u^{aux} - u|
\]

If it is bigger than a threshold, we go back into Eq. C.8 taking,

\[
u^{aux} = u \quad \text{(C.9)}
\]

This process must be repeated until get a value for dif smaller than a given limit.

Eq. C.9 could incorporate a relaxation factor to improve the iterative process or maybe, for making it possible, because not always one gets a convergent solution.
MATLAB codes

D.1. MATLAB code for calculating performance curves

```matlab
% This script computes the value of the beam ion energy cost (eV/beam ion) as a function of the propellant utilisation efficiency.
%
clear
clc

%-------------------CONSTANTS-------------------
const.e=1.60217662e-19;  % Electron charge (C)
const.me=9.10938356e-31;  % Electron mass (kg)
const.kb=1.3806505e-23;  % Boltzmann constant (J/K) //
const.kt=11604.505;  % Constant to convert from eV to k
const.Uion=12.12984;  % Xenon first ionisation energy (eV)
const.Ul=8.315;  % Xenon first excitation energy (eV)
const.Tw=400;  % Wall temperature (K)
const.M=2.1802225e-25;  % Mass of a Xenon atom (kg)
const.d=0.12;  % Diameter of ion thruster (m)
const.Ag=pi()*const.d^2/4;  % Area of grids through which the ion beam...
const.V=1.5825e-4;  % Volume m^3
const.phi_o=0.16;  % SHAG
const.phi_i=0.8;  % Transparency to ions
const.le=1;  % Primary electron containment length (m)
const.Uex=0.5*(const.Ul+const.Uion);  % Lumped excitation energy (V)
const.Va=2;  % It is the difference between plasma potential and anode potential (V)

Vd=40;  % Discharge voltage (V)
Vc=0;  % Plasma potential from which the e- are supplied.
       % Hollow cathode potential (V)
dm=1;  % Propellant flow rate (A eq)
fc=0.1;  % Fraction of ion current to cathode surfaces
fb=0.5;  % Extracted ion fraction

% Function that returns the value of Co for given Vd and Vc and constants dependent of the geometry of the thruster and propellant employed.
Co = calc_Co(Vd,Vc,const);  % Primary electron utilisation factor

nu=0.1:0.01:0.95;  % propellant utilisation efficiency
% Calculation of the beam ion cost and as a function of nu for an specific configuration.
```
Annex D

\[
[EpsB] = \text{calc\_EpsB}(\text{const}, V_d, V_c, \nu, d_m, f_b, f_c, C_0);
\]

```matlab
function [C_0] = calc_Co(V_d, V_c, const)

\[
vo = (8*\text{const}.k_b*\text{const}.T_w/pi()/\text{const}.M)^0.5; \quad \% \text{Neutral atom velocity}
\]

sigma_exP, sigma_ionP = cross_sections(V_d, V_c); \quad \% \text{Cross sections obtention}

sigma_inelastic = sigma_exP + sigma_ionP; \quad \% \text{Total inelastic collision cross section at the primary electron energy (m}^2\text{)}

\% We compute the value of the primary electron utilisation factor as a function of the engine design.
\text{Co}=4*sigma_inelastic*\text{const}.le/\text{const}.e/vo/\text{const}.Ag/\text{const}.phi_o;
end

```

```matlab
% Function that returns the value of the beam ion cost given the main performance parameters as a function of nu.
function [EpsB] = calc_EpsB(const,V_d,V_c,nu,dm,fb,fc,Co)

\[
v_P = (2*\text{const}.e*(V_d-V_c)/\text{const}.m_e)^0.5; \quad \% \text{Primary electron velocity (m/s)}
\]

sigma_exP, sigma_ionP = cross_sections(V_d, V_c); \quad \% \text{Function that returns the value of the excitation and ionisation collision cross sections at primary electron energy.}

sigma_inelastic = sigma_exP + sigma_ionP; \quad \% \text{Total inelastic collision cross section at the primary electron energy (m}^2\text{)}

for i=1:length(nu);
    \% We call a function that iterate in order to obtain \text{Tm} and EpsBas for each specific thruster configuration
    [EpsBas(i), Tm(i)] = Tm_Iteration(const, V_d, V_c, sigma, v_P, nu(i), dm);
    \% We finally obtain the value of EpsB for each nu
    EpsB(i) = (EpsBas(i)/(1-exp(1)^(-Co*dm*(1-nu(i))))) / fb ... + fc*V_d/fb;
end
end

