POLITECNICO DI TORINO

Politecnico di Torino

Dipartimento di Ingegneria Meccanica e Aerospaziale

Multidisciplinary Design Optimisation of Launchers coupling the Design of the Propulsion System with the Trajectory Optimisation

Supervisor
prof. Paolo Maggiore

Co-Supervisor
prof. Edmondo Minisci

Candidate
Chiara de Cataldo

A thesis submitted for the Master's degree in
Aerospace Engineering

December 2020

Alle mie nonne,
le mie stelle fisse,
e ad ognuno dei fili sottili che reggono il mio mondo.

Abstract

In this work, a bi-level Multidisciplinary Design Optimisation process was implemented to design the propulsion system of a single stage HTOHL spaceplane.
The test case used as a reference for dimensions and aerodynamics is the CFASTT-1, a reusable launch vehicle developed for a SSTO mission whose concept is similar to the most recent ongoing project Skylon.
The engine modelisation was carried out through HyPro, a medium-fidelity modular model which employs the equations of fluid dynamics to design the main components of a hybrid engine in all its configurations, airbreathing (ejector ramjet, ramjet and scramjet) and rocket.
Due to its complexity, though, HyPro is too computationally expensive for the MDO process, so Surrogate Models were used instead.
Once set the range of work of the four configurations, HyPro was used to create a database for each engine, inclusive of significative and uniformly distributed values of Mach, altitude and the sensitive areas, i.e. those cross sectional areas of the modules which most affect the thrust and mass flow output in each engine mode.
These databases were the training sets for the Surrogate Models: neural networks specifically trained on the HyPro models data but much cheaper and more suitable for a process involving multiple iterations.
The main objective of the MDO process is to optimise the propulsion system for a given mission profile: an inner loop solves the control problem and finds the optimal trajectory, an outer loop optimises the engine performances, i.e. its thrust, mass flow, and volume (and, as a consequence, its weight).
The target driving the whole optimisation process is the objective function, which can be modified according to the purposes of the design process. In this specific case, the goal was to maximise the payload of the mission, thus finding the optimum hybrid propulsion system.
This process was tested on a typical SSTO mission profile for access to space.

Acknowledgements

First of all, my gratitude goes to both my supervisor prof. Paolo Maggiore and co-supervisor prof. Edmondo Minisci for the opportunity I had to be a part of this project at the University of Strathclyde, Glasgow.
I am particularly thankful for prof. Minisci's dedication and help, this experience gave me the chance to learn so much and I could not be more grateful for everything he taught me and encouraged me to delve into on my own.
I wish to thank prof. Ian Taylor and PhD student Mark De Luca from Glasgow University, communication and confrontation with them was extremely important in the first part of my work, especially for what regards the software HyPro and its features.
Last but not least, I am extremely thankful for the ICE Lab team members, who made me feel welcome right from the beginning and never failed to give me a piece of advice when I needed one. Working with them and being in touch with them, even if just through a webcam, was the incentive I needed in these strange and difficult times to keep on studying and doing my best. Ad maiora!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This section outlines the purpose and objective of this work of thesis. Starting with the general process of spacecraft design, an introduction to the Multidisciplinary Design Optimisation follows, with its concept and methods, together with an introduction to the software HyPro and the test case developed.

1.1 Spacecraft Design Process

A spacecraft can be referred to as an integrated system of functional components, or subsystems, which are in charge of different duties through the whole mission. The purpose of design phase is to correctly outline said subsystems and integrate them so that the mission objectives are correctly and efficiently achieved.

Generally speaking, the subsystems are the following:

- Payload: the reason why the mission was set up in the first place. It usually consists of instruments to carry on scientific experiments or to collect frequencies/images/samples;
- Structure: the shell, the "box" containing all the other physical subsystems;
- Thermodynamics: includes thermal control on board and all the devices for thermal protection for the re-entry phase;
- Propulsion: the system involved in the ascending phase, both in atmosphere and in orbit. Together with attitude control system, is appointed to orbital manoeuvring;
- Attitude control: dedicated to the pointing of the spacecraft;
- Mission analysis: essential to study all the phases of the mission, from launching to orbiting to descending (according to the type of mission);
- Power: the source that provides electrical power to the devices on board;
- Communications: the link to the ground control, used for uplinking commands and downlinking mission data.

- Cost evaluation: the whole business planning and budget meeting study.

As we can see, the design process is a multidisciplinary study that needs to take into account multiple needs and constraints, without exceeding the budget limitations.

1.2 Multidisciplinary Design Optimisation

MDO is an evolving methodology for the design of engineering systems involving many interacting subsystems and parts.
The concept of this method starts from a mathematical model of the system at hand, goes through a design oriented analysis, which allows a better understanding of the elements the optimisation will be focused on, and a sensitivity analysis, i.e. a study of the parameters whose variations mostly affect the behaviour of the system, and ends with the actual optimisation procedure.
The interdependencies between all the subsystems are the real protagonists of the MDO: only studying them carefully it is, in fact, possible to create an optimisation process that suits all the aspects of the design.
The Trajectory Optimisation is a set of methods used to find solutions to the Optimal Control problem. The solution depends on the objective function that describes the dynamics of the object and the constraints imposed.

The methods and algorithm used will be further discussed in the second chapter.

1.3 HyPro

The Hybrid Propulsion Optimizer is an engineering-level modular software package for the design and analysis of advanced hybrid propulsion systems [12].
This software has a modular structure, each module is connected to the other by nodes, which are C++ class containing information about the thermodynamic conditions, the velocity and the cross sectional area of a specific section within the engine.

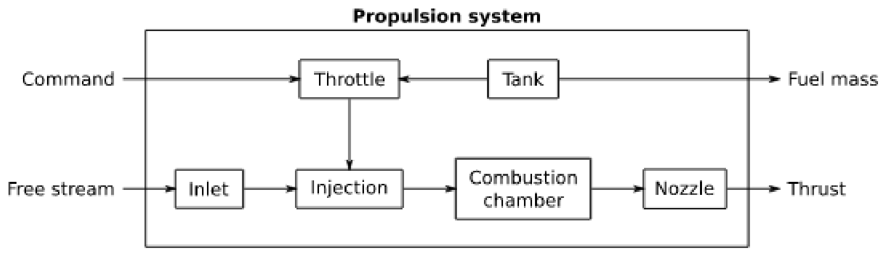

Figure 1.1: Example of the modular structure of the propulsion model for a scramjet/ramjet.

The gas dynamics modules which constitute the core of the software are the following:

- isentropic one-dimensional compression/expansion
- Fanno flow
- Rayleigh flow
- General one-dimensional balance equation solver

The modules are independent, which means that the combinations can be many and varied, and, in fact, the software has been used for many applications trying different configurations and design solutions of hybrid engines.

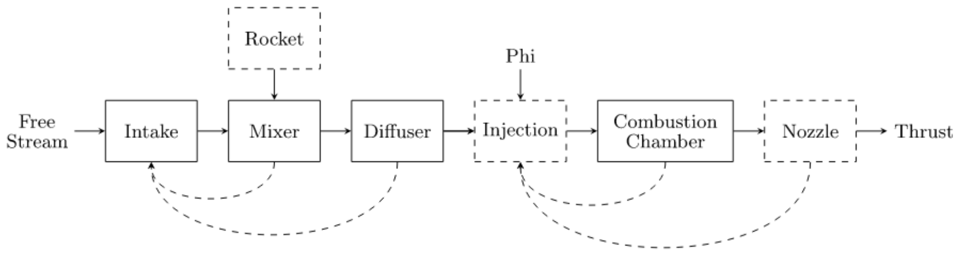

Figure 1.2: Information flow through the propulsion system model.

The information flows from the inlet to the nozzle, as does the flow, but, as we can see from figure 1.2, there's a mechanism which guarantees that the choking conditions do not ruin the model forecasts. In fact, it modifies the upstream flow so that the conditions for chocking within the system are precisely met.
Once the flow path is fully determined, the overall thrust produced by the engine

$$F = \dot{m}_2 u_2 - \dot{m}_1 u_1 + (p_2 - p_1)A_2 - D \tag{1.1}$$

can be calculated by performing a momentum balance across the appropriate control volume.
It goes without saying that, to improve the fidelity required for this HyPro model, a CFD model has been developed to produce a convenient data set for the validation and verification of the software.
To do so, OpenFOAM has been extended and developed to contain a number of algorithms appropriate to modeling high-speed propulsion systems.

Many tests were implemented to verify the accuracy of the model. It has been tested for a whole engine, first for a scramjet configuration, then against a similar package called SSCREAM, In the second case, the evaluation was performed on the ejector ramjet engine that was proposed for Hyperion (Figure 1.3)

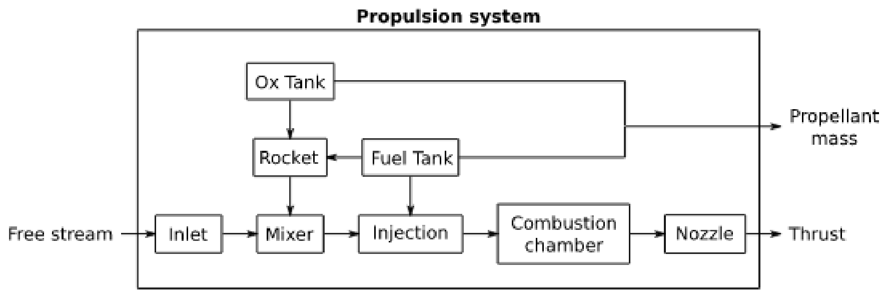

Figure 1.3: Example of the modular structure of the propulsion model for an RBCC engine.

It turned out the thrust predicted by HyPro was comparable with the one SS-CREAM had predicted. HyPro has also been validated against flight measurements of a ramjet test vehicle, its predictions turned out to be pretty close to the reference data with an error always lower than 10%.

The advantages in using this software are linked to its flexibility in representing a plurality of engine configurations. It provides the capability to be into an optimization loop, thus allowing designers to find the best engine for a selected mission.

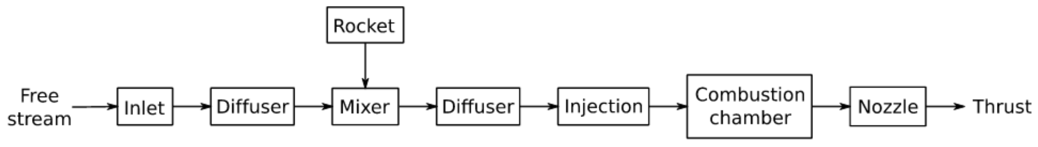

Figure 1.4: HyPro model of the Hyperion ejector ramjet engine.

1.4 Thesis structure and objective

The purpose of this work of thesis is developing an algorithm of bi-level optimisation which allows the trajectory optimisation in the inner loop and a performance optimisation in the outer loop. The details of this strategy will be further discussed and outlined in detail throughout these pages.

The first part of this work consisted in research on the topic and literary review on the methods of optimisation and the strategies of optimal control and trajectory optimisation.

The second part of this work involved a more accurate analysis of the software HyPro and its features. A series of tests have been carried out, then a first optimisation for all the configurations, allowing a better understanding of the sensitive parameters affecting the thrust and mass flow computations(sensitivity analysis)

The third and most significative part involved the creation of the bi-level optimisation loop: three different surrogate models were created for the most complex engine configurations(ejector ramjet, ramjet, scramjet), the rocket configurations was modeled with the simple rocket equation for thrust computation. Then the two interfacing loops were developed, the inner one concerning the solution of the optimal control problem, and so the trajectory optimisation, the outer one related to the engine performance.

We will discuss these phases more in details through the dedicated sections of this thesis.

Chapter 2

Background and Literature review

The access to space has always represented a huge challenge for human beings. It started with the first challenges in the '60s, this attitude of curiosity and this tension towards technologies always new almost never stopped.

In the constant search for new ways to get access to space, we are facing a growing attention towards re-usability of supplies and the considerable savings that follow. One of the strategies that are being investigated involves the so-called "space-plane" paradigm, which means that, hopefully, the future space vehicles will have an airliner-like mission profile. To do so, there's urgent need for new propulsion technologies and new ways to study and optimize the wholly new trajectory these vehicles will engage.

As previously described, the software used in this work was born exactly for this reason, which is modeling and optimizing the trajectory of a hybrid propulsion system for the access to space. In this section we will investigate the background of our research, i.e. the different techniques used in the recent years to carry out such modelisation and optimisation.

2.1 Optimisation under Uncertainty

The inclusion of a certain level of uncertainty is pivotal in the formulation of optimisation problems, it accounts for what cannot be surely determined and quantified, for whatever complexity is found in representing the system at hand.
The definition of uncertainty, in fact, embraces a whole set of factors: some bias of measurement, the absence of information, a partial or incomplete understanding of the mechanisms lying under the system.
Under these circumstances, a deterministic approach during the design phase results in a conservative design with high costs. This is why the field of optimization under uncertainty has been explored.

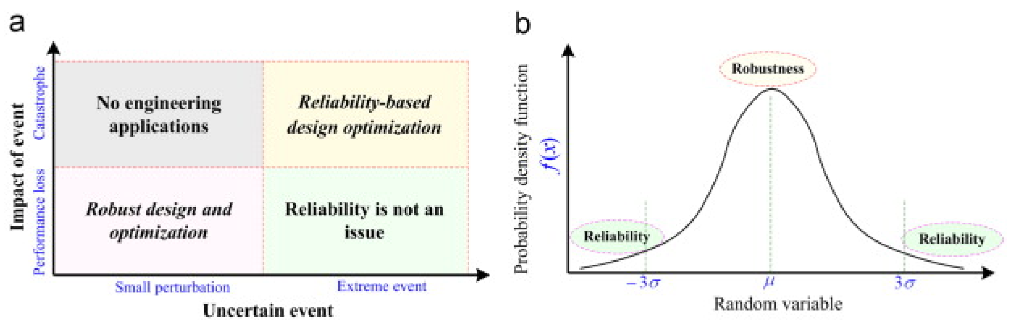

Figure 2.1: Two categories of uncertainty-based design [1]: (a) uncertainty-based design domains and (b) robustness and reliability in terms of probability density function

- *Stochastic programming*
  Includes stochastic programming with recourse and chance constraint programming.They give a stochastic representation of uncertainty, which is not always feasible.

