Micropropulsion

AE4S07

Full analysis and optimisation of the operational envelope of the Delfi-PQ micro-propulsion demonstrator

Project #7

by

Kathleen Blyth

(4446526)

Contents

Nomenclature

Symbols

$\dot{m}$	Mass flow	$[kg/s]$
c_p	Specific heat	$[J/K/kg]$
l	Length	$[m]$
M	Mass	$[kg]$
p	Pressure	$[Pa]$
t	Time	$[s]$
A_t	Throat area	$[m^2]$
D	Diameter	$[m]$
F_t	Thrust	$[N]$
I	Total impulse	$[Ns]$
I_{sp}	Specific impulse	$[s]$
L_h	Latent heat of vaporisation	$[J/kg]$
P	Power	$[W]$
R	Specific gas constant	$[J/K/kg]$
T	Temperature	$[K]$
V	Nitrogen volume	$[m^3]$
η	Efficiency	$[-]$
γ	Ratio of specific heats	$[-]$
Γ	Vandenkerckhove function	$[-]$
ρ	Density	$[kg/m^3]$
ξ	Quality factor	$[-]$

Physical constants

g_0	Gravitational acceleration	$9.81\,\mathrm{m/s^2}$
M_w	Water molecular mass	$18.0153\,\mathrm{kg/mol}$
R_A	Gas constant	$8.314\,\mathrm{J/K/mol}$

Subscripts

0	Initial value
1	Reference value
c	Chamber conditions
g	Gaseous
l	Liquid
$exit$	Exit conditions
vap	Vaporisation conditions
LPM	Low Pressure Micro-resitojet
VLM	Vaporising Liquid Micro-resitojet

1 Introduction

This aim of this paper is to perform a more in depth analysis of the dual micro-resitojet demonstrator payload currently being developed by the TU Delft. This aim is to validate this payload in-flight on board the Delfi-PQ satellite.

This demonstrator consists of two thrusters; a Vaporising Liquid Micro-resitojet (VLM) and a Low-Pressure Micro-resitojet (LPM). The two resitojets use liquid water as a propellant, which they heat by electrical resistance in the thruster chamber. The propellant for both thrusters is stored in the same tube, and pressurised by nitrogen. The VLM thruster is fired first, and involves heating water to above its vaporisation temperature, and then expelling it through a convergent divergent nozzle. When the pressure in the propellant tube has been significantly reduced, the LPM thruster is fired making use of the remainder of the propellant and pressurant.

There currently exists a preliminary analysis of the operational envelope. In this analysis, equations from ideal rocket theory were used to determine the performance of the thrusters, and primarily focuses on the performance of the VLM. As will be discussed in chapter 3, the model considers two possibilities; either the thruster maintains a constant chamber temperature, or the thrusters operates at the varying vaporisation temperature.

This report will start by first looking at the required inputs for determining the operational envelope. Then the current model will be fully analysed in chapter 3, providing an overview of the approach as well as a discussion to its validity. Finally, an updated model of the operational envelope will be presented in chapter 4. This will build upon the work of the previous model, with the aim of presenting a more accurate and optimised solution.

2 Model input data

In this chapter, the required inputs for the operations envelope will be analysed, and acceptable ranges determined based on the requirements for the Delfi-PQ. This is to answer the following research question:

What are the input data required to fully characterise the operational envelope, and what are suitable ranges of values for them?

2.1 Delfi-PQ requirements

There are a number of requirements listed for the design of the Delfi-PQ [1]. Out of these, there are five which have the largest influence on the design of the operational envelope. These will form the constraints of the optimisation problem, as well as fix the input parameters for the propellant.

PROP-PERF-200 The thrust provided by the propulsion system shall be 3 mN as a maximum.

PROP-PERF-210 The thrust provided by the propulsion system shall be at least 0.12 mN

PROP-FUN-100 The micro-propulsion system shall have at least two modes:idle mode with a maximum power consumption of 15 mW and a full thrust mode with a maximum power consumption of 4 W.

PROP-FUN-200 The thruster shall be able to operate on gaseous N_2 as well as on liquid H_2O

PROP-RAMS-200 The internal pressure of all propulsion system components shall not be higher than 10 bar.

Based on these requirements, the key limits of the operational envelope can be defined as follows:

- Max power of 4W
- Minimum thrust of 0.12 mN at the end of the VLM thrust

In addition, one more limit must be set. This is for the amount of propellant remaining after the VLM thrust, to ensure there is sufficient for the LPM firing. The requirements do not specify how much this should be, but the previous study [2] takes the following limit, which shall be used for the purpose of this report:

- A minimum of 0.2 g of propellant remaining for the LMP thrust

2.2 Operational envelope output data

The selection of the operational envelope input parameters will be done by analysing the data output by the model. It will focus on the following parameters: the remaining propellant mass $M_p(t)$, the nitrogen pressure $p(t)$, the exit pressure $p_e(t)$, the total thrust time t_{burn}, the chamber temperature $T_c(t)$, the input power $P(t)$, and the thrust $F_t(t)$. The operational envelope will look at these parameters across the entire burn profile of the VLM thruster.

2.3 Input data

A number of the inputs for the model are already fixed by the previously selected hardware, as well as the conditions on board the satellite. These characteristics are summarised in a table below.

Table 2.1: Pre-fixed inputs [2]

Variable		**Value**	**Units**
Nozzle throat area	A_t	4.5×10^{-9}	m^2
Tubing length	l	0.3	m
Tubing inner diameter	d	1.57×10^{-3}	m
Ambient temperature	T_0	283	K
Propellant	–	$H_2O(l)$	–

In order to analyse the operational envelope, a number of characteristics of the propellant must be known. Based on the selection of liquid water as the propellant, Table 2.2 can be formed, which contains constants which will be used as inputs for the model. It should be noted there are slight variations in these input variables given in [2] due to here being considered at the ambient temperature of 283 K.

Table 2.2: Water properties at 283 K[3][4]

Variable		**Value**	**Units**
Molar mass	M_w	18.0153×10^{-3}	kg/mol
Heat of vaporisation	L_h	2256×10^3	J/kg
Specific heat (liquid)	C_{p_l}	4187	J/K/kg
Vanderkerckhoven function	Γ	0.6712	–
Specific gas constant	R	461.5	J/K/kg
Density	ρ	999.7	kg/m^3
Ideal specific impulse @ 550 K	I_{sp}	135.4	s
Real specific impulse @ 550 K	I_{sp}	94.9	s

As will be discussed in the next chapter, the original model has three design inputs. These were the initial pressure, the initial volume of pressurant, and the chamber temperature. By assuming that the chamber temperature is always equal to the propellant vaporisation temperature, the temperature becomes a function of the initial pressure, and the model can be reduced to the two design variables given in Table 2.3.

Table 2.3: Operational envelope deign variables

Symbol	**Parameter**
$V(0)$	Initial volume of nitrogen
$p(0)$	Initial pressure of nitrogen

2.4 Operation envelope matrix

A matrix can be designed which highlight the relationships between the inputs and the outputs in the determination of the operational envelope. It continuously determines the relationships shown in Figure 2.1 until one of the limits in section 2.1 is met.

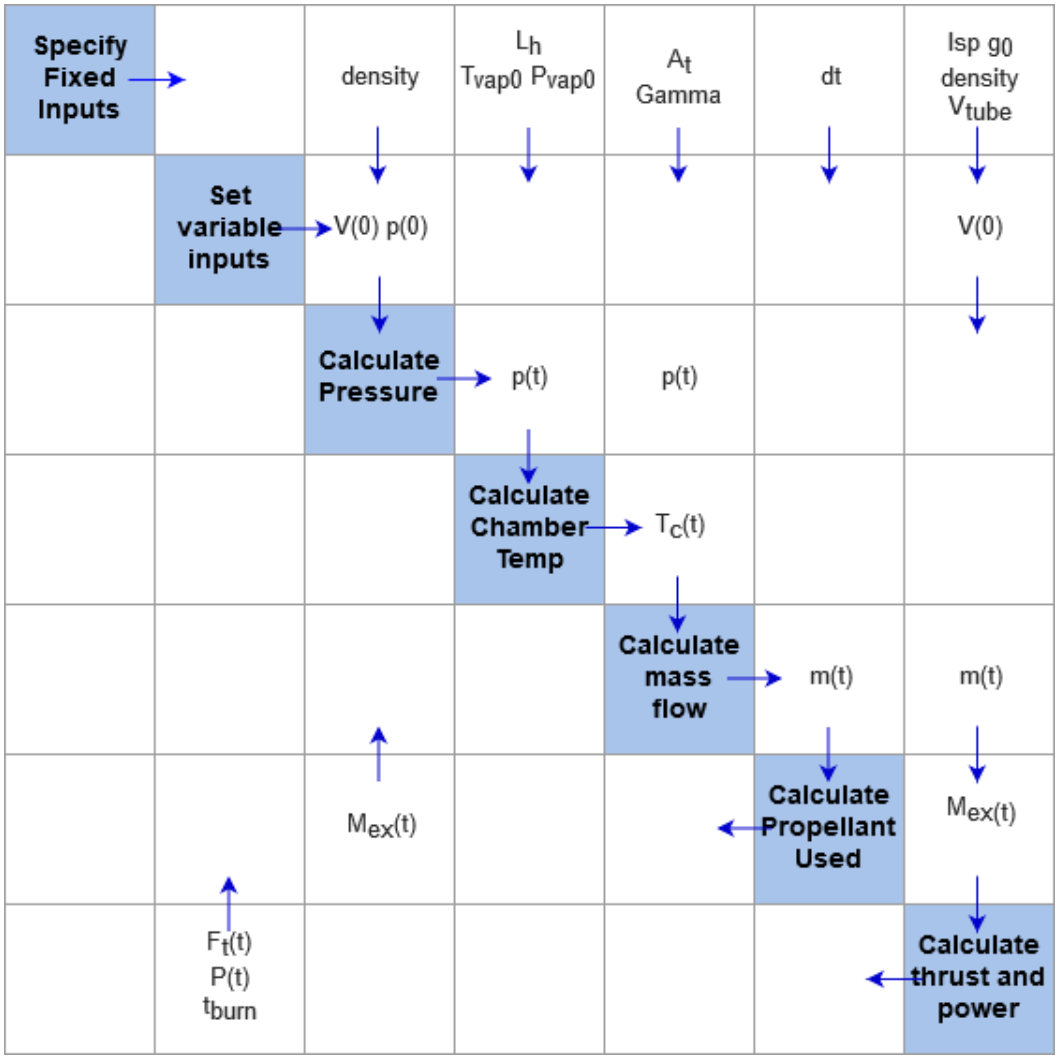


Figure 2.1: Operational envelope design matrix

2.5 Acceptable ranges

Based on the key limiting requirements, it is possible to determine the acceptable ranges for the design input variables. As will be seen these acceptable ranges are dependant upon the heating efficiency of the thruster, as well as the model selected. However, this section shall still provide a good indication of the ranges which can be used.

