Politecnico di Milano
Department of *Aerospace Science and Technology*
Aerospace Propulsion Course (A.Y. 2017/18)

Final Degree Project

Falcon 9: 2nd Stage Redesign using LOX/LH_2 Bipropellant

Maria Francesca Palermo (849330)
mariafrancesca.palermo@mail.polimi.it

Massimo Piazza (847535)
massimo.piazza@mail.polimi.it

Michele Rampini (844471)
michele.rampini@mail.polimi.it

Last update: September 2, 2019 at 16:10

Abstract

Falcon 9 is a two-stage partially reusable rocket launcher, designed and built by *SpaceX*. Except for a few high-energy launches, the rocket's first stage is completely recovered and reused, while the 2nd stage is always expended.
The redesign of a 2nd stage using LOX/LH_2, instead of the original LOX/RP-1 combination, will be investigated: cryogenic propellants do not allow reusability, but at the same time they provide the highest specific impulse obtainable with conventional liquid propellants, which is a major determining factor for upper stages performance.

Contents

Nomenclature & Acronyms

Nondimensional Quantities

α	Learning rate	$[-]$
γ	Ratio of specific heats $\left(\frac{c_p}{c_v}\right)$	$[-]$
λ	Regularization parameter	$[-]$
ε	Nozzle expansion area ratio $\left(\frac{A_e}{A_t}\right)$	$[-]$
C_D	Discharge coefficient	$[-]$
C_F	Thrust coefficient	$[-]$
L_f	Nozzle fractional length	$[-]$
M	Mach number	$[-]$
r	Oxidizer to fuel ratio $\left(\frac{\dot{m}_{\text{ox}}}{\dot{m}_f}\right)$	$[-]$

Physical Quantities

$\mathcal{F}$	Thrust	[N]
$\mathfrak{M}$	Mean molar weight	[g/mol]
$\dot{m}$	Mass flow rate	[kg/s]
ρ	Density	$[\text{kg/m}^3]$
a	Speed of sound	[m/s]
c^*	Characteristic velocity	[m/s]
c_p	Isobaric specific heat	$[\text{J/(kg} \cdot \text{K)}]$
c_v	Isochoric specific heat	$[\text{J/(kg} \cdot \text{K)}]$
D	Diameter	[m]
h	Height	[m]
I_{sp}	Gravimetric specific impulse	[s]
I_t	Total impulse	$[\text{N} \cdot \text{s}]$
I_v	Volumetric specific impulse	$[\text{kg} \cdot \text{s/m}^3]$
L	Length	[m]

m	Mass	[kg]
p	Pressure	[kPa]
Q	Volumetric mass flow rate	[m^3/s]
R	Specific gas constant $\left(\frac{R_u}{\mathfrak{M}}\right)$	[$J/(kg \cdot K)$]
R_u	Universal gas constant	[$J/(mol \cdot K)$]
T	Temperature	[K]
t_b	Burning time	[s]
V	Volume	[m^3]
v	Velocity	[m/s]
W	Weight	[N]
z	Altitude	[m]
z^*	Optimum expansion altitude	[m]
PL	Payload	[kg]

Subscripts/Superscripts

a	Atmospheric
b	Burning
c	Combustion chamber
cyl	Cylindrical
e	Exit/exhaust
ell	Ellipsoidal
f	Fuel
gg	Gas-generator
inj	Injection
int	Internal
opt	Optimum expansion
ox	Oxidizer
p	Propellant
pred	Prediction

res	Resulting
sl	Sea level
T	Total quantity
t	Throat
vac	Vacuum

Acronyms

BFGS	Broyden-Fletcher-Goldfarb-Shanno
CEA	Chemical Equilibrium for Applications
DLR	Deutsches zentrum für Luft und Raumfahrt
FT	Full Thrust version
GD	Gradient Descent
GTO	Geosynchronous Transfer Orbit
I/S	InterStage
L-BFGS	Limited-memory BFGS
LEO	Low Earth Orbit
LH2SS	Liquid Hydrogen Second Stage
LOX	Liquid OXygen
LRE	Liquid Rocket Engine
LSQ	Least SQuares
LV	Launch Vehicle
M1DV	Merlin 1D Vacuum
M1D	Merlin 1D
MECO	Main Engine Cut-Off
NE	Normal Equation
O/F	Oxidizer to Fuel ratio
PAF	Payload Adapter Fitting
RP1SS	RP-1 Second Stage
SECO	Sustainer Engine Cut-Off

VTOL	Vertical Take-Off and Landing
w.r.t.	With Respect To

Chapter 1

Introduction

The *Falcon 9* launch vehicle has been designed with reusability in mind, hence minimizing costs, complexity and refurbishment time, leading to a configuration employing semi-cryogenic propellants (LOX/RP-1) on both stages. Such a choice reflects *SpaceX*'s philosophy, aiming at the highest possible cost reduction, yet guaranteeing good performance.
The high-energy liquid propellants LOX/LH_2 yield a specific impulse approaching the theoretical maximum of the fluorine/hydrogen combination: the extreme toxicity of the latter makes its use unfeasible, while the combustion product of the former is just harmless water vapor. Nevertheless, the use of hydrogen is also characterized by some drawbacks, the most critical being its very low density (requiring cryogenic temperatures) and the high flame temperature, which lead to bulkier stages and jeopardize reusability due to thermal stress.
The aforementioned disadvantages are practically nonexistent for upper stages, where a high I_{sp} plays a very important role. For the case of the *Falcon 9* rocket, whose second stage is not recovered, the use of hydrogen would yield a substantial net performance gain, which in turn could also allow the recovery of the first stage with higher payloads.
For such reasons, the redesign of a LOX/LH_2 *Falcon 9* 2nd stage will be investigated.

The starting point for the redesign will be the estimated performance of the actual 2nd stage, determined from the whole amount of data available about the rocket. Such data are provided both by official and non-official sources[(1)] and are grouped in Tab. 1.1.

[(1)] which are all listed in the bibliography at the end of this report

Table 1.1: *Falcon 9 Data (v1.2 "Full Thrust")*

	1st Stage	**2nd Stage**	**Fairing**
m_{wet}	$\sim$ 433.2 Mg	$\sim$ 112.5 Mg	$\sim$ 2 Mg
m_{dry}	$\sim$ 22.2 Mg	$\sim$ 4 Mg	$\sim$ 2 Mg
$m_{\text{dry}}^{\text{eng}}$	470 kg	490 kg	-
L	40.67 m	15.95 m (including I/S)	13.38 m
D	3.66 m	3.66 m	5.2 m
$\mathcal{F}_{\text{sl}}$	7607 kN	-	
$\mathcal{F}_{\text{vac}}$	8227 kN	934 kN	
$I_{\text{sp}}^{\text{sl}}$	282 s	-	
$I_{\text{sp}}^{\text{vac}}$	311 s	348 s	
t_b	162 s	397 s	
ε	16	165	
m_{ox}	287 430 kg	75 200 kg	
m_f	123 570 kg	32 300 kg	
V_{ox}	234.7 m^3	61.4 m^3	
V_f	143.9 m^3	37.6 m^3	
r	2.33	2.33	
p_c	110 bar	110 bar	
Engine	Merlin 1D+ (9 engines)	Merlin 1D Vac+ (1 engine)	
Propellants	LOX/RP-1	LOX/RP-1	
$PL_{\text{LEO}}^{\text{max}}$	22 800 kg[(2)]		
$PL_{\text{GTO}}^{\text{max}}$	8300 kg		
ρ_{ox}	1.22467 g/cm^3		
ρ_f	0.85872 g/cm^3		
T_{ox}	-207 °C		
T_f	-7 °C		

[(2)] restricted to 10 886 kg due to PAF structural limit

Chapter 2

Problem Analysis

2.1 Second Stage Sizing

The redesign of the 2nd stage will be based on two main **assumptions**:

- LH2SS and RP1SS have the same expansion pressure p_e (i.e. same optimum expansion altitude)
- LH2SS and RP1SS have the same vacuum thrust $\mathcal{F}_{\text{vac}}$

In this section, the following procedure will be carried out.

1. Thermochemical characterization of the LOX/RP-1 propellant combination, in order to estimate the expansion pressure of the actual engine.
2. Thermochemical characterization of the LOX/LH_2 propellant combination. From the p_e value, we can now obtain the expansion area ratio of the redesigned engine.
3. Performance calculation for LH2SS.
4. Bell nozzle sizing, using Rao's parabolic approximation.
5. Tanks sizing.

Throughout the whole analysis, the Fortran code `NASA CEA` has to be extensively used. The output it provides is a simple text file, which may turn out to be difficult to visualize. For this reason, the software `MatlabCEA` has been developed by the team: by combining Matlab, Python and Unix scripts it is possible to dynamically plot the CEA output, completely within the Matlab environment (i.e. executing a simple Matlab script like C.14).

Code Workflow

1. Matlab script assigns the CEA input `<fileName>.inp`
2. Matlab script executes the Python code `ceaCall.py` (C.15) using a *Unix* command. The result is:
   (a) CEA execution using the specified input file
   (b) parsing of `<fileName>.plt` (i.e. the CEA output)

(c) `<fileName>.plt` is converted to a `.csv` file

3. Matlab script imports `<fileName>.csv`, creating a data matrix

4. Matlab script plots every imported output data for each of the 3 nozzle regions (Combustion Chamber; Throat; Exhaust)

2.1.1 Thermochemical Properties

Before running `NASA CEA` for determining the molar fractions of the LOX/RP-1 combustion products, the fuel enthalpy has to be estimated. The thermochemical properties of a substance at specified conditions can be determined using the ***Shomate Equation*** for enthalpy and isobaric specific heat:

$$H = RT \cdot \left(a_1 + \frac{a_2}{2} \cdot T + \frac{a_3}{3} \cdot T^2 + \frac{a_4}{4} \cdot T^3 + \frac{a_5}{5} \cdot T^4 + \frac{a_6}{T} \right) \tag{2.1}$$

$$c_p = a_1 + a_2 \cdot T + a_3 \cdot T^2 + a_4 \cdot T^3 + \frac{a_5}{T^2} \tag{2.2}$$

Where $\{a_1, a_2, a_3, a_4, a_5, a_6\}$ is a set of LSQ coefficients obtained from experimental data, which are used to accurately represent the thermodynamic properties of elements and compounds, through a 4th order polynomial. Each set is specified for a certain substance and a range of temperatures. Such coefficients are extensively tabulated in [Cha98].

The thermochemical characterization of RP1SS was made assuming a one-formula surrogate fuel ($C_{12}H_{24}$), using the fitting coefficients taken from [Wan96], which are given in A.1.
Using (2.1) evaluated at $T_f = 266$ K[(1)] yields:

$$H_{\text{RP1}} = -2.253736 \cdot 10^6 \text{ J/mol}$$

Which is used as a CEA input in C.17.

The constant γ (ratio of specific heats) has been calculated for every combustion product of the reaction between fuel (RP-1 or H_2) and oxidizer (O_2):

RP1SS: $\text{RP-1} + O_2 \longrightarrow CO + CO_2 + H_2O + H_2$
LH2SS: $H_2 + O_2 \longrightarrow H_2O + OH + H_2 + O_2 + O + H$

$$\begin{cases} \gamma = \frac{c_p}{c_v} \\ R = c_p - c_v \end{cases} \implies \gamma = \frac{c_p}{c_p - R}$$

The CEA output file provides the molar fraction of combustion products (Tab. 2.1, 2.2), which can be used as weighting coefficients for obtaining the average c_p value of the fuel immediately downstream of the combustion chamber:

[(1)] Falcon 9 FT has the capability of *sub-cooling* the RP-1 fuel

Table 2.1: *Molar fraction of RP1SS combustion products*

	CO	**CO_2**	**H_2O**	**H_2**
Molar Fraction	0.40432	0.30694	0.27644	0.01221
γ	1.29268	1.15687	1.18388	1.30476

Table 2.2: *Molar fraction of LH2SS combustion products*

	H_2O	**OH**	**H_2**	**O_2**	**O**	**H**
Molar Fraction	0.91651	0.03899	0.03617	0.00444	0.00202	0.00182
γ	1.17029	1.28558	1.27809	1.25620	1.65071	1.66668

From a simple weighted average, it follows that:

$$\gamma_{\text{RP1}} = 1.221067$$

$$\gamma_{\text{H}_2} = 1.180937$$

One of the first value needed to size LH2SS is the **expansion pressure**: the **optimum expansion altitude** is assumed to be the same both for the original RP1SS and for the LH2SS.
The area expansion ratio ε can be expressed as:

$$\frac{1}{\varepsilon} = \left(\frac{\gamma+1}{2}\right)^{\frac{1}{\gamma-1}} \left(\frac{p_e}{p_c}\right)^{\frac{1}{\gamma}} \sqrt{\frac{\gamma+1}{\gamma-1}\left[1-\left(\frac{p_e}{p_c}\right)^{\frac{\gamma-1}{\gamma}}\right]} \tag{2.3}$$

The expansion pressure of RP1SS can be obtained by numerically solving (2.3) for p_e, which yields: $p_e = 3471.7$ Pa.

Assuming for LH2SS the same value of p_e, we can use again (2.3) in order to determine the expansion area ratio of the redesigned stage, obtaining:

$$\varepsilon = 195.28$$

2.1.2 Performance Calculation

For LH2SS a **mixture ratio** $r = 6.0$ was selected, according to the results obtained from `MatlabCEA`. The CEA input file C.18 was used, producing the following results:

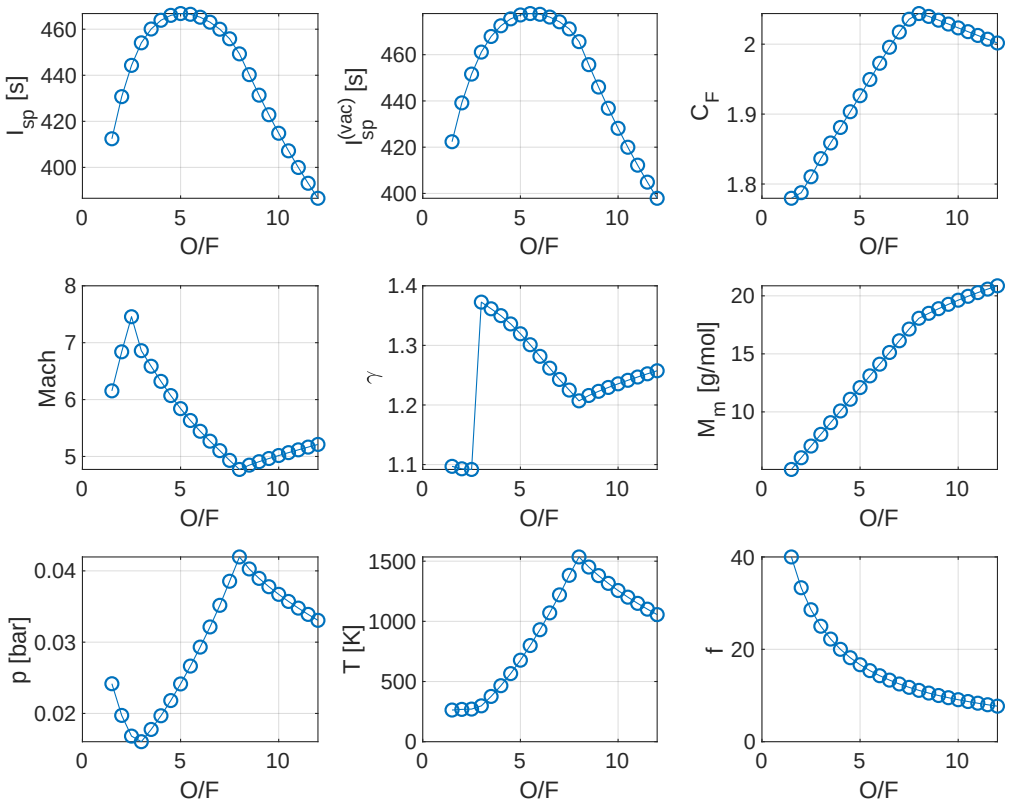
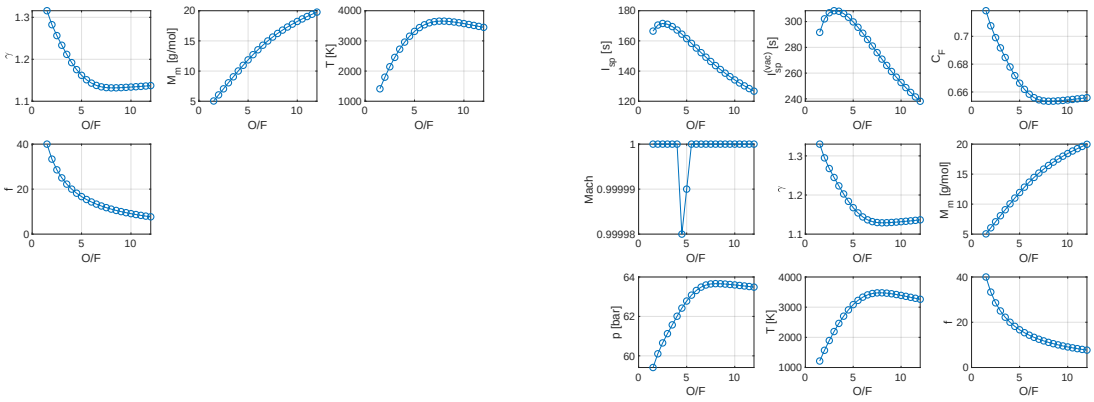


Figure 2.1: *LOX/LH_2 performance variation with O/F (Exhaust)*



(a) *Combustion Chamber* (b) *Throat*

Figure 2.2: *LOX/LH_2 performance variation with O/F (C.C. and T.)*

It can be inferred that the optimum-I_{sp} mixture ratio is approximately $4.5 \div 5$. Nevertheless, propellant density plays a very important role in the choice of the O/F, given an oxidizer density which is 16 times that of the fuel. In order to obtain tanks with a reasonable size, it is very common to select O/F equal to 6.0, sacrificing some engine performance but significantly reducing the overall stage weight.