- *Robust Optimisation*
  The robust design optimization (RDO) strategy accounts for the effects of variation and uncertainties by simultaneously optimizing the objective function and minimizing performance parameter variations [13].

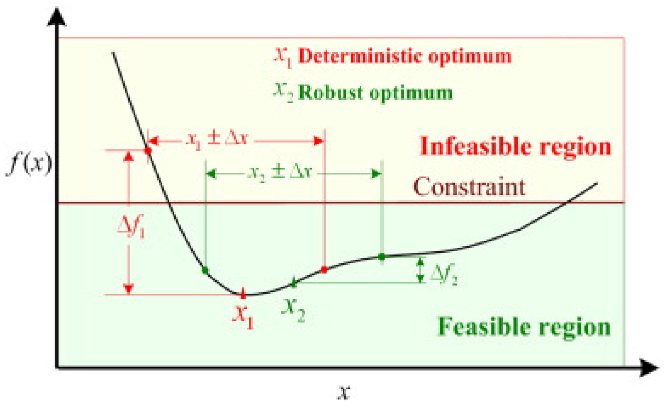

Figure 2.2: Graphic illustration of RDO [1]

- *Reliability Based Design Optimisation*
  The idea is to solve the design optimization problem with reliability constraints:the RBDO process deals with two optimisation models simultaneously. The first one is a design optimisation model which searches the feasible solution in an original random space.The other one is a reliability analysis model The uncertainties are modeled with a probability distribution, and the optimal solution has a chance of failure which lies in a pre-determined allowable range.

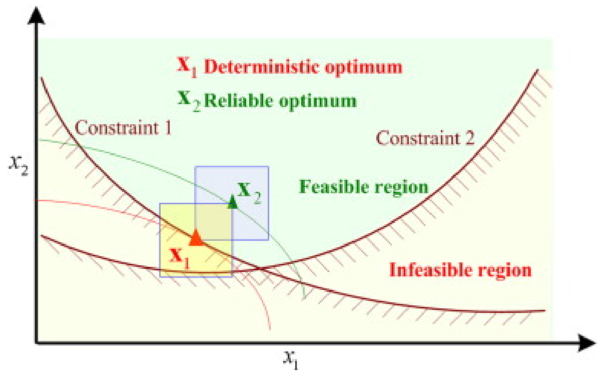

Figure 2.3: Graphic illustration of RBDO [1]

2.2 Design Optimisation

In the past decades, a whole suite of multidisciplinary analysis and optimization architectures has emerged. Given the multidisciplinary nature of the case we are considering, we have to carefully take into account various aspects of computation. In this environment, computational performance is just one of the aspects for architecture selection. Integration and communication among any disciplines need to be considered carefully

Optimisation architectures

One-variable-at-a-time Using RSM(Response Surface Method), feasible design are computed at numerous points in the design space and a surface is fitted to these points. So the optimisation is performed on this approximate representation of the design space.

Traditional multi-disciplinary design optimisation approach It's the standard optimisation approach, where interdisciplinary compatibility is enforced through some form of loop convergence criterion. It turns out there's need for more than one analysis to produce a single design candidate.

All-at-once-architecture The iterative loop of the standard approach is removed through the use of auxiliary variables and compatibility constraints.

Collaborative optimisation The design is decomposed into a number of subspaces and coordinated by a system-level optimizer. The gs we see on the diagram are the interdisciplinary compatibility functions, used to ensure that a consistent design solution is achieved upon convergence on the system-level optimisation problem.

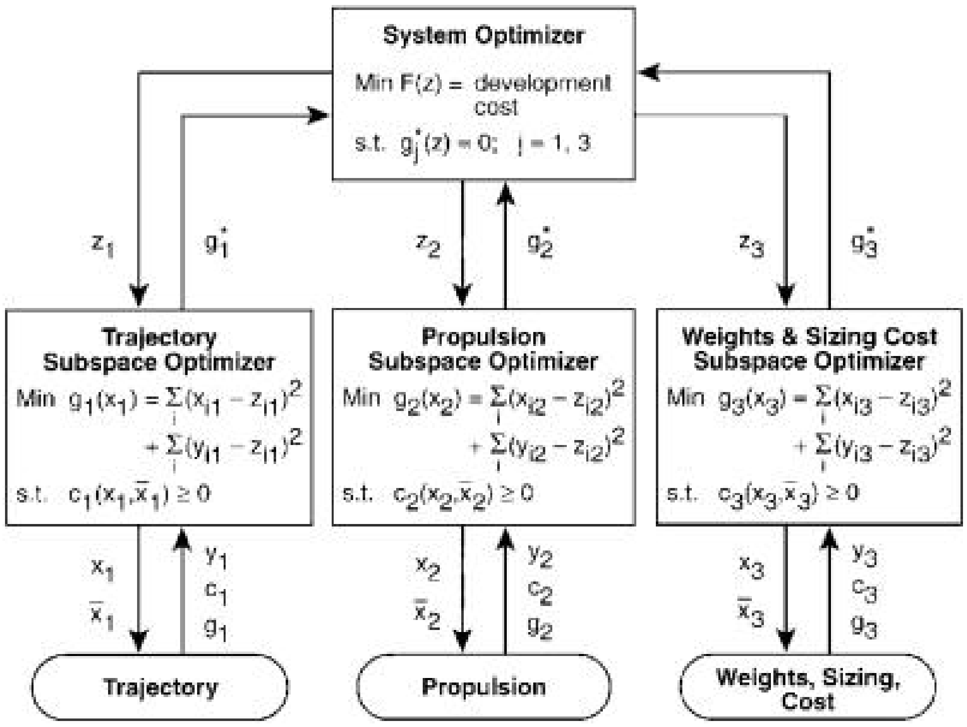
System Optimizer
Min F(z) = development cost
s.t. $g_j^*(z) = 0;\ j = 1, 3$
z_1
g_1^*
z_2
g_2^*
z_3
g_3^*
Trajectory Subspace Optimizer
Min $g_1(x_1) = \sum_i (x_{i1} - z_{i1})^2 + \sum_i (y_{i1} - z_{i1})^2$
s.t. $c_1(x_1, \bar{x}_1) \geq 0$
Propulsion Subspace Optimizer
Min $g_2(x_2) = \sum_i (x_{i2} - z_{i2})^2 + \sum_i (y_{i2} - z_{i2})^2$
s.t. $c_2(x_2, \bar{x}_2) \geq 0$
Weights & Sizing Cost Subspace Optimizer
Min $g_3(x_3) = \sum_i (x_{i3} - z_{i3})^2 + \sum_i (y_{i3} - z_{i3})^2$
s.t. $c_3(x_3, \bar{x}_3) \geq 0$
x_1 $\bar{x}_1$
y_1 c_1 g_1
x_2 $\bar{x}_2$
y_2 c_2 g_2
x_3 $\bar{x}_3$
y_3 c_3 g_3
Trajectory
Propulsion
Weights, Sizing, Cost

Figure 2.4: Collaborative optimisation architecture for launch vehicle design. [2]

Bi-level optimisation This technique couples an evolutionary multi-objective algorithm with a direct transcription method for optimal control problems.
We see it applied in a problem of optimisation of trajectory and shape of a reentry vehicle, and the process is described as follows: "an outer multi-objective evolutionary optimisation procedure manages the parameters defining the shape of the vehicle and considers as objectives and constraints the mean and variance of the values of the performance indices coming from an inner optimisation process handling the optimal control problem.

So there is an inner loop in which the optimal control problem is solved, computing the values of the performance. Once a deterministic solution is found in the inner loop, uncertainties are introduced by perturbing the deterministic quantity by a certain error. Statistical moments are then computed and sent to the outer loop for the determination of shape. And again, until the optimal shape for the optimal flight performance is reached.

2.3 Surrogate Based Optimisation

The simulation of articulated systems with multiple input and output is a complex, expensive and time-consuming process.The system under study is often a black box, with little information about the inner mechanisms and the way the output is generated.[14].

The purpose of surrogate modeling is to create a model that mimics the original system, but has less computational costs and is quicker.

The model is created from a database, constructed performing simulations (called samples) at key points in the domain. The model will have to approximate those samples and the overall behaviour of the system. [14]. For this reason, the surrogate models are also known as *metamodels* or *reduced-order models*.

In *local* surrogate modeling, local models are used to guide the optimisation algorithm towards a global optimum and are then disposed of. They only provide information about the impact of inputs on the outputs within the small region [15].The *global* surrogate modeling aims at creating a model that simulates the behaviour of the whole system and that can be used *instead of* the original model. Global surrogate modeling together with global optimisation algorithms can find a solution which can be considered global taking into account the accuracy of the models used and the range of data used to generate them. In fact, using a model in a domain that contains values which do not belong to the database it was constructed from cannot be considered feasible.

So local modeling is basically a tool to find the optimum, to guide the algorithm to make the optimisation quicker. Global modeling is a way to overcome the long computational time of the simulator by providing a fast but accurate approximation.[14].

Once this great distinction between these two macro categories has been cleared, we shall now focus on the main types of surrogate models we can find in applications and in literature.

In the aforementioned article dealing with the bi-level optimisation technique the dynamic characteristics of the vehicle is approximated by a data-fitted surrogate model.

Surrogate Based Optimisation (SBO) is widely used and we can find many examples of it in literature.
A surrogate is mostly employed when the need for quick and cheap computation prevails on the need for accuracy of the results (i.e. when the "real" model is too slow and expensive and therefore not suitable for an optimisation loop).

Given two different fidelity models [3]

high fidelity	**low fidelity**
f(x)	$\hat{f}(\hat{x})$
c(x)	$\hat{c}(\hat{x})$

To use a low fidelity model with a different number of design variables from the high fidelity function to be optimised it is necessary to find a relationship between the two sets of design vectors

$$\hat{x} = P(x)$$

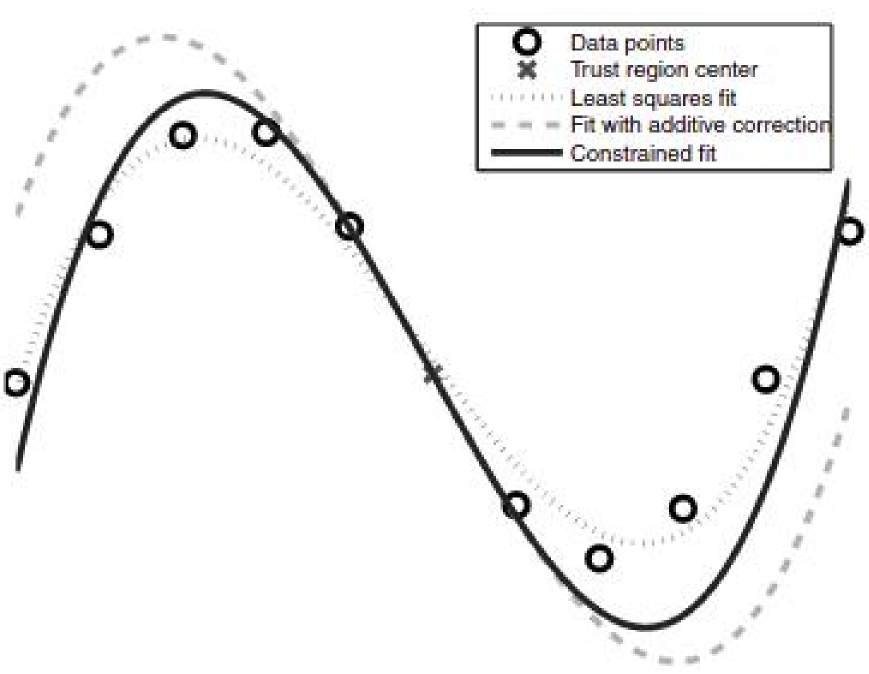

Figure 2.5: Simultaneous vs sequential dara fitting and enforcement of first order accuracy. [3]

To link the design variables in a variable parametrisation model it is necessary to *map* them. Mapping, can be quite easy, for example when the high and low fidelity models have the same set of physical equations.
Basically, the optimisation process is carried out on the low fidelity space, the mapping is then *inverted* to find a trial point in the high fidelity space. New data points near the trial point are then used to create the map for the next iteration. The method we just described is **TRMM**(Trust Region Model Management), which solves a sequence of optimisation sub-problems using the low-fidelity, provided that the solution of this dwells in a specific trust region.
However, space mapping does not always prove to converge in a TRMM framework [3], firs order accuracy is needed. One way to reach it is performing a corrected space mapping, which consists of basically running the mapping and the correction in parallel, embedding the correction within the space mapping.

In the Response Surface Methods (**RSMs**) [16], on the other hand, feasible designs are computed at numerous statistically selected points in the design space and a surface is fitted to these points.
In this case, the optimisation process is performed on this approximate representation on the design space.

Artificial Neural Networks (**ANNs**) can work with minimal knowledge about the structure of the function. This kind of model needs to be trained and updated

during the deign process by means of multi-fidelity evolution technique.
This is the basic idea of Evolution Control (**EC**), evaluating both the true and the surrogate model in order to reduce the computational time and preserve the accuracy of the final solution.