As will be described in more depth in section 4.3, the maximum power usage occurs at the beginning of the thrust. If the temperature in the chamber is assumed to be equal to the vaporisation temperature of the propellant, the initial input power becomes a function of only the initial system pressure. As the power usage must not exceed 4 W, the maximum initial pressure can be found to be 2.075 bar when assuming perfect heating efficiency. In reality, the heating efficiency will be lower than 100% which means the maximum acceptable initial pressure will also be lower.

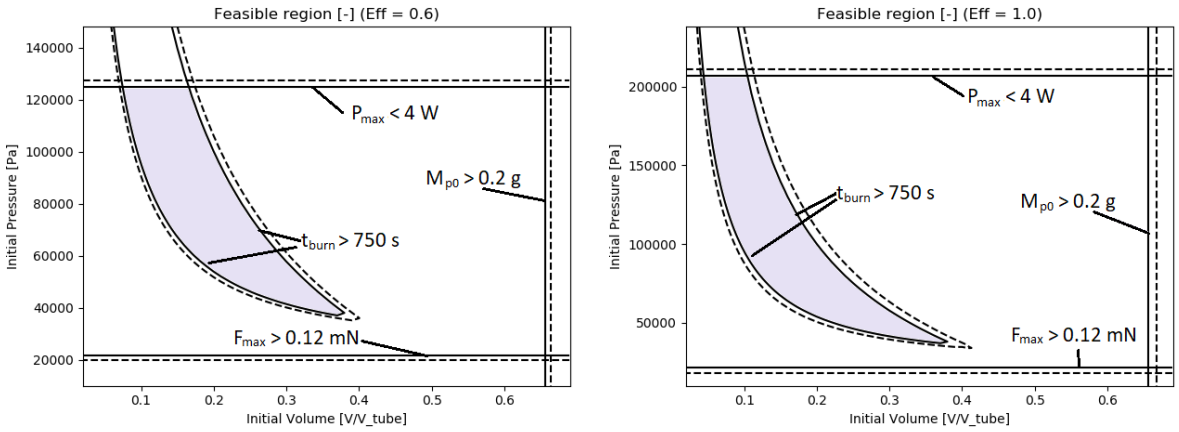
A minimum pressure limit is also required, in order to ensure the minimum thrust requirements

are met, and that the VLM has a reasonable burn time to ensure sufficient data can be collected. If only the minimum thrust requirement is considered, to have an initial thrust greater than 0.12 mN an initial pressure of 0.2 bar is needed.

As can be seen from section 2.1, there is no requirement on a minimum burn time for the VLM thruster. However, for the design to be feasible, it needs to be able to fire for long enough to record significant data. However, "sufficient" is a vague requirement, and therefore at a later stage a clear number should be defined. For the purpose of this report, a semi-arbitrary value of 750 s shall be taken for the minimum burn time, a value which is a little over half the burn time determined by the previous study [2].

Finally, the acceptable range of the initial volume needs to be looked at. This will be limited by the required burn time and the minimum amount of propellant required for the LPM burn. Considering only the latter, for the total initial propellant to be greater than 0.2 g, the initial pressurant volume of 66%.

All of the values above are based on the extreme limits, and don't feature the influence of the heating efficiency. As such contour plots can be generated to highlight the set of feasible solutions to the problem, including the effects of a minimum burn time. This region is highlighted in Figure 2.2, for both a heating efficiency of 60% and 100%. From this it can be seen that the solutions are only limited by the minimum burn time and the maximum heating power.



(a) $\eta = 60$ % (b) $\eta = 100$ %

Figure 2.2: Feasible problem solutions

As such, suitable ranges for the initial pressure and initial volume are summarised in Table 2.4 below. Not all values in these ranges meet the criteria, but it reduces the number of points considered speeding up computation time.

Table 2.4: Acceptable ranges for the design variables

Heating efficiency		**60 %**		**100 %**		
		Min	**Max**	**Min**	**Max**	**Units**
Initial pressure	$p(0)$	0.4	1.25	0.4	2.07	bar
Initial N2 volume	$V(0)$	7.5	38	4	38	% V_{tube}

3 Current operational envelope model

As has been previously mentioned, there exists a first draft of the operational envelope for the thrusters, which is based on ideal, simplified equations [2]. Through out this chapter, this model will be explored and the following research question shall be answered:

Is the current draft operational envelope of the micro-propulsion demonstrator payload the best one in terms of simplicity, completeness and effectiveness of the in-orbit demonstration and, if not, which changes would need to be made to it?

3.1 Overview of the current model

The current draft operational envelope focuses primarily on the behaviour of the VLM thruster. In it two approaches are considered, one in which the chamber temperature is fixed at 600 K, and one in which the temperature is equal to the water vaporisation temperature which changes with pressure over time. Of the two, the variable chamber temperature model was selected to be used, as while a constant temperature of 600 K provided slightly more thrust, for an almost identical power curve the lower temperatures of the second model were deemed to be better for the hardware in the system.

Both of these models used equations 3.1, 3.2 and 3.4 to determine the operational envelope, which are taken from ideal rocket theory. Equation 3.3 was only used for the second model, as it determines the vaporisation temperature of water for a given pressure

$$p(t) = \frac{V(0)}{V(0) + \frac{m_{exit}(t)}{\rho}} \cdot p(0) \tag{3.1}$$

$$\dot{m} = \frac{p(t) \cdot A_t \cdot \Gamma}{\sqrt{R \cdot T_C}} \tag{3.2}$$

$$T_c = \frac{T_1 \cdot L_h}{T_1 \cdot R \cdot \ln\left(\frac{p_1}{p(t)}\right) + L_h} \tag{3.3}$$

$$m_{exit}(i) = m_{exit}(i-1) + \dot{m} \cdot \Delta t \tag{3.4}$$

Using the matlab code provided at the end of the report[2], a pseudo version of the setup is provided in Algorithm 3.1 to show how the second model was set up. It highlights how the initial pressure, initial pressurant volume, and the thrust time of the VLM are all manual inputs, from which the operational envelope is calculated over the pre-determined time.

Algorithm 3.1 Variable Temperature Psuedocode

$time = 0 : \Delta t : 1200$
$p(1) = p_0$
calculate $T_c(1)$
calculate $\dot{m}(1)$
$m_{exit}(1) = 0$

for i = 2 : length($time$)
    $p(i) = V_0 \cdot p_0/(V_0 + m_{exit}(i-1)/rho)$
    $\dot{m}(i) = p(i-1) \cdot A_t \cdot Gamma/sqrt(R \cdot T_c(i-1))$
    $T_c(i) = L_h \cdot T_1/(T_1 \cdot R \cdot \ln(p_1/p(i-1)) + L_h)$
    $m_{exit}(i) = m_{exit}(i-1) + \dot{m}(i-1) \cdot \Delta t$

From these, the thrust, input power, and amount of propellant remaining in the tubing can all be found as a function of time.

$$V(t) = V(0) \cdot \frac{p(0)}{p(t)} \tag{3.5}$$

$$m_{tube}(t) = \frac{V_{tube} - V(t)}{\rho} \tag{3.6}$$

$$F_t = \dot{m} \cdot I_{sp} \cdot g_0 \tag{3.7}$$

$$P_{input} = \dot{m} \cdot \left[c_{p_l} \cdot (T_{vap} - T_0) + L_h + c_{p_g} \cdot (T_c - T_{vap})\right] \tag{3.8}$$

This power equation does not account for the heating efficiency. However it can be rewritten to include this efficiency and to account for the fact that the propellant is only heated to the minimum vaporisation temperature, as follows:

$$P_{input} = \frac{\dot{m}}{\eta_h} \cdot [c_{p_l} \cdot (T_{vap} - T_0) + L_h] \tag{3.9}$$

The current design arbitrarily assumed the heating efficiency to be 60 %. Based on this, the main inputs being the thrust time of the VLM, the initial pressure, and the initial percentage of the tube which is pressurant, were varied until certain criteria were met. These were, that the input power remained below 4 W, the thrust of the VLM must exceed 0.12 mN at all points, and that at the end of the VLM testing phase, there must still be 0.2 g of propellant remaining for the LPM testing. The selected parameters were:

Table 3.1: Final inputs for the operational envelope

Variable		**Value**	**Unit**
Initial pressure	p_0	1.1×10^5	Pa
Initial N_2 volume	V_0	$0.12 \cdot V_{tube}$	m^3
VLM thrust time	t_{VLM}	1200	s

3.2 Analysis of the current model

While the current draft of the operational envelope provides a good initial analysis, it is not without its faults. It appears to be inefficiently optimised, and looks at only a small combination of possible initial values.

One of the large issues with the current code is that the burn time is an input parameter. The result of this is that the thrust at the at the switch over between the VLM and the LPM is only 0.114 mN, which is smaller than the required 0.12 mN. This occurs as the burn time was set only focusing on the requirement for there to be at least 0.2 g of propellant left for the LPM.

This is also without accounting for any quality factors, which would likely result in the in the expected thrust being even lower than this value. The maximum expected thrust is 0.58 mN, occurring at the very beginning of the testing.

3.2.1 Quality factors

As was previously mentioned, this first draft is modelled using equations from ideal rocket theory. This ideal performance only has limited validity in comparison to the real performance, but can be made more realistic with the use of quality or correction factors [5].

$$\xi_F = \frac{F_{real}}{F_{ideal}} = C_d \cdot \xi_s \qquad C_d = \frac{\dot{m}_{real}}{\dot{m}_{ideal}} \qquad \xi_s = \frac{(I_{sp})_{real}}{(I_{sp})_{ideal}} \tag{3.10}$$

While the current draft did not directly state that it included quality factors in the model, in some cases real values were used instead of ideal, making the analysis more valid. The reference I_{sp} was taken to be 94.9 at 550 K [2], however this is the real I_{sp} found via testing the nozzle, while the ideal I_{sp} is 135.4 s [4]. This means $\xi_s = 70$ %. However, it should be noted that this was for a chamber pressure of 5 bar, and the current draft operates at around 1 bar. It can be seen that the pressure does have an effect on this model, as at 7 bar the quality factor $\xi_s = 73.1$ % [4].