In order to evaluate several performance parameters it is convenient to define the Vandenkerckhove function:

$$\Gamma = \sqrt{\gamma_{\mathrm{H}_2}\left(\frac{2}{\gamma_{\mathrm{H}_2}+1}\right)^{\frac{\gamma_{\mathrm{H}_2}+1}{\gamma_{\mathrm{H}_2}-1}}} = 0.6448 \tag{2.4}$$

An efficiency index of the thermochemical energy conversion process inside the combustion chamber is given by the **characteristic velocity**, defined as

$$c^* = \frac{p_c A_t}{\dot{m}_p} \tag{2.5}$$

which can also be re-expressed as a function of Γ:

$$c^* = \frac{\sqrt{RT_c}}{\Gamma} = 2285.5 \text{ m/s} \tag{2.6}$$

The **exhaust velocity** can be written as the product between two terms: the nozzle expansion efficiency (ν_n) and the maximum theoretical velocity (v_{lim}) that would be obtained at the exhaust of an infinite nozzle operating in vacuum, namely

$$v_e = \underbrace{\sqrt{1-\left(\frac{p_e}{p_c}\right)^{\frac{\gamma_{\mathrm{H}_2}-1}{\gamma_{\mathrm{H}_2}}}}}_{\nu_n} \cdot \underbrace{\sqrt{\frac{2\gamma_{\mathrm{H}_2}}{\gamma_{\mathrm{H}_2}-1}R\cdot T_c}}_{v_{\text{lim}}} = 4483.8 \text{ m/s}$$

The exhaust Mach number can be determined from the relation between total and static pressures, where the former is $\approx p_c$[(2)], while the latter is p_e:

$$\frac{p_c}{p_e} = \left[1+\frac{\gamma_{\mathrm{H}_2}-1}{2}M^2\right]^{\frac{\gamma_{\mathrm{H}_2}}{\gamma_{\mathrm{H}_2}-1}} \Longrightarrow M_e = \sqrt{\frac{2}{\gamma_{\mathrm{H}_2}-1}\left[\left(\frac{p_c}{p_e}\right)^{\frac{\gamma_{\mathrm{H}_2}-1}{\gamma_{\mathrm{H}_2}}}-1\right]} = 5.192$$

From the definition of Mach number $a_e = \dfrac{v_e}{M_e} = 863.6$ m/s
Furthermore, the speed of sound in an ideal gas is given by

$$a = \sqrt{RT\gamma} \tag{2.7}$$

hence, we can obtain the **exhaust temperature** from (2.7):

$$T_e = \frac{a_e^2}{\gamma_{\mathrm{H}_2} R} = 1027.7 \text{ K}$$

Another important performance parameter is the **specific impulse**, which for a rocket is defined as

$$I_{\text{sp}} = \frac{\mathcal{F}}{\dot{m}_p g_0} \tag{2.8}$$

[(2)] assuming a neglectable velocity in the combustion chamber

Let us now consider the rocket **thrust equation**:

$$\mathcal{F} = \dot{m}_p v_e + (p_e - p_a)A_e \tag{2.9}$$

At optimum expansion conditions $p_e = p_a$, thus $\mathcal{F} = \dot{m}_p v_e \quad \longrightarrow \quad \dfrac{\mathcal{F}}{\dot{m}_p} = v_e$

so that the specific impulse reduces to:

$$I_{\text{sp}}^{\text{opt}} = \frac{v_e}{g_0} = 457.2 \text{ s}$$

In order to account for the inevitable **losses** that are involved in any real process, we may assume a nozzle isentropic efficiency of $\eta_n^{\text{is}} \overset{\text{HP}}{=} 0.97 \xrightarrow{(\,)\cdot\eta_n^{\text{is}}} I_{\text{sp}}^{\text{opt}} = 443.5$ s

The **thrust coefficient** is a measure of thrust increase due to the divergent region, normalized w.r.t. the static thrust at throat section

$$C_F = \frac{\mathcal{F}}{p_c A_t} \tag{2.10}$$

which can also be re-expressed as a function of Γ:

$$C_F = \Gamma \cdot \nu_n \cdot \sqrt{\frac{2\gamma_{\text{H}_2}}{\gamma_{\text{H}_2} - 1}} + \left(\frac{p_e - p_a}{p_c}\right)\cdot\varepsilon \tag{2.11}$$

At optimum expansion conditions the last term drops out ($p_e = p_a$):

$$C_F^{\text{opt}} = \Gamma \cdot \nu_n \cdot \sqrt{\frac{2\gamma_{\text{H}_2}}{\gamma_{\text{H}_2} - 1}} \quad \xrightarrow{(\,)\cdot\eta_n^{\text{is}}} \quad \text{actual value:} \quad C_F^{\text{opt}} = 1.903$$

while for vacuum expansion ($p_a = 0$):

$$C_F^{\text{vac}} = \Gamma \cdot \nu_n \cdot \sqrt{\frac{2\gamma_{\text{H}_2}}{\gamma_{\text{H}_2} - 1}} + \left(\frac{p_e}{p_c}\right)\cdot\varepsilon \quad \xrightarrow{(\,)\cdot\eta_n^{\text{is}}} \quad \text{actual value:} \quad C_F^{\text{vac}} = 1.963$$

Solving Eq. (2.10) for the throat area (vacuum condition) yields

$$A_t = \frac{\mathcal{F}_{\text{vac}}}{p_c \cdot C_F^{\text{vac}}} = 0.0433 \text{ m}^2 \quad \Longrightarrow \quad A_e = \varepsilon \cdot A_t = 8.4475 \text{ m}^2$$

so that the corresponding **diameters** are $\quad D_t = 0.2347$ m, $\quad D_e = 3.2796$ m

The **throat temperature** can be determined from the relation between total and static temperatures, where the former is $\approx T_c$[(3)], while the latter is T_t:

$$\frac{T_c}{T_t} = \left[1 + \frac{\gamma_{\text{H}_2} - 1}{2} M_t^2\right] \quad \xrightarrow{(M_t = 1)} \quad T_t = \frac{2}{\gamma_{\text{H}_2} + 1} T_c = 3240.8 \text{ K}$$

[(3)] assuming a neglectable velocity in the combustion chamber

$$\implies a_t = \sqrt{\gamma_{\mathrm{H}_2} R T_t} = 1533.6 \text{ m/s}$$

Solving Eq. (2.9) for the **mass flow rate** of propellant, through the combustion chamber:

$$\dot{m}_p = \frac{\mathcal{F}_{\text{vac}} - p_e A_e}{v_e} = 201.76 \ \frac{\text{kg}}{\text{s}}$$

However, for our design, we will consider a **gas-generator** cycle configuration[(4)], having to account for an additional mass flow rate through the gas-turbine. According to [Sut16], such value is typically in the range of $1.5 \div 7\%$ of total propellant flow, hence a reasonable assumption would be:

$$\dot{m} = 1.05 \cdot \dot{m}_p = 211.85 \ \frac{\text{kg}}{\text{s}}$$

In turn, the overall engine **specific impulse** becomes slightly lower than that of the thrust chamber:

$$I_{\text{sp}} = \eta_n^{\text{is}} \cdot \frac{\mathcal{F}}{\dot{m} g_0} = 436.1 \text{ s}$$

In addition, given the average propellant density

$$\overline{\rho} = \frac{\rho_{\text{ox}} V_{\text{ox}} + \rho_f V_f}{V_{\text{ox}} + V_f} = \frac{\cancel{\rho_{\text{ox}}} \frac{m_{\text{ox}}}{\cancel{\rho_{\text{ox}}}} + \cancel{\rho_f} \frac{m_f}{\cancel{\rho_f}}}{\frac{m_{\text{ox}}}{\rho_{\text{ox}}} + \frac{m_f}{\rho_f}} = \frac{(\dot{m}_{\text{ox}} + \dot{m}_f)}{\frac{\dot{m}_{\text{ox}} \rho_f + \dot{m}_f \rho_{\text{ox}}}{\rho_{\text{ox}} \rho_f}}$$

$$\xrightarrow{( ) \cdot \frac{\frac{1}{\dot{m}_f}}{\frac{1}{\dot{m}_f}}} \quad \overline{\rho} = \frac{(1+r)\rho_{\text{ox}} \rho_f}{\rho_{\text{ox}} + r \rho_f} = 367.96 \ \frac{\text{kg}}{\text{m}^3}$$

the volumetric specific impulse can be determined as well:

$$I_v = \overline{\rho} I_{\text{sp}} = 160\,463.58 \ \frac{\text{kg} \cdot \text{s}}{\text{m}^3} \tag{2.12}$$

Combustion Chamber

Assuming a combustion chamber Mach number of $M_c \approx 0.2$, it follows that:

$$a_c = \sqrt{\gamma_{\mathrm{H}_2} R T_c} = 1601.5 \ \frac{\text{m}}{\text{s}} \quad \longrightarrow \quad v_c = M_c a_c = 320.3 \ \frac{\text{m}}{\text{s}}$$

$$A_c = \frac{\dot{m}}{\rho_c v_c} \quad \implies \quad R_c = \sqrt{\frac{A_c}{\pi}} = 0.196 \text{ m}$$

The characteristic chamber length is defined as:

$$L^* = \frac{V_c}{A_t} \tag{2.13}$$

and its value is roughly the same for many different rocket engines using the same propellant combination.

(4) a small amount of propellant is burned in a gas-generator and expanded through a turbine, in order to power the engine's pumps. This configuration is typical of cryogenic/semi-cryogenic propellants and is also used in the original M1DV engine (RP-1)

Propellant Combination	Combustion Chamber Characteristic Length (L*), in.
Chlorine trifluoride/hydrazine-base fuel	20-35
Liquid fluorine/hydrazine	24-28
Liquid flourine/liquid hydrogen (GH_2 injection)	22-26
Liquid fluorine/liquid hydrogen (LH_2 injection)	25-30
Hydrogen peroxide/RP-1 (including catalyst bed)	60-70
Nitric acid/hydrazine-base fuel	30-35
Nitrogen tetroxide/hydrazine-base fuel	30-35
Liquid oxygen/ammonia	30-40
Liquid oxygen/liquid hydrogen (GH_2 injection)	22-28
Liquid oxygen/liquid hydrogen (LH_2 injection)	30-40
Liquid oxygen/RP-1	40-50

Figure 2.3: *Characteristic chamber lengths*

From [Huz92] (Fig. 2.3), it can be seen that a typical value for a LOX/LH_2 combination would be $L^* = 30 \div 40$ in. We will therefore assume $L^* = \frac{35 \cdot 2.54}{100} = 0.8890$ m, in order to determine the thrust **chamber length**:

$$V_c = L^* A_t = 0.0385 \text{ m}^3 \quad \Longrightarrow \quad L_c = \frac{V_c}{A_c} = 0.3188 \text{ m}$$

2.1.3 Nozzle Sizing

Bell-shaped nozzles are the most common configurations employed in liquid-propellant rocket engines. The contour of an **optimum-thrust** nozzle can be accurately determined by applying the so called *method of characteristics*:[(5)] this allows to balance the oblique expansion waves[(6)] with oblique compression waves[(7)], hence minimizing energy losses. Such a procedure is described in [Rao58] by G. V. R. Rao, who first suggested a very accurate **parabolic approximation** of the nozzle contour in [Rao61].

Rao's Assumptions

1. The contour immediately upstream of the throat is chosen as a circular arc with a radius of $1.5R_t$
2. The region immediately downstream of the throat is a circular arc with a radius of $0.382R_t$
3. The final divergent contour, from the inflection point up to the exit, is a parabola

The contour angle is maximum at the inflection point (θ_i), afterwards, the cross section increases at a diminishing rate and the contour angle reaches the minimum value θ_e at the exit. As the flow is turned back, divergence losses are reduced by aligning the flow with the axial direction.

Bell nozzles are typically specified in terms of equivalent 15° *conical* nozzles.

(5) after solving for the exit flow field required to deliver optimum thrust
(6) occurring in the region immediately downstream of the throat, up to the inflection point
(7) occurring from the inflection point up to the exit

Fig. 2.4 shows the initial and final angles[8] versus ε, for a given fractional length[9] L_f. The reduced length provided by a bell-shape also yields slight improvements in terms of mass ratio. However, for $L_f < 70\%$, the energy losses due to internal oblique shock waves can become significant, as shown by Fig. 2.5. We will therefore select an **80%** equivalent length.

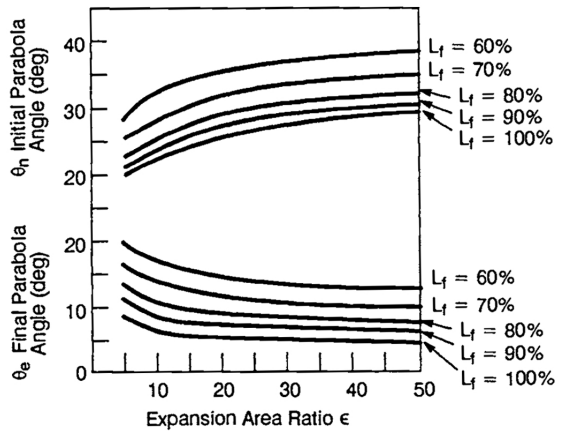


Figure 2.4: *Initial and final parabola angle (from [Huz92])*

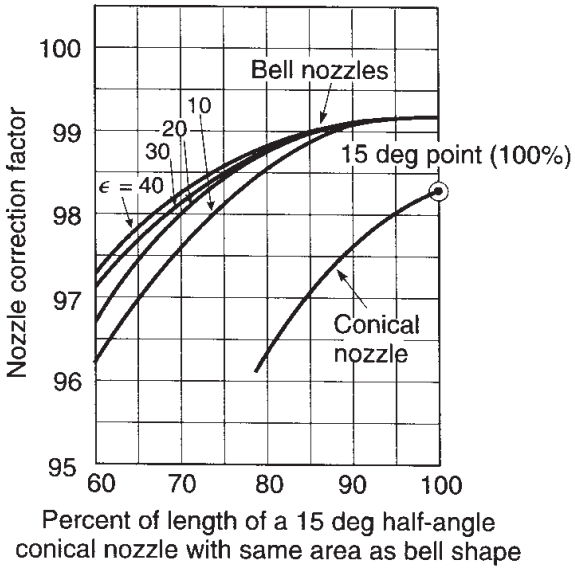


Figure 2.5: *Nozzle losses in terms of a correction factor (from [Sut16])*

Given the limited range of area ratios covered by the plot in Fig. 2.4, further data have to be extrapolated, in order to obtain the value of $\theta_i|_{\substack{L_f=80\% \\ \varepsilon=195}}$.
For this purpose, the following procedure has been adopted.