```

```matlab
% Function that returns the value of the excitation and ionisation collision cross sections at primary electron energy (E=f(Vd,Vc))
function [sigma_exP, sigma_ionP] = cross_sections(V_d, V_c)

% Experimental data
CSion=[12.5 1.099e-21 ; 13.5 4.123e-21 ; 17.5 1.670e-20 ; 21 2.488e-20 ; 25 3.613e-20 ;
13.5 1.073e-20 ; 16.5 1.802e-20 ; 20.5 2.831e-20; 23 2.928e-20; 24 3.095e-20; 26 3.367e-20; 28 3.613e-20;...
14.5 7.420e-21 ; 15 9.055e-21 ; 15.5 1.231e-20 ; 16.5 1.380e-20 ; 17 1.529e-20 ; 17.5 1.670e-20 ; 18 1.802e-20 ;
18.5 1.925e-20; 19 2.048e-20; 20 2.277e-20; 20.5 2.382e-20; 21 2.488e-20; 22.5 2.831e-20; 23 2.928e-20;
24 3.095e-20; 26 3.367e-20; 28 3.613e-20;...
```

14 / 24
Annex D

30 3.851e-20 ; 32 4.044e-20; 34 4.185e-20 ; 36 4.290e-20;
38 4.387e-20; 40 4.475e-20 ;45 4.677e-20; 50 4.835e-20 ];

CSex=[12.5  1.05e-20;13  1.28e-20; 14  1.7e-20;...
15  2.14e-20; 16  2.55e-20; 18  3.35e-20;...
20  3.73e-20; 25  3.85e-20; 30  3.57e-20; 40  2.85e-20;...
50  2.4e-20];

% We interpolate into tables above in the case we do not have the exact
% value
sigma_exP=interp1(CSion(:,1),CSion(:,2),Vd-Vc);
sigma_ionP=interp1(CSex(:,1),CSex(:,2),Vd-Vc);
end

% We call the function that provides the baseline plasma ion
% energy cost as a function of the bulk plasma electrons
% temperature and the operating conditions (Vd, Vc)
function [EpsBas,PtoM] = calc_EpsBas(Tm, const, Vd, Vc)

  % Calculation of the rate factor
  A=vP*sigma_ionP*(Vd*sigma.inelastic/EpsBas/sigma_ionP-1);
  B=vP*Vd*sigma.inelastic*const.V/(0.15*const.e*EpsBas*vo*vb*...%
   (const.Ag)^2*const.phi_o*const.phi_i);
  Qo=A/(B*dm*(1-nu)-1);

  % Calculation of the new Maxwellian electrons temperature given
  % the rate factor
  Tm=Tm_from_Qo(Qo, const);

  dif=Tmi-Tm; % We update the control parameter of the iteration
  Tmi=Tmi-dif*0.1; % We update the temperature for next iteration
end

end

% We call the function that provides the baseline plasma ion
% energy cost as a function of the bulk plasma electrons
% temperature and the operating conditions (Vd, Vc)
function [EpsBas, PtoM] = calc_EpsBas(Tm, const, Vd, Vc)
Annex D

\[ \varepsilon M = 2 \left( \frac{2}{3} T_m \right) + \text{const}. V_a; \]
% Average energy of Maxwellian electrons leaving the plasma at the anode (eV)

\[ v_P = \left( \frac{2 \text{const}. e (V_d - V_c)}{\text{const}. m e} \right)^{0.5}; \]
% Primary electron velocity (m/s)

\[ \text{Energia}_P = 0.5 m e v_P^2 / e; \]
% Primary electron energy (eV)

\[ v_e = \left( \frac{8 \text{const}. e T_m}{\pi} / \text{const}. m e \right)^{0.5}; \]
% Maxwellian electrons velocity in bulk plasma: m(kg), \( k b(J/K) \), \( k t(K/eV) \), \( k b \times k t = e) \), T(eV), v(m/s)