At high Reynolds numbers, the discharge coefficient (C_d) can be said to be equal to one, making $\xi_F = \xi_s$ [5]. However, due to the very small throat size of the nozzle, the throat Reynolds number is very low, so this assumption cannot be made [6].

When not accounting for the efficiency of the heating, the maximum required input power is 2.1 W, and when accounting for an efficiency of only 60%, a maximum input power of 3.5 W is found. The requirement specified an input power of less than 4 W, so the criteria is met, but the 60% is an arbitrary value.

3.2.2 Comparing the two model approaches

In the previously done work [2], there was some graphs comparing the a fixed temperature of 600 K and the variable temperature scenario. The graphs considered the mass flow of the propellant and power transferred to the propellant over time. From the graphs, it was seen that there was little difference between the two scenarios, and the variable temperature was selected due to the lower temperature.

However, what was not considered, was a lower fixed temperature, more comparable to the variable temperature. As such, the original code was used to produce more comparison graphs, shown in

Figure 3.1 and Figure 3.2 below. In these a third scenario of a fixed temperature of 373 K was considered (the vaporisation temperature at 1.1 bar [3]), and the outputs were different being: the VLM thrust, input power and thrust to power ration over time.

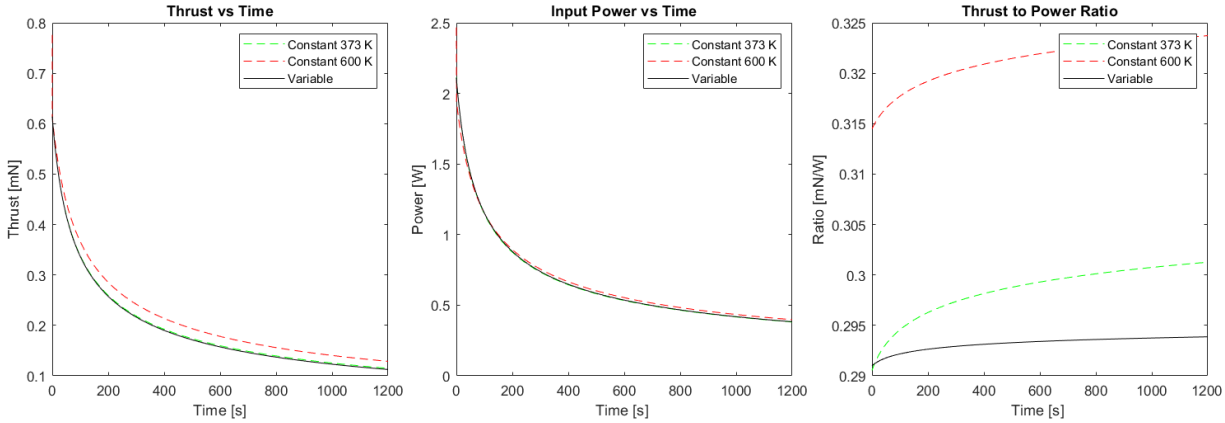


Figure 3.1: Graphs comparing model approaches ($p(0) = 1.1$ bar $V(0) = 0.12V_{tube}$, $\eta = 1.0$

It can be seen that for a heating efficiency of both 60% and 100%, the scenario in which the temperature is fixed at 600 K results in the highest thrust to power ratio. When the temperature is fixed at 373 K the thrust to power ratio power ratio is marginally higher than the variable case, but as can be seen from the thrust graph and the power graph, the difference is so small the option picked doesn't matter. This is due to the two scenarios with constant thrust resulting in a marginally higher thrust

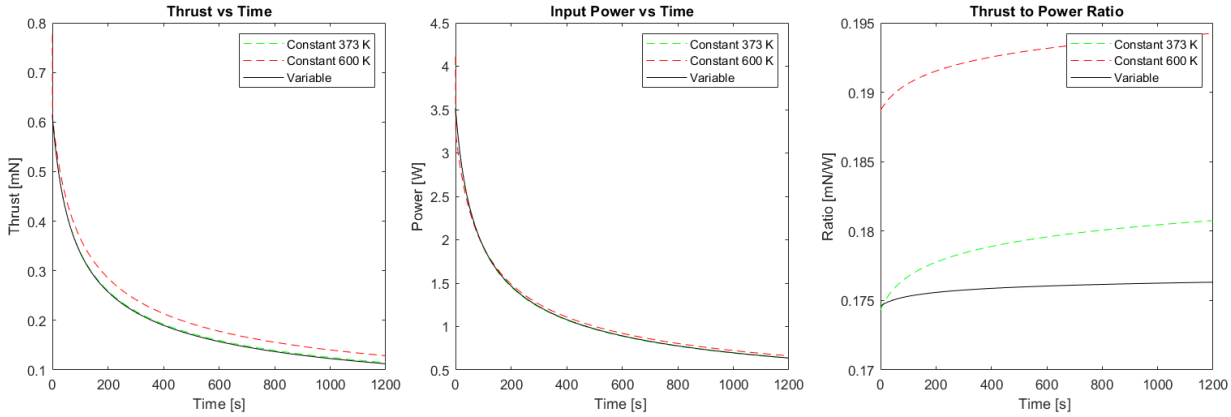


Figure 3.2: Graphs comparing model approaches ($p(0) = 1.1$ bar $V(0) = 0.12V_{tube}$, $\eta = 0.6$

As was mentioned above, while it appears there is a larger difference in the thrust to power ratios than when comparing the thrust or power alone, this is in fact not the case. The small differences between the ratios are simply more noticeable due the to the scale of the graph. Both the thrust and power ratio have large decreases as the burn time increases, while the thrust to power stays more constant, allowing the differences in the curves to be seen more clearly.

4 Updated operational envelope model

In this chapter, improvements shall be made to the model in order to gain a more realistic understanding of the operational envelope. This model will be used to select a number of initial points and compare them in order to find the optimal operational envelope. It should be noted that the new model will be written in python, unlike the previous model which was written in Matlab, hence the large difference in graph appearance.

4.1 Assumptions and additional inputs

Some of the assumptions made in the previous model, will be also be used in the updated version of the model

- No mixing between propellant and pressurant
- Propellant is expelled before the pressurant during operation of the propulsion system
- Ideal gas law of no friction between pressurant molecules
- Liquid propellant is fully incompressible
- Isothermal expansion of the gas within the tubing

The original model made two additional assumptions, which will not be made for this updated operational envelope. These two are as follows:

- The pressure in the tubing is equal to the pressure in the thruster (blowdown mode for he thruster)
- No boundary layer effects accounted for which is especially of importance for the throat of the nozzle.

The first of these assumptions is not made, as while the pressure will be almost equal, the valve and the injection into the thruster will result in a small pressure drop. The magnitude of this will be determined later in the chapter. The second assumption that no boundary layer effects are accounted for, and hence no quality factors, is not made as a result of the low Reynolds number in the throat. In reality, the nozzle is going to be far from ideal [6]. However, while assumptions can be made as to the effects of the low Reynolds number, to exactly quantify the performance losses, testing must be done. A better approximation could also be generated through the means of a CFD analysis. As such for graphs in this report the discharge coefficient C_d will at this stage be taken as 1, but the code will be easily adaptable at a later stage.

In addition to these assumptions, a number of fixed input variables need to be defined based on the properties of the hardware and the selected propellant. The numbers laid out in section 2.3 will be used for this updated model. Here is was determined that the real specific impulse of the thruster was 94.9 s at 550 K. However it is known that the specific impulse is proportional to the square root of the chamber temperature[5], as such the the I_{sp} won't be assumed to be a constant input but will be updated each iteration as follows:

$$I_{sp} = \frac{I_{sp,ref}}{\sqrt{T_{c,ref}}} \cdot \sqrt{T_c} = \frac{94.9}{\sqrt{550}} \cdot \sqrt{T_c} \tag{4.1}$$

4.2 Design variables

In the previous model, the initial pressure, the initial volume of pressurant and the total VLM thrust time, were all constant inputs at the beginning of the program. For the updated model, the only initial pressure and volume of the nitrogen will be used, as the thrust time is determined from these results. The ranges considered for these were presented in Table 2.4.

4.3 Modelling the VLM behaviour

There is also a constrain on the maximum power that can be used. The maximum power usage occurs at the beginning of the burn, and as such the initial power required can be written simply as a function of the initial pressure. This can be solved with the algorithm below. Assuming an efficiency of 60% the maximum initial pressure permitted is 1.25 bar. The exact maximum pressure was 125015 Pa it was rounded down to 125000 Pa for input feasibility.

Algorithm 4.1 Determining maximum pressure

syms p_0
$T_{c\,0} = L_h \cdot T_1/(T_1 \cdot R \cdot \ln(p_1/p_0) + L_h)$
$\dot{m}_0 = p_0 \cdot A_t \cdot \Gamma/sqrt(R \cdot T_{c\,0})$
$eqn = P_{0\,max} == \dot{m}_0/\eta_{th} \cdot [c_{pl} \cdot (T_{c\,0} - T_0) + L_h]$
solve(p_0, eqn)

The two limitations taken from [2] are that at the end of the VLM thrusting period, there must be a minimum of 0.2 g of propellant remaining, and that the minimum thrust must not be below 0.12 mN.