From literature [17] it appears that it is hard to get an idea, in terms of savings, of when it is useful to invest the additional effort of creating and using multi-fidelity surrogates. This is because MF surrogates combine the information of multiple models with a different cost and accuracy. Basically, we use this kind of surrogates when we have a complicated function and we either need a cheaper way to build a surrogate or we have a cheap low-fidelity function with a similar behaviour.
Fitting multi-fidelity surrogates isn't that obvious as well, because sometimes those varied levels of fidelity could actually cost more then the high-fidelity alone! The functions involved must be carefully evaluated, and the discrepancy functions need to have a low range of variation.

2.4 Trajectory Optimisation

To better understand how the problem of optimisation of the trajectory has been addressed, we will consider ascent trajectory optimisation for a single-stage-to-orbit vehicle with hybrid propulsion.
We can see that two different types of optimisation approaches are applied to this case.The idea is to develop a model-based software tool for the preliminary design.

The system is discretised into a finite set of unknowns. The dynamical system is transcribed into a problem with a finite set of variables, following which the resultant finite dimensional problem is solved by using a numerical parameter optimisation method.

The aim of the whole process is re-writing the optimal control problem as a non-linear programming problem (NLP).

Direct Shooting Method The system control equations are discretised and the dynamical system equations are integrated from initial to final conditions. The problem is that the objective functions can be evaluated *just at the end* of the iteration.

Another approach is the direct collocation method:this time both the controls and the state variables defining the system are discretised. The result is a large but sparse NLP, which can be solved with a sequential quadratic programming.
The approach proposed in the reference paper [18] is a mixed formulation which combines a population-based stochastic algorithm with a deterministic gradient-based method.

As we can see from the scheme below, there are many phases we have to go through to solve the NLP and reach the optimal results.

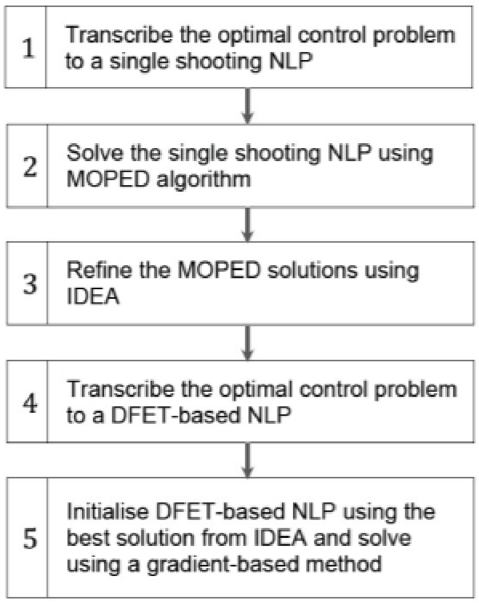
1
Transcribe the optimal control problem to a single shooting NLP
2
Solve the single shooting NLP using MOPED algorithm
3
Refine the MOPED solutions using IDEA
4
Transcribe the optimal control problem to a DFET-based NLP
5
Initialise DFET-based NLP using the best solution from IDEA and solve using a gradient-based method

Figure 2.6: General scheme of the trajectory optimisation process

1. Transcribe the optimal control problem into a single shooting NLP.

2. **2)** Solve the single NLP using MOPED. The NLP is solved using a hybrid Evolutionary Algorithm obtained coupling the MOPED (Multi Objective Parzen-based Estimation of Distribution) and a modified version of Inflationary Differential Evolution Algorithm.

3. Refine MOPED solution using IDEA.

4. **4)** Transcribe the optimal control problem using DEFT (method based on Finite Elements in Time).

5. **5)** Initialise DEFT-based NLP using best solution from IDEA and solve using gradient method

The research carried out in this thesis investigates two distinct operational phases, the first using air-breathing propulsion mode and the second using rocket propulsion.
The uncertainties both in atmospheric and aerodynamic models were accounted for and, from the results, it appeared that, for a good estimation of the performance of the vehicle, two types of uncertainty based analysis are carried out. The first evaluates the robustness of the nominal control law against the estimated uncertainties.

The other, instead, evaluates the optimal performance of the vehicle subject to uncertainties within the optimisation loop.
The output of the atmospheric and aerodynamic models are the deterministic values, which are then perturbed according to the following relation:

$$x_{unc} = x_{nom} + \varepsilon S_E x_{nom} \tag{2.1}$$

Where S_E is the sampling surface mapping.
As we mentioned, single shooting methods for the transcription of the optimal control problem are appropriate for dynamical systems which are subject to instabilities in a range of values of their control parameters.
Multiple shooting, instead, divides the time in multiple shooting segments $[t_0, t_1, ..., t_M]$ where the trajectory is integrated numerically within the interval $[t_i, t_{i+1}]$, with initial conditions x_i for all $i = 0, .., M-1$.

$$\dot{x}(t) = f(x(t), u(t)) \tag{2.2}$$
$$x(ti) = x_i \qquad t \in [t_i, t_{i+1}] \tag{2.3}$$

Then the interval $[t_i, t_{i+1}]$ is further discretised in NC control nodes

$$u_0^1, ..., u_{NC}^i \quad \text{for} \quad i = 0, .., M-1. \tag{2.4}$$

Continuity constraints are enforced

$$x_i = F([t_{i-1}, x_{i-1}]), \tag{2.5}$$
$$u_{NC}^{i-1} = u_0^1. \tag{2.6}$$

$F([t_{i-1}, t_i], x_{i-1})$ is the final state of the numerical integration on the interval $[t_{i-1}, t_i]$ with initial conditions x_{i-1}.
The first guess for an ascent optimisation problem is the quick run of a stochastic global search.
Then the NLP is solved using the Sequential Quadratic Programming algorithm.
For the ascent we have the following control vector $c = [t, \alpha, \mu, \tau]$ (t=time, α= angle of attack, μ= bank angle, τ=throttle).
If D is the search space and $\dot{m} = -\dot{m}_p$ ($\dot{m}_p$ is the propellant mass flow), then the objective function to solve is

$$\min_{c \in D} m(t = t_f),$$

since the purpose of the optimisation is to maximise the payload mass which could be carried to orbit.
The ascent trajectory is optimised to achieve a circular orbit while satisfying all the constraints.

2.5 Numerical Methods for Optimal Control

As previously mentioned, trajectory optimisation is a set of mathematical techniques used to find the "best" behaviour for a dynamical system [19].
The plurality of techniques is due to the plurality of problems and dynamical systems

that are to be encountered and optimised, according to the singular features of each of them.
First of all, we have to bear in mind that the core of any optimisation problem, including the trajectory optimisation, is the objective function, whose logic drives the whole process. The objective of the optimisation, then, is to **minimize** the objective function under some constraints.
The main assumption given in this process is that the time is continuous, which means that, given that x is the state variable, then

$$\dot{x} = \mathbf{f}(t, \mathbf{x}, \mathbf{u}) \tag{2.7}$$

The simplest problems are set on a single phase trajectory, which is not the case of the work developed here, though.
In fact, as was made clear enough in the previous chapters, the spacecraft works with an hybrid engine, which can count on four different configurations, so there will be a **switch** between the trajectory phases. The mechanism will be better outlined further along this dissertation, but is made clear to the reader that the single phase simplification used in this paragraph is just for clarity's sake in this first introduction to numerical methods applied to optimal control.

The trajectory optimisation problem, given the assumption previously mentioned, can be expressed in the following way

$$\min_{t_0, t_F, \boldsymbol{x}(t), \boldsymbol{u}(t)} J\left(t_0, t_F, \boldsymbol{x}(t_0), \boldsymbol{x}(t_F)\right) + \int_{t_0}^{t_F} w\left(\tau, \boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\right) d\tau \tag{2.8}$$

where the first term is the boundary objective function, the second term is the integral objective, while the term w expresses some quantity along the trajectory, $\boldsymbol{x}(t)$ is the *state* and $\boldsymbol{u}$(t) is the control. This type of problem assumes the form of a *Bolza* form: a formulation with just the first term is called a *Mayer* problem, while the second term only poses a *Lagrange* problem.
In the theory of optimal control we usually find problems expressed in the Mayer form.
Equation (2.4) comes with a series of constraints and boundaries:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}\left(t, \boldsymbol{x}(t), \boldsymbol{u}(t)\right) \qquad \textbf{system dynamics} \tag{2.9}$$

The system dynamics basically describes how the dynamical system changes in time.
Then we have the path constraint, which has to be fulfilled during the whole time and enforces limitations on the trajectory:

$$\boldsymbol{h}\left(t, \boldsymbol{x}(t), \boldsymbol{u}(t)\right) \leq 0 \qquad \textbf{path constraint} \tag{2.10}$$

As to the initial and final state of the trajectory we have

$$\boldsymbol{g}\left(t_0, t_F, \boldsymbol{x}(t_0), \boldsymbol{x}(t_F)\right) \leq 0 \qquad \textbf{boundary constraint} \tag{2.11}$$

Other constraints on the state and/or the control are

$$\boldsymbol{x}_{low} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{upp} \qquad \textbf{path bound on state} \tag{2.12}$$
$$\boldsymbol{u}_{low} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{upp} \qquad \textbf{path bound on control} \tag{2.13}$$
$$t_{low} \leq t_0 < t_F \leq t_{upp} \qquad \textbf{bounds on initial and final time} \tag{2.14}$$
$$\boldsymbol{x}_{0,low} \leq \boldsymbol{x}(t_0) \leq \boldsymbol{x}_{0,upp} \qquad \textbf{bound on initial state} \tag{2.15}$$
$$\boldsymbol{x}_{F,low} \leq \boldsymbol{x}(t_F) \leq \boldsymbol{x}_{F,upp} \qquad \textbf{bound on final state} \tag{2.16}$$

2.5.1 Transcription methods

Numerical methods used for solving optimal control problems are many and much varied: the first great distinction is between **indirect** and **direct** methods.
The **indirect methods** are based on the *calculus of variations*: a multi-point boundary value problem with piece-wise differential equations is solved, the solution being a candidate optimal trajectory. The objective is finding the minimum between those candidates, identifying them from maximum and saddle point.
The **direct methods** works, so to say, in the opposite way: state and control are discretised first. The problem is, then, transcribed to a **non linear programming** problem (**NLP**), whose form is something of the kind:

$$\min_{\boldsymbol{z}} f(\boldsymbol{z}) \tag{2.17}$$

subject to

$$\boldsymbol{g}(\boldsymbol{z}) \leq 0 \tag{2.18}$$
$$\boldsymbol{h}(\boldsymbol{z}) = 0 \tag{2.19}$$

and

$$\boldsymbol{z}_{low} \leq \boldsymbol{z} \leq \boldsymbol{z}_{upp} \tag{2.20}$$

2.5.2 Indirect methods

The calculus of variations is applied to the optimal control problem. The following first order necessary conditions for an optimal trajectory are valid [?]: there exists an n-vector function of adjoint variables $\lambda(t)$ and an m-vector function $\nu(t)$, such that with the so-called Hamiltonian function

$$H = \lambda^T f + \nu^T g \tag{2.21}$$

a multi-point boundary value problem in canonical form with piecewise defined differential equations results for $t_0 \leq t \leq t_f$

$$\dot{x} = \frac{\partial H}{\partial \lambda} = f, \tag{2.22}$$
$$\dot{\lambda} = -\frac{\partial H}{\partial x} = -\lambda^T \frac{\partial f}{\partial x} - \nu^T \frac{\partial g}{\partial x}, \tag{2.23}$$
$$g \leq 0 \tag{2.24}$$

The optimal control is determined by minimizing H with respect to u. For example, for H nonlinear in u and $t_0 \leq t \leq t_f$

$$\frac{\partial H}{\partial u} = \lambda^T \frac{\partial f}{\partial u} + \nu^T \frac{\partial g}{\partial u} = 0 \tag{2.25}$$

Any solution ($\mathbf{x}$(t),$\mathbf{u}$(t), $\boldsymbol{\lambda}$(t), $\boldsymbol{\mu}$(t), $\boldsymbol{\nu}$) is called an extremal: in an indirect method they are determined numerically.

Indirect shooting

An initial guess of the unknown boundary conditions at one end of the interval is made. Then the Hamiltonian is integrated to the other end. Then the terminal conditions obtained are compared to the ones given, and the process is repeated until the difference between the actual terminal conditions and the ones computed is lower then a certain threshold.

Indirect Multiple-Shooting Method

When the optimal control problem is *hyper-sensitive* (i.e. when time interval of interest is long in comparison with the time-scales of the Hamiltonian system in a neighborhood of the optimal solution [4]) indirect simple shooting method is not adequate any more. In the modified numerical method called *the multiple shooting method*, the time interval $[t_0, t_f]$ is divided into $M + 1$ sub-intervals, on which the shooting method is applied.
Continuity constraints are enforced at the interface of the sub-intervals:

$$\mathbf{y}(t_i^-) = \mathbf{y}(t_i^+) \tag{2.26}$$

where $\mathbf{y}(t)$ is the combined state - co-state vector:

$$\mathbf{y}(t) = \begin{bmatrix} \mathbf{x}(t) \\ \lambda(t) \end{bmatrix} \tag{2.27}$$

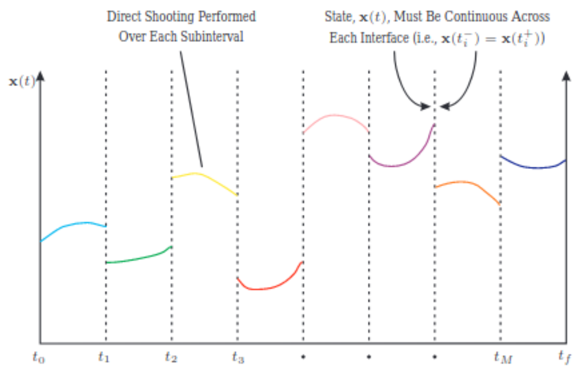
Direct Shooting Performed Over Each Subinterval
State, $\mathbf{x}(t)$, Must Be Continuous Across Each Interface (i.e., $\mathbf{x}(t_i^-) = \mathbf{x}(t_i^+)$)
$\mathbf{x}(t)$
t_0
t_1
t_2
t_3
t_M
t_f

Figure 2.7: Schematic of the Indirect Multiple-Shooting Method [4]

Indirect Collocation Methods

The state and co-state are parametrized using polynomials, and the problem is, then, solved using a root finding problem where the vector of unknown coefficients $\mathbf{z}$ consists of the coefficients of the piecewise polynomial [4].