\[
\begin{align*}
\sigma_{\text{exP}}, & \sigma_{\text{ionP}} \} = \text{cross_sections}(V_d, V_c); \\
\sigma_{\text{inelastic}} &= \sigma_{\text{exP}} + \sigma_{\text{ionP}}; \quad \% \text{Total inelastic collision cross section at the primary electron energy (m}^2) \\
\text{%Ionisation and Excitation Reaction Rates for Xenon in Maxwellian Plasmas} \\
\text{if } T_m < 5 \\
\quad & \sigma_{\text{ion}} = 10^{-20} \times (3.97 + 0.643 \times T_m - 0.0368 \times (T_m^2)) \times e^{-12.127 / T_m}; \quad \% < 5 \text{eV} \\
\text{else} \\
\quad & \sigma_{\text{ion}} = 10^{-20} \times (1.031 \times 10^{-4} \times T_m^2 + 6.386 \times e^{-12.127 / T_m}); \quad \% \geq 5 \text{eV} \\
\text{end} \\
\sigma_{\text{ex}} &= 1.93 \times 10^{-19} \times e^{-11.6 / T_m} \times T_m^{-0.5}; \\
\% \text{Resolution of the system of equations by iteration} \\
\text{PtoM} &= 1; \quad \% \text{Initialisation of np/nm ratio with an aleatory value} \\
\text{dif} &= 1; \quad \% \text{Initialisation of the control parameter} \\
\text{while } \text{abs} (\text{dif}) > 10^{-6} \\
\quad & \text{EpsBasi} = \left( \text{const}. U_{\text{ion}} + \varepsilon M + \left( (\text{PtoM} \times \sigma_{\text{exP}} \times v_P + \sigma_{\text{ex}} \times v_e) \times \text{const}. U_{\text{ex}} \right) / \left( (\text{PtoM} \times \sigma_{\text{ionP}} \times v_P + \sigma_{\text{ion}} \times v_e) \right) / \left( 1 - (V_c + \varepsilon M) / V_d \right) \right); \\
\quad & \text{PtoMf} = \sigma_{\text{ion}} \times v_e / \sigma_{\text{ionP}} \times v_P / (V_d / \text{EpsBasi} \times \sigma_{\text{inelastic}} / \sigma_{\text{ionP}} - 1); \\
\quad & \text{dif} = (\text{PtoM} - \text{PtoMf}); \\
\quad & \text{PtoM} = \text{PtoM} - \text{dif}; \quad \% \text{We assign the new value of np/nm for next iteration} \\
\text{end} \\
\% \text{When solution has converged we return the values of np/nm and EpsBasi} \\
\text{PtoM} &= \text{PtoMf}; \\
\text{EpsBasi} &= \text{EpsBasi}; \\
\% \text{Solver to obtain electron temperature from ionisation reaction rate by means of an iterating process}
Annex D

%Qo: valid from 1e-45 (Tm=0.17eV) until 2.5e-13 (Tm=60eV)

function [Tm] = Tm_from_Qo(Qo,const)

if (Qo<8.2918e-15)
    Tmi=4;
    dif=1;
    while abs(dif)>1e-5
        Tm=Tmi;
        Tmf=((Qo/(1e-20.*((3.97+0.643.*Tm-0.0368.*(Tm.^2))... 
          .*exp(1).^(-12.127./Tm))))^2)*pi()*const.me/8/const.e;
        dif=Tmf-Tmi;
        Tmi=Tmi+dif*0.01;
    end
else
    Tmi=4;
    dif=1;
    while abs(dif)>1e-5
        Tm=Tmi;
        Tmf=((Qo/(1e-20.*(-(1.031e-4).*Tm.^2+6.386.*exp(1)... 
          .^(-12.127./Tm))))^2)*pi()*const.me/8/const.e;
        dif=Tmf-Tmi;
        Tmi=Tmi+dif*0.01;
    end
end