Algorithm 4.2 Variable Temperature Psuedocode

while $F_t > 0.12$ mN **and** $M_p > 0.2$ g
$p = [p\,;\, V_0 \cdot p_0/(V_0 + m_{exit}(end)/density)]$
$\dot{m} = [\dot{m}\,;\, p(end-1) \cdot A_t \cdot \Gamma/\sqrt{(R \cdot T_c(end))} \cdot C_d]$
$T_c = [T_c\,;\, L_h \cdot T_1/(T_1 \cdot R \cdot \ln(p_1/p(end-1)) + L_h)]$
$m_{exit} = [m_{exit}, m_{exit}(end) + \dot{m}(end-1) \cdot \Delta t]$
$t = t + \Delta t$
$F_t = \dot{m}(end) \cdot I_{sp} \cdot g_0$
$M_p = M_{p\,0} - m_{exit}(end)$

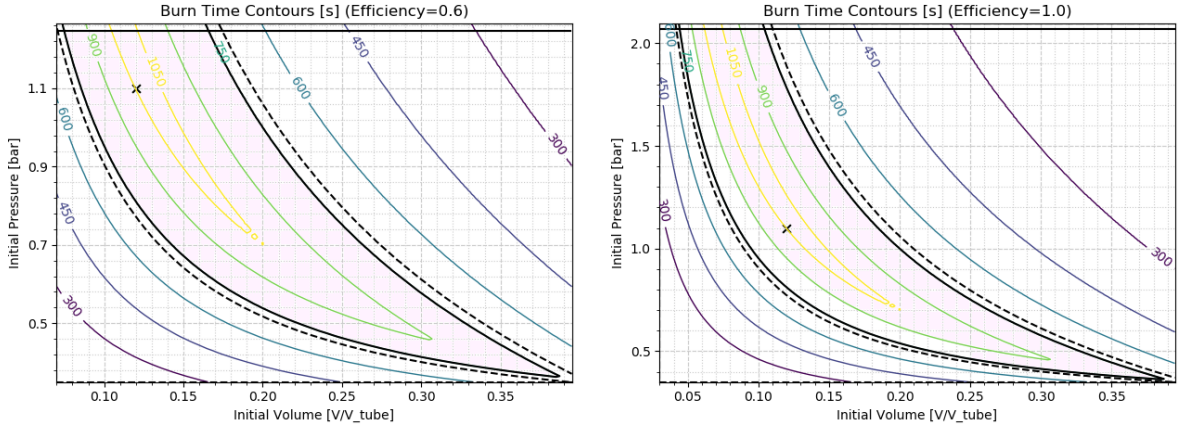
This leave the only input to be varied, to be the initial volume of nitrogen in the tube. A higher initial volume will result in a higher thrust across the burn, but

$$\begin{aligned} \max_{p(0),V(0)} \quad & F_t/P \\ s.t. \quad & \max(P(t)) \leq 4\,\mathrm{W} \\ & \min(F_t(t)) \geq 0.12\,\mathrm{mN} \\ & \min(M_p(t)) \geq 0.2\,\mathrm{g} \\ & 500\,\mathrm{s} \leq t_{burn} \leq 3000\,\mathrm{s} \end{aligned}$$

4.4 Contour plots of output variable

In this section a number of contour plots are presented, showing how the thrust, burn time, and thrust to power ratios all vary with the initial nitrogen pressures and volumes.

On each graph the original design of an initial pressure of 1.1 bar and volume of 12% [2] is included, denoted as a cross on each plot. This is for easy comparison with the previous design. For each output considered, there will be two contour plots; one for a heating efficiency of 60% and one for 100%. This is because the exact efficiency is not know, so a range is considered. Finally, each plot contains a shaded area highlighting the feasible region specified by the conditions in section 4.3.



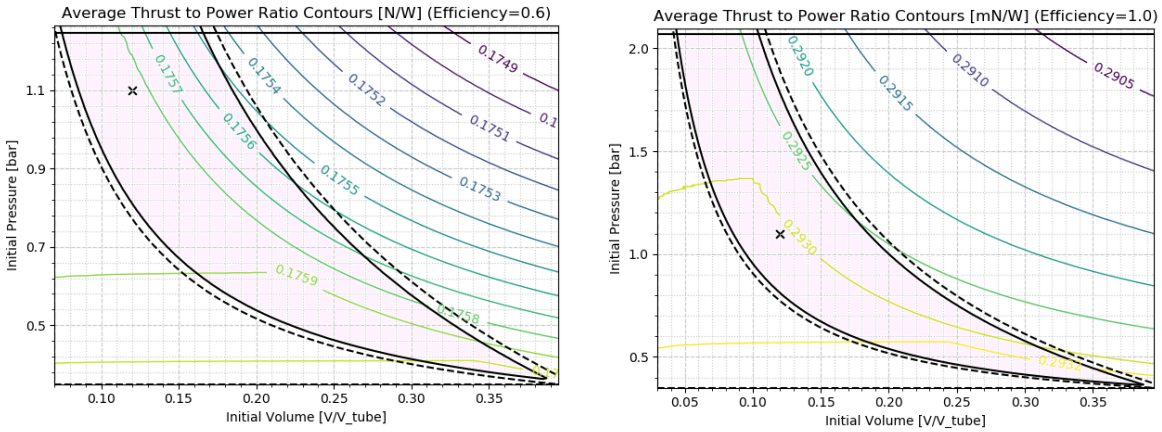
(a) $\eta = 60\%$ (b) $\eta = 100\%$

Figure 4.1: Burn time [s] contours

Figure 4.1 shows the burn time contours. It can be seen that the longest achievable burn times are a little over 1100 seconds. This is indicated by the yellow contours, which appear to be discontinuous due to the step sizes used in the initial pressure and volume ranges. The model in chapter 3 indicated the initial point resulted in a burn time of 1200 s, however, the plot below indicates it results in a burn time of only 1050 s.

One key difference between these plots and the other contour plots shown in this section, is that for a given initial pressure, the same contour value achieved at two different initial volumes. This is a result of the way the model is developed, with the simulation ending with either the remaining propellant reaching 0.2 g or the thrust falling below 0.12 mN. These two parameters are functions of the initial mass of water and the water mass flow rates.

The initial mass of water is only a function of the initial volume of the nitrogen, as water is incompressible at the given ambient temperature. Similarly the water mass flow rate increases as the pressure increases. As such, at low initial pressures and pressurant volumes, there is a low initial mass flow and larger mass of water. This results in the thrust falling below the set limit in a shorter amount of time. At higher initials pressures and volumes, there are higher mass flows but a lower initial mass of water, meaning the total burn time is also reduced. This is why the same contour can appear twice in plot in Figure 4.1.

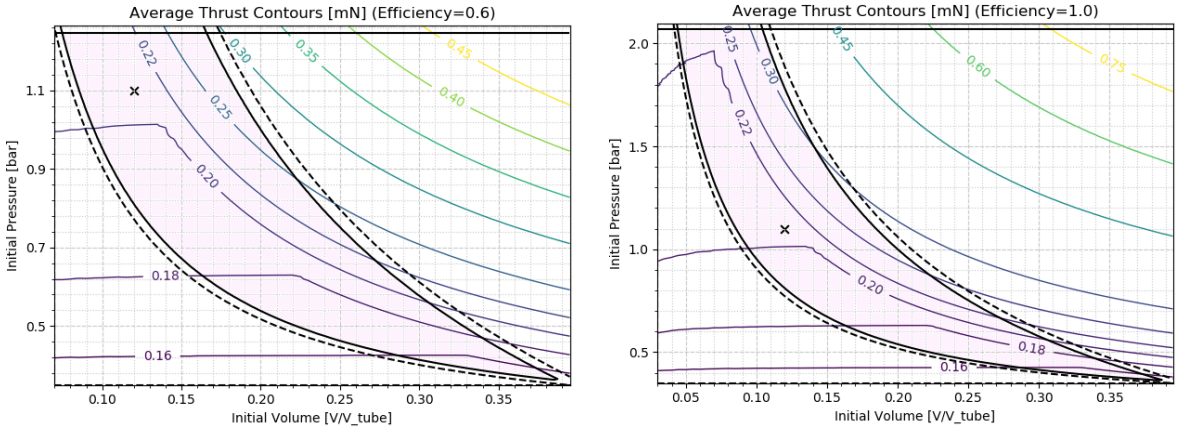


(a) $\eta = 60\%$ (b) $\eta = 100\%$

Figure 4.2: Average thrust to power [mN/W] contours

Similarly we can look at the contours for the average thrust to power ratio, as are shown in Figure 4.2. From these contours it can be seen that the lower the initial pressure of the system the more efficient the thruster. However, it can be seen that the difference between the contours is negligibly small, indicating the thrust to power ratio may not be the most worthwhile value to consider.

As such, the average thrust across the burn can also be considered, as are show in Figure 4.3. From this we can see the average thrust over the total burn increases as both the initial volume and initial pressure increase.



(a) $\eta = 60\%$ (b) $\eta = 100\%$

Figure 4.3: Average thrust [mN] contours

4.5 Trade-off between various envelopes

As was shown by the from the contours above, there is a large range of possible options for the initial nitrogen pressure and volume. However, only a few of these will be selected and traded-off in this section.

4.5.1 Selection of options

For the trade-off, four point will be considered, and for each a heating efficiency of 60% will be assumed. The points will be the max average thrust, the max burn time, the max thrust to power ratio, and the point selected in the previous analysis discussed in chapter 3. The previous point was $p_0 = 1.1$ bar and $V_0 = 0.12$, and is included as a bench mark to highlight the changes made to the model.

The maximum burn time is considered as it would allow the VLM to operate for longer. This is a burn time of 1128 seconds, and corresponds to a initial point of $p_0 = 1.22$ bar and $V_0 = 0.115$. The highest average thrust over the burn possible with an efficiency of 60% is 0.288 mN. This is an initial point of $p_0 = 1.25$ bar and $V_0 = 0.165$. Finally the highest average thrust to power ratio, the point where the thruster is acting most efficiently, is 0.1760 mN/W, at an initial point of $p_0 = 0.365$ bar and $V_0 = 0.385$. All four of the selected points selected are shown in Figure 4.4.

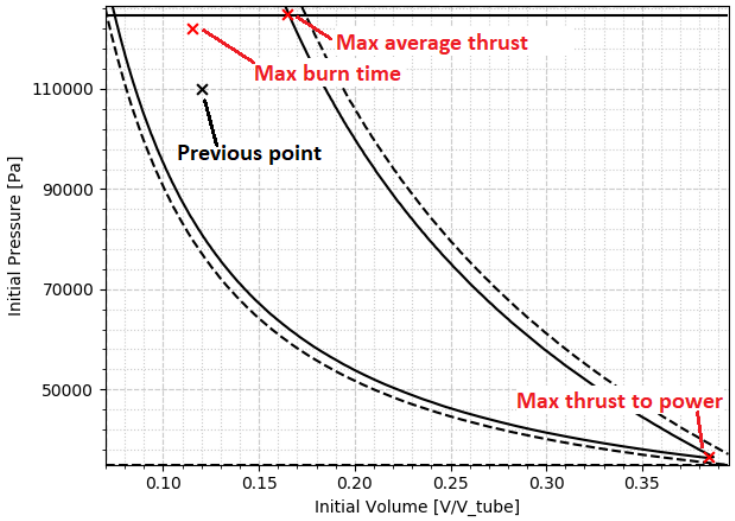


Figure 4.4: Chosen points for trade-off

4.5.2 Comparison of operational envelopes

In order to trade off the four initial points, the data for thruster behaviour across the entire burn for all the options will be compared. This is shown in Figure 4.5 (following page), which shows each of the full operational envelopes.

From these plots, some of the key features of the operational envelope can be determined, allowing for the different operational points to be easily compared. These key values are summarised in Table 4.1, as they will be used for the trade-off in the following subsection.