2.5.3 Direct methods

As previously mentioned, indirect methods discretize the trajectory optimization problem converting it into a non linear problem. In this case too we have many types of numerical methods.

Direct shooting method

The trajectory is approximated using a simulation. The decision variables in the non-linear programming are an open-loop parametrization of the control along the trajectory [19].

Direct multiple shooting

Also called *parallel shooting*. In this method the trajectory is divided into segments, onto which the shooting method is applied. It is usually employed for more challenging trajectory optimisation problems, because it is more robust.

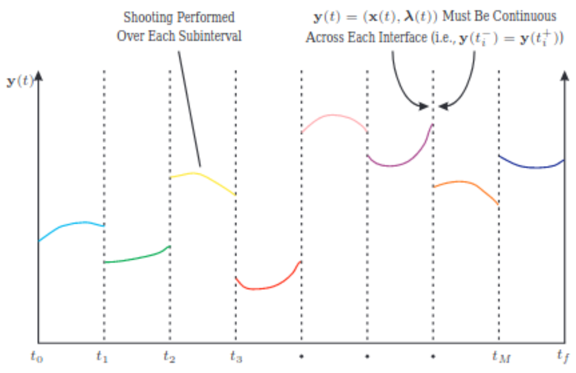
Shooting Performed
Over Each Subinterval
$\mathbf{y}(t) = (\mathbf{x}(t), \boldsymbol{\lambda}(t))$ Must Be Continuous
Across Each Interface (i.e., $\mathbf{y}(t_i^-) = \mathbf{y}(t_i^+)$)
$\mathbf{y}(t)$
t_0
t_1
t_2
t_3
t_M
t_f

Figure 2.8: Schematic of the Direct Multiple-Shooting Method [4]

Pseudospectral (Global Orthogonal Collocation) Methods

A pseudospectral method is a global form of orthogonal collocation, i.e. in a pseudospectral method the state is approximated using a global polynomial and collocation is performed at chosen points [4].

The continuous Bolza problem is transcribed to a nonlinear programming problem.
First, the original time interval $I = [t_0, t_f]$ is divided into S sub-intervals or segments I_x, (s=1,....,S) such that $I_s = [t_{s-1}, t_s]$ and

$$\bigcap_{s=1}^{S} I_s = 0, \qquad \bigcup_{s=1}^{S} I_s = I \tag{2.28}$$

where 0 is the empty set. Within each sub-interval, the time $t^{(s)}$ is transformed to the interval [-1,1] via the affine transformation

$$\tau = \frac{2t^{(s)}}{t_s - t_{s-1}} - \frac{t_s + t_{s-1}}{t_s - t_{s-1}} \tag{2.29}$$

where $\tau \in [-1, 1]$ and denotes the transformed time in I_s. Next, assuming that the approximation has the same order in each sub-interval I_s, the state is approximated using a basis of $N + 1$ Lagrange interpolating polynomials $\mathcal{L}_i(\tau)(i = 0, ..., N)$ as

$$(\mathbf{x})^{(s)}(\tau) \approx \mathbf{X}^{(s)}(\tau) = \sum_{i=0}^{N} \mathbf{X}^{(s)}(\tau_i)\dot{\mathcal{L}}_i(\tau) \tag{2.30}$$

where N is the number of Legendre-Gauss(LG) points [defined as the roots of the N_{th} degree Legendre polynomial $P_N(\tau)$] in segment $s \in [1, ..., S]$. Differentiating Eq. (2.25) gives

$$\dot{\mathbf{x}}_{(s)}(\tau) \approx \dot{\mathbf{X}}^{(s)}(\tau) = \sum_{i=0}^{N} \mathbf{X}^{(s)}(\tau_i)\dot{\mathcal{L}}(\tau) \tag{2.31}$$

The derivative of the ith Lagrange polynomial at the LG points $\dot{\mathcal{L}}_i(\tau_k)$, (k=1,...,N), can be represented in a differential approximation matrix $D \in \mathbb{R}^{Nx(N+1)}$ and is determined offline. The dynamic constraint is transcribed into algebraic constraints via the differential approximation matrix as

$$\sum_{i=0}^{N} D_{ki}\mathbf{X}_i^{(s)} - \frac{t_s - t_{s-1}}{2}\mathbf{f}(\mathbf{X}_k^{(s)}, \mathbf{U}_k^{(s)}, \tau_k; t_{s-1}, t_s) = \mathbf{0}, \qquad (k = 1, ..., N) \tag{2.32}$$

where $\mathbf{X}_k^{(s)} \equiv \mathbf{X}^{(s)}(\tau_k) \in \mathbb{R}^n$ and $\mathbf{U}_k^{(s)} \equiv \mathbf{U}^{(s)}(\tau_k) \in \mathbb{R}^m$, (k=1,...,N). The final state of each segment $\mathbf{X}_f^{(s)}$ is also included as a variable in the NLP, where $\mathbf{X}_f^{(s)}$ is defined in terms of $\mathbf{X}_k^{(s)}$, $(k = 0, ..., N)$ and $\mathbf{U}_k^{(s)}$, (k=1,...,N) via the Gauss quadrature

$$\mathbf{X}_f^{(s)} \equiv \mathbf{X}_0^{(s)} + \frac{t_s - t_{s-1}}{2}\sum_{k=1}^{N} w_k\mathbf{f}\left(\mathbf{X}_k^{(s)}, \mathbf{U}_k^{(s)}, \tau_k; t_{s-1}, t_s\right) \tag{2.33}$$

The N LG points plus the initial and final point define the set of n= $N+2$ discretization points or nodes. Also note that the control is not discretized at the boundaries. The continuous cost function is approximated using a Gauss quadrature as

$$J = \Phi\left(\mathbf{X}_0^{(1)}, t_0, \mathbf{X}_f^{(S)}, t_f\right) + \sum_{s=1}^{S} \frac{t_s - t_{s-1}}{2} \sum_{k=1}^{N} w_k g\left(\mathbf{X}_k^{(s)}, \mathbf{U}_k^{(s)}, \tau_k; t_{s-1}, t_s\right) \qquad (2.34)$$

where w_k are the Gauss weights associated with the LG points. Next, the boundary and linkage constraints are expressed as

$$\phi\left(\mathbf{X}_0^{(s-1)}, t_{s-1}, \mathbf{X}_f^{(s)}, t_s\right) = 0 \qquad (s = 1, ..., S) \qquad (2.35)$$

Furthermore, the path constraint

$$\mathbf{C}[\mathbf{x}(t), \mathbf{u}(t), t] \leq 0 \qquad (2.36)$$

is evaluated at the LG points as

$$\mathbf{C}\left(\mathbf{X}_k^{(s)}, \mathbf{U}_k^{(s)}, \tau_k; t_{s-1}, t_s\right) \leq 0 \qquad (k = 1, ..., N; s = 1, ..., S) \qquad (2.37)$$

The cost function of Eq. (2.310) and the algebraic constraints of Eqs (2.28), (2.29), (2.31), (2.33) define an NLP whose solution is an approximate solution to the continuous Bolza problem.
If the sub-interval used is *one*, the GPM is employed as a global collocation method.

Chapter 3

Hybrid propulsion technology

3.1 Combined Cycle Propulsion

The HYbrid Propulsion Optimizer is an engineering-level, modular software package for the design and analysis of advanced aircraft propulsion systems, in particular, combined cycle propulsion system.[20] In fact, the flexibility of the software makes it possible to re-create a variety of engine configurations, from ramjet to rocket and, in general, varied combinations of propulsive components.

Combined Cycle Propulsion is one of the most promising solutions for space-plane propulsion and it includes the **Rocket Based Combined Cycle** and the **Turbine Based Combined Cycle**.

The RBCC operates in ejector mode from take-off to Mach $2 \div 3$, then it operates in ramjet mode until around Mach 5, then scramjet until Mach 11, then rocket.
The TBCC differs from the RBCC for the presence of turbo-machineries, the engine cycle is turbo-based. Essentially, while in the RBCC rocket and ramjet share the same flow path, this is not true for the TBCC, thus the need for extra structural components and weight.
The design process for these kinds of engine involves airframe/engine integration techniques , cooling systems, thermal protection, efficient atomizing, mixing, flame holding and combustion organization technique of inlet and nozzle.[21]

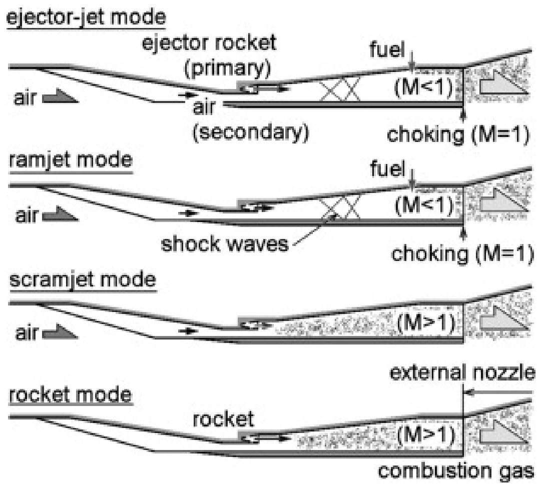

Figure 3.1: Operating conditions for RBCC engine.

Multi-cycle propulsion concepts include separate systems for each operating mode(e.g. a turbojet system and a rocket system). These systems may be operated separately or in parallel in order to maximize the performance of the overall vehicle.
Combined cycle propulsion systems integrate various operating modes into a single set of hardware components to minimize redundant systems and reduce propulsion system weight. [22]

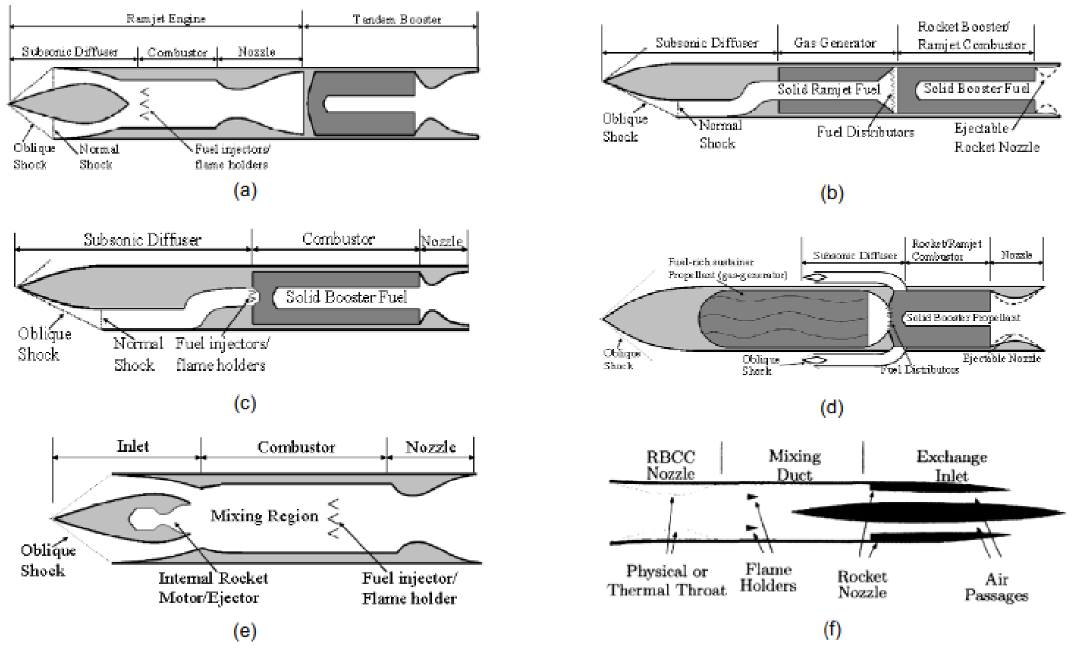

Figure 3.2: Change from a combined propulsion system to a combined-cycle engine system.

The amount of research on combined-cycle propulsion systems is outstanding, many space-focused firms have destined their funds to the improvement of the technology involved and there's plenty of studies on the best optimisation strategies to apply on those specific designs.
Even though not a single SSTO mission has still been completed, RBCC seems to be the most promising architecture to reach this ambitious goal.

3.2 RBCC

The most important thing to keep in mind when talking about RBCC architectures is that there is a huge difference between combined propulsion and combined-cycle propulsion. The former is, in fact, a juxtaposition of engines which work on their own in different flight conditions.[23]
Figure 3.2 shows some examples of combined propulsion systems together with combined-cycle propulsion systems: the first one, Figure 3.2a) is a tandem solid booster with a canister ramjet, in which the rocket is mounted behind the ramjet and works as booster. Figure 3.2b) shows a solid fuel integral rocket ramjet (IRR), in which the ramjet and rocket share one common flowpath. Similarly, there is a liquid fuel IRR configuration in Figure 3.2c), which works likewise. Figure 3.2d) shows an air-ducted rocket: when the ramjet is operating a gas-generator creates fuel-rich propellant, which then reacts with the air in the combustor and produces thrust.
Must be noted, the solid rocket in these engines is disposable, so once it burns out it is of no use any more. So, it goes without saying, the whole concept of reusability the RBCC technology seeks to achieve does not apply to the propulsive architectures Figure 3.2 a-d).