Tm=Tmf;
end
D.2. MATLAB code for optimised performance parameters obtainment

This code makes use of the same functions already presented in Annex D.1.

```matlab
function [ nuopt,uopt,EpsBopt,Ml_Miopt] = nuopt_uopt_dV_t(dV,t,dm)

%-----------------CONSTANTS----------------------
const.e=1.60217662e-19;   % Electron charge (C)
const.me=9.10938356e-31;  % Electron mass (kg)
const.kb=1.3806505e-23;   % Boltzman constant (J/K) //
const.kt=11604.505;       % Constant to convert from eV to k
const.Uion=12.12984;      % Xenon first ionisation energy (eV)
const.Ui=8.315;           % Xenon first excitation energy (eV)
const.Tw=400;             % Wall temperature (K)
const.M=2.1802225e-25;    % Mass of a Xenon atom (kg)
const.d=0.12;             % Diameter of ion thruster (m)
const.Ag=pi()*const.d^2/4; % Area of grids through which the ion beam...
const.V=1.5825e-4;        % Volume m^3
const.phi_o=0.16;         % SHAG
const.phi_i=0.8;          % Transparency to ions
const.le=2.5;             % Primary electron containment length (m)
const.Uex=0.5*(const.Ui+const.Uion); % Lumped excitation energy (V)
const.Va=2;               % It is the difference between plasma
                          % potential and anode potential (V)

%------------------------------------------
Vd=40;                     % Discharge voltage (V)
Vc=10;                     % Plasma potential from which the e- are supplied.
                          % Hollow cathode potential (V)
fc=0.1;                    % Fraction of ion current to cathode surfaces
fb=0.5;                    % Extracted ion fraction
Mpo=200;                   % Total mass of propellant for a given mission (kg)
dmp=dm*const.M/const.e;    % We define the propellant flow rate in kg/s from
                          % that given in A eq.
                          % that flow rate is constant.
alpha=0.03;                % Power plant specific mass (kg/W). Obtained from
                          % typical values of real thrusters

% Function that returns the value of Co for given Vd and Vc and constants
% dependent of the geometry of the thruster and propellant employed.
Co=8;

nu=0.1:0.001:0.96;        % propellant utilisation efficiency
% Calculation of the beam ion cost as a function of nu for an specific
% configuration.
```

Annex D
% For each value of alpha we find the double optimized value of Ml/Mi and
% the corresponding nu, which will be plotted

EpsB = calc_EpsB(const,Vd,Vc,nu,dm,fb,fc,Co);
a=dV./nu;
b=alpha.*nu/2/t;
c=2*const.e/const.M.*EpsB;

for i=1:length(nu)
    u(i)=solve_u(a(i),b(i),c(i));
    Ml_Mi(i)=exp(-dV./nu(i)/u(i))-alpha*u(i)^2*nu(i)/(2*t)*...
                  (1+2*const.e*EpsB(i)/(const.M*u(i)^2))*(1-exp(-dV./nu(i)/u(i)));
end

% Instead of derivate the expression, we look for the maximums
pos=find(Ml_Mi==max(Ml_Mi));
Ml_Miopt=Ml_Mi(pos);
EpsBopt=EpsB(pos);
nuopt=nu(pos);
uopt=u(pos);
end

% Solver which returns the optimum exhaust velocity
function [u] = solve_u(a,b,c)

ui=20000;
dif=1;

while abs(dif)>1e-5;
    u=(a/2/ui^2/b*(1+b*(c+ui^2))/(exp(a/ui)-1));
    dif=u-ui;
    ui=ui+dif*0.1;
end
end
D.3. MATLAB code for solving orbital motion
differential equations

% Function that returns the necessary time to perform the specific
% manoeuver given the thrust (N), initial mass (kg) of the spacecraft and
% the mass flux rate of the propulsion system (kg/s).
function [time] = time_T(T,Mi,dm)

dt=100;       % We define a time step
tf=1e8;      % Total time of integration
t=0:dt:tf;   % We define the time vector
s=zeros(length(t),3);  % We define a position vector
v=s;

%----Initial conditions that define the initial orbit---------

s(1,:)=[6671,0,0];  %Initial position conditions
v(1,:)=[0,7.74345,0];  %Initial velocity conditions

%--------Constants

G=6.67e-20;         % Gravitational constant
M=6e24;              % Earth mass
mu=G*M;             % Gravitational parameter of the Earth
el=1.60217662e-19;  % Electron charge
ma=2.1802225e-25;   % Atomic mass of xenon

time=tf;
circ=0;
i=1;

% Loop where we solve the differential equations at each instant t through
% the Runge Kutta 4th order method

while circ==0 & i<length(t)
  % We define de acceleration provided to the spacecraft from the thrust
  % and the spacecraft mass at each moment
  aT=(T/(Mi-dm*ma*t(i)/(el)))/1000;

  % Position  and velocity vectors determination at each instant by means
  % of the runge kutta method.
  [s(i+1,:),v(i+1,:)]=rk4(s(i,:),v(i,:),t(i),dt,aT);