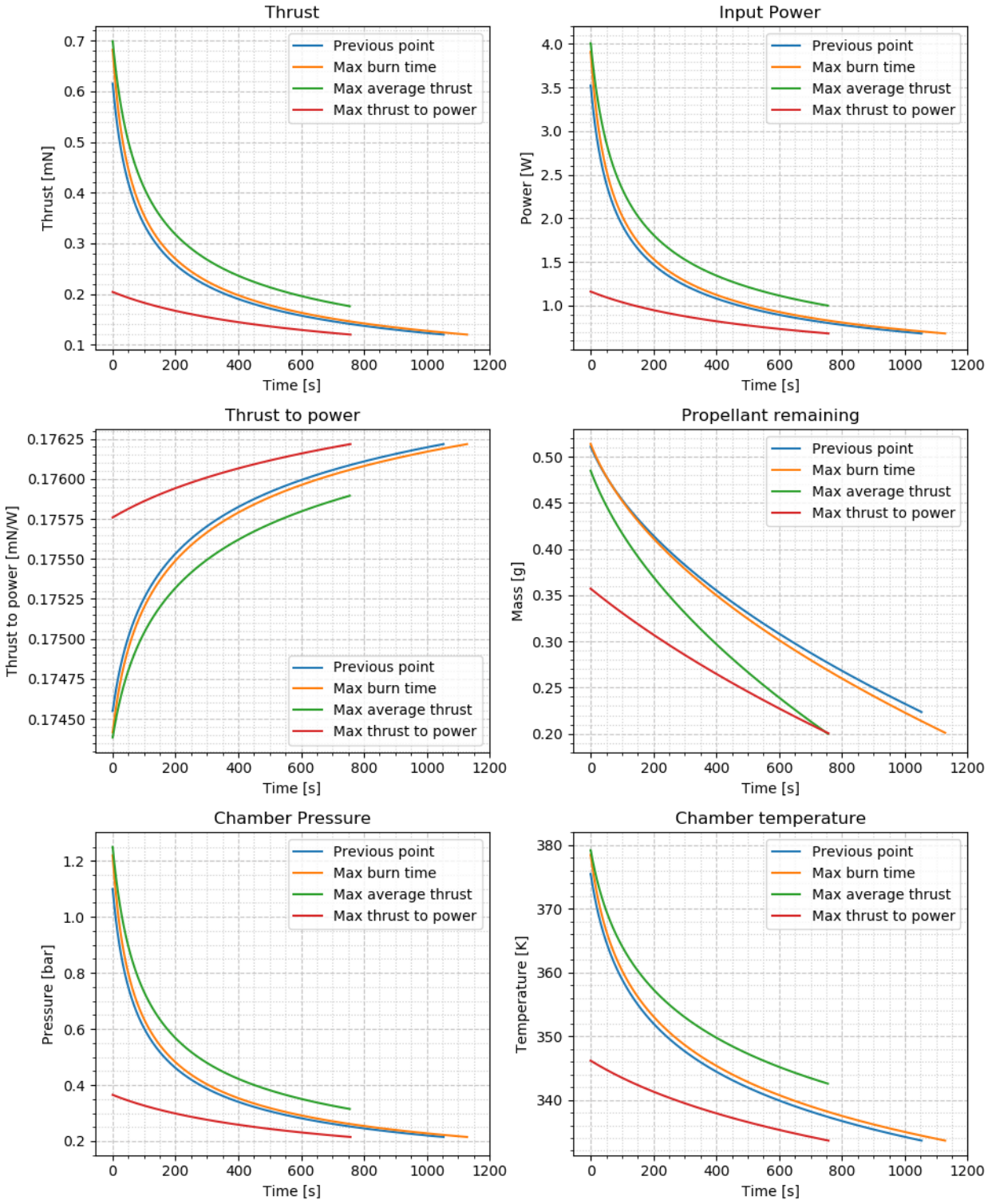


Figure 4.5: Operating envelopes

Table 4.1: Trade off summary

	p_0 [bar]	V_0 [V/V_{tube}]	t_{burn} [s]	I [mN·s]	F_{avg} [mN]	P_{avg} [W]	P_{max} [W]
Previous point	1.1	0.12	1053.3	212.93	0.202	1.151	3.523
Max burn time	1.22	0.115	1128.2	231.93	0.206	1.170	3.911
Max average thrust	1.25	0.165	754.9	213.36	0.283	1.612	4.000
Max thrust to power	0.365	0.385	757.0	114.52	0.151	0.859	1.161

4.5.3 Trade-off

The four considered scenarios for the nitrogen initial pressure and volume shall now be traded-off in order to select the optimal solution. For this a graphical trade-off table shall be used, as is shown in Table 4.2.

This trade-off shall be based on four criteria; the total burn time, the average thrust, the average input power required, and the total impulse. Of these the average thrust and burn time are considered to be the most important, as indicated by the wider column. The total impulse is deemed least important being so heavily linked to the burn time and thrust, so has the narrowest column.

For each criteria the four options are scored. Often with graphical the colours are used to indicate whether a design is considered good with respect to the design requirements, However, as all the initial points were taken from within the feasible region, it will be used to rank the options compared to each other. Here ✓✓ indicates the best option, ✓ indicates an option which is nearly as good as the best one, ∼ indicates the option is not good, but not particularly bad, whilst ✗ means in comparison the option does very poorly.

Table 4.2: Graphical Trade off Table

Criteria / **Option**	**Burn Time**	**Thrust**	**Power**	**Total Impulse**
Previous Point	✓	✓	✓	✓
Max Burn Time	✓✓	✓	✓	✓✓
Max Average Thrust	✗	✓✓	∼	✓
Max Thrust to Power	✗	✗	✓✓	✗

As can be seen from the trade-off table in Table 4.2, the best option for the VLM thruster is the point allowing for the maximum possible burn time. This point is $p_0 = 1.22$ bar and $V_0 = 0.115 V_{tube}$.

4.6 Operational envelope of design point

In section 4.5 an initial point of $p_0 = 1.22$ bar and $V_0 = 0.115V_{tube}$ was selected. The operational envelope for this point with an assumed heating efficiency of 60% was also presented.

As 60% is not certain to be the heating efficiency, a range of other values must also be considered. The heating efficiency only effects the input power required, meaning the only graphs in which a difference will be seen will be the input power and thrust to power ratio graphs. The other outputs of the operational envelope for the design point were presented in section 4.5 but are summarised in Table 4.3 below.

Table 4.3: Operational envelope for $p_0 = 1.22$ bar, $V_0 = 0.115V_{tube}$

Parameter	**Value**	**Units**
Burn time	1128.2	s
Average thrust	0.206	mN
Max thrust	0.682	mN
Total impulse	231.93	mN·s
Initial propellant	0.514	g
Remaining propellant	0.2	g

As was mentioned the input power and thrust to power ratio vary with varying heating efficiencies, as is shown in Figure 4.6. Here efficiencies of 40%, 60%, 80%, and 100% are considered.

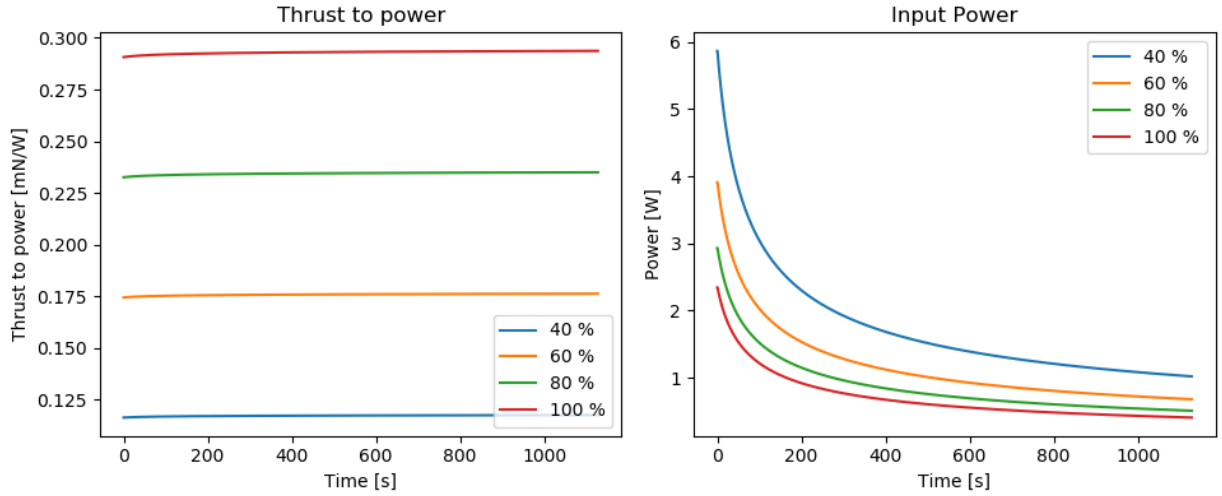


Figure 4.6: The effect of heating efficiency at the chosen point

From this it can be seen that for a heating efficiency lower than 60%, the input power required is greater than 4 W. This is expected as the design was done assuming an efficiency of 60%, but would violate **PROP-FUN-100**. If the efficiency of the thruster is determined to be less than 60%, the initial pressure and volume of the nitrogen would have to be redetermined.

Finally, the compliance of the design with the requirements must be analysed. This is shown in Table 4.4 below. All requirements but the power one can be said to be met. The power requirement

cannot be said to be met at this stage, as was explained above.

Table 4.4: Requirements compliance matrix

ID	Requirement	Met?	Justification
PROP-PERF-200	The thrust provided by the propulsion system shall be 3 mN as a maximum.	✓	The max thrust is only 0.682 mN.
PROP-PERF-210	The thrust provided by the propulsion system shall be at least 0.12 mN.	✓	The min thrust is just above 0.12 mN, at the end of the burn.
PROP-FUN-100	The micro-propulsion system shall have at least two modes:idle mode with a maximum power consumption of 15 mW and a full thrust mode with a maximum power consumption of 4 W.	❊	As long as the efficiency is greater or equal to 60% the condition is met. But the efficiency is not fully know.
PROP-FUN-200	The thruster shall be able to operate on gaseous N_2 as well as on liquid H_2O	✓	All calculations were assuming these two propellants
PROP-RAMS-200	The internal pressure of all propulsion system components shall not be higher than 10 bar	✓	The max pressure is 1.22 bar.

4.7 Retrospective on the modelling approach

While the modelling approach taken allowed for a preliminary design to be made, it is not yet the most optimal or accurate method.