1. Vector data extraction from the `.pdf` diagram, using the software `WebPlotDigitizer` ([Ank18])
2. Data **fitting** (Fig. 2.6) using a 2nd degree polynomial for the logarithmic-transformed data
3. Extrapolation of the initial parabola angle through the fitted diagram: $\theta_i = 32.65°$

[8] of the tangent to the parabolic contour, w.r.t. the axial direction
[9] e.g. $L_f = 80\% \rightarrow$ bell-nozzle 20% shorter than a 15° conical nozzle with the same A_t and A_e

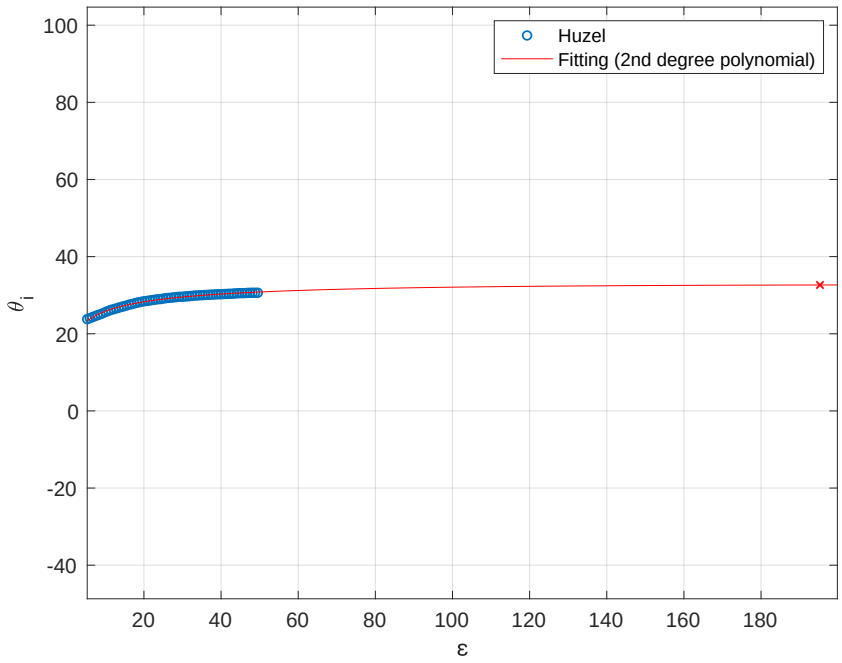


Figure 2.6

Now, assuming $L_f = 80\% \rightarrow L_{\text{bell}} = L_f \cdot \frac{R_t(\sqrt{\varepsilon}-1)}{\tan(15^\circ)} = 4.545$ m.

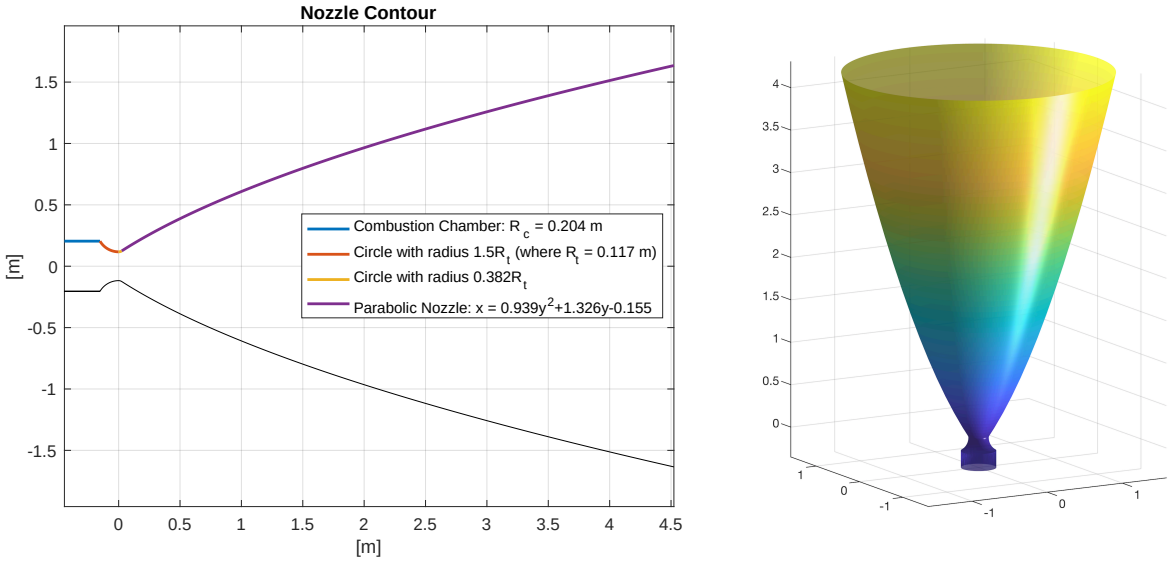
From the obtained values, we are able to calculate the analytical parabola expression by solving a linear system:

$$x = ay^2 + by + c$$

$$\begin{cases} \underbrace{\dfrac{\mathrm{d}x}{\mathrm{d}y}}_{1/\tan(\theta)} = 2ay + b \longrightarrow 2y_i a + b = \frac{1}{\tan(\theta_i)} \\ x_e = ay_e^2 + by_e + c \longrightarrow R_e^2 a + R_e b + c = L_{\text{bell}} \\ y_i^2 a + y_i b + c = x_i \end{cases}$$

$$\begin{bmatrix} 2y_i & 1 & 0 \\ R_e^2 & R_e & 1 \\ y_i^2 & y_i & 1 \end{bmatrix} \begin{Bmatrix} a \\ b \\ c \end{Bmatrix} = \begin{Bmatrix} \frac{1}{\tan\theta_i} \\ L_{\text{bell}} \\ x_i \end{Bmatrix} \quad \Longrightarrow \quad \begin{Bmatrix} a \\ b \\ c \end{Bmatrix} = \begin{Bmatrix} 0.939 \\ 1.326 \\ -0.155 \end{Bmatrix} \text{ m} \tag{2.14}$$

The results obtained in this section yield the following nozzle contour:



(a) *2D View* (b) *3D View*

Figure 2.7: *Nozzle Contour*

2.1.4 Tank Sizing

In order to determine the propellant mass of LH2SS we first need to set our performance goal. The **total impulse** is basically proportional to the total energy released by the propulsion system; thus, it can be inferred that two different stages characterized by the same value of I_t would result in very similar performances, in terms of launch-mission requirements. If we set a 5% performance increase:

$$\underbrace{I_t}_{Ft_b} = 1.05 \cdot I_t^{\text{RP1}} = (\underset{\downarrow}{W_p}\ I_{\text{sp}})^{\text{RP1}} \overset{\text{[Tab. 1.1]}}{=} 385\,223\,864\ \text{N}\cdot\text{s} \implies t_b = 412.4\ \text{s}$$
$$[9.807 \cdot (m_{\text{ox}} + m_f)^{\text{RP1}}]$$

from which

$$\dot{m}_{\text{ox}} = \frac{r}{1+r}\dot{m} = 181.59\ \frac{\text{kg}}{\text{s}} \qquad \dot{m}_f = \frac{1}{1+r}\dot{m} = 30.26\ \frac{\text{kg}}{\text{s}}$$
$$m_{\text{ox}} = 1.07 \cdot \dot{m}_{\text{ox}} t_b = 80\,137.8\ \text{kg} \qquad m_f = 1.07 \cdot \dot{m}_f t_b = 13\,356.3\ \text{kg}$$
$$V_{\text{ox}} = \frac{m_{\text{ox}}}{\rho_{\text{ox}}} = 65.44\ \text{m}^3 \qquad V_f = \frac{m_f}{\rho_f} = 188.65\ \text{m}^3$$

$$\left(\text{where} \qquad \rho_{\text{ox}} = 1224.67\ \frac{\text{kg}}{\text{m}^3}, \quad \rho_f = 70.81\ \frac{\text{kg}}{\text{m}^3}\right)$$

For obtaining a more accurate sizing, a correction coefficient of 1.07 was used in order to estimate the extra propellant that has to be loaded into the tanks, in order to account for:

- propellant *vaporization*, needed to pressurize its own tank
- propellant *evaporation* by means of a heat exchanger, during the time elapsing between the end of the "topping-off"[10] procedure and MECO

[10] just before launch, the evaporated propellant is vented overboard and continuously replaced with fresh cryogenic propellant

- *residual propellant*, that adheres to walls or remains trapped in valves
- start and stop *transients*
- other uncertainty factors, ...

LH2SS and RP1SS will share the same **tandem** tank configuration, with common bulkhead and ellipsoidal ends. Given the much lower fuel-density of the redesigned stage, in order to obtain a reasonable stage length, its external diameter is increased such as to match that of the fairing (5.2 m). We can therefore assume an internal diameter $D_{\text{int}} \approx 5$ m for the cylindrical portion of the tanks.
The tank internal volumes are actually slightly larger than V_{ox} and V_f, in order to account for the **ullage** volume[(11)], which is typically 2.5% of propellant volume:

$$V_{\text{ox}}^t = 1.025 \cdot V_{\text{ox}} = 67.07 \text{ m}^3 \qquad V_f^t = 1.025 \cdot V_f = 193.36 \text{ m}^3$$

The ellipsoidal-tank-end volume can be expressed as

$$V_{\text{ell}} = \frac{2\pi}{3} a^2 b$$

where a, b are respectively the major and minor half-diameters of the ellipsoidal ends. We will consider $b = 0.6 \cdot a$; in addition $a \equiv \frac{D_{\text{int}}}{2} = 2.5$ m.
The 2nd stage geometry is now univocally determined, hence we are able to calculate its length. Let L_f^{cyl} be the length of the cylindrical portion of the fuel tank, then we have

$$V_f^t = \frac{\pi D_{\text{int}}^2}{4} L_f^{\text{cyl}} + \underbrace{2 \cdot \left[\frac{2\pi}{3} \cdot \underbrace{a^2 b}_{\frac{D_{\text{int}}^2}{4} \cdot 0.3 D_{\text{int}}} \right]}_{0.1\pi D_{\text{int}}^3}$$

$$\implies \quad L_f^{\text{cyl}} = \frac{4V_f^t}{\pi D_{\text{int}}^2} - 0.4 D_{\text{int}} = 7.85 \text{ m}$$

$$L_f = L_f^{\text{cyl}} + \underbrace{2b}_{0.6 D_{\text{int}}} = 10.85 \text{ m}$$

$$V_{\text{ox}}^t = \pi \frac{D_{\text{int}}^2}{4} L_{\text{ox}} \quad \implies \quad L_{\text{ox}} = \frac{4V_{\text{ox}}^t}{\pi D_{\text{int}}^2} = 3.42 \text{ m}$$

The overall tank size is therefore $L_{\text{tanks}} = L_f + L_{\text{ox}} = 14.26$ m.

From graphical analysis of images of the Falcon 9 FT vehicle, it was estimated that the length of the remaining portion of RP1SS, which encases the M1DV engine (i.e. the I/S), should be somewhere around 6.8 m. Assuming a similar size of the injection system and given our longer divergent nozzle (4.52 m vs. 3.2 m), it can be estimated that the I/S length of LH2SS will be ~ 1.32 m longer, that is:

$$L_{\text{I/S}} = 8.12 \text{ m}$$

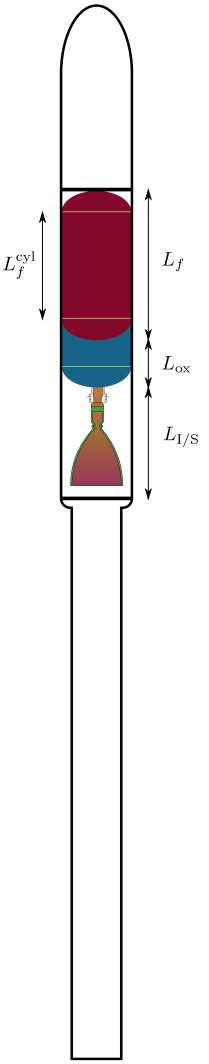


Figure 2.8

[(11)] cryogenic tanks are never completely filled, allowing the expansion of cold liquid propellant

The previous result would hence yield an overall 2nd stage length of:

$$L_{\text{LH2SS}} = L_{\text{tanks}} + L_{\text{I/S}} = 22.39 \text{ m}$$

2.1.5 Injection Analysis

The last design aspect considered in this report is related to the injector plate. It was chosen to analyze the case of a simple *unlike doublet* impinging-stream injection pattern.[12] Fig. 2.9 lists some of the orifice geometries that are commonly used for realizing the injection holes: let us consider a short tube with conical entrance:

$$D_{\text{inj}} = 3.18 \text{ mm} \quad \longrightarrow \quad C_D \approx 0.8$$

It is also reasonable to assume $\Delta p \approx 0.15 \cdot p_c = 165$ kPa.

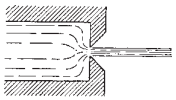
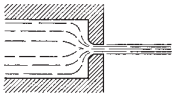
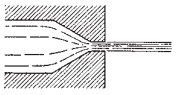
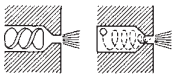
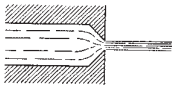
Orifice Type	Diagram	Diameter (mm)	Discharge Coefficient
Sharp-edged orifice		Above 2.5 Below 2.5	0.61 0.65 approx.
Short tube with rounded entrance $L/D > 3.0$		1.00 1.57 1.00 (with $L/D \sim 1.0$)	0.88 0.90 0.70
Short tube with conical entrance		0.50 1.00 1.57 2.54 3.18	0.7 0.82 0.76 0.84–0.80 0.84–0.78
Short tube with spiral effect		1.0–6.4	0.2–0.55
Sharp-edged cone		1.00 1.57	0.70–0.69 0.72

Figure 2.9: *Injector discharge coefficients (from [Sut16])*

The discharge coefficient is defined as

$$C_D = \frac{Q}{Q_{\text{id}}} \tag{2.15}$$

$$\overset{\text{turbulent flow}}{\rightarrow} = \frac{Q}{A\sqrt{\frac{2\Delta p}{\rho}}} \tag{2.16}$$

[12] RP1SS uses instead a more complex coaxial *pintle-injector*, which is quite adequate for deep throttling requirements

$$\longrightarrow \underbrace{Q}_{\frac{\dot{m}}{\rho}} = C_D A\sqrt{\frac{2\Delta p}{\rho}} \quad \Longrightarrow \quad \begin{cases} A_{\text{ox}}^{\text{tot}} = \dfrac{\dot{m}_{\text{ox}}}{C_D\sqrt{2\rho_{\text{ox}}\Delta p}} & = 3570.51 \text{ mm}^2 \\ A_f^{\text{tot}} = \dfrac{\dot{m}_f}{C_D\sqrt{2\rho_f\Delta p}} & = 2474.98 \text{ mm}^2 \end{cases}$$

So that, if a single hole has a minimum section $A_{\text{inj}}^{\text{min}} = \pi\frac{D_{\text{inj}}^2}{4} = 7.94$ mm^2, then the number of injection holes can be computed as:

$$N_{\text{ox}}^{\text{max}} = \text{floor}\left(\frac{A_{\text{ox}}^{\text{tot}}}{A_{\text{inj}}^{\text{min}}}\right) = 449 \qquad N_f^{\text{max}} = \text{floor}\left(\frac{A_f^{\text{tot}}}{A_{\text{inj}}^{\text{min}}}\right) = 311$$

$$N_{\text{inj}} = \min\left(N_{\text{ox}}^{\text{max}},\, N_f^{\text{max}}\right) = 311$$

and the overall number of holes through the injector plate is 622, half for the oxidizer and the other half for injecting fuel.
From the above obtained number, we can now determine the exact hole dimensions and thus also their relative injection velocities, which are required to guarantee the desired mass flow rate.

$$A_{\text{ox}}^{\text{inj}} = \frac{A_{\text{ox}}^{\text{tot}}}{N_{\text{inj}}} = 11.48 \text{ mm}^2 \qquad A_f^{\text{inj}} = \frac{A_f^{\text{tot}}}{N_{\text{inj}}} = 7.96 \text{ mm}^2$$

$$D_{\text{ox}}^{\text{inj}} = \sqrt{\frac{4A_{\text{ox}}^{\text{inj}}}{\pi}} = 3.82 \text{ mm} \qquad D_f^{\text{inj}} = \sqrt{\frac{4A_f^{\text{inj}}}{\pi}} = 3.18 \text{ mm}$$

$$v_{\text{ox}} = \frac{\dot{m}_{\text{ox}}}{\rho_{\text{ox}}A_{\text{ox}}^{\text{tot}}} = 41.53 \ \frac{\text{m}}{\text{s}} \qquad v_f = \frac{\dot{m}_f}{\rho_f A_f^{\text{tot}}} = 172.72 \ \frac{\text{m}}{\text{s}}$$

In order to minimize losses due to undesired velocity components of the injected flow, the inclination δ of the jet resulting from the impingement (indicated in Fig. 2.10) has to be $\delta = 0$.