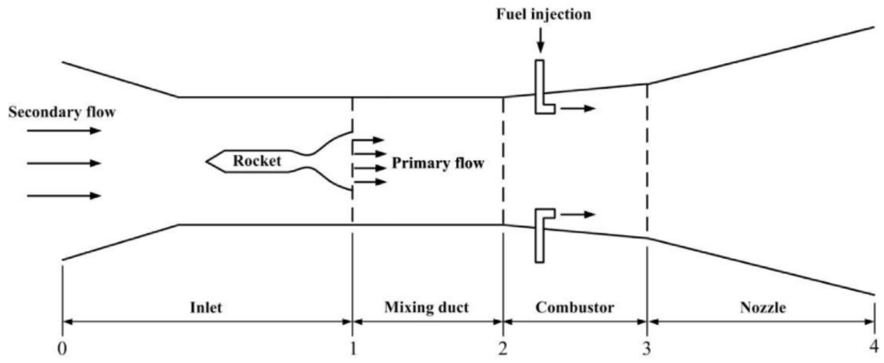

Figure 3.3: Schematic of the RBCC engine.

The RBCC, as previously stated, is designed to work in four modes: ejector ramjet, ramjet, scramjet and rocket. The aircrafts that make use of this technology are

designed to take off from a runway. So, from take-off to approximately Mach 2 the engine works in ejector mode: the eject rocket is ignited and the fuel rich plume mixes and reacts with the secondary flow in the main combustor.
Once the aircraft has gained speed, usually between Mach 2 and Mach 5, the engine switches to ramjet mode.
The free flow is compressed and slowed down after a normal shock wave in the inlet, before mixing and reacting with the fuel in the combustor. The eject rocket is shut down, but can also supply fuel in the combustor.
From Mach 5 the engine switches to scramjet mode, in which, even if the flow has slowed down while passing through the inlet, it is still supersonic in the combustion chamber. The fuel is injected by a transverse jet in the combustion chamber.
This whole process takes place while the spacecraft is ascending, the altitude is increasing and the air density is decreasing.
The transition from scramjet to pure rocket happens when the inlet is no longer capable to capture enough airflow to keep the combustion going in the chamber. This is when the eject rocket is re-ignited and the inlet is closed.
In rocket mode there is no airbreathing, no secondary flow and the thrust is entirely produced by the rocket.
The transition from one mode to the other needs to be carefully calibrated, the engine architecture must guarantee continuity in the production of thrust, thus ensuring the correct operation of the engine throughout the mission.
We shall now study the main engine configurations in detail.

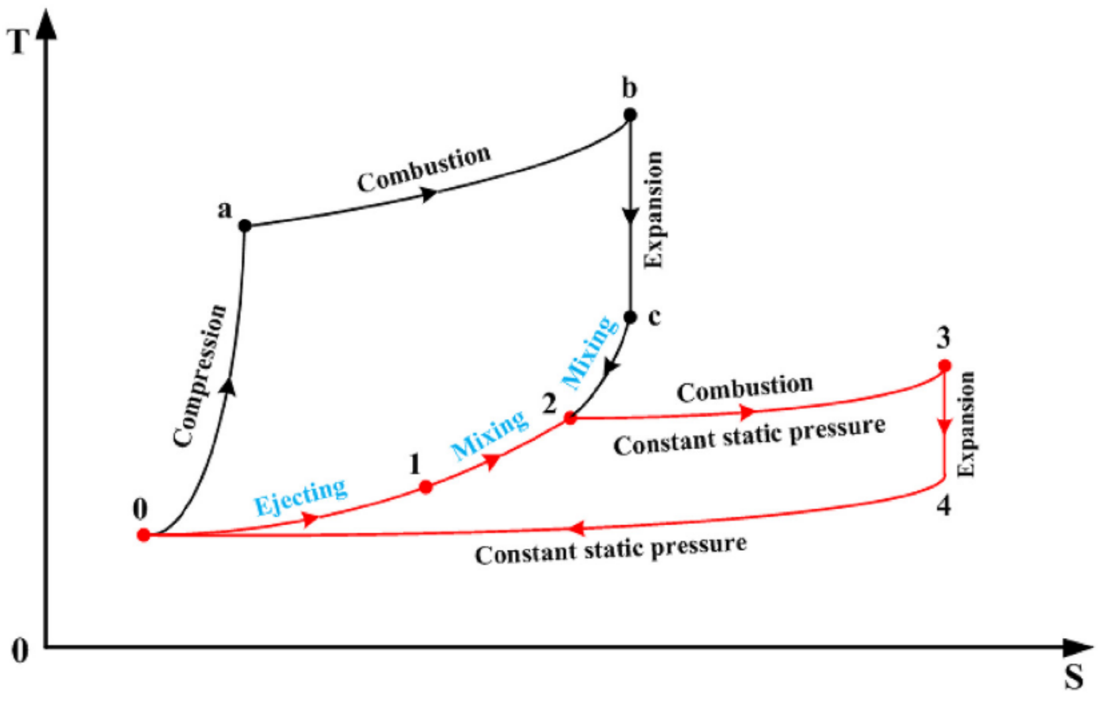

Figure 3.4: Ideal thermodynamic cycle of the RBCC ejector mode [5].

3.2.1 Ejector Ramjet

The ejector-ramjet, also called air-augmented rocket, is a technology developed to take advantage of the rocket propulsion in the air during flight in the atmosphere, basically combining a rocket and a ramjet in the same system[24].
From take-off through to the low-supersonic flight regime, the ejector mode is used to accelerate and lift the vehicle in the first stage of the mission, taking advantage of the engine height thrust-to-weight ratio [6].
The rocket acts as primary, and generates thrust. Then the thrust is augmented through the fuel afterburning with the ejected air in air duct downstream of the rocket. The air is mixed with the products of the rocket and then the gases are diffused to a low subsonic Mach number.
The thrust produced in the ejector mode is higher than the one the single rocket can produce.
The concept of the RBCC itself was born together with the development of air-augmented rocket technologies, and the ejector ramjet is still considered the most distinctive part of this combined-cycle propulsion system. First of all, it is the engine that "puts everything together" from the beginning, combining ramjet (typical airbreathing technology) with rocket. Take off would not be possible with a ramjet only, but the clever combination of a rocket plume and a chamber makes all the magic work (at least theoretically).

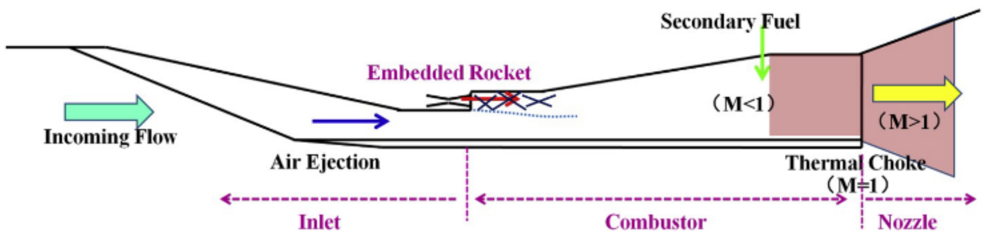

Figure 3.5: Schematic of ejector mode operation of RBCC engines [6].

Following the evolution of research of this type of engine we can see how the structure changed, the improvement probably enhanced by new materials, better simulation and CFD models, optimisation algorithms and so forth. The preliminary design phase has known a huge development thanks to the introduction of mathematical modeling tools and one-dimensional/quasi-one dimensional analysis, which make it possible to foresee issues and ruptures which may occur during the operational life of the engine. Comprehensive analysis include many of the anticipated effects during the practical implementation of RBCC engines[6], and thanks to the modeling studies it is possible now to summarize the main ways to improve the performances of an ejector ramjet. First of all, the rocket should eject as much air as possible into the RBCC engine. The amount of air which then reacts with

propellant and is then expelled from the nozzle determines the amount effectively propelled by the engine and, as a consequence, its thrust.
Figure 3.6 shows how the air ejection ratio affects the thrust and how the quality of good air suction and just as good diffusion can have a positive impact on the performances of the engine.

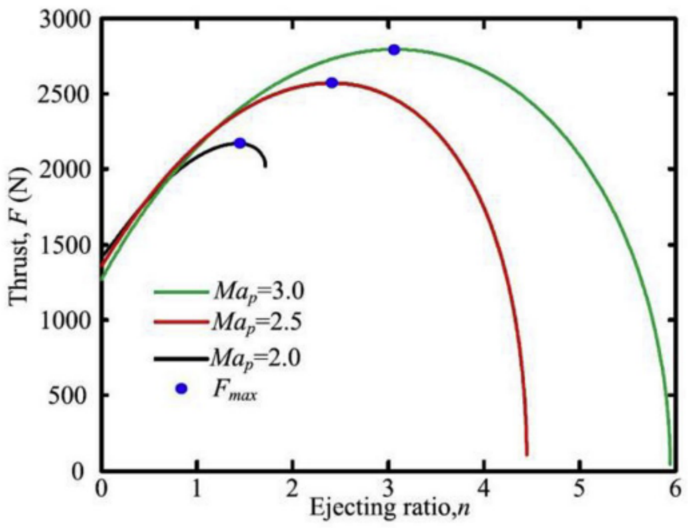

Figure 3.6: Influence of the air ejection ratio on the engine thrust [7].

Another pivotal aspect to ensure is the efficient mixing of the air with the rocket plume and a good and effective combustion. The generation of a thermal choke in the chamber is important to guarantee a high pressure level in the chamber, for the most efficient combustion.
According to the need to satisfy the aspects just outlined and in the search for a good propulsion quality for the RBCC, many researches have been conducted and technologies developed (or are being currently developed). The reader will find plenty of examples in the references mentioned in this thesis and not reported here for the sake of brevity and not to go too far from the purposes of this work.

3.2.2 Ramjet

The ramjet engine is the simplest air-breathing engine.
It consists of a diffuser, a combustion chamber, and an exhaust nozzle. Air enters the diffuser, where it is compressed before it is mixed with the fuel and burned in the combustion chamber[9].
The ramjet cannot operate from a standing start, but must first be accelerated to a high speed by another means of propulsion (in the RBCC, this is the case of the ejector ramjet).

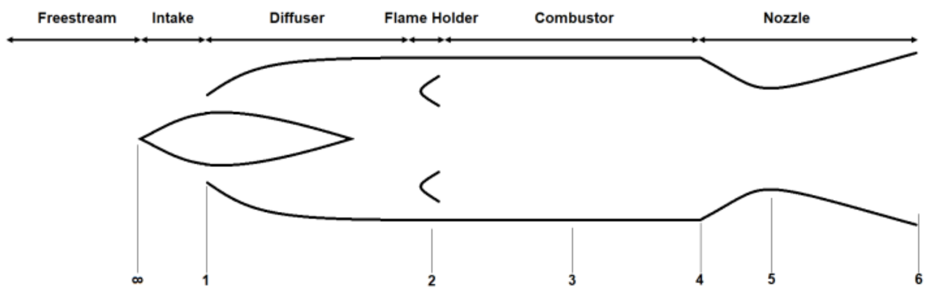

Figure 3.7: Schematic diagram of a ramjet engine.

The ramjet itself does not have moving parts of its own, and depends solely on its forward motion to compress the air [25]. However, unlike a turbojet or rocket engine, the ramjet requires an auxiliary boost system to accelerate it to its supersonic operating regime.
Although ramjets can actually operate at subsonic flight speeds, they are more suitable for the supersonic regime because of the increasing pressure rise due to higher flight speeds.
The flow entering the inlet is compressed through a series of shocks (supersonic compression). The isentropic deceleration taking place in this phase is crucial because the flow entering the combustion chamber cannot be supersonic. It is actually extremely difficult to get the desired pressure ratio between ambient and combustion chamber because the stagnation pressure losses due to the shocks are not negligible.
Afterwards, the air is mixed with the fuel sprayed by the injectors and enters the combustion chamber, where it is possible to find a flameholder, i.e. a system to stabilize the flame. The products of the combustion expand at high speed in the nozzle and produce thrust.
One of the main disadvantages of the ramjet engine is that the pressure ratio is strictly limited by flight speed and efficiency of the diffuser, which means that the ramjet cannot develop static thrust and - as previously mentioned - cannot accelerate a vehicle from a standing start [9].

3.2.3 Scramjet

Scramjets are supersonic combustion chambers, a technology that makes it possible for the ramjet to be applied for hypersonic flight[9]. The use of supersonic combustion requires fuel to be injected into and mixed with a supersonic stream, without excessive losses.
Unlike the subsonic combustion ramjet's terminal normal shock system, the combined effect of heat addition and diverging area in the scramjet's combustor, plus the absence of a geometric exit nozzle throat, generate a shock train located at and upstream of the combustor entrance. The strength of this shock system depends on the flight conditions, inlet compression or inlet exit Mach number M_4, overall engine

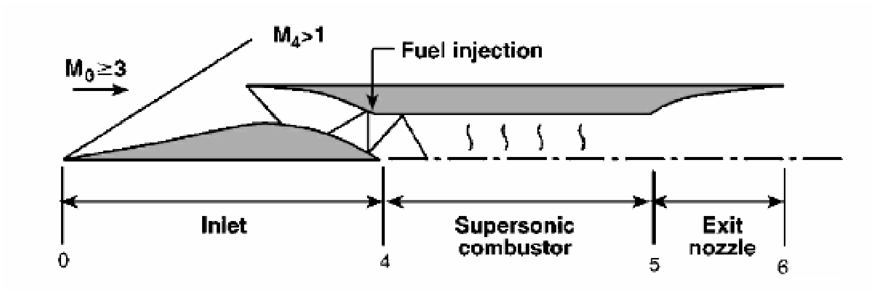

Figure 3.8: Schematic diagram of a scramjet engine.

fuel/air ratio ER_0, and supersonic combustor area ratio.[25].
Thanks to the combination of heat addition in a supersonic airstream and the absence of a geometric throat, the scramjet operates efficiently over a wide range of flight conditions, $M_0 > 5$.