One key issue is that the trade-off considered only three extreme points, two of which lie on the minimum time contour. This is an issues partially as the time limit was selected somewhat arbitrarily, but more importantly it only considers three extreme points, none of which may be the most optimal value.

Manually trading-off more values with the graphical table would be both slow and may still not lead to the optimal value being selected. It could also be possible to brute-force an optimal solution, by considering every set of initial values in the feasible range and then performing a weighted trade-off to find the optimal solution. While this would find the most optimal point, it is both inelegant and computationally slow. As such some form of optimisation algorithm should be implemented. This would most likely involve starting by selecting random areas in the domain and then searching for the optimal value. Care should be taken when selecting the algorithm to be used, as it is not know if there are local optimums on which the algorithm could accidentally converge.

The other main issues comes from the minimum burn time limit. As was mentioned, the value taken was not a defined requirement. In this model it was needed as it provided the limit for two of the extreme values used in the trade-off. However, for the optimisation algorithm it would not be required to specify this, as without it there is still a finite range of possible initial values. More overly, having a high burn time is the highest weighted trade-off criteria, which should prevent the chosen point having a low burn time. The only reason in which this limit would be worth keeping is that it reduces the size of the considered domain, which would reduces the computational time.

5 Conclusion and recommendations

Through out this report, the operational envelope of the a Vaporising Liquid Micro-resitojet thruster for the Delfi-PQ satellite was analysed. This VLM thruster uses water as a propellant with nitrogen as a pressurant.

This analysis began by identifying the key inputs required, as well as analysing the thruster requirements in order to establish suitable ranges for the initial values. Here it was established that the behaviour would be limited by three main requirements; a maximum power usage of 4 W, a minimum thrust of 0.12 mN, and that there must be at least 0.2 g of propellant remaining for the LPM thrust. It was also established that the two main key initial values were the initial volume and pressure of the nitrogen.

The next stage involved analysing the existing model for the VLM operational envelope. This model selected an initial pressure of 1.1 bar and a volume of 12 % of the tube volume, with the aim of a burn time of 1200 s. This model was predominately based on equations ideal rocket theory, which can be used to represent a more realistic scenario through the inclusion of quality factors. Some of these were already included as for instance the real specific impulse was used, but others were not. This model did compare a few initial values for the pressure and volume to pick the best, but not a sufficient amount for the optimal point to be found.

Finally, the an updated model for the operational envelope was presented. This strived to not only compare a larger range of initial points, but also to use more realistic values. This was done by producing a number of contour plots to explore the effect the initial values selected had on the thrust time, thrust and input power required for the VLM. A trade-off was then performed which determined the initial point corresponding to the longest thrust time was the most optimal. This was an initial pressure of 1.22 bar and volume of 11.5% of the tube volume, giving a max thrust if 0.682 mN and a thrust time of 1128.2 s for the VLM.

However, as was explained in chapter 4 this was assuming a heating efficiency of 60% and discharge coefficient of 1. Therefore the value presented can only be seen as an initial idea of the operational envelope. After further testing to determine better estimate for these values, the process described in chapter 4 with the code provided in the appendix should be replicated for a more accurate operational envelope.

5.1 Recommendations for improvements in future models

As there is still some doubt as to the accuracies of the current operational envelope model, a number of improvements could be made to improve this.

1. **Determine a more accurate value for the specific impulse**
   The value taken for the I_{sp} was for a chamber pressure of 5 bar and temperature of 550 K. While the varying temperature has been accounted for, the much lower pressure has not yet been.

2. **Determine the discharge coefficient of the nozzle**
   Currently the real mass flow is taken to be equal to the real mass flow, but is know that due

to the low Reynolds number in the throat this won't be the case. Further analysis of the nozzle would be needed to account for this.

3. **Determine the heating efficiency**
   The heating efficiency has a large impact on the power required for the VLM thruster, and at this stage only an assumed value is used.

4. **Determine the pressure drop between the tube and the chamber**
   In both models this was assumed to be small enough to be neglected. However, this is not certain, no was enough about the geometry know to perform orifice calculations, nor if there was any pressure loss due to friction. As such, further modelling should be done to check this still holds.

5. **Specify the amount of propellant left for the LPM firing**
   The limit taken was based on assumptions made in the previous study of the envelope. However, this is not clearly defined by the requirements.

Bibliography

[1] V. Pallichadath. *Propulsion Subsystem Requirements for the Delfi-PQ Satellites*. DFF-TUD-PROP-REQ. 2017.

[2] L.A. Turmaine. *A technology demonstration payload for micro-resistojet thrusters on Delfi-PQ*. MSc thesis. TU Delft, 2018.

[3] NIST Chemistry Webbook. *Water*. Last accessed 01/05/2019. URL: https://webbook.nist.gov/cgi/cbook.cgi?ID=C7732185&Units=SI.

[4] A, Cervone et al. *Green micro-resistojet research at Delft University of Technology: the new frontiers of Cubesat propulsion* (2016).

[5] B.T.C. Zandbergen. *Thermal Rocket Propulsion*. Version 2.07. AE4S01. 2018.

[6] V. Pallichadath et al. *In-orbit micro-propulsion demonstrator for pico-satellite applications*. International Astronautical Congressl. 2018.

A Python code

A.1 Operational_envelope.py: The main file for determining the operational envelope, by comparing the performance for a set of initial points.

A.2 Contour_Plots.py: The scripts for creating contour plots of the feasible region.

A.3 Final_Envelope.py: Plot the chosen operational envelope at a range of efficiencies.

A.4 Behaviour.py: The python class containing the functions needed to calculated each characteristics

A.5 Axes_setup.py The class containing functions for formatting the graphs

A.6 Progress_bar_tool.py A loading bar called in the contour plots file, as they take time to produce

A.1 Operational_envelope.py

```
"""
Kathleen Blyth (2019) for Micropropulsion
The main file for determining the  best operational envelope by comparing a
range of initial points
"""
import numpy as np
import matplotlib.pyplot as plt
import time
from Behaviour import ThrusterBehaviour
from Axes_setup import AxesConfig

# ============================================================================
# ------------------------------ User Inputs ---------------------------------
# ============================================================================

efficiency = 0.6                                    # Thruster efficiency [-]
initial_pressure = [1.1, 1.22, 1.25, 0.365]         # Initial pressure [bar]
initial_volume = [0.12, 0.115, 0.165, 0.385]        # Initial volume [V/V_tube]
dt = 0.1                                            # Time step [s]

# ============================================================================
# ------------------- Determining operational envelope -----------------------
# ============================================================================

axis_setup = AxesConfig()
behaviour = ThrusterBehaviour()
start_time = time.time()

initial_pressure = [x * 1e5 for x in initial_pressure]
envelopes = {}

for i in range(4):
    key = "Point " + str(i+1)
    envelopes[key] = behaviour.envelope(initial_volume[i], initial_pressure[i],
```

```
                                        efficiency, dt, 2)

# ============================================================================
# ------------------------- Generating plots ---------------------------------
# ============================================================================

# --> Defining the 4 axes
fig, axs = plt.subplots(3, 2, figsize=(10, 12))
ax1, ax2, ax3, ax4, ax5, ax6 = \
    axs[0, 0], axs[0, 1], axs[1, 0], axs[1, 1], axs[2, 0], axs[2, 1]

# --> Plotting the data for all scenarios
for i in range(4):
    key = "Point " + str(i+1)
    burn_time = np.arange(0, envelopes[key]["burn_time"], dt)
    ax1.plot(burn_time, [x * 1000 for x in envelopes[key]["thrust"]])
    ax2.plot(burn_time, envelopes[key]["power"])
    ax3.plot(burn_time, [x * 1000 for x in envelopes[key]["thrust_to_power"]])
    ax4.plot(burn_time, [x * 1000 for x in envelopes[key]["mass"]])
    ax5.plot(burn_time, [x / 100000 for x in envelopes[key]["pressure"]])
    ax6.plot(burn_time, envelopes[key]["temperature"])

# --> Formatting the plot axis
labels = ['Previous point', 'Max burn time', 'Max average thrust',
          'Max thrust to power']
axis_setup.axes_config(
    axs, ax1, labels, "upper right", "Time [s]", "Thrust [mN]", "Thrust")
axis_setup.axes_config(
    axs, ax2, labels, "upper right", "Time [s]", "Power [W]", "Input Power")
axis_setup.axes_config(axs, ax3, labels, "lower right", "Time [s]",
                       "Thrust to power [mN/W]", "Thrust to power")
axis_setup.axes_config(axs, ax4, labels, "upper right", "Time [s]", "Mass [g]",
                       "Propellant remaining")
axis_setup.axes_config(axs, ax5, labels, "upper right", "Time [s]",
                       "Pressure [bar]", "Chamber Pressure")
axis_setup.axes_config(axs, ax6, labels, "upper right", "Time [s]",
                       "Temperature [K]", "Chamber temperature")
plt.show()
```

A.2 Contour_Plots.py

```
"""
Kathleen Blyth (2019) for Micropropulsion
The main file for producing contours plots of the feasible region
"""
from Behaviour import ThrusterBehaviour
import numpy as np
import matplotlib.pyplot as plt
from Progress_bar_tool import ProgressBar
import time

start_time = time.time()
behaviour = ThrusterBehaviour()
# ============================================================================
# ------------------------ User Inputs and Setup -----------------------------
# ============================================================================

dt = 1                          # Time step [s]
efficiency = 1.0                      # Thruster efficiency [-]

# Select which contours to add
contour_to_plot = 1                    # Index from list below [0-3]
contour_options = [None,                                      # Index 0
                   "Burn Time",                               # Index 1
                   "Average Thrust",                          # Index 2
                   "Average Thrust to Power Ratio"]           # Index 3

# ==> Contour levels - if desired specify, if not set as None, otherwise set
# range and rounding
# contour_levels = None
contour_levels = [[[300, 450, 600, 750, 900, 1050], '%1.0f'],
                  [[0.16, 0.18, 0.20, 0.22, 0.25, 0.3, 0.35, 0.4, 0.45],
                   '%1.2f'],
                  [np.arange(0.1749, 0.1761, 0.0001), '%1.4f']]


#  Initial pressure range [Pa]
initial_pressure_range = np.arange(0.35e5, 1.27e5, 0.05e4)
# Initial volume range [m2]
initial_volume_range = np.arange(0.05, 0.4, 0.005)

#  Initial pressure range [Pa]
initial_pressure_range = np.arange(0.35e5, 2.1e5, 0.05e4)
# Initial volume range [m2]
initial_volume_range = np.arange(0.03, 0.4, 0.005)


#     initial_pressure_range = np.arange(0.35e5, 2.12e5, 0.1e4)
#     initial_volume_range = np.arange(0.03, 0.4, 0.01)


# ============================================================================
# ---------------------------- Generating contour data -----------------------
# ============================================================================