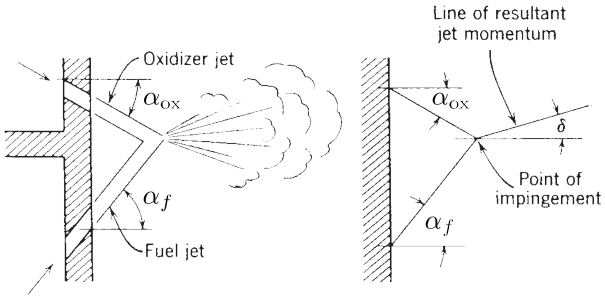


Figure 2.10: *Doublet impinging-stream injection pattern*

The conditions corresponding to this situation can be obtained from the principle of **conservation of momentum**, equating the velocity components before and after the impingement:

$$\begin{cases} x: & \dot{m}_{ox} v_{ox} \cos\alpha_{ox} + \dot{m}_f v_f \cos\alpha_f = \dot{m}_{res} v_{res} \cos\delta \\ y: & -\dot{m}_{ox} v_{ox} \sin\alpha_{ox} + \dot{m}_f v_f \sin\alpha_f = \dot{m}_{res} v_{res} \sin\delta \end{cases}$$

$$\xRightarrow[\frac{(\cdot)_y}{(\cdot)_x}]{} \quad \tan\delta = \frac{-\dot{m}_{ox} v_{ox} \sin\alpha_{ox} + \dot{m}_f v_f \sin\alpha_f}{\dot{m}_{ox} v_{ox} \cos\alpha_{ox} + \dot{m}_f v_f \cos\alpha_f}$$

So that $\delta = 0 \rightarrow \tan\delta = 0 \quad \Longrightarrow \quad \dot{m}_{ox} v_{ox} \sin\underset{?}{\alpha_{ox}} = \dot{m}_f v_f \sin\underset{?}{\alpha_f}$

One of the two unknown angles needs to be fixed, hence it was chosen $\alpha_{ox} = 30°$, which corresponds to

$$\alpha_f = \arcsin\left(\frac{\dot{m}_{ox} v_{ox}}{\dot{m}_f v_f} \sin\alpha_{ox}\right) = 46.16°$$

2.2 Mass Estimation

The exact calculation of the mass of a launch vehicle requires a careful and accurate design of the structure and all its subsystems, which in a preliminary design phase may turn out to be very time consuming and hence not recommended.
Our approach will make use of historical data from 61 different LOX/LH_2 rocket stages, to train a model with the aim of predicting the dry mass of LH2SS. Such data will be fitted using **multivariate polynomial regression**, taking into account 6 different features: propellant mass, specific impulse, no. engines, length, diameter, maiden flight year[13] (i.e. technological progress).

Let n be the number of features, then, if our hypothesis were linear,[14] it would be in the form:

$$h_\theta(\underline{\mathbf{x}}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{2.17}$$
$$= \underline{\boldsymbol{\theta}}^\intercal \underline{\mathbf{x}} \tag{2.18}$$

where $x_0 = 1$ for convenience.

A non-linear hypothesis can still be expressed by means of an inner product like in Eq. (2.18), simply by creating new artificial features through transformations and/or non-linear combinations of the already existing ones. For instance, by selecting a quadratic hypothesis, with 2 original features, we might have $x_0 = 1$, $x_1 = x_1$, $x_2 = x_2$, $x_3 = x_1^2$, $x_4 = x_2^2$, $x_5 = x_1 x_2$.

Given a set of data $\left\{(\underline{\mathbf{X}}^{(i)}, y_i)\right\}_{i=1}^m$, the **cost function** $J(\underline{\boldsymbol{\theta}})$ is defined as the halved[15] mean square error

[13] for LVs that never flew, the planned date will be considered
[14] i.e. the prediction we are trying to make: the dry mass
[15] just as a convenience, for the computation of the gradient descent, i.e. the derivative term of the square will cancel out with $\frac{1}{2}$

$$J(\underline{\boldsymbol{\theta}}) = \frac{1}{2m}\sum_{i=1}^{m}\left[h_\theta(\underline{\underline{\mathbf{X}}}^{(i)}) - y^{(i)}\right]^2 \tag{2.19}$$

$$= \frac{1}{2m}(\underline{\underline{\mathbf{X}}}\,\underline{\boldsymbol{\theta}} - \underline{\mathbf{y}})^{\mathsf{T}}(\underline{\underline{\mathbf{X}}}\,\underline{\boldsymbol{\theta}} - \underline{\mathbf{y}}) \tag{2.20}$$

and it represents a measure of the hypothesis accuracy. In Eq. (2.20), $\underline{\underline{\mathbf{X}}}$ is called the data matrix and its i^{th} row $\underline{\underline{\mathbf{X}}}^{(i)}$ contains the i^{th} training example.

In order to prevent overfitting issues,[16] we will actually be using a slightly modified version of Eq. (2.19), (2.20), namely the regularized cost function

$$J(\underline{\boldsymbol{\theta}}) = \frac{1}{2m}\Bigg\{\underbrace{\sum_{i=1}^{m}\left[h_\theta(\underline{\underline{\mathbf{X}}}^{(i)}) - y^{(i)}\right]^2}_{(\underline{\underline{\mathbf{X}}}\,\underline{\boldsymbol{\theta}}-\underline{\mathbf{y}})^{\mathsf{T}}(\underline{\underline{\mathbf{X}}}\,\underline{\boldsymbol{\theta}}-\underline{\mathbf{y}})} + \lambda\sum_{j=1}^{n}\theta_j^2\Bigg\} \tag{2.21}$$

where the newly introduced λ is called the regularization parameter and has the role of "penalizing" every parameter except θ_0, while solving the related optimization problem. Increasing λ mitigates overfitting, but after overcoming a certain threshold we would incur underfitting issues.

The choice of values $\{\theta_i\}_{i=1}^{n}$ for obtaining the best (polynomial) fit to our data, corresponds to the minimization of the cost function. So that our problem reduces to

$$\min_{\theta_1,\theta_2,\ldots,\theta_n}\Big(J(\underline{\boldsymbol{\theta}})\Big)$$

For solving the aforementioned minimization problem, there are basically two possible approaches:

- the Normal Equation method (NE), which is highly computationally expensive[17] for very high-dimensional problems, however not in this specific case
- iterative algorithms such as Gradient Descent (GD), Conjugate GD, BFGS, L-BFGS

Given our relatively low-dimensional problem, it actually turns out that the NE method is slightly more efficient, however it was decided to implement GD, because it scales better to larges sizes and hence can efficiently work in any condition, without non-invertibility issues.[18]

Multivariate Gradient Descent

1. Before executing the algorithm, a pre-processing step is required, namely feature scaling[19] (with mean normalization), according to:

[16] which are typical when using a wide set of features

[17] due to the computation of the inverse matrix $[\underline{\underline{\mathbf{X}}}^{\mathsf{T}}\underline{\underline{\mathbf{X}}}]^{-1}$, with $\texttt{flops} \underset{\text{LU fact.}}{=} \mathcal{O}(\frac{8}{3}n^3)$

[18] that are encountered whenever two of the chosen features are linearly dependent

[19] this may drastically affect convergence speed

$$\underline{\underline{\mathbf{X}}}^{(i)} \mapsto \frac{\underline{\underline{\mathbf{X}}}^{(i)} - \mu^{(i)}\underline{\underline{\mathbb{I}}}}{\mathrm{std}(\underline{\underline{\mathbf{X}}}^{(i)})}$$

where $\mu^{(i)}$ is the average of the elements of vector $\underline{\underline{\mathbf{X}}}^{(i)}$, while $\mathrm{std}(\cdot)$ indicates the standard deviation

2. Weights are all initialized to zero: $\underline{\boldsymbol{\theta}} = \underline{\mathbf{0}} \in \mathbb{R}^{n+1}$
3. $\underline{\boldsymbol{\theta}}$ is updated at each iteration, until convergence

$$\texttt{while} \quad (\mathrm{tol} > \varepsilon \quad \wedge \quad \mathrm{iter} < \mathrm{iter}_{\max}) \ \{$$

$$\begin{aligned}\underline{\boldsymbol{\theta}} &= \underline{\boldsymbol{\theta}} - \alpha \frac{\partial J}{\partial \underline{\boldsymbol{\theta}}} \\ &= \underbrace{\left(1 - \alpha\frac{\lambda}{m}\right)}_{\lesssim 1} \underline{\boldsymbol{\theta}} - \frac{\alpha}{m}\underline{\underline{\mathbf{X}}}^{\intercal}\left(\underline{\underline{\mathbf{X}}}\,\underline{\boldsymbol{\theta}} - \underline{\mathbf{y}}\right)\end{aligned}$$

$$\}$$

The parameter $\alpha \in \mathbb{R}$ is called the learning rate and it has to be carefully selected, in order to make the algorithm converge, preferably as quick as possible. For small values of α we have a slower convergence rate; by increasing this parameter, the algorithm starts converging faster, until it diverges: α is indeed preferably set to the highest possible value still guaranteeing convergence. The effect of α for the current analysis is highlighted in Fig. 2.11.
The choice of the polynomial degree d and the regularization parameter λ can instead be performed by minimizing the validation error on a small set of data, with the condition that h_θ is not fit to it.

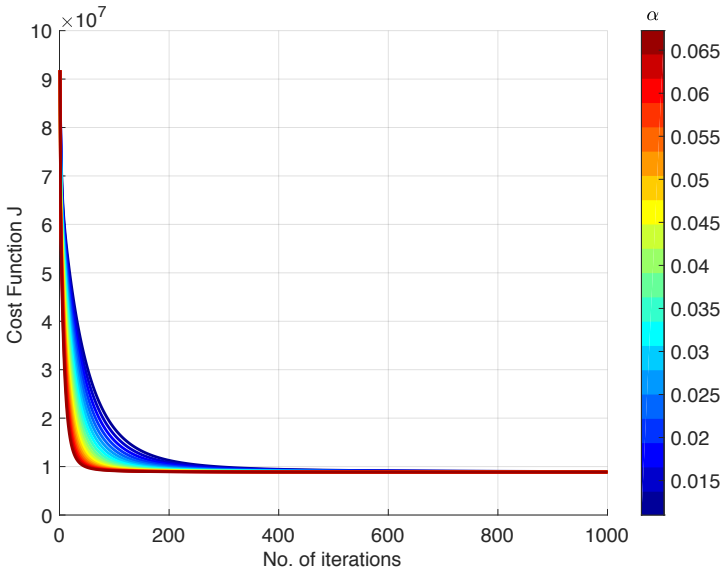


Figure 2.11: *GD convergence parametrized w.r.t.* α

After proper vectorization of the code, the algorithm can be easily implemented, as shown in C.8, taking advantage of `Matlab`'s linear algebra libraries.

- The historical data used to build the model can be found in Tab. B.1.

- The parameters for executing the gradient descent algorithm (C.8) were set to $\alpha = 0.067275$, $\lambda = 8.192$.
- The polynomial degree was selected as $d = 8$ and additional features were created by raising to the power of 1-to-d each of the original $n = 6$ features, and eventually combining them into all possible couples (i.e. $\binom{n}{2} = \frac{n!}{2!\cdot(n-2)!} = 15$ further features), for a total of $n + n \cdot (d-1) + \binom{n}{2} = 63$ features. The creation of artificial features is executed in C.9.

The final result of the training is a predictive model using $63 + \underset{\substack{\downarrow \\ \text{bias unit } (\theta_0)}}{1}$ learned parameters $\theta_0, \theta_1, \theta_2, \ldots, \theta_{63}$ indicated in Tab. B.3.

At this point, the mass prediction for a given launch vehicle (represented by a certain feature vector) can be easily performed, just with the inner product $\underline{\boldsymbol{\theta}}^{\mathsf{T}}\underline{\mathbf{x}}_{\text{pred}}$, where $\underline{\mathbf{x}}_{\text{pred}} \in \mathbb{R}^{64}$ is the "extended" feature vector, obtained from the original features as previously explained. For our redesigned LV, the original features are:

$$\underline{\mathbf{x}}_{\text{pred}}^{\text{LH2SS}}(1:6) = [93\,494 \quad 436 \quad 1 \quad 22.39 \quad 5.2 \quad 2015]^{\mathsf{T}}$$

$$m_{\text{dry}}^{\text{LH2SS}} = \underline{\boldsymbol{\theta}}^{\mathsf{T}}\underline{\mathbf{x}}_{\text{pred}}^{\text{LH2SS}} = 9054.2 \text{ kg}$$

The accuracy of this result can be confirmed by [Dum17]: in this paper by DLR, a LOX/LH_2 VTOL solution very similar to our redesign is investigated. The 2nd stage of the LV has a propellant mass of 77 000 kg (vs. 93 494 kg) and the authors ended up predicting a dry mass of 8500 kg which, as expected, is slightly lower than our value.

Chapter 3

Conclusions

The main results obtained in Chap. 2 are summarized and compared with the specifications of RP1SS, in Tab. 3.1.

Table 3.1: *2nd stages comparison*

	RP1SS	**LH2SS**
m_{wet}	$\sim$ 112.5 Mg	102.548 Mg
m_{dry}	$\sim$ 4 Mg	9.054 Mg
L	15.95 m (including I/S)	22.39 m (including I/S)
D	3.66 m	5.2 m
$\mathcal{F}_{\text{vac}}$	934 kN	934 kN
$I_{\text{sp}}^{\text{vac}}$	348 s	436 s
t_b	397 s	412 s
ε	165	195
m_{ox}	75 200 kg	80 138 kg
m_f	32 300 kg	13 356 kg
V_{ox}	61.4 m^3	65.4 m^3
V_f	37.6 m^3	188.6 m^3
r	2.33	6
p_c	110 bar	110 bar
Propellants	LOX/RP-1	LOX/LH_2

The first and most intuitive evaluation metric for comparing the two stages is their Δv. Given that they operate in vacuum, the Tsiolkovsky equation will give us very accurate results:

$$\Delta v = I_{\text{sp}} g_0 \ln\left(\frac{m_{\text{initial}}}{m_{\text{final}}}\right) \tag{3.1}$$

where

$$m_{\text{initial}} = m_{\text{ox}} + m_f + m_{\text{dry}} + m_{\text{fairing}} + PL \qquad m_{\text{final}} = m_f + m_{\text{dry}} + m_{\text{fairing}} + PL$$

Considering the maximum payload allowed by the PAF, i.e. $PL = 10\,886$ kg, yields:

$$\Delta v_{\text{RP1SS}} = 6845.6 \ \frac{\text{m}}{\text{s}} \qquad \Delta v_{\text{LH2SS}} = 7103.7 \ \frac{\text{m}}{\text{s}}$$

The performance increase is therefore confirmed, with a 4% higher Δv, which may have allowed the recovery of the 1st stage during some of the past expendable missions. This higher velocity increase performed by the 2nd stage determines a lower propellant consumption before MECO, potentially leaving an additional amount that is sufficient for the *boost-back*, *re-entry* and *landing* burns.