Chapter 4

Software insight

This chapter is dedicated to the software HyPro, the purpose of its creation and development as a preliminary design tool and its architecture and structure.

4.1 Structure overview

As briefly mentioned in the introduction to this thesis, the software *HyPro* has a modular structure, which allows a better simulation of a variety of engine configurations.
Modules are connected by nodes, which are defined by constraints on the equations of the fluid dynamics controlling the flow through the engine. The intake, as it is modeled in this software, is made up of many modules, to better include the complexity of the physics of this component.
Depending on the flight speed and the mass flow demanded by the engine, the inlet may have to operate with a wide range of incident stream conditions. The node N_1 in Figure 4.1 represents the conditions of the flow stream external to the inlet (the pre-intake conditions).

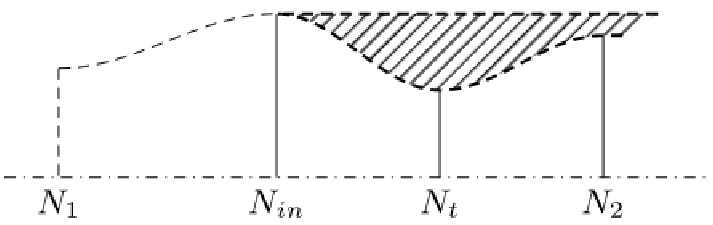

Figure 4.1: Definition of nodes of the convergent-divergent intake model

The "capture area" is strongly affected by the flight conditions, and so is the pre-intake node, especially for high speed flight.
N_1 represents the intake mode, N_t is the throat.
The importance of this node is due to a very specific problem typical of inlet in supersonic flight, which is the **starting**. It must not be forgotten that a supersonic

flow cannot enter the engine. This difficulty is due to shocks that arise during an abrupt deceleration process, and it needs not to be related to boundary layer behaviour.

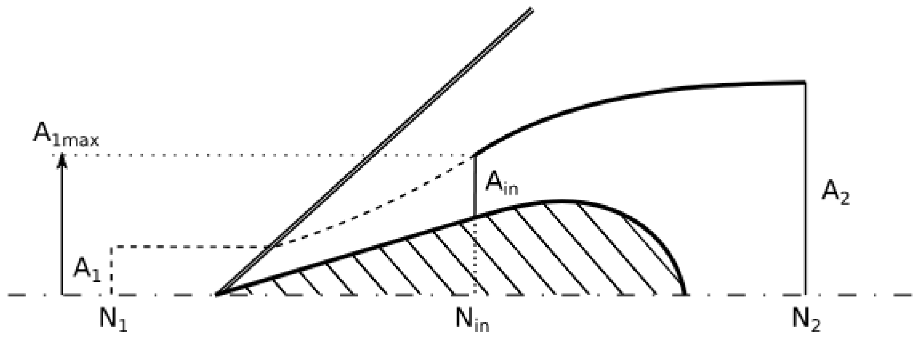

Figure 4.2: Definition of nodes for the central body intake model

We will now consider a converging-diverging diffuser that is one-dimensional and isentropic. In supersonic conditions, in order for the flow to "sense" the inlet and the flow around it, the spilled air must be reduced to subsonic speed upstream of the inlet plane. Then, once the air speed increases, the shock enters the convergence and at the designed speed the shock enters the throat.
The shock is digested and positioned in the divergent with a much lower pressure drop (inlet starting).
That is the reason why the throat cross sectional area is an important geometrical parameter, because the ratio $\frac{A_i}{A_t}$ depends on the capability of the inlet to swallow the shock when there is an acceleration or if there's the need for other strategies, like, for example, variation of geometry at constant speed.

Pinch Point represents a point along the streamline where congestion occurs or is likely to occur.
It is modeled to take into account the discontinuities in the flow and the tridimensional effect that may take place in specific parts of the engine, especially where we have junctions, rotations and mixture of flows (see injection, for example).
The nodes called primary are related to the primary flow which comes from the rocket system, while the secondary flow is the streamline coming from the inlet, when the engine is still flying in the atmosphere and there is air-breathing propulsion.
Primary and secondary merge into the mixer, represented in Figure 4.3. As can be seen from the picture, in HyPro the structure of the mixer is such that in between N_1 and N_2 lies a third node, which models the injection of the primary flow into the mixer.
The injection is the mixing between the injected fuel and the main air stream.

Last but not least, a model which really needs to be accounted for is the rocket: it

1	Pre-Intake
2	Intake
3	Throat
4	Pinch Point
5-6	Primary(In&Out)
7	Mixer End
8	Rocket Outlet
9	End Diffuser
10	Injection
11	End Chamber
12	Nozzle

Table 4.1: HyPro nodes nomenclature.

is created combining a combustion chamber module and a nozzle module. As for the chamber, the assumption of complete combustion is given. To take into account any incompleteness, there is a combustion efficiency factor.

The nozzle is the final component of the engine: its geometry is fixed in the study at hand, but the parameters of the model make it possible to account for a variability of the geometry and the incidence of the ejected flow.We will see further ahead that the cross sectional area of the nozzle, as the inlet one, will have a pivotal importance in the computation of the engine performances.

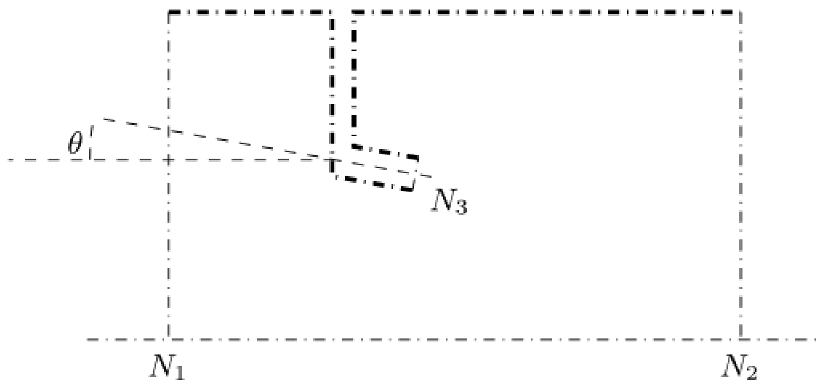

Figure 4.3: Definition of nodes for the mixer module

4.2 Matlab Interface

HyPro is written in C++ and is launched from command window in an Ubuntu environment.

For an easiest integration with the optimisation algorithms and a future improvement of the applications of the software, a Matlab interface was created.

Instead of editing the main *cpp* files containing the values of the cross sectional areas, Mach, altitude and mode, the script was modified so that the inputs are all read from a file, written from Matlab.

The source files do not have to be compiled every time a test is performed, and this considerably cuts the times.

This makes it possible to run the software faster and from a much easier interface, which can be integrated to other functions, as, in fact, it was for the purpose of this thesis.

4.3 Sensitivity Analysis

We shall now qualify and quantify these uncertainties: to do so, we carried out a sensitivity analysis, whose purpose was to distribute this uncertainty over some sources in the model.

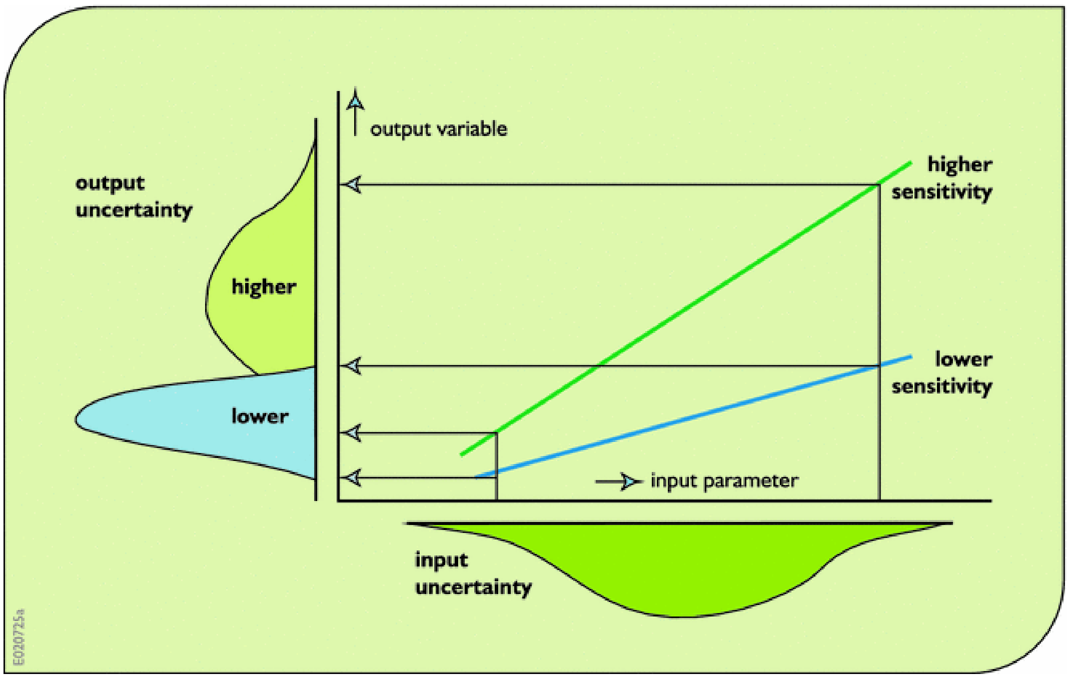

Figure 4.4: Uncertainty Analysis [8]

When dealing with a model whose output is affected by uncertainty, the uncertainty analysis and the sensitivity analysis are usually carried out in tandem. In fact, the first determines what is the extent of the variability of the output in response to the variation of the input, and the second focuses on finding out which factors/elements are responsible for said variability.

Using the MATLAB Optimization Toolbox™we set up an optimisation for the four different engines to analyse how the thrust and mass flow changed accordingly to the variation of the inputs.

4.3.1 Function settings

The function used was `fmincon` function.

`x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)` (4.1)

Where `x0` is the vector of initial cross sectional areas, the t_0 of the optimisation process. A and `b` are the linear inequality constraints, `Aeq` and `beq` are the linear equality constraints, `lb` and `ub` are the lower and upper boundaries, i.e. the limitations set to the values the cross sectional areas can have. These boundaries were chosen after previously running some simulation and seeing how the variation of specific areas affected the output. In fact, the idea was to avoid values which might cause negative feedback actions (for example choking) in some modules due to the unnecessary small value of the interface areas. It is true that in some flight conditions issues must come out and compromise the operation of the engine, but since we are dealing with a medium fidelity model and the purpose of this work is the optimisation of the engine on a specific trajectory and not an in depth study of the HyPro model, we can avoid these criticality and work with the engine at its best.
Then, `c` is the nonlinear constraint and is a function that accepts a vector or array `x` and returns two arrays, `c(x)` and `ceq(x)`.

`function[c, ceq]= mynonlincon(x)` (4.2)

In our case the non linear constraints were both void. Lastly, `fun` is the objective function, the logic driving the optimisation: maximising the thrust and minimising the mass flow.
Additional output `grad` shows the gradient of `fun` at the solution x. To better grasp the slope of the gradient curve, the step size can be reduced until convenience.

The risk in these optimisation problems (see figure 4.5) is to end up finding a local minimum instead of the desired global minimum, that is why studying the gradient of the function can be helpful not to fall for a wrong value of x as optimum. It can help to change starting points of the optimisation, as well.

Studying the gradient it was possible to see that the functions (thrust and/or mass flow) had major variations in correspondence with some specific nodes, and

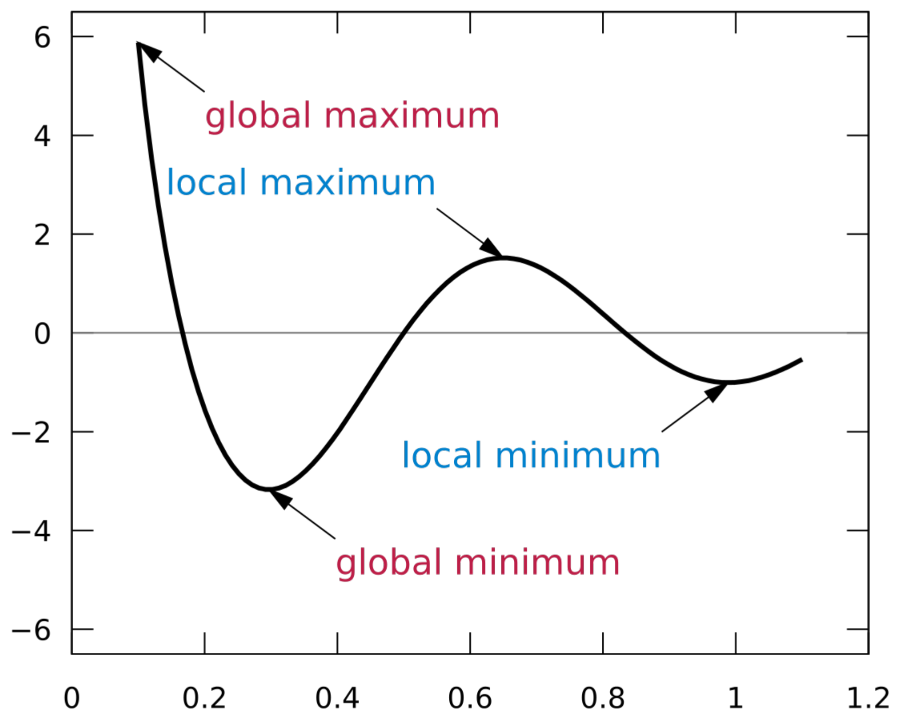

Figure 4.5: Gradient descent

this behaviour depended on Mach and altitude input values.
The analysis was carried out for all four engines and it was clear that the sensitive areas changed according to the engine configuration considered.