```

```
55 units = ["s", "mN", "mN/W"]                # Units of three considered contours
56 contour = contour_options[contour_to_plot]     # Index contour
57
58 # ==> Specify plot title
59 if contour is None:
60     title = f"Operational envelope (Efficiency = {efficiency})"
61 else:
62     title = f"{contour} Contours [{units[contour_to_plot-1]}] " \
63         f"(Efficiency={efficiency})"
64
65 # ==> Set up dictionary of lists for the contour plots
66 contours = {"Power limit": [],
67             "Time limit": [],
68             "Thrust limit": [],
69             "Volume limit": [],
70             "Time": [],
71             "Thrust": [],
72             "Thrust to power": []
73             }
74
75 # ==> Initialise the progress bar
76 bar = ProgressBar(len(initial_pressure_range),
77                   label="",
78                   process_count=True,
79                   progress_percent=True,
80                   run_time=True,
81                   eta=True,
82                   overwrite_setting=True,
83                   bar_type="Equal",
84                   activity_indicator=False,
85                   rainbow_bar=True)
86
87 # ==> Produce contour plots
88 for pressure in initial_pressure_range:
89     power_limit_lst = []
90     time_limit_lst = []
91     thrust_limit_lst = []
92     volume_limit_lst = []
93     time_lst = []
94     thrust_lst = []
95     thrust_to_power_lst = []
96
97     for volume in initial_volume_range:
98         # Calling operational envelope
99         envelope_outputs = behaviour.envelope(volume, pressure, efficiency, dt)
100        # Updating lists
101        power_limit_lst.append(envelope_outputs["Power limit"])
102        time_limit_lst.append(envelope_outputs["Time limit"])
103        thrust_limit_lst.append(envelope_outputs["Thrust limit"])
104        volume_limit_lst.append(envelope_outputs["Mass limit"])
105        time_lst.append(envelope_outputs["Burn time"])
106        thrust_lst.append(envelope_outputs["Average thrust"]*1000)
107        thrust_to_power_lst.append(
108            envelope_outputs["Average thrust to power"] * 1000)
109        bar.update_activity()
110
```

```
111     # Updating list of lists
112     contours["Power limit"].append(power_limit_lst)
113     contours["Time limit"].append(time_limit_lst)
114     contours["Thrust limit"].append(thrust_limit_lst)
115     contours["Volume limit"].append(volume_limit_lst)
116     contours["Time"].append(time_lst)
117     contours["Thrust"].append(thrust_lst)
118     contours["Thrust to power"].append(thrust_to_power_lst)
119     bar.update_progress()
120
121 initial_pressure_range = [x/10**5 for x in initial_pressure_range]
122 # ================================================================================
123 # ------------------------- Generating contour plot ------------------------------
124 # ================================================================================
125
126 fig, ax = plt.subplots()
127 if contour is not None:
128     # --> Plot main contours.  Set up as plt.contour(lst, lst, lst of lists)
129     if contour_levels is not None:
130         main_contours = plt.contour(
131             initial_volume_range, initial_pressure_range,
132             contours["Time", "Thrust", "Thrust to power"][contour_to_plot-1],
133             levels=contour_levels[contour_to_plot-1][0],
134             linewidths=1)
135         plt.clabel(main_contours, main_contours.levels, inline=True,
136                    fmt=contour_levels[contour_to_plot-1][1], fontsize=10)
137     else:
138         main_contours = plt.contour(
139             initial_volume_range, initial_pressure_range,
140             contours["Time", "Thrust", "Thrust to power"][contour_to_plot - 1],
141             linewidths=1)
142         plt.clabel(main_contours, main_contours.levels, inline=True,
143                    fontsize=10)
144
145 # --> Plotting limit contours and a dashed line to highlight the feasible region
146 # Power limit
147 plt.contour(initial_volume_range, initial_pressure_range,
148             contours["Power limit"], levels=[0], colors=('k',))
149 plt.contour(initial_volume_range, initial_pressure_range,
150             contours["Power limit"], levels=[0.02],
151             colors=('k',), linestyles=('--',))
152 # Time limit
153 plt.contour(initial_volume_range, initial_pressure_range, contours["Time limit"],
154             levels=[0], colors=('k',))
155 plt.contour(initial_volume_range, initial_pressure_range, contours["Time limit"],
156             levels=[0.055], colors=('k',), linestyles=('--',))
157 # Thrust limit
158 plt.contour(initial_volume_range, initial_pressure_range,
159             contours["Thrust limit"], levels=[0.000], colors=('k',))
160 plt.contour(initial_volume_range, initial_pressure_range,
161             contours["Thrust limit"], levels=[0.075],
162             colors=('k',), linestyles=('--',))
163 # Volume limit
164 plt.contour(initial_volume_range, initial_pressure_range,
165             contours["Volume limit"], levels=[0.000], colors=('k',))
166 plt.contour(initial_volume_range, initial_pressure_range,
```

```
167             contours["Volume limit"], levels=[0.025],
168             colors=('k',), linestyles=('--',))
169 # Marker for the previous point
170 plt.scatter(0.12, 1.1, s=40, c='k', marker='x')
171
172 # --> Formatting labels
173 plt.xlabel("Initial Volume [V/V_tube]")
174 plt.ylabel("Initial Pressure [bar]")
175 plt.title(title)
176
177 # --> Adding grid lines
178 # Set ranges for ticks
179
180 if efficiency == 0.6:
181     x_major_ticks = np.arange(0.1, max(initial_volume_range), 0.05)
182     x_minor_ticks = np.arange(min(initial_volume_range),
183                               max(initial_volume_range), 0.01)
184     y_major_ticks = np.arange(50000, max(initial_pressure_range), 20000)
185     y_minor_ticks = np.arange(38000, max(initial_pressure_range), 4000)
186
187
188 else:
189     x_major_ticks = np.arange(0.05, max(initial_volume_range), 0.05)
190     x_minor_ticks = np.arange(min(initial_volume_range),
191                               max(initial_volume_range), 0.01)
192     y_major_ticks = np.arange(0.5, max(initial_pressure_range), 0.5)
193     y_minor_ticks = np.arange(0.4, max(initial_pressure_range), 0.1)
194
195
196 # Specify ticks
197 ax.set_xticks(x_major_ticks)
198 ax.set_xticks(x_minor_ticks, minor=True)
199 ax.set_yticks(y_major_ticks)
200 ax.set_yticks(y_minor_ticks, minor=True)
201
202 # Formatting grid lines
203 ax.grid(which='major', color='#CCCCCC', linestyle='--')
204 ax.grid(which='minor', color='#CCCCCC', linestyle=':')
205
206 plt.show()
207
208 print(f"Total time = {round(time.time()-start_time, 2)} seconds")
```

A.3 Final_Envelope.py

```
"""
Kathleen Blyth (2019) for Micropropulsion
Plot the final operational point for a range of efficiencies
"""
from Behaviour import ThrusterBehaviour
import numpy as np
import matplotlib.pyplot as plt
from Axes_setup import AxesConfig

# ================================================================================
# ------------------------------ User Inputs -------------------------------------
# ================================================================================

efficiency = [0.4, 0.6, 0.8, 1.0]       # List of possible efficiencies [-]
initial_pressure = 1.22e5               # Initial pressure [bar]
initial_volume = 0.115                  # Initial volume [V/V_tube]
dt = 0.1                                # TIme step [s]

# ================================================================================
# ------------------- Determining operational envelope ---------------------------
# ================================================================================

axis_setup = AxesConfig()
behaviour = ThrusterBehaviour()

envelopes = {}

for i in range(len(efficiency)):
    key = "Point " + str(i+1)
    envelopes[key] = behaviour.envelope(initial_volume, initial_pressure,
                                        efficiency[i], dt, 2)

# ================================================================================
# ---------------------------- Generating  plots ---------------------------------
# ================================================================================

# --> Defining plot axes
fig, axs = plt.subplots(3, 2, figsize=(10, 12))
ax1, ax2, ax3, ax4, ax5, ax6 = \
    axs[0, 0], axs[0, 1], axs[1, 0], axs[1, 1], axs[2, 0], axs[2, 1]

# --> Plotting data
for i in range(4):
    key = "Point " + str(i+1)
    burn_time = np.arange(0, envelopes[key]["burn_time"], dt)
    ax1.plot(burn_time, [x * 1000 for x in envelopes[key]["thrust"]])
    ax2.plot(burn_time, envelopes[key]["power"])
    ax3.plot(burn_time, [x * 1000 for x in envelopes[key]["thrust_to_power"]])
    ax4.plot(burn_time, [x * 1000 for x in envelopes[key]["mass"]])
    ax5.plot(burn_time, [x / 100000 for x in envelopes[key]["pressure"]])
    ax6.plot(burn_time, envelopes[key]["temperature"])

x_major_ticks = np.arange(0, 1300, 200)
x_minor_ticks = np.arange(0, 1250, 50)
```

```
x_ticks = [x_major_ticks, x_minor_ticks]
ticks_ax2 = x_ticks + [np.arange(0, 7, 1), np.arange(0, 6.0, 0.2)]
ticks_ax3 = x_ticks + [np.arange(0.10, 0.35, 0.05),
                       np.arange(0.10, 0.31, 0.01)]

# --> Formatting plots
labels = ['40 %', '60 %', '80 %', '100 %']
axis_setup.axes_config(axs, ax1, labels, "upper right", "Time [s]",
                       "Thrust [mN]", "Thrust")
axis_setup.axes_config(axs, ax2, labels, "upper right", "Time [s]", "Power [W]",
                       "Input Power", ticks_ax2)
axis_setup.axes_config(axs, ax3, labels, "lower right", "Time [s]",
                       "Thrust to power [mN/W]", "Thrust to power", ticks_ax3)
axis_setup.axes_config(axs, ax4, labels, "upper right", "Time [s]", "Mass [g]",
                       "Propellant remaining")
axis_setup.axes_config(axs, ax5, labels, "upper right", "Time [s]",
                       "Pressure [bar]", "Chamber Pressure")
axis_setup.axes_config(axs, ax6, labels, "upper right", "Time [s]",
                       "Temperature [K]", "Chamber temperature")

plt.show()
```