Whether or not the redesigned configuration could have resulted in a net cost reduction for SpaceX is however still debatable.

The first aspect that stands out is the much larger - and hence more expensive - 2nd stage which in turn causes a higher fixed cost at each launch.

The M1D and M1DV engines used by RP1SS are basically the same, with minor differences, due to the fact that the latter operates in vacuum, thus having a much higher area expansion ratio. Employing two different propellant combinations, inevitably translates into a practically doubled R&D effort, for designing very different engine configurations, also somehow doubling the risk of failure.

In addition, two different rocket engines also means two separate testing campaigns, aimed at certifying them for flight. The related procedures are very expensive, time consuming and need to be carried out after each slight upgrade/modification of the configuration. This would invariably double the effort and slow down the pace of SpaceX.

Within the continuation of this study it is planned to carry out a detailed economical analysis and to determine which of the past expendable missions would have instead allowed the recovery. The related savings need to be compared with an estimate of the additional above mentioned expenses, for which further data is required.

Appendix A

Thermochemistry

A.1 Thermochemical Characterization of RP1SS

Table A.1: *LSQ coefficients of RP-1 and its main combustion products*

	RP-1	**CO**	**CO_2**	**H_2**	**H_2O**
a_1	3.9508691	35.1507	58.16639	43.41356	41.96426
a_2	0.10207987	0.001300095	0.002720074	-0.004293079	0.008622053
a_3	1.31E-05	-2.06E-07	-4.92E-07	1.27E-06	-1.50E-06
a_4	-7.66E-08	1.36E-11	3.88E-11	-9.69E-11	9.81E-11
a_5	3.45E-11	-3282780	-6447293	-20533862	-11157640
a_6	-52093.574	-127.8375	-425.9186	-38.515158	-272.1797
a_7	21.980951	231.712	263.6125	162.081354	219.7809
a_8	0	-110.5271	-393.5224	0	-241.8264

A.2 Thermochemical Characterization of LH2SS

Table A.2: *LSQ coefficients of H_2 and its main combustion products*

	$\mathbf{H_2}$	$\mathbf{O_2}$	**OH**	$\mathbf{H_2O}$
a_1	43.41356	20.91111	28.74701	41.96426
a_2	-0.004293079	0.01072071	0.004714489	0.008622053
a_3	1.27E-06	-2.02E-06	-8.15E-07	-1.50E-06
a_4	-9.69E-11	1.46E-10	5.47E-11	9.81E-11
a_5	-20533862	9245722	-2747829	-11157640
a_6	-38.515158	5.337651	26.41439	-272.1797
a_7	162.081354	237.6185	214.1166	219.7809
a_8	0	0	38.98706	-241.8264

Note: *for O and H, given the very slight variation with temperature of their c_p value, the LSQ coefficients have not been used, but instead a constant isobaric specific heat was assumed.*

Appendix B

Mass Prediction

B.1 Historical Data

The following pages contain the total amount of data used for training and validation of the predictive model, on which is based the dry mass estimate of the redesigned LH2SS. All of these features have been indeed used, with the only exception of I_{tot}.

Table B.1: *Historical data of LOX/LH$_2$ stages: training set*

Stage	m_p	I_{tot}	I_{vac}	No. Engines	L	D	Maiden Flight	m_{dry}
Angara Stage 2	66000	441000	455	1	16.00	3.90	2004	9000
Ariane 5 EPC	173300	724100	434	1	30.50	5.46	2005	12700
Ariane 5 ESC B	24100	107730	467	1	10.00	5.46	2018	3400
Ariane 5-1	158100	656146	430	1	30.50	5.46	2005	12700
Centaur B-X	16780	152426	470	2	10.10	3.05	1998	2358
Centaur C	13604	114765	425	2	9.14	3.05	1960	1996
Centaur D/E	13627	123349	444	2	9.60	3.05	1962	2631
Centaur G	21105	183500	444	2	9.00	4.33	1989	2775
Centaur G Prime	16501	161480	444	2	8.87	4.33	1984	3000
Centaur G STS	13727	123312	444	2	5.95	4.33	1986	2600
Centaur I	13900	118027	444	2	9.15	3.05	1990	1700
Centaur II	16780	143277	444	2	10.10	3.05	1991	2053
Centaur IIA	16780	145049	449	2	10.10	3.05	1992	2293
Centaur IIIB	20830	182453	451	2	11.68	3.05	2000	2130
Centaur V2	20800	172606	451	2	12.68	3.05	2002	2250
Chang Cheng 1	27000	210700	230	1	24.00	4.50	1992	30000
CZ-3-3	8500	141120	425	4	7.48	2.25	1984	2000
CZ-3A-3	18200	146640	440	2	12.38	3.00	1994	2800
DC-X	9120	133502	373	4	11.89	3.05	1993	7200
Delta 3 - 2	16824	77021	462	1	8.80	2.44	1998	2476

Delta 4H - 2	27220	123806	462	1	12.00	2.44	2004	3490
Energia EUS	70000	313924	455	1	16.47	5.70	1990	7000
Energia M	244000	1078000	455	1	20.00	7.70	1993	28000
H-1-2	8800	38073	450	1	10.32	2.49	1986	1800
H-2-1	86200	372987	446	1	28.00	4.00	1994	11900
H-2-2	14000	72900	452	1	10.60	4.00	1994	2700
H-2A-1	100000	428220	440	1	37.20	4.00	2001	13600
H-2A-2	16600	73158	447	1	9.20	4.00	2009	3000
Jarvis-2	130000	541791	425	1	10.00	8.38	1985	15000
Mustard 1	117007	462250	405	1	30.00	4.00	1968	24036
N1 Block S	50000	211680	440	1	16.00	6.70	1967	8000
N1 Block Sr	66400	564902	441	2	16.50	5.20	1974	11500
Nova 59-4-4	59000	237216	420	1	11.90	3.00	1959	9000
Nova 60-8-3	204000	839372	425	1	30.50	6.70	1960	23000
Nova A-3	36000	751200	427	5	13.10	5.80	1960	4000
Project 921-2	50000	210700	440	1	10.00	4.50	1992	7000
Sanger I-2	13500	57330	430	1	20.50	2.20	1965	18000
Sanger II-2	79400	381441	490	1	27.60	5.50	1985	32600
SASSTO	91308	467430	464	1	18.80	6.60	1967	6668
Saturn IVB	106609	490192	425	1	17.80	6.61	1961	13311
Saturn IVB (S-IB)	105900	490010	421	1	17.80	6.61	1961	12900
Saturn IVB (S-V)	106600	490010	421	1	17.80	6.61	1961	13300
Saturn IVB C-3B	45343	184090	420	1	15.40	5.59	1961	6801

Saturn IVB C-5A	90686	369070	420	1	19.00	5.59	1961	11563
Saturn IVB-A	105900	430245	421	1	17.80	6.61	1965	12900
Saturn MS-IVB-1	106600	490010	421	1	18.08	6.61	1965	13900
Saturn MS-IVB-1A	158800	644750	421	1	22.82	6.61	1967	20400
Saturn MS-IVB-2	158800	685236	447	1	23.12	6.61	1965	17800
Saturn MS-IVB-3B	158800	684877	447	1	22.82	6.61	1967	20400
Saturn MS-IVB-4(S)B	104300	423988	421	1	17.79	6.61	1967	14100
Saturn MS-IVB-x	158700	644784	421	1	18.08	6.61	1965	36300
Saturn S-IVC	181373	717228	425	1	22.86	6.61	1962	22671
Sea Horse-2	13000	53361	425	1	10.00	3.00	1962	2000
SLS Stage A	53000	222331	424	1	15.00	4.28	1961	6000
Titan C-2	51400	199800	403	1	19.50	4.07	1961	6000
Vulkan Blok V	127000	563070	460	1	16.50	6.70	1975	15000

Table B.2: *Historical data of LOX/LH$_2$ stages: validation set*

Stage	m_p	I_{tot}	I_{vac}	No Engines	L	D	Maiden Flight	m_{dry}
Delta 4 - 2	21320	93543	462	1	12.00	2.44	2002	2850
Ares Stage 2	158800	712320	465	1	16.00	10.00	2019	13200
CZ-NGLV-HO	22900	187200	448	2	12.00	3.35	2004	3100
Centaur C-X	16780	73128	450	1	10.10	3.05	1998	2358
Mustard 2	117241	464400	405	1	30.00	4.00	1968	24943

B.2 Trained Model

Table B.3: *Learned parameters*

Components	$\underline{\theta}$			
θ_{0-3}	9093.064556331	674.925638096451	-1597.74696787808	-122.260506527634
θ_{4-7}	966.363091274269	-34.0362298982093	-93.2452044483517	604.202830712997
θ_{8-11}	-1164.89748841528	144.996888456976	741.557318152466	80.8242537390301
θ_{12-15}	-110.214492767153	344.213373973634	-681.414887689445	221.367152485161
θ_{16-19}	391.012416646232	133.333565539392	-127.137810302955	126.386765243343
θ_{20-23}	-181.553804212605	158.524269297941	58.3225142132354	159.229774270255
θ_{24-27}	-144.01348393944	8.50907874225066	312.575579682314	33.8952045035571
θ_{28-31}	-185.938375705219	176.811625134224	-160.839829255242	-24.0495799581849
θ_{32-35}	791.061474103998	-104.223058818554	-329.132714683561	191.810343782209
θ_{36-39}	-177.615150881875	-1.04533024953471	1251.66397806114	-233.037217793518
θ_{40-43}	-386.754956324669	204.167803738353	-194.337742873889	52.1387321179375
θ_{44-47}	1695.6544503611	-344.272829620171	-382.587016211129	212.591527056247
θ_{48-51}	-211.005889093096	569.696105078911	532.287456360677	-692.214370747629
θ_{52-55}	775.557818228936	626.257545235748	-437.859724290955	770.490371757632
θ_{56-59}	-305.246009209413	-1557.37098671937	450.641132758919	-265.890404418351
θ_{60-63}	-112.946220652396	-140.123120597097	920.777226557989	-30.3122309033616