The gradient vector allows us to locate the most sensitive spots of the function for every suit of conditions. Where the gradient is different from zero, there the function is more likely to be affected by changes of the variables.

4.4 Sensitive areas

Ejector Ramjet

The sensitive areas for this engine are the Throat, the Primary, The Rocket and the End Diffuser. The throat is the area that determines the mass flow entering the engine, because, as mentioned in paragraph 4.1, on the dimension of the cross sectional area of the throat depends the starting of the inlet.
The Primary is the area indicating the section of the engine where primary flow gets out to be mixed with the rocket plume(whose entity clearly depends on the cross sectional area of the rocket), they merge in the mixer, are sprayed with fuel and then are ignited and ejected through the diffuser.
It is easy to see how as the physical phenomenon which mostly affect the performance of the engine are the two flows mixing and their ejection and diffusion, so the model behaves accordingly.

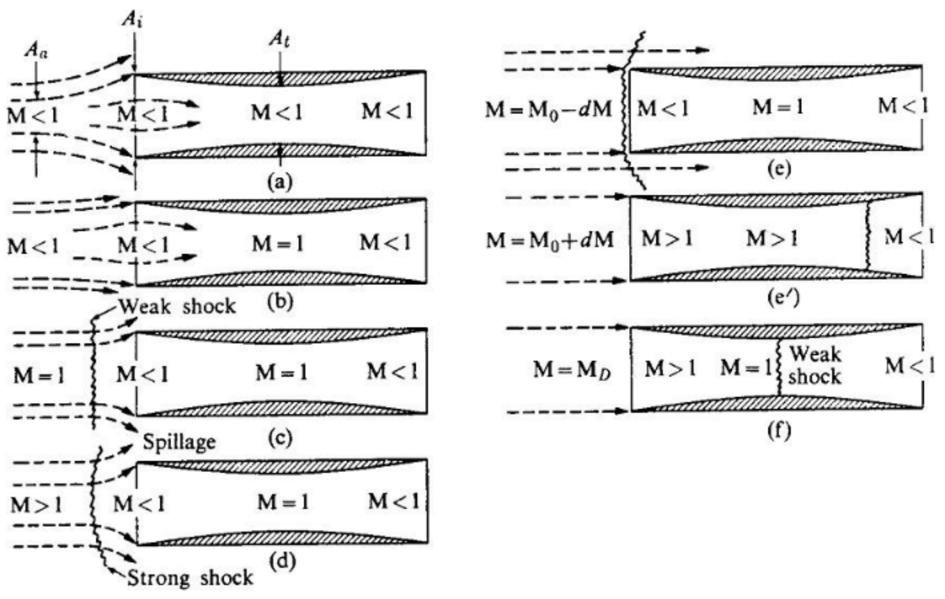

Figure 4.6: Successive steps in the acceleration and overspeeding of a one-dimensional supersonic inlet. [9]

Ramjet

In this case, the stress is on the Intake, the Throat, the End Diffuser and the Nozzle. It is worth saying that, in this specific case, the sensitive areas had a fast variability with Mach and altitude. The four mentioned above are the most recurring ones.
The explanation is simple: the mass flow injected by the intake and then goes through the throat is the mass of air that will then be propelled and pushed through the diffuser and, then, the nozzle.
The more air is swallowed by the intake, the more will be used to generate thrust and to carry the spacecraft up. It is the most simple basic concept of ramjet: a duct used to accelerate a mass of air to produce thrust, and so, by definition, the entrance and the exit of said duct must be the main parameters to modulate the thrust and the mass flow.

Scramjet

Since the scramjet is an evolution of the ramjet concept, the physics of the production of thrust and the relative sensitive areas cannot be too different. Here the cross sectional areas whose contribution to the variability of the engine performance is most evident are the pre intake, the End diffuser and the Nozzle. The Pre-Intake is basically the section of the air stream which has not been swallowed by the inlet yet, but already "sees" the inlet. We must bear in mind that the scramjet works in high supersonic regime, the Mach number can reach the value of 10 or even 15.
From the thermodynamics of inlet we know that that accelerating a one-dimensional

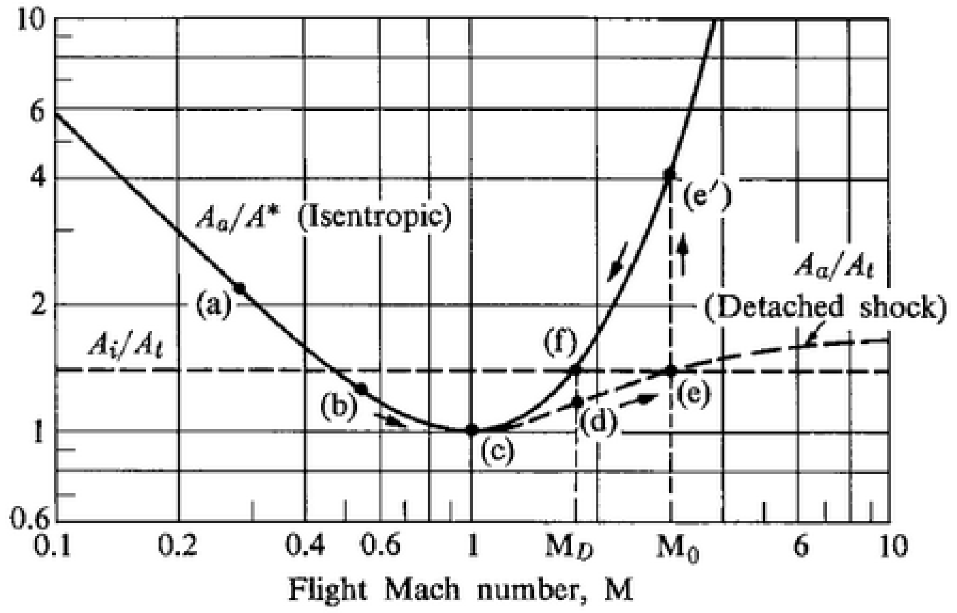

Figure 4.7: Accelerating and overspeeding of a one-dimensional supersonic inlet. [9]

supersonic inlet, for sonic or supersonic flight speeds the upstream capture area A_a is smaller than the inlet area A_i and therefore, spillage will occur around the inlet.

The spilled air is reduced to a subsonic velocity upstream of the inlet plane through a shock wave, that settles at the right position to allow the required spillage. The plot on Figure 4.4 indicates that the inlet area A_i will remain too large and spillage will continue even beyond the design Mach number M_D unless the inlet can be overspeeded to a Mach number M_0, when the inlet can ingest the whole mass flow without spillage [9].

The other two sensitive areas for the scramjet configuration are the End Diffuser and the Nozzle, which are the areas involved in the actual propulsive action of the engine. It is worth noticing that in this case the stress is on the very first area and the very last, i.e. the geometrical parameters that manage the air flow from the beginning to the end, at supersonic speed.

Rocket

Last but not least, the rocket most sensitive cross section is, of course, just one, the Rocket area, which does not need much of an explanation.

Chapter 5

Surrogate Models

As previously mentioned in the introduction to this thesis and in the following literature review, a surrogate model is a metamodel created from a set of significant data with the purpose of simulating the behaviour of the system in the best way possible. This procedure is particularly important for iterative processes such as the optimisation because running the original simulator multiple times would mean huge costs and an extremely high amount of time.
Surrogate modeling allowed us to perform the optimisation of our hybrid engine test case with reasonable means of time and processing power, and the purpose of this chapter is describing the procedure that led to their construction and validation.

5.1 Artificial Neural Network

Artificial Neural Networks can be described as structures comprised of densely interconnected adaptive simple processing elements (called artificial neurons or nodes) that are capable of performing massively parallel computations for data processing and knowledge representation [26].

The basic structure of an ANN consists of artificial neurons that are grouped into layers.

These simple elements simulate the behaviour of brain neurons, which send signals to each other and process information continuously. The idea of a neural network is basically the same, computing inputs through layers of neurons. The inputs are weighted before being processed: these weights are negative or positive quantities, and they can either inhibit or activate the neuron. The neuron sums all the signals it receives, multiplied by their respective weights.

$$v = (\omega_1 x1) + (\omega_2 x_2) + (\omega_3 x_3) + b = \omega x + b \tag{5.1}$$

The activation function determines the behaviour of the node.

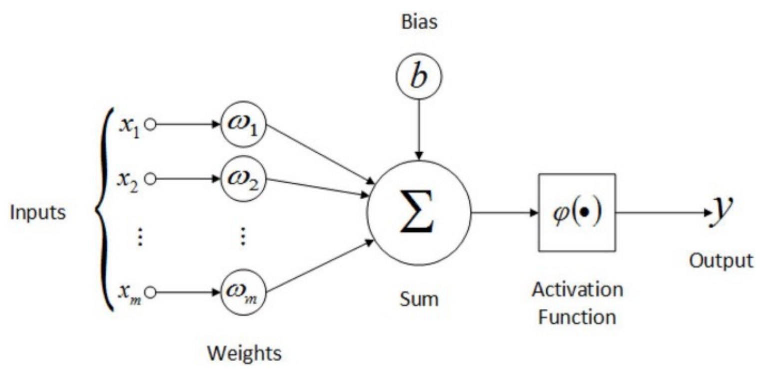

Figure 5.1: Structure of the artificial neuron.

So the output y will be the activation function evaluated for the weighted inputs plus the bias:

$$y = \phi(v) = \phi(\omega x + b) \tag{5.2}$$

These weights are initialised at the beginning of the procedure, then, after the first computation, the output of the net are compared to the target output and the error is computed. In the following iteration, the weights will be adjusted to reduce the error, and so on.

Database creation

The first step for the construction of any metamodel is the creation of the samples. Since the purpose of our optimisation is the maximisation of the performance of the engine and the payload it can carry on the trajectory, our focus is on Thrust and Mass Flow and their variation with altitude and Mach and engine configuration.
At first, the volume (and so the weight) of the engine was considered as well, but to avoid the creation of too many modules we set some boundaries on the cross sectional areas of the engine and worked with that, therefore making sure that the volume of the engine would not be excessively large and maximising its performance at the same time.

The first generation of databases considered seven equally spaced values for each variable. This means that for the ejector and for the ramjet, which both have four sensitive areas plus altitude and Mach as input variables to consider, there were more then a hundred thousands samples.
This turned out to be pretty expensive to compute, the training tool `nntraintool` took more then ten days and when it was forced to a halt the net had not reached a satisfactory value of performance.

So, for ejector ramjet and ramjet, we opted for a smaller quantity of samples, five values for each variable, while for the scramjet, which has less sensitive areas, we kept seven values each.

The computational time was noticeably reduced and we were able to generate not so accurate but still adequate neural networks.
For the next step, which has to do with the actual creation of the model and how it is performed, a little background on the structure of networks is needed.

Constructing the ANN

The choice of the artificial neural network for our surrogate based optimisation was mainly due to the availability of software packages and apps to easily build one.
In this work we used `Matlab Deep Learning Toolbox` .
The data set to train the neural network will be divided in three parts:

- training set: it is the set of samples used to tune the weights of connection between neurons;
- validation set: it is used to test the predictive ability of the model. It is useful as the training of the neural network should be stopped when the error on the validation datasets increases, as it is a sign of imminent overfitting.
- test dataset: used to evaluate the fitting of the network on the dataset.

On the validation process depends the number of hidden layers and neurons of the net. If an ANN is overtrained, its predictive capability may be affected and the net might fit only to the samples of the dataset. On the other hand, if a neural network has not been running enough training iterations it may settle to a local minimum, rather than to the global minimum solution and so the error might still be far from being minimized.
To know when to stop the training is a crucial part of the whole ANN building process and it needs several trial and errors attempts, with different parameters and times of training.

In this work we used this partition of the data set

$$\texttt{net.divideParam.trainRatio = 90/100;} \tag{5.3}$$
$$\texttt{net.divideParam.valRatio = 5/100;} \tag{5.4}$$
$$\texttt{net.divideParam.testRatio = 5/100;} \tag{5.5}$$

After many attempts to get the best training possible, the proportion 90/5/5 turned out to be the right choice.
The number of neurons varies from model to model, according to the level of accuracy each and every one of them could reach with the data set given.

Fine tuning

The first generation of neural networks created proved mildly accurate.

The following pictures show the performances of two different ejector ramjet model (**blue**) against the evaluation of HyPro (**red**) of the thrust for different values of Mach, altitude and sensitive areas.

Adding neurons to the layers - from 25 to 80 - on a same data set improved the performances of the model and made it quite satisfactory. Unfortunately, as mentioned earlier, the whole procedure of training and validation of these models took days and days, and, since the models to create were at least six and each of them had to be tested and proved adequate, the data set was reduced.
Figure 5.3 shows the same test case performed with the new 80 neurons model trained on the smaller data set.
It is clear that, for less computational costs, the accuracy of the net was slightly compromised.

A *fine tuning* of the net was needed, and different strategies were performed.