A.4 Behaviour.py

```
"""
Kathleen Blyth (2019) for Micropropulsion
Files containing all the functions needed to determine the thruster behaviour
"""

import numpy as np
from numba import jit

# ============================================================================
# ------------------------------ Inputs --------------------------------------
# ============================================================================

gas_constant = 8.3144598 / 18.0153e-3   # Specific gas constant
throat_area = 4.5e-9                    # Throat area [m^2]
specific_heat_ratio = 1.33              # Ratio of specific heats [-]
latent_heat = 2256e3                    # Latent heat of vapourisation
temp_ref = 373.15                       # Reference temperature [K]
ambient_temp = 283                      # Satellite ambient temperature [K]
pressure_ref = 1.0142e5                 # Reference pressure [Pa]
isp_ref = 94.9 / np.sqrt(550)           # Reference Isp [s/K^0.5]
specific_heat_liquid = 4187             # Specific heat (liquid) [J/K/kg]
g0 = 9.80665                            # Gravitational acceleration [m/s^2]
density = 999.7                         # Water density [m3/kg]
tube_length = 0.30                      # Tube length [m]
tube_diameter = 1.57e-3                 # Tube diameter [m]
tube_volume = (tube_length * np.pi * tube_diameter**2) / 4  # Tube volume [m^3]


class ThrusterBehaviour:
```

```
30      # ---> Determining chamber pressure [Pa]
31      @staticmethod
32      @jit(nopython=True)
33      def __find_pressure(initial_pressure, initial_volume, prop_used):
34          pressure = initial_volume / (initial_volume + prop_used/density) \
35                     * initial_pressure
36          return pressure
37
38      # ---> Determining Vandenkerchove function value [-]
39      @staticmethod
40      @jit(nopython=True)
41      def __find_vandenkerckhove():
42          return np.sqrt(specific_heat_ratio) * (2 / (specific_heat_ratio+1)) \
43                 ** ((specific_heat_ratio+1)/(2*(specific_heat_ratio-1)))
44
45      # ---> Determining propellant mass flow [kg/s]
46      def __find_mass_flow(self, pressure, chamber_temp):
47          return pressure * throat_area * self.__find_vandenkerckhove() \
48                 / np.sqrt(gas_constant*chamber_temp)
49
50      # ---> Determining chamber temperature [K]
51      @staticmethod
52      @jit(nopython=True)
53      def __find_chamber_temp(pressure):
54          return temp_ref * latent_heat \
55                 / (temp_ref * gas_constant * np.log(pressure_ref/pressure)
56                    + latent_heat)
57
58      # ---> Determining volume of pressurant [m^3]
59      @staticmethod
60      @jit(nopython=True)
61      def __find_nitrogen_volume(pressure, initial_pressure, initial_volume):
62          return initial_volume * initial_pressure / pressure
63
64      # ---> Finding initial propellant mass [kg]
65      @staticmethod
66      @jit(nopython=True)
67      def __find_initial_mass(initial_volume):
68          return (tube_volume - initial_volume) * density
69
70      # ---> Determining thrust produced [kg]
71      @staticmethod
72      @jit(nopython=True)
73      def __find_thrust(mass_flow, chamber_temp, mass_flow_quality=1.0):
74          isp = isp_ref * np.sqrt(chamber_temp)
75          return isp * g0 * mass_flow * mass_flow_quality
76
77      # ---> Determining amount of propellant which has been used [kg]
78      @staticmethod
79      @jit(nopython=True)
80      def __find_prop_used(prop_used, mass_flow, dt=0.01):
81          prop_used += mass_flow * dt
82          return prop_used
83
84      # ---> Determining the input power required [W]
85      @staticmethod
```

```
86      @jit(nopython=True)
87      def __find_power(mass_flow, chamber_temp, efficiency):
88          return mass_flow / efficiency \
89                 * (specific_heat_liquid*(chamber_temp-ambient_temp) + latent_heat)
90
91      # ---> Using above eqs to determine operational envelope of a specified point
92      def envelope(self, initial_volume, initial_pressure, efficiency, dt=1,
93                   option=1):
94          """
95          :param initial_volume: Initial volume of N2 in the tube [V_n2/V_tube]
96          :param initial_pressure: The initial pressure of the N2 [Pa]
97          :param efficiency: The heating efficiency of the thruster [-]
98          :param dt: Time step [s]
99          :param option: Choose which set of outputs to call
100         :return: outputs: A dictionary containing: "Thrust_to_power_max",
101             "burn_time", "max_thrust", "avg_thrust", "Thrust_to_power_avg",
102             "Power_limit", "Time_limit", "Thrust_limit"
103         """
104
105         # Initial N2 volume [m3]
106         initial_volume = initial_volume * tube_volume
107         # Initial N2 mass [kg]
108         initial_mass = self.__find_initial_mass(initial_volume)
109
110         # ---> Initialising lists to store data
111         pressure_lst = [initial_pressure]
112         prop_remaining_lst = [initial_mass]
113         chamber_temp_lst = [self.__find_chamber_temp(initial_pressure)]
114         prop_used_lst = [0]
115         mass_flow_lst = \
116             [self.__find_mass_flow(initial_pressure, chamber_temp_lst[0])]
117         thrust_lst = [self.__find_thrust(mass_flow_lst[0], chamber_temp_lst[0])]
118         power_lst = [self.__find_power(mass_flow_lst[0], chamber_temp_lst[0],
119                                        efficiency)]
120         thrust_to_power_lst = [thrust_lst[0] / power_lst[0]]
121         time = 0
122
123         # ---> Main while loop performing analysis
124         while thrust_lst[-1] > 0.12e-3 and prop_remaining_lst[-1] > 0.2e-3:
125             pressure_lst.append(self.__find_pressure(initial_pressure,
126                                                      initial_volume,
127                                                      prop_used_lst[-1]))
128             mass_flow_lst.append(
129                 self.__find_mass_flow(pressure_lst[-2], chamber_temp_lst[-1]))
130             chamber_temp_lst.append(self.__find_chamber_temp(pressure_lst[-2]))
131             prop_used_lst.append(
132                 self.__find_prop_used(prop_used_lst[-1], mass_flow_lst[-2], dt))
133             power_lst.append(self.__find_power(mass_flow_lst[-2],
134                                                chamber_temp_lst[-2],
135                                                efficiency))
136             thrust_lst.append(
137                 self.__find_thrust(mass_flow_lst[-2], chamber_temp_lst[-2]))
138             prop_remaining_lst.append(prop_remaining_lst[0] - prop_used_lst[-1])
139             thrust_to_power_lst.append(thrust_lst[-1] / power_lst[-1])
140             time += dt
141
```

```
142         if option == 1:
143             outputs = {"Thrust to power max": thrust_to_power_lst[0],
144                        "Burn time": time,
145                        "Max thrust": max(thrust_lst),
146                        "Average thrust": np.average(thrust_lst),
147                        "Average thrust to power":
148                            np.average(thrust_to_power_lst),
149                        "Power limit": max(power_lst) / 4 - 1,
150                        "Time limit": 1 - time / 750,
151                        "Time limit1": time / 1500 - 1,
152                        "Thrust limit": 1 - thrust_lst[0] / 0.12e-3,
153                        "Mass limit": 1 - initial_mass / 2e-4}
154 
155             return outputs
156 
157         else:
158             outputs = {"burn_time": time,
159                        "thrust": thrust_lst,
160                        "power": power_lst,
161                        "mass": prop_remaining_lst,
162                        "thrust_to_power": thrust_to_power_lst,
163                        "temperature": chamber_temp_lst,
164                        "pressure": pressure_lst
165                        }
166 
167             return outputs
```

A.5 Axes_setup.py

```
1  """
2  Kathleen Blyth (2019) for Micropropulsion
3  Contains functions for formatting the graphs
4  """
5  import numpy as np
6 
7 
8  class AxesConfig:
9      def axes_config(self, axs, axes, legend_labels, legend_pos, x_label, y_label,
10                     title, ticks=None):
11         axes.set_title(title)
12         axes.set_xlabel(x_label)
13         axes.set_ylabel(y_label)
14         axes.set_xlim(0)
15         axes.legend(legend_labels, loc=legend_pos)
16         if ticks is None:
17             x_major_ticks, x_minor_ticks, y_major_ticks, y_minor_ticks = \
18                 self.axis_ticks(axs, axes)
19         else:
20             x_major_ticks, x_minor_ticks, y_major_ticks, y_minor_ticks = \
21                 [tick for tick in ticks]
22 
23         axes.set_xticks(x_major_ticks)
24         axes.set_xticks(x_minor_ticks, minor=True)
```

```
25          axes.set_yticks(y_major_ticks)
26          axes.set_yticks(y_minor_ticks, minor=True)
27
28          # Formatting grid lines
29          axes.grid(which='minor', color='#CCCCCC', linestyle=':', zorder=19)
30          axes.grid(which='major', color='#c6c4c4', linestyle='--', zorder=21)
31
32      @staticmethod
33      def axis_ticks(axs, axes):
34          ax1, ax2, ax3, ax4, ax5, ax6 = \
35              axs[0, 0], axs[0, 1], axs[1, 0], axs[1, 1], axs[2, 0], axs[2, 1]
36          x_major_ticks = np.arange(0, 1300, 200)
37          x_minor_ticks = np.arange(0, 1250, 50)
38          if axes == ax1:
39              y_major_ticks = np.arange(0.1, 0.71, 0.1)
40              y_minor_ticks = np.arange(0.1, 0.74, 0.02)
41          elif axes == ax2:
42              y_major_ticks = np.arange(1.0, 4.5, 0.5)
43              y_minor_ticks = np.arange(0.5, 4.3, 0.1)
44
45          elif axes == ax3:
46              y_major_ticks = np.arange(0.17450, 0.17630, 0.00025)
47              y_minor_ticks = np.arange(0.17430, 0.17635, 0.00005)
48
49          elif axes == ax4:
50              y_major_ticks = np.arange(0.2, 0.55, 0.05)
51              y_minor_ticks = np.arange(0.18, 0.53, 0.01)
52
53          elif axes == ax5:
54              y_major_ticks = np.arange(0.2, 1.3, 0.2)
55              y_minor_ticks = np.arange(0.15, 1.3, 0.05)
56
57          else:
58              y_major_ticks = np.arange(340, 381, 10)
59              y_minor_ticks = np.arange(332, 384, 2)
60
61          return x_major_ticks, x_minor_ticks, y_major_ticks, y_minor_ticks
```

A.6 Progress_bar_tool.py

Some of the characters used in this file would not load into LaTeX. As such it can be found at https://github.com/vguillet/Better_loading_bar.

This script was written as a collaboration for another project, and was imported to improve user friendliness.