Appendix C

Source Code

C.1 Matlab Code

C.1.1 Second Stage Sizing

Thermochemistry

Listing C.1: *MAIN.m*

```
clc
close all
clear all

set(0, 'defaultTextInterpreter', 'tex' )
set(0, 'DefaultAxesFontSize', 12)


%% RP1 - THERMOCHEMICAL PROPERTIES
run('Thermochemistry/thermochemicalProperties_RP1.m')
kRP1


%% EXPANSION PRESSURE CALCULATION
% The redesigned LH2 Second Stage (LH2SS) will have the same p_e of the actual RP-1
% second stage (RP1SS) (i.e. the same optimum expansion altitude)
p_c = 110*10^5; % Combustion chamber pressure [Pa] (we will assume it to be the same for LH2SS)
epsilon = 165; %Area expansion ratio [-]
%Let us define x = p_e/p_c
f = @(x) (1/epsilon) - ((kRP1+1)/2)^(1/(kRP1-1)) * (x).^(1/kRP1) .* sqrt( ((kRP1+1)/(kRP1-1)) * (1-x.^((kRP1-1)/kRP1)) );
syms x
```

```
22 x0 = 0; % initial guess
23 p_e2p_c = fsolve(f,x0)
24 p_e = p_e2p_c * p_c %Expansion pressure
25
26
27 %%
28 p_e
29 T_c = 3534; %Combustion temperature; [K] ************ CEA
      OUTPUT (LOX/LH2) ************
30 F_vac = 934e+3; %Vacuum thrust [N] (we will assume the same
      value of RP1SS)
31
32 run('Thermochemistry/thermochemicalProperties_H2.m')
33 kH2
34
35 M_m = 0.01353; %Molecular mass [g/mol] ************ CEA
      OUTPUT (LOX/LH2) ************
36 R_u = 8.3145;
37 R = R_u / M_m; %[J/(kg*K)]
38 rho_c = p_c / (R * T_c); %combustion gases density [kg/m^3]
39 g0 = 9.8065; %Gravity constant [m/s^2]
40
41 % Let us define x = p_e/p_c
42 epsilon = (((kH2+1)/2)^(1/(kH2-1)) * (p_e2p_c).^(1/kH2) .*
      sqrt( ((kH2+1)/(kH2-1)) * (1-p_e2p_c^((kH2-1)/kH2)) ))^-1
      %(LOX/LH2)
43
44
45 z_coesa = [1:1:25000];
46 [~, ~, p_coesa, ~] = atmoscoesa(z_coesa);
47 [~, z_optIndex] = min(abs(p_e-p_coesa));
48 z_opt = z_coesa(z_optIndex) % Optimum expansion altitude [m]
49
50
51 %% PERFORMANCE CALCULATION
52
53 Gamma = sqrt( kH2 * ( 2/(kH2+1) ) ^ ((kH2+1)/(kH2-1))) %
      Vandenkerckhove function
54
55 c_star = sqrt(R*T_c)/Gamma %Characteristic velocity [m/s]
56
57 nu_n = sqrt( 1-(p_e/p_c)^((kH2-1)/kH2) ) %Nozzle (expansion)
      efficiency [-]
58
59 v_lim = sqrt( (2*kH2/(kH2-1)) * R * T_c) %Maximum theoretical
       velocity [m/s]
60
```

```
v_e = nu_n * v_lim %Exhaust velocity [m/s]

M_e = sqrt( (2/(kH2-1)) * ( (p_c/p_e)^((kH2-1)/kH2) -1) ) % Exhaust Mach number

a_e = v_e/M_e %Exhaust sound speed [m/s]
T_e = a_e^2/(kH2*R) %Exhaust temperature [K]
%Alternative meth.: T_e = T_c/(1+((k-1)/2)*M_e^2)

I_sp_opt = v_e / g0; %Specific impulse (optimum expansion) [s]

% ************ Search from literature ************
eta_n_is = 0.97; %Nozzle isentropic efficiency [-]
% ************ Search from literature ************

I_sp_opt = eta_n_is * I_sp_opt %real value


C_F_opt = Gamma * nu_n * sqrt(2*kH2/(kH2-1)) %Thrust coefficient [-] (optimum expansion)
C_F_opt = eta_n_is * C_F_opt %real value



C_F_vac = Gamma * nu_n * sqrt(2*kH2/(kH2-1)) + (p_e/p_c)*epsilon %Thrust coefficient [-] (vacuum expansion)
C_F_vac = eta_n_is * C_F_vac %real value

A_t = F_vac/(p_c*C_F_vac); %Throat area [m^2]
A_e = epsilon*A_t; %Exhaust area [m^2]
D_t = 2*sqrt(A_t/pi) %Throat diameter [m]
R_t = D_t/2;

D_e = 2*sqrt(A_e/pi) %Exhaust diameter [m]
R_e = D_e/2;


T_t = (2/(kH2+1))*T_c %Throat temperature [K]
a_t = sqrt(kH2*R*T_t) %Sound speed at throat [K]


m_dot_c = (F_vac - p_e*A_e)/v_e % Mass flow rate of propellant through c.c.[kg/s]
m_dot = 1.05*m_dot_c % real value  --> given by (m_dot_c + m_dot_gg),  i.e. chamber + gas generator
```

```
% N.B. from Sutton (Table 6.6): For a Gas-Generator Cycle turbine flow = 1.5 to 7% of total flow

I_sp = F_vac/(m_dot*g0);
I_sp = eta_n_is * I_sp


r = 6; % Oxidizer to Fuel ratio (chosen design value) [-]
% from Sutton, Tab. 7.1:
rho_ox = 1224.67; % Oxidizer density [kg/m^3]
rho_f = 70.8;     % Fuel density     [kg/m^3]



rho_avg = rho_ox*rho_f*(1+r) / (rho_ox + r*rho_f) % Average propellant density (LH2SS) [kg/m^3]
I_v = rho_avg * I_sp % Volumetric specific impulse [kg*s=m3]


% -------------------------------------------------------
% ---------------- Combustion Chamber -------------------
% -------------------------------------------------------

M_c = 0.2; % Combustion chamber Mach number [-]
% Such a value is a reasonable ASSUMPTION


a_c = sqrt(kH2*R*T_c);
v_c = M_c * a_c;

A_c = m_dot/(rho_c * v_c)
R_c = sqrt(A_c/pi)

L_star = 35*2.54/100 %Characteristic chamber length [m]
% From Huzel: Table 4.1 (pag. 72) -> L*=30to40'' for (LOX/LH2)

V_c = L_star*A_t

% ************ Check with typical values (if reasonable) ************
L_c = V_c/A_c
% ************ Check with typical values (if reasonable) ************


%% Nozzle Sizing (Rao's Parabolic Approximation)
```

```
143
144 % Read data obtained through WebPlotDigitizer from Huzel (Fig
    . 4.16)
145 for i = 1:2
146     figure
147     % Let us convert our data into logarithmic coordinates (
        this is quite appropriate since the curve saturates)
148     eps_theta_i = csvread(sprintf('Rao/Rao_theta_%d_80%%
        _Huzel.csv', i));
149     x_eps = eps_theta_i(:,1);
150     y_theta = eps_theta_i(:,2);
151     x_eps_log = log(x_eps);
152     y_theta_log = log(y_theta);
153     plot(x_eps_log, y_theta_log, 'o', 'MarkerSize', 5)
154     axis equal
155     grid on
156     hold on
157     title('Logarithmic Fitting')
158
159     % Fitting logarithmic-coordinates data
160     z = [log(x_eps(1)):.01:log(200)];
161     fitOrd = 2;
162     P = polyfit(x_eps_log,y_theta_log,fitOrd);
163     Pz = polyval(P,z);
164     plot(z,Pz)
165     legend('Huzel', sprintf('Fitting (%dnd degree polynomial)
        ', fitOrd));
166
167
168     fig = figure;
169     plot(x_eps, y_theta, 'o', 'MarkerSize', 5)
170     hold on
171     grid on
172     axis equal
173     plot(exp(z),exp(Pz), 'r')
174     myTheta = plotEvaluate(exp(z),exp(Pz),epsilon)
175     plot(epsilon, myTheta, 'rx')
176     xlabel(char(949)) %Unicode for \varepsilon
177     if (i==1)
178         sectAngName = 'i';
179     elseif (i==2)
180         sectAngName = 'e';
181     end
182     ylabel(['\theta_' sectAngName])
183     legend('Huzel', sprintf('Fitting (%dnd degree polynomial)
        ', fitOrd));
184
```

```
185
186
187         if (i==1)
188         theta_i = plotEvaluate(exp(z),exp(Pz),epsilon)
189         theta_i = deg2rad( theta_i );
190         run('resizePDFPaper.m')
191         print(fig, 'theta_i.pdf', '-dpdf')
192     elseif (i==2)
193         theta_e = plotEvaluate(exp(z),exp(Pz),epsilon)
194         theta_e = deg2rad( theta_e );
195         fig.PaperPositionMode = 'auto';
196         fig_pos = fig.PaperPosition;
197         fig.PaperSize = [fig_pos(3) fig_pos(4)];
198         run('resizePDFPaper.m')
199         print(fig, 'theta_e.pdf', '-dpdf');
200     end
201 end
202
203
204 beta = deg2rad(30); %Convergent angle [rad]
205 alpha = deg2rad(15); %Divergent angle [rad]
206 L_d_con = ((D_e-D_t)/(2*tan(alpha))) %Nozzle divergent lenght
    (corresponding conical nozzle) [m]
207 theta_con = deg2rad(15) %Nozzle divergent angle (
    corresponding conical nozzle) [rad]
208
209 K = 0.8; %Percentage of the corresponding conical nozzle
    lenght [-]
210
211 L_bell_tot = K*(sqrt(epsilon)-1)*R_t/tan(theta_con) %Nozzle
    divergent lenght of the chosen bell nozzle [m]
212
213
214 % Let us find the angle of the 1.5*R_t arc connecting the C.C
    . with the
215 % nozzle
216 f = @(x) (1.5*R_t*(sind(270-x) + 1) + R_t)   -   R_c;
217 syms x
218 x0 = beta; % initial guess
219 angle_1_5_R_t = fsolve(f,x0)
220
221
222
223 % 1.5 R_t circle
224 h = .01; %Plot discretization
225 thetad_1 = [270-angle_1_5_R_t:h:270]; % we will use [deg] in
    order to avoid the otherwise present floating point errors
```

```
    (e.g. cos(pi) = 6.12323399573677e-17)
x_1 = 1.5*R_t*cosd(thetad_1);
y_1 = 1.5*R_t*(sind(thetad_1) + 1) + R_t;

% Combustion chamber
x_cc = [x_1(1):-h:-L_c+x_1(1)];
y_cc = ones(1,length(x_cc))*y_1(1);


fig = figure;
plot(x_cc, y_cc, 'LineWidth', 2)
hold on
grid on
axis equal
plot(x_1, y_1, 'LineWidth', 2)


thetad_2 = [thetad_1(end):h:270+rad2deg(theta_i)];
x_2 = 0.4*R_t*cosd(thetad_2);
y_2 = 0.4*R_t*(sind(thetad_2) + 1) + R_t;
plot(x_2, y_2, 'LineWidth', 2)

x_N = x_2(end)
y_N = y_2(end)
x_E = L_bell_tot-abs(x_1(1))
y_E = sqrt(epsilon)*R_t

% x = ay^2+by+c
A_parabSys = [2*y_N   1 0;
              R_e^2 R_e 1;
              y_N^2 y_N 1]
b_parabSys = [1/tan(theta_i) L_bell_tot x_N]'
parCoeffs = A_parabSys\b_parabSys

y_3 = [y_2(end):h:y_E];
x_3 = parCoeffs(1)*y_3.^2 + parCoeffs(2)*y_3 + parCoeffs(3);
plot(x_3, y_3, 'LineWidth', 2)

% Mirror the semi-nozzle
x_123 = [x_cc x_1 x_2 x_3];
y_123 = [y_cc y_1 y_2 y_3];
plot(x_123, -y_123, 'k');

xlabel('[m]')
ylabel('[m]')
legend(sprintf('Combustion Chamber: R_c = %.3f m', R_c),
   sprintf('Circle with radius 1.5R_t (where R_t = %.3f m)',
```

```
   R_t), 'Circle with radius 0.382R_t', sprintf('Parabolic
   Nozzle: x = %.3fy^2%+4.3fy%+4.3f', parCoeffs(1), parCoeffs
   (2), parCoeffs(3)), 'Location', 'east' )
title('Nozzle Contour')
run('resizePDFPaper.m')
print(fig, 'nozzle_2d', '-dpdf')



theta_e = atan( (y_3(end)-y_3(end-1))/(x_3(end)-x_3(end-1)) )
   ;
sprintf('theta_i=%.2f deg theta_e=%.2f deg', rad2deg(theta_i)
   , rad2deg(theta_e))


% deriv=diff(y_3)./diff(x_3)
% tang=(y_3-y_3(end))*deriv(end)+y_3(end)
% hold on
% plot(x_3,tang)
% scatter(x_3(end),y_3(end))
%
% deriv=diff(y_3)./diff(x_3)
% tang=(y_3-y_3(1))*deriv(1)+y_3(1)
% hold on
% plot(x_3,tang)
% scatter(x_3(1),y_3(1))


% y_123 = [x_1 x_2 x_3]';
%     order = 2;
%     [rx,ry]=meshgrid( -order/2:order/2 , -order/2:order/2 )
   ;
%     r = hypot( rx , ry );
%     halfb = y_123((length(y_123)-1)/2 + 1 : .01 : end );
%     vq = interp1( 0:(length(halfb)-1) , halfb , r(:) , '
   linear' , 0 );
%     W=r; W(:)=vq;

%% 3D NOZZLE PLOT
z = x_123';
z = z(1:10:end);
y = y_123';
y = y(1:10:end);
myTheta = linspace(0,2*pi,length(z));
[TH, Z] = meshgrid(myTheta, z);
RAD = y;

```

```
311 [X,Y,Z] = pol2cart(TH,RAD,Z);
312
313
314
315
316 fig = figure;
317 % UNCOMMENT for high res. .png render
318 % set(fig, 'DefaultAxesFontSize', 100)
319 % set(fig, 'DefaultAxesLineWidth',  6)
320 nSurf = surf(X,Y,Z,sqrt(X.^2+Y.^2));
321 axis equal
322 view([-30,12])
323 shading interp
324 lightangle(50,-30)
325 nSurf.FaceLighting = 'gouraud';
326 nSurf.AmbientStrength = 0.5;
327 nSurf.DiffuseStrength = 0.8;
328 nSurf.SpecularStrength = 0.9;
329 nSurf.SpecularExponent = 25;
330 nSurf.BackFaceLighting = 'lit';
331 nSurf.FaceAlpha = 0.75;
332
333 % fig.Renderer = 'Painters' % Allows vector .pdf export
334 % (by default, above 180x180 points surf() becomes a bitmap)
335 run('resizePDFPaper.m')
336 print(fig, 'nozzle_3d', '-dpdf')
337
338 % UNCOMMENT for high res. .png render
339 % ppi = 150; % pixels per inch
340 % set(gcf, 'PaperUnits', 'inches', 'PaperPosition', [0 0
     10000 10000]/ppi);
341 % print(gcf,'-dpng',sprintf('-r%d',ppi), 'Nozzle (3D Render)
     _highRes');
342
343
344
345 %% TANKS AND INJECTION
346
347 run('Injection/tanksAndInjection.m')
```

Listing C.2: *plotEvaluate.m*

```
1 function [y_out] = plotEvaluate(x,y,x_ex)
2
3 [~,n] = min(abs((x-x_ex)));
4
5 y_out = y(n);
6
```

```
end
```

Listing C.3: *resizePDFPaper.m*

```

fig.PaperPositionMode = 'auto';
fig_pos = fig.PaperPosition;
fig.PaperSize = [fig_pos(3) fig_pos(4)];
```

Listing C.4: *thermochemicalProperties_H2.m*

```
%% THERMOCHEMICAL PROPERTIES FOR: THE LOX/LH2 COMBUSTION
   PRODUCTS


Ru = 8.3145; % [J/mol K]

% Shomate Equation for isobaric SPECIFIC HEAT
cp = @(v) v(1) + v(2)*T_c + v(3)*T_c^2 + v(4)*T_c^3 + v(5)/
   T_c^2;


% LSQ coefficients for H2 (2500 to 6000 K) [Source: NIST]
v_h2 = [43.413560
         -4.293079e-03
         1.272428e-06
         -0.096876e-09
         -20.533862e+06
         -38.515158
         162.081354
         0.0];

Mm_h2= 0.002;
cp_h2 = cp(v_h2)/Mm_h2;
R_h2 = Ru/Mm_h2;
cv_h2 = cp_h2 - R_h2;
k_h2 = cp_h2/cv_h2



% LSQ coefficients for H2O (1700 to 6000 K) [Source: NIST]
v_h2o = [41.96426
         8.622053e-03
         -1.499780e-06
         0.098119e-09
         -11.15764e+06
         -272.1797
         219.7809
```

```
36          -241.8264];
37
38 Mm_h2o= 0.018;
39 cp_h2o = cp(v_h2o)/Mm_h2o;
40 R_h2o = Ru/Mm_h2o;
41 cv_h2o = cp_h2o - R_h2o;
42 k_h2o = cp_h2o/cv_h2o
43
44
45 % LSQ coefficients for O2 (2000 to 6000 K) [Source: NIST]
46 v_o2 = [20.91111
47          10.72071e-03
48          -2.020498e-06
49          0.146449e-09
50          9.245722e+06
51          5.337651
52          237.6185
53          0.0];
54
55 Mm_o2= 0.032;
56 cp_o2 = cp(v_o2)/Mm_o2;
57 R_o2 = Ru/Mm_o2;
58 cv_o2 = cp_o2 - R_o2;
59 k_o2 = cp_o2/cv_o2
60
61
62 % LSQ coefficients for OH (radical) (1600 to 6000 K) [Source:
       NIST]
63 v_oh = [28.74701
64          4.714489e-03
65          -0.814725e-06
66          0.054748e-09
67          -2.747829e+06
68          26.41439
69          214.1166
70          38.98706];
71
72 Mm_oh = 0.017;
73 cp_oh = cp(v_oh)/Mm_oh;
74 R_oh = Ru/Mm_oh;
75 cv_oh = cp_oh - R_oh;
76 k_oh = cp_oh/cv_oh
77
78
79 Mm_o = 0.016;
80 cp_o = 21.092 /Mm_o; %Isobaric specific heat for O (radical)
     at 3500K (very slight variation with temperature) [J/kg]
```

```
                    % [Source: Chase]
R_o = Ru/Mm_o;
cv_o = cp_o - R_o;
k_o = cp_o/cv_o



Mm_h = 0.001;
cp_h = 20.78603/Mm_h; %Isobaric specific heat for H (radical) at 3500K (very slight variation with temperature) [J/kg]
                     % [Source: Chase]
R_h = Ru/Mm_h;
cv_h = cp_h - R_h;
k_h = cp_h/cv_h



molarFracts = [0.91651 0.03899 0.03617 0.00444 0.00202 0.00182]'; % ************ CEA OUTPUT (LOX/LH2) ************
                    %[  H2O       OH       H2       O2       O       H    ]


% Let us add the fraction of neglected combustion products to the most abudant
% one, so that sum(molarFracts)=1
kProds = [k_h2o k_oh k_h2 k_o2 k_o k_h]'
missingPerc = 1 - sum(molarFracts);
[maxFrac, posMax] = max(molarFracts);
molarFracts(posMax) = molarFracts(posMax) + missingPerc;


molarFracts
sum(molarFracts)



kH2 = sum(molarFracts.*kProds) %Ratio of specific heats = c_p/c_v [-]
                               %(weighted average of
                                  combustion products)
```