  % Obtention of the orbital elements from the position and velocity
  % vectors
  [a,e(i+1),in,o,w,vt] = vecele(mu,s(i+1,1),s(i+1,2),s(i+1,3),...
  v(i+1,1),v(i+1,2),v(i+1,3));

  % We stop solving the differential equations when the following conditions
  % are given. Only one condition can be activated at the same time

  if norm(s(i+1,:))>42336
    circ=1;
    time=t(i);
    months=time/24/30/3600;
  elseif norm(e(i+1))>0.05
    circ=1;
    time=t(i);
    months=time/24/30/3600;
  end

end
% If we are analysing the transfer from SSTO, we check if the orbit is
% circular by means of the excentricity

if e(i+1)<0.01
    circ=1;
    time=t(i);
    months=time/24/30/3600;
end
i=i+1;
end

% We obtain the new position and velocity vectors applying runge kutta
% method
function [s,V] = rk4(r,v,t,dt,T)
s=zeros(1,3);
V=s;

    dx1=dt*v;
    R=norm(r(1:3));
    for i=1:3
        dv1(i)=dt*F(t,r(i),v(i),R,i,r,v,T);
    end

    dx2=dt*(v+dv1/2);
    d=r+dx1/2;
    R=norm(d);
    for i=1:3
        dv2(i)=dt*F(t+dt/2,r(i)+dx1(i)/2,v(i)+dv1(i)/2,R,i,d,v,T);
    end

    dx3=dt*(v+dv2/2);
    d=(r+dx2/2);
    R=norm(d);
    for i=1:3
        dv3(i)=dt*F(t+dt/2,r(i)+dx2(i)/2,v(i)+dv2(i)/2,R,i,d,v,T);
    end

    dx4=dt*(v+dv3);
    d=(r+dx3);
    R=norm(d);
    for i=1:3
        dv4(i)=dt*F(t+dt,r(i)+dx3(i),v(i)+dv3(i),R,i,d,v,T);
    end

    dx=(dx1+2*dx2+2*dx3+dx4)/6;
    dv=(dv1+2*dv2+2*dv3+dv4)/6;
Annex D

s=r+dx;
V=v+dv;
end

function [ res ] =F(t,x,v,R,j,d,V,T)
G=6.67e-20;
M=6e24;%kg
mu=G*M;

vu=V/norm(V);  % Unit vector in the direction of the velocity
% If transfer is spiral climb, thrust goes on the velocity vector direction
if j==1
    res=-mu*x/(R^3)+T*vu(j);
end
if j==2
    res=-mu*x/(R^3)+T*vu(j);
end
if j==3
    res=-mu*x/(R^3)+T*vu(j);
end
% If transfer is from SSTO, there is only thrust in j=2 direction (y)

% if j==1
%    res=-mu*x/(R^3);
% end
% if j==2
%    res=-mu*x/(R^3)-T;
% end
% if j==3
%    res=-mu*x/(R^3);
% end
end
Annex D

D.4. Mission analysis MATLAB code

This code makes use of functions presented in Annexes D.1, D.2 and D.3 in order to compute the optimum thruster’s performance parameters for a specific mission as a function of the propellant mass flow rate.

```matlab
% Script that computes the optimum thruster’s performance parameters for a specific mission as a function of the propellant flow rate
clear
clc

dm=0.8:0.4:1.6;
Mi=2205;
e=1.60217662e-19;
ma=2.1802225e-25;

for i=1:length(dm)
    dif=1;
    nuopt=0.9;
    uopt=30000;
    while dif>0.01
        uant=uopt;
        T=4*dm(i)*nuopt*uopt*ma/e;
        tic
        [time] = time_T(T,Mi,4*dm(i));
        toc
        dV=log(Mi/(Mi-4*dm(i)*ma/e*time))*uopt*nuopt;
        tic
        [nuopt,uopt,EpsBopt,Ml_Miopt] = nuopt_uopt_dV_t(dV,time, dm(i));
        toc
        dif=abs(uant-uopt)/uopt
    end
    nuopti(i)=nuopt;
    uopti(i)=uopt;
    Ti(i)=T;
    dVi(i)=dV;
    EpsBopti(i)=EpsBopt;
    Ml_Miopti(i)=Ml_Miopt;
    timei(i)=time;
end
```
Annex D

Bibliography