Listing C.5: *thermochemicalProperties_RP1.m*

```

% CALCULATION OF THERMOCHEMICAL PROPERTIES FOR: RP1 AND THE LOX/RP1 COMBUSTION PRODUCTS
```

```
3
4
5  %% RP1
6
7  % LSQ (least-squares) coefficients for RP1 (range: 300 to
       1000 K) [Source: NASA]
8  % Note: a one formula surrogate-fuel (C12H24) was assumed
9  v_rp1 = [0.39508691e+01
10      0.10207987e+00
11      0.13124466e-04
12     -0.76649284e-07
13      0.34503763e-10
14     -0.52093574e+05
15      0.21980951e+02];
16
17
18 T = 266; %[K]
19 Ru = 8.3145; %[J/(mol*K)]
20 Mm = 0.175; %[kg/mol]
21 R = Ru / Mm; %[J/(kg*K)]
22
23 % Shomate Equation for ENTHALPY [J/mol]
24 H = @(v_rp1) R*T*(  v_rp1(1)+ v_rp1(2)*T/2+ v_rp1(3)*T^2/3 +
       v_rp1(4)*T^3/4 + v_rp1(5)*T^4/5 + v_rp1(6)/T  );
25
26 % RP1 enthalpy [J/mol]
27 H_rp1 = H(v_rp1)
28
29
30 % ************************************************
31 % fileName = 'RP-1'; % SET YOUR INPUT FILENAME
32 % unix(sprintf('python %s/ceaCallRP1.py %s RP-1 %.3f',
       ceaWorkingDir_unix, ceaWorkingDir, H_rp1/1000)); % N.B.
       Convert to [kJ/mol]
33 % RP1Data = csvread(sprintf('%s.csv', fileName), 1, 1);
34 % T_c_RP1 = RP1Data(1,9) %[K]  ************ CEA OUTPUT (LOX/
       RP1) ************
35 % ************************************************
36
37
38
39
40 % The enthalpy value calculated, yields the following CEA
       output:
41 molarFracts = [0.40432 0.30694 0.27644 0.01221]'; %
       ************ CEA OUTPUT (LOX/RP1) ************
42              %[ 'CO'    'CO2'    'H2O'    'H2' ]
```

```
43 T_c_RP1 = 2420; %[K]  ************ CEA OUTPUT (LOX/RP1)
       ************
44
45
46
47
48
49 %% LOX/RP1 COMBUSTION PRODUCTS
50
51 % Shomate Equation for isobaric SPECIFIC HEAT
52 cp = @(v) v(1) + v(2)*T_c_RP1 + v(3)*T_c_RP1^2 + v(4)*T_c_RP1
       ^3 + v(5)/T_c_RP1^2;
53
54 % LSQ coefficients for CO (1300 to 6000 K) [Source: NIST]
55 v_co = [35.15070
56          1.300095e-03
57         -0.205921e-06
58          0.013550e-09
59         -3.282780e+06
60         -127.8375
61          231.7120
62         -110.5271];
63
64 Mm_co= 0.028;
65 cp_co = cp(v_co)/Mm_co;
66 R_co = Ru/Mm_co;
67 cv_co = cp_co - R_co;
68 k_co = cp_co/cv_co
69
70
71
72 % LSQ coefficients for CO2 (1200 to 6000 K) [Source: NIST]
73 v_co2 = [58.16639
74           2.720074e-03
75          -0.492289e-06
76           0.038844e-09
77          -6.447293e+06
78          -425.9186
79           263.6125
80          -393.5224];
81
82 Mm_co2= 0.044;
83 cp_co2 = cp(v_co2)/Mm_co2;
84 R_co2 = Ru/Mm_co2;
85 cv_co2 = cp_co2 - R_co2;
86 k_co2 = cp_co2/cv_co2
87
```

```
88
89
90  % LSQ coefficients for H2O (1700 to 6000 K) [Source: NIST]
91  v_h2o = [41.96426
92           8.622053e-03
93          -1.499780e-06
94           0.098119e-09
95          -11.15764e+06
96          -272.1797
97           219.7809
98          -241.8264];
99
100 Mm_h2o= 0.018;
101 cp_h2o = cp(v_h2o)/Mm_h2o;
102 R_h2o = Ru/Mm_h2o;
103 cv_h2o = cp_h2o - R_h2o;
104 k_h2o = cp_h2o/cv_h2o
105
106
107 % LSQ coefficients for H2 (1700 to 6000 K) [Source: NIST]
108 v_h2 = [43.413560
109         -4.293079e-03
110          1.272428e-06
111         -0.096876e-09
112         -20.533862e+06
113         -38.515158
114          162.081354
115          0.0];
116
117 Mm_h2= 0.002;
118 cp_h2 = cp(v_h2)/Mm_h2;
119 R_h2 = Ru/Mm_h2;
120 cv_h2 = cp_h2 - R_h2;
121 k_h2 = cp_h2/cv_h2
122
123
124
125
126 kProds = [k_co k_co2 k_h2o k_h2]'
127
128
129 % Let us add the fraction of neglected combustion products to
        the most abudant
130 % one, so that sum(molarFracts)=1
131 missingPerc = 1 - sum(molarFracts);
132 [maxFrac, posMax] = max(molarFracts);
133 molarFracts(posMax) = molarFracts(posMax) + missingPerc;
```

```
134
135 molarFracts
136 sum(molarFracts)
137
138
139
140 kRP1 = sum(molarFracts.*kProds); %Ratio of specific heats =
       c_p/c_v [-]
141                                   %(weighted average of
                                         combustion products)
```

Tanks & Injection

Listing C.6: *tanksAndInjection.m*

```
1
2
3 %% Performance Constraints:
4
5 m_ox_RP1 = 75200; % Oxidizer mass (RP1SS) from data-table [kg
      ]
6 m_f_RP1 = 32300;  % Fuel mass (RP1SS) from data-table     [kg
      ]
7 W_p_RP1 = 9.807 * (m_ox_RP1 + m_f_RP1); % Propellant weight (
      RP1SS)       [N]
8 I_sp_RP1 = 348;                        % Vacuum specific
      impulse (RP1SS) [s]
9
10 I_tot_RP1 = I_sp_RP1 * W_p_RP1; % Total impulse (RP1SS) [N*s]
11
12 % Let us set a 5 percent performance increase
13 I_tot_H2 = 1.05*I_tot_RP1; % Total impulse (LH2SS) [N*s]
14 t_b_H2 = I_tot_H2/F_vac % Burning time (LH2SS)     [s]
15
16
17 % Tanks
18 m_dot_ox = r/(1+r)*m_dot % Oxidizer mass flow rate [kg/s]
19 m_dot_f = 1/(1+r)*m_dot % Fuel mass flow rate      [kg/s]
20
21 V_dot_ox = m_dot_ox/rho_ox; % Oxidizer volume flow rate [m^3/
      s]
22 V_dot_f = m_dot_f/rho_f;    % Fuel volume flow rate     [m^3/
      s]
23
24
25
26
27 % Let us consider an overall 7% margin, since from Sutton:
```

```
%'For a nominal burning time t, a 1% residual propellant,
% and a 6% overall reserve factor, give a formula for the
%amount of fuel and oxidizer propellant required with
%constant propellant flow. Ignore stop and start transients,
%thrust vector control, and evaporation losses.'
m_ox = 1.07*m_dot_ox*t_b_H2
m_f =  1.07*m_dot_f*t_b_H2

V_ox = m_ox/rho_ox
V_f = m_f/rho_f

D_int = 5; % internal diameter of the cylidrical tank portion
    of LH2SS [m]
a = D_int / 2;
b = 0.6 * a;

% actual tank volumes
ullagePerc = 2.5;
V_oxT = V_ox * (1+ullagePerc/100)
V_fT = V_f * (1+ullagePerc/100)

L_f_cyl = (V_fT - 4*pi/3*a^2*b) / (pi*D_int^2/4)
L_f = L_f_cyl + 2*b

L_ox = 4*V_oxT / (pi*D_int^2)

L_tanks = L_f + L_ox

L_IS = 6.8 + x_3(end) - 3.2 % Estimated length of the LH2SS
   portion encasing the engine [m]

L_LH2SS = L_tanks + L_IS


%%
% Injection Holes

D_inj = 0.00318; % Orifice diameter (from Sutton Tab. 8.2) [m
   ]
C_D = 0.8; % Discharge coefficient (from Sutton Tab. 8.2)
   [-]
delta_p = 0.15*p_c; % Sutton: 'The pressure drop across the
   injector is usually
% set at values between 15 and 25% of the chamber pressure

A_tot_ox = m_dot_ox/(C_D*sqrt(2*rho_ox*delta_p))
A_tot_f = m_dot_f/(C_D*sqrt(2*rho_f*delta_p))
```

```
70
71 A_inj = pi*(D_inj^2)/4;
72
73 N_ox = A_tot_ox/A_inj
74 N_f = A_tot_f/A_inj
75 N_inj = floor(min(N_ox,N_f))
76
77 A_inj_ox = A_tot_ox/N_inj;
78 A_inj_f = A_tot_f/N_inj;
79
80 D_inj_ox = sqrt(4*A_inj_ox/pi)
81 D_inj_f = sqrt(4*A_inj_f/pi)
82
83 v_inj_ox = m_dot_ox/(rho_ox*A_tot_ox)
84 v_inj_f  = m_dot_f/(rho_f*A_tot_f)
85
86 alpha_ox = deg2rad(30); % We choose a 30 deg oxidizer
      injection angle
87 alpha_f = asin((m_dot_ox*v_inj_ox*sin(alpha_ox))/(m_dot_f*
      v_inj_f));
88 sprintf('alpha_ox=%.2f deg, alpha_f=%.2f deg', rad2deg(
      alpha_ox), rad2deg(alpha_f))
89
90
91
92
93 %% Engine Mass Prediction
94 % fileName = 'rocketEngines_LH2';
95 % data = csvread(sprintf('%s.csv', fileName), 1, 1); % data
      to plot
96 %
97 %
98 % colNum = length(data(1,:)) + 1;
99 % rowNum = length(data(:,1));
100 % % Header extraction
101 % fid = fopen(sprintf('%s.csv', fileName), 'rt');
102 % H = textscan(fid, '%s ', 'delimiter', ',', '
      MultipleDelimsAsOne', 1);
103 % H = H{1};
104 % H = H(2:colNum);
105 %
106 % featureSel = [1 2 3 5];
107 % p = 10;
108 % lambda = 3;
109 %
110 % X = data(:,featureSel);
111 % X = polyFeatures(X,p);
```

```
% y = data(:,end);
% X_F9 = [F_vac p_c*1e-5 epsilon I_sp t_b_H2];

% ****** TEST SET ******
% X_test = [180000, 60.8, 240, 465, 800]; % Vinci (Ariane 6): 560 kg
% N.B. Actually Safran declares a maximum allowable t_b of 900 s, -> margin
% of safety -> 800s (https://www.safran-group.com/media/140-successful-tests-and-several-firsts-vinci-engine-ariane-6-20180215)

%X_test = [843500 127 52 446 346]; % LE-7 (H-II (Japan)): 1,714kg
%X_test = [1140000 100 45.1 431 605]; % Vulcain HM-60 (Ariane 5): 1300 kg
% ****** TEST SET ******

%X_F9 = X_F9(featureSel);
%X_F9 = polyFeatures(X_F9,p);


%[massEng_LH2SS, theta] = predictMassFromDataset(X,y,lambda, X_F9,1);
%fprintf(['Predicted MASS of LH2SS Engine: %.1f kg   (vs. RP1SS: 490 kg)\n'], massEng_LH2SS);
% To be compared with the RP1SS mass of 490 kg




%% LH2SS Dry Mass Prediction
fileName = 'secondStages';
data = csvread(sprintf('%s.csv', fileName), 1, 1); % data to plot
%data(:,7) = log( 1 + data(:,7) - min(data(:,7)) )

colNum = length(data(1,:)) + 1;
rowNum = length(data(:,1));
% Header extraction
fid = fopen(sprintf('%s.csv', fileName), 'rt');
H = textscan(fid, '%s ', 'delimiter', ',', 'MultipleDelimsAsOne', 1);
H = H{1};
H = H(2:colNum);

featureSel = [1 3 4 5 6 7];
```

```
149 d = 8;
150
151
152 % ---------------------
153
154
155 X = data(:,featureSel);
156 X = polyFeatures(X,d);
157 y = data(:,end);
158 % ---------------------
159
160
161 % ********************* PREDICTION TARGET
    **********************
162 X_F9 = [(m_ox+m_f) I_tot_H2 I_sp 1 L_LH2SS D_int+.2 2015]; %
     Falcon 9 (LH2SS)
163 mass_RP1 = 4000;
164 X_F9 = X_F9(:,featureSel);
165 X_F9 = polyFeatures(X_F9,d);
166 % ********************* PREDICTION TARGET
    **********************
167
168
169 % ********************* VALIDATION set **********************
170 X_val1 = [21320, 93543, 462, 1, 12.00, 2.44, 2002];   %
    Delta IV
171 actualMass1 = 2850;
172
173 X_val2 = [158800, 712320, 465, 1, 16.00, 10.00, 2019];   %
    Ares
174 actualMass2 = 13200;
175
176 X_val3 = [22900, 187200, 448, 2, 12.00, 3.35, 2004];   %  CZ-
    NGLV-HO
177 actualMass3 = 3100;
178
179 X_val4 = [16780, 73128, 450, 1, 10.10, 3.05, 1998];    %
    Centaur C-X
180 actualMass4 = 2358;
181
182 X_val5 = [117241, 464400, 405, 1, 30.00, 4.00, 1968];  %
    Mustard 2
183 actualMass5 = 24943;
184
185 X_val = [X_val1; X_val2; X_val3; X_val4; X_val5];
186 y_val = [actualMass1 actualMass2 actualMass3 actualMass4
    actualMass5]';
```

```
187 X_val = X_val(:,featureSel);
188 X_val = polyFeatures(X_val,d);
189 % ******************** VALIDATION set ********************
190
191
192 [m_dry_LH2SS, theta] = predictMassFromDataset(X, y, X_val,
       y_val, X_F9, 1);
193 fprintf(['Predicted DRY MASS: %.1f kg   (vs. RP1SS: %.0f kg)\
       n'], m_dry_LH2SS, mass_RP1);
194
195
196
197
198
199
200 PL = 10866; % Payload mass [kg]
201 m_fair = 2000; % Fairing mass [kg]
202 m_wet_RP1SS = 112500;
203 m_dry_RP1SS = 4000;
204
205 DeltaV_RP1SS = I_sp_RP1*g0*log((m_wet_RP1SS+(m_fair+PL)) / (
       m_dry_RP1SS+(m_fair+PL)))
206
207 DeltaV_LH2SS = I_sp*g0*log(((m_ox+m_f+m_dry_LH2SS)+(m_fair+PL
       )) / (m_dry_LH2SS+(m_fair+PL)))
```

Listing C.7: *predictMassFromDataset.m*

```
1
2 function [prediction, theta] = predictMassFromDataset(X, y,
       X_val, y_val, x2Pred, convergencePlot)
3
4 m     = size(  X   ,1);
5 m_val = size(X_val,1);
6 n     = size(  X   ,2);
7
8 lambda_vec = [0.001*2.^[0:13]]';
9 alpha_vect = .01 * 1.1.^[1:20]';
10
11
12 if nargin <= 4
13     convergencePlot = 0;
14 else
15     convergencePlot = 1;
16 end
17
18 if nargin >= 4
19     computePrediction = 1;
```

```
20 else
21     computePrediction = 0;
22 end
23
24
25 %% ================ Feature Normalization ================
26
27 [X mu sigma] = featureNormalize(X);
28
29 % Add intercept term to X
30 X = [ones(m, 1) X];
31
32 % Normalize X_val using mu, sigma   and add ones
33 X_val = bsxfun(@minus, X_val, mu);
34 X_val = bsxfun(@rdivide, X_val, sigma);
35 X_val = [ones(m_val, 1), X_val];
36
37
38
39 %% ================ Validation Curve ================
40     clear alpha
41     alpha = alpha_vect(end)
42     [J_train, J_cv, lambda_opt] = validationCurve(X, y, X_val
          , y_val, lambda_vec, alpha);
43 %      figure
44 %      plot(lambda_vec, J_train, '-o', lambda_vec, J_cv, '-o')
   ;
45 %      legend('Train', 'Cross Validation');
46 %      xlabel('\lambda');
47 %      ylabel('J (Error)');
48 %
49       lambda = lambda_opt
50
51
52 %% ================ Gradient Descent ================
53
54 if convergencePlot
55     figure
56     hold on
57     grid on
58     xlabel('No. of iterations');
59     ylabel('Cost Function J');
60
61     myCMap = jet(length(alpha_vect));
62     set(gcf,'DefaultAxesColorOrder', myCMap);
63     colormap(myCMap);
64     hcb = colorbar;
```

```
65      caxis([min(alpha_vect) max(alpha_vect)])
66      set(get(hcb,'Title'),'String','\alpha');
67  end
68
69  for i = 1:length(alpha_vect)
70
71      alpha = alpha_vect(i);
72      itermax = 1000;
73
74      % Initialize theta before running gradient descent
75      theta = zeros(n+1, 1);
76      [theta, J_history] = gradientDescent(X, y, lambda, theta,
            alpha, itermax);
77
78
79      % Plot the convergence graph
80      if convergencePlot
81          plot(1:numel(J_history), J_history, 'LineWidth', 2, '
                Color', myCMap(i,:));
82          drawnow
83      end
84  end
85
86
87  if computePrediction
88      % Estimate the engine dry mass
89      x2pred = (x2Pred - mu)./sigma;
90      x2pred = [1 x2pred]';
91      prediction = theta'*x2pred;
92  else
93      prediction = nan;
94  end
95
96
97  %% Learning Curve
98  % learningCurve(X, y, X_val, y_val, lambda, alpha);
99
100
101 end
```

Listing C.8: *gradientDescent.m*

```
1   function [theta, J_history] = gradientDescent(X, y, lambda,
        theta, alpha, itermax)
2   %   Performs gradient descent to learn theta
3   %   theta = GRADIENTDESCENT(X, y, theta, alpha, num_iters)
        updates theta by
4   %   taking num_iters gradient steps with learning rate alpha
```

```
5
6
7  m = length(y); % number of training examples
8  J_history = zeros(itermax, 1);
9
10 for iter = 1:itermax
11
12     h = X*theta; % Prediction vector
13     theta = (1-alpha*lambda/m)*theta - (alpha/m) * X'*(h-y);
14
15     J = sum( (h-y).^2 ) / (2*m);
16
17     % Save the cost J in every iteration
18     J_history(iter) = J;
19
20 end
21
22 end
```

Listing C.9: *polyFeatures.m*

```
1  function [X_poly] = polyFeatures(X, d)
2
3  % d = degree of polynomial
4
5  X_poly = [];
6
7  for i = 1:d
8      X_poly = [X_poly X.^i];
9  end
10
11
12 n = size(X,2);
13 for i = 1:n-1
14     for j = i+1:n
15         X_poly = [X_poly X(:,i).*X(:,j)];
16     end
17 end
18
19
20
21 end
```

Listing C.10: *featureNormalize.m*

```
1  function [X_norm, mu, sigma] = featureNormalize(X)
2  %       Normalizes the features in X
3  %   FEATURENORMALIZE(X) returns a normalized version of X
```

```
   where
%   the mean value of each feature is 0 and the standard
   deviation
%   is 1.


mu = mean(X,1); % mean by column
sigma = std(X,1); % std by column

X_norm = (X-mu)./sigma;




end
```

Listing C.11: *linearRegCostFunction.m*

```
function [J, grad] = linearRegCostFunction(X, y, theta,
   lambda)

%   [J, grad] = LINEARREGCOSTFUNCTION(X, y, theta, lambda)
   computes the
%   cost of using theta as the parameter for linear
   regression to fit the
%   data points in X and y

m = length(y); % number of training examples


h = X*theta;
J = 1/(2*m) * (  sum((h-y).^2) + lambda*sum(theta(2:end).^2)
   );
grad = (1/m) * (  X'*(h-y) + lambda*[0; theta(2:end)] );



end
```

Listing C.12: *learningCurve.m*

```
function [] = learningCurve(X, y, X_cv, y_cv, lambda, alpha)
%LEARNINGCURVE Generates the train and cross validation set
   errors needed
%to plot a learning curve
%

% Number of training examples
m = size(X, 1);
```

```
n = size(X, 2);


% Error vectors
J_train_vec = zeros(m, 1);
J_cv_vec   = zeros(m, 1);

for i = 1:m

    % Let us consider the training subset
    X_train = X(1:i, :);        y_train = y(1:i);
    itermax = 200;
    theta0 = zeros(n, 1);
    [theta, ~] = gradientDescent(X, y, lambda, theta0, alpha,
        itermax);
    % Setting lambda = 0, so that we can reuse the function:
    [J_train, ~] = linearRegCostFunction(X_train, y_train,
       theta, 0);
    [J_cv, ~] = linearRegCostFunction(X_cv, y_cv, theta, 0);


    J_train_vec(i) = J_train;
    J_cv_vec(i)    = J_cv;
end


figure
plot([1:m], J_train_vec, '-o', [1:m], J_cv_vec, '-o');
legend('Train', 'Cross Validation');
xlabel('m');
ylabel('J (Error)');

end
```

Listing C.13: *validationCurve.m*

```
function [J_train_vec, J_cv_vec, lambda_opt] = ...
    validationCurve(X, y, X_cv, y_cv, lambda_vec, alpha)
%VALIDATIONCURVE Generate the train and validation errors
   needed to
%plot a validation curve that we can use to select lambda


m = size(X,1);
m_val = size(X_cv,1);
n     = size(  X  ,2);


```

```
for i = 1:length(lambda_vec)

    itermax = 200;
    lambda = lambda_vec(i);
    theta0 = zeros(n, 1);
    [theta, ~] = gradientDescent(X, y, lambda, theta0, alpha,
        itermax);


    % Setting lambda = 0 we can reuse the function:
    [J_train, ~] = linearRegCostFunction(X, y, theta, 0);
    [J_cv, ~] = linearRegCostFunction(X_cv, y_cv, theta, 0);
    %   N.B. J is NOT regularized, since we are using it to
       evaluate the
    %   misclassification error

    J_train_vec(i) = J_train;
    J_cv_vec(i)    = J_cv;

end

    J_train_vec = J_train_vec(:);
    J_cv_vec = J_cv_vec(:);

    [~, opt_index] = min(J_cv_vec);
    lambda_opt = lambda_vec(opt_index);
    lambda_opt = lambda_opt(end);


end
```

C.1.2 MatlabCEA

The following script works in conjunction with C.15.

Listing C.14: *ceaRun*

```
close all
clear all

ceaPath = '~/CEA/'; % SET YOUR WORKING DIRECTORY
fileName = 'myInp'; % SET YOUR INPUT FILENAME

set(0,'defaultTextInterpreter','tex')


unix(sprintf('python %sceaCall.py', ceaPath));

```

```
12
13 A = csvread(sprintf('%s.csv', fileName), 1, 1) % data to plot
14
15
16 pcn = 1; % Variable parameter column number
17
18 colNum = length(A(1,:));
19 rowNum = length(A(:,1));
20 O_F = A([3:3:rowNum],pcn) % Variable parameter
21 % Header extraction
22 fid = fopen(sprintf('%s.csv', fileName), 'rt');
23 H = textscan(fid, '%s ', 'delimiter', ',', '
      MultipleDelimsAsOne', 1);
24 H = H{1};
25 H = H(1:colNum)
26 paramName = H(pcn)
27
28
29     if (pcn == 1)
30         A = A(:,2:end)
31         H = H(2:end)
32
33     elseif (pcn == colNum)
34         A = A(:,1:end-1)
35         H = H(1:end-1)
36     else
37         A = A(:,[1:pcn-1, pcn+1:end])
38         H = H([1:pcn-1, pcn+1:end])
39     end
40
41
42
43
44
45
46 %% PLOT 'Combustion Chamber', 'Throat' ans 'Exhaust' data
47
48 sectNames = ["Combustion Chamber", "Throat", "Exhaust"] %
      Note: string arrays require double quotes
49 n = length(sectNames);
50
51 for k = 1:n
52
53     A_sect = [];
54     for i = k:n:rowNum-n+k
55         A_sect = [A_sect; A(i,:)];
56     end
```

```
57
58      A_sect
59
60      sectName = sectNames(k);
61      fig = figure('Name', sectName);
62      title('Nozzle')
63      subPlRows = 3;
64      subPlNum = length(H);
65      subPlIndx = 1;
66      for j = 1:subPlNum
67          Y = A_sect(:,j); % The considered quantity to plot
68          if ( min(Y)~= max(Y) ) % Do not plot constant
               quantities
69              subplot(subPlRows,ceil(subPlNum/subPlRows),
                   subPlIndx)
70              plot(O_F, Y, '-o')
71              xlabel(paramName)
72              ylabel(H(j))
73              grid on
74              subPlIndx = subPlIndx+1;
75          end
76      end
77
78
79      run('resizePDFPaper.m');
80      print(fig, sprintf('H2_perf_OF_%d', k), '-dpdf');
81
82
83  end
```

C.2 Python Code

Listing C.15: *ceaCall.py*

```
import subprocess
import time



def subprocess_cmd(command):
    process = subprocess.Popen(command, stdout=subprocess.
       PIPE, shell=True)
    proc_stdout = process.communicate()[0].strip()
    print proc_stdout



inpName = 'myInp'
workingDir = '/Users/massimopiazza/CEA'
inpDir = workingDir + '/' + inpName


import os
if os.path.isfile(inpDir + '.plt'):
    os.remove(inpDir + '.plt')
if os.path.isfile(inpDir + '.csv'):
    os.remove(inpDir + '.csv')
# print file list
path = '.'  # Change this as you need.
abspaths = []
for fn in os.listdir(path):
    abspaths.append(os.path.abspath(os.path.join(path, fn)))
print("\n".join(abspaths))



subprocess_cmd('cd ' + workingDir + '; printf "' + inpName +
   '" | ./FCEA2')
#e.g. on  my computer: subprocess_cmd('cd /Users/
   massimopiazza/CEA; printf "myInp" | ./FCEA2')


#time.sleep(.5)

def vspace(vertSpace):
    print('\n'*vertSpace)
vspace(1)
```

```
41
42 # Print .out file
43 outFile = open(inpDir + '.out', 'r')
44 outFile_contents = outFile.read()
45 print (outFile_contents)
46 outFile.close()
47
48
49
50 # Print .plt file
51 pltFile = open(inpDir + '.plt')
52 pltFile_contents = pltFile.read()
53 print (pltFile_contents)
54 pltFile.close()
55
56
57
58 # Parse .plt file
59 pltFile = open(inpDir + '.plt')
60
61 lines = pltFile.readlines()
62 result = []
63 headers = []
64 counter = 0;
65
66
67 import re
68
69
70 # Create 2D list from .plt file
71 for x in lines:
72     counter = counter + 1;
73     if (counter == 1):
74         headers = re.split(' +' ,x)[1:-1] # ' +' is added in
              order to treat consecutive delimiters as one, in
              addition the range is 1:-1 in order to exclude
              both the elements # and \n
75         ispCol = headers.index('isp') # Let us identify it in
               order to later convert I_sp into seconds
76         ivacCol = headers.index('ivac')
77         headers = [w.replace('p', 'p [bar]').replace('mw', '
              M_{m} [g/mol]').replace('mach', 'Mach') for w in
              headers]
78         headers = [w.replace('o/f', 'O/F').replace('%f', 'f')
              .replace('t', 'T [K]').replace('isp', 'I_{sp} [s]'
              ).replace('ivac', 'I_{sp}^{(vac)} [s]').replace('
              cf', 'C_{F}').replace('gaM', '\gamma')  for w in
```

```
            headers]
79          headers = [w.replace('gam', '\gamma') for w in
               headers]
80          headers = [w.replace('I_{sp} [s] [bar]', 'I_{sp} [s]'
               ) for w in headers]
81      elif (counter > 1) & (counter < len(lines)):
82          result.append(x.split('  ')[1:])  # from col 1 to end
83
84  # Convert string-elements to float-elements
85  def myFloat(myList):
86      return map(float, myList)
87  []
88  result = map(myFloat, result)
89  print result
90
91  # Convert list to NumPy array
92  import numpy as np
93  data = np.asarray(result, dtype=np.float64)
94  print data
95  for row in data:
96      row[ispCol] = row[ispCol] / 9.807
97      row[ivacCol] = row[ivacCol] / 9.807
98
99  # Convert NumPy array to .csv file
100 import pandas as pd
101 df = pd.DataFrame(data)
102 df.to_csv(inpDir + '.csv', header=headers)
103
104 print headers
105
106
107
108 pltFile.close()
```

C.3 CEA Input

Listing C.16: *H2.inp*

```
1
2       prob
3 case=LOX/LH2 o/f=1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,
4 5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10.0,10.5,11.0,11.5,12.0
5     rocket  equilibrium  tcest,k=2500
6 p,bar=110
7 sup,ae/at=195
8       reac
9 oxid=O2(L) t,k=90
10 fuel=H2(L) t,k=20
11
12      outp
13 massf short plot o/f isp ivac cf mach gam mw p t %f
14
15      end
```

Listing C.17: *RP-1.inp*

```
1 problem   case=LOX/RP-1 o/f=2.33
2     rocket  equilibrium  tcest,k=2500
3   p,bar=110,
4   sup,ae/at=165,
5 react
6   oxid=O2(L) t,k=90
7   fuel=RP-1sc  t,k=266
8     h,kj/mol=-2253.736   C 12 H 24
9
10    outp
11 massf short plot o/f isp ivac cf mach gam mw p t %f
12      end
```

Listing C.18: *myInp.inp*

```
1
2       prob
3 case=LOX/LH2 o/f=1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0,
4 5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10.0,10.5,11.0,11.5,12.0
5     rocket  equilibrium  tcest,k=2500
6 p,bar=110
7 sup,ae/at=195
8       reac
9 oxid=O2(L) t,k=90
10 fuel=H2(L) t,k=20
11
```

```
12      outp
13 massf short plot o/f isp ivac cf mach gam mw p t %f
14
15      end
```

Bibliography

[101] Spaceflight 101. *http://spaceflight101.com/spacerockets/falcon-9-ft/*.

[Ank18] Ankit Rohatgi. *WebPlotDigitizer*. Version 4.1. Jan. 8, 2018. URL: https://automeris.io/WebPlotDigitizer/.

[Ast] Astronautix. *http://www.astronautix.com*.

[Cha98] MW Chase Jr. "NIST-JANAF thermodynamical tables". In: *J. Phys. Chem. Reference, Monograph* 9 (1998).

[Din08] Aaron Dinardi, Peter Capozzoli, and Gwynne Shotwell. "Low-cost launch opportunities provided By the falcon family of launch vehicles". In: *The Fourth Asian Space Conference*. 2008.

[Dum17] Etienne Dumont et al. "Evaluation of Future Ariane Reusable VTOL Booster stages". In: *68th International Astronautical Congress (IAC), Adelaide, Australia*. 2017.

[Eck17] Tobias Ecker et al. "A Numerical Study on the Thermal Loads During a Supersonic Rocket Retro-Propulsion Maneuver". In: *53rd AIAA/SAE/ASEE Joint Propulsion Conference*. 2017, p. 4878.

[Huz92] Dieter K Huzel. *Modern engineering for design of liquid-propellant rocket engines*. Vol. 147. AIAA, 1992.

[Lar92] Wiley J Larson and James Richard Wertz. *Space mission analysis and design*. Tech. rep. Microcosm, Inc., Torrance, CA (US), 1992.

[Lar95] Wiley J Larson, Gary N Henry, and Ronald W Humble. *Space propulsion analysis and design*. McGraw-Hill, 1995.

[Rao58] G V R Rao. "Exhaust nozzle contour for optimum thrust". In: *Journal of Jet Propulsion* 28.6 (1958), pp. 377–382.

[Rao61] G V R Rao. "Recent developments in rocket nozzle configurations". In: *ARS journal* 31.11 (1961), pp. 1488–1494.

[Rep] Space Launch Report. *http://www.spacelaunchreport.com/falcon9ft.html*.

[Smi07] Joshua John Smith. "High pressure LOX/H_2 rocket engine combustion." PhD thesis. 2007.

[Spa15] SpaceX. *Falcon 9 Launch Vehicle: Payload User's Guide*. Rev. 2. Oct. 2015.

[Sta] NIST: National Institute of Standards and Technology. *http://webbook.nist.gov/chemistry/form-ser/*.

[Sut16] George P Sutton and Oscar Biblarz. *Rocket propulsion elements*. John Wiley & Sons, 2016.

[Wan96] Ten-See Wang. "Thermo-kinetics characterization of kerosene/RP-1 combustion". In: *32nd Joint Propulsion Conference and Exhibit*. 1996, p. 2887.

[Wax17] Günther Waxenegger et al. "Implications of Cycle Variants, Propellant Combinations and Operating Regimes on Fatigue Life Expectancies of Liquid Rocket Engines". In: *7th European Conference for Aeronautics and Space Sciences, Milan, Italy*. 2017, pp. 3–6.