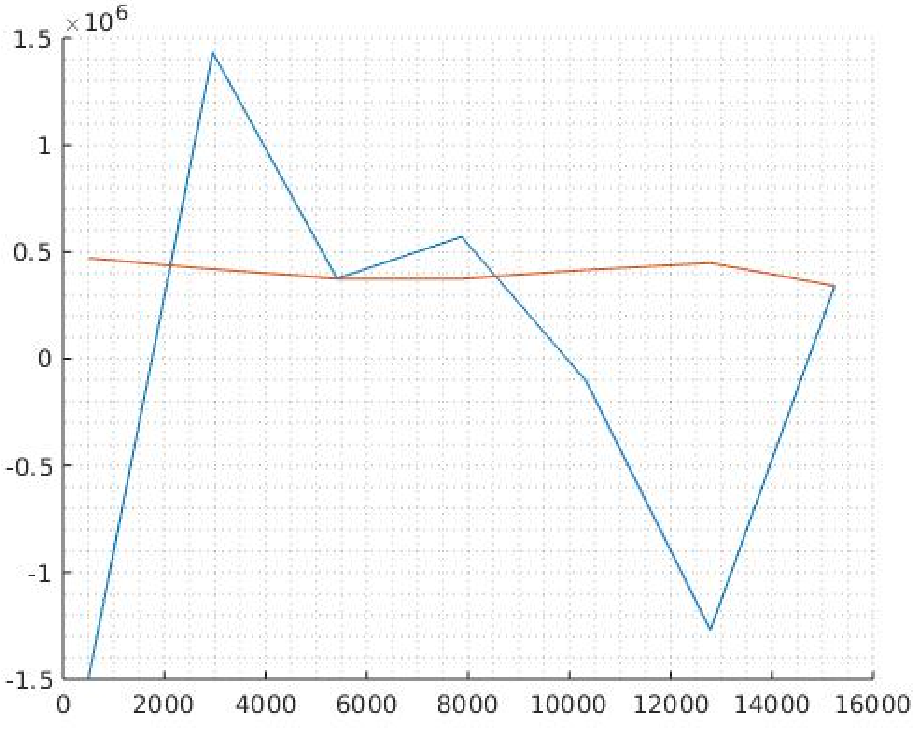

Figure 5.2: Original Database - 25 neurons.

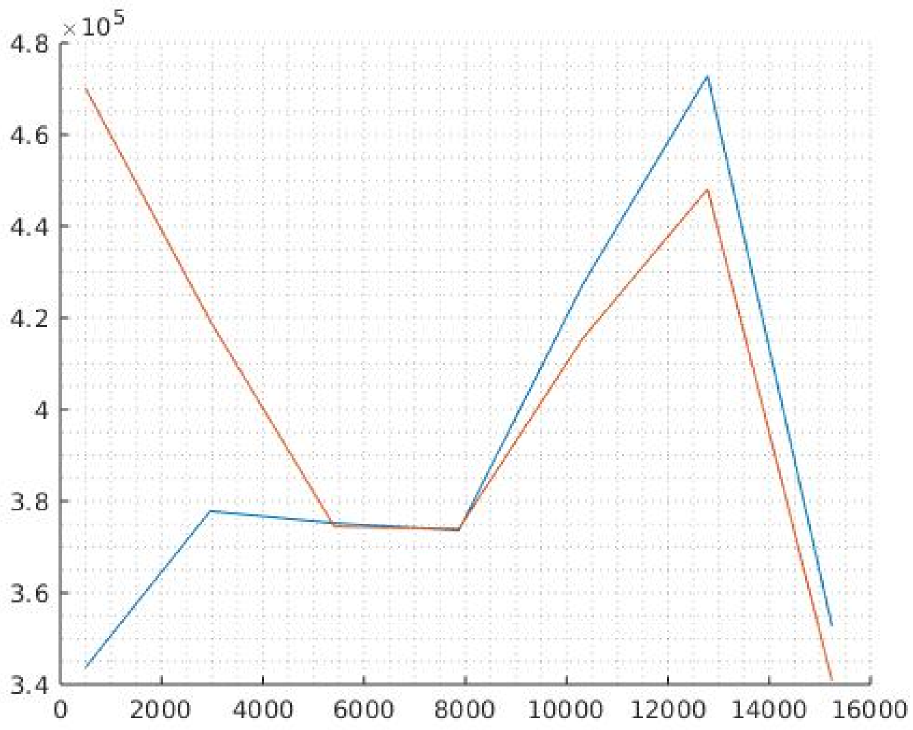

Figure 5.3: Original Database - 80 neurons.

Neuron fitting

One of the first attempts involved adding neurons to the layers, as it proved effective with the first database. However, it is important to keep in mind that when increasing the number of neurons of a net, the risk of overfitting is just around the corner.
Figures 5.4 and 5.5 show the performance of 100 neurons and 220 neurons neural network.

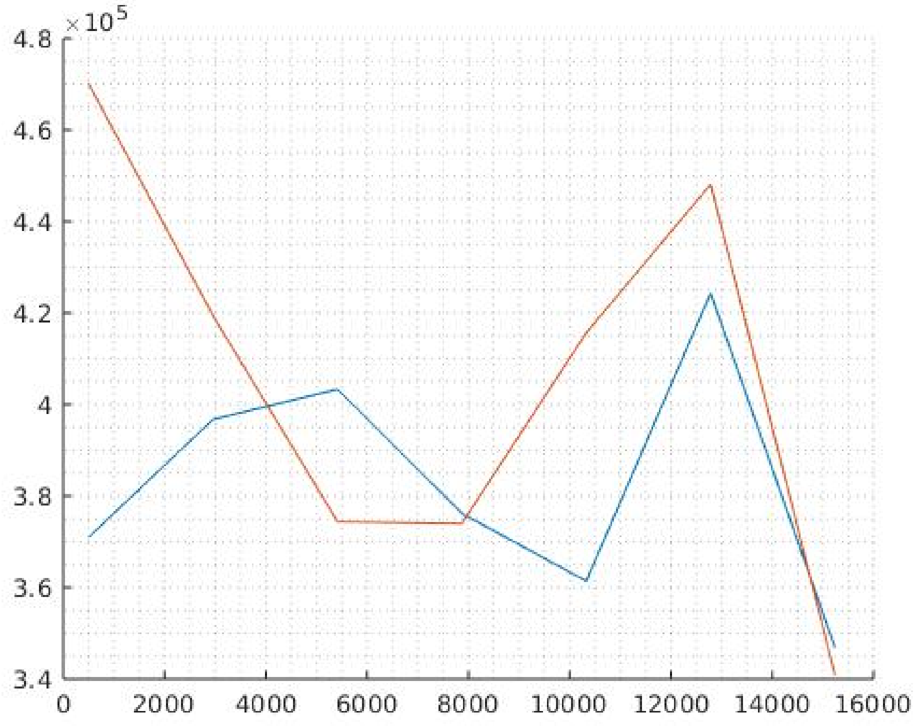

Figure 5.4: Reduced Database - 100 neurons.

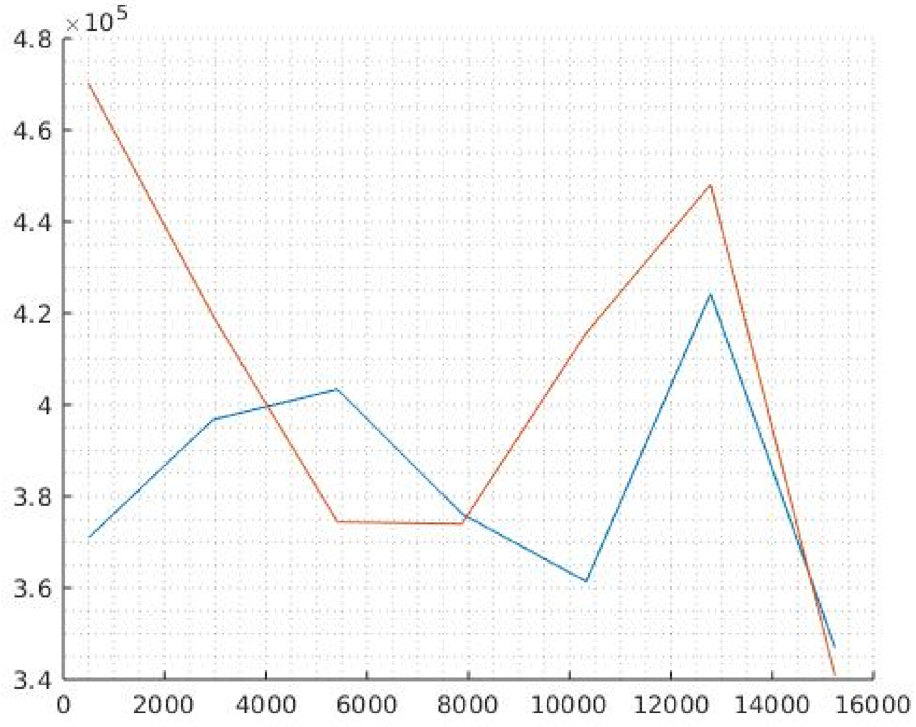

Figure 5.5: Reduced Database - 220 neurons.

Clearly the increased amount of neurons did not improve the performance of the net, the error is just slightly lower but not enough to be appreciated and to consider the neuron fitting strategy effective.

Transfer learning

This procedure consists in taking an already trained network and performing a new training on it to make it learn another task.
It is mostly used for Convolutional Neural Networks (CNNs) which are trained for image recognition and similar activities.
In these nets, the final layers are replaced by new ones assigned to the new learning process, then the training is performed again, and so on, until the model has all the "knowledge" needed to perform correctly its duties.

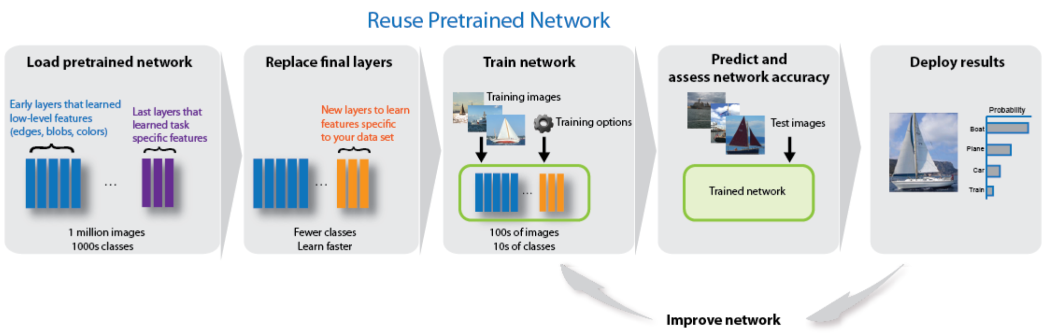

Figure 5.6: Transfer Learning - from MathWorks website.

However, while it is apparently very effective and extensively discussed in the literature, for the ANNs employed in this thesis transfer learning was not such a simple and successful procedure.

The first attempt with transfer learning followed this procedure:

1. Split the extended database in smaller sections;
2. Create and train the Artificial Neural Network with the reduced database;
3. Re-train the ANN on single sections of the extended database.

This strategy proved ineffective because of the high computational costs, and the training had to be stopped when it was still far from a satisfactory convergence.

The second attempt was slightly more complex, as the following steps show:

1. Split the extended database in smaller sections;
2. Create and train an ANN with the reduced database;

3. Re-train the ANN on the extended database sections **mixed with a significant batch of the reduced database**.

In this case, the process came to an end displaying the two performance plots displayed in Figure 5.7 and 5.8.

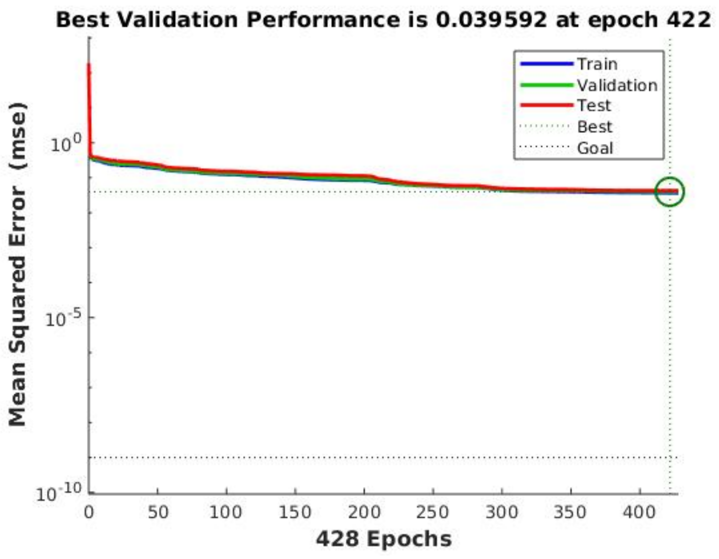

Figure 5.7: Best performance plot.

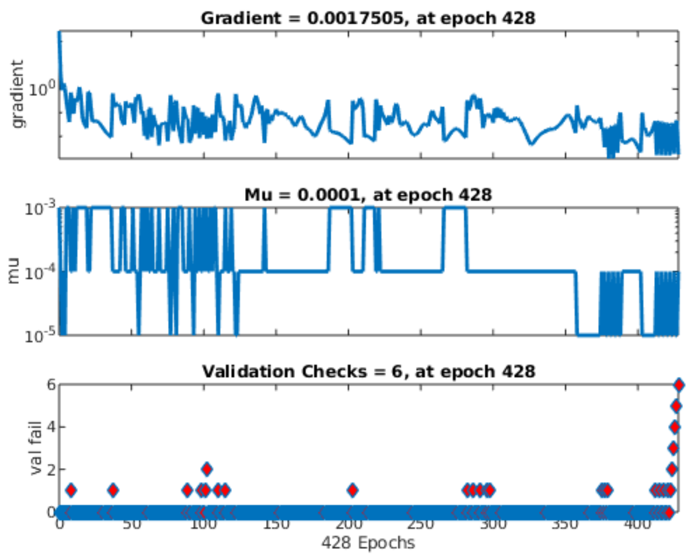

Figure 5.8: Gradient and validation performance.

The first one shows that after a certain number of epochs the training reached its best performance, that was not enough to consider the training completed successfully, though.

In the second one it is clearly visible the trend of the validation process: multiple local minima are found on the curve, then at last, right before the training stopped, the curve started rising to an extent that forced the process to quit, because, as was explained earlier, that is a sign of imminent overfitting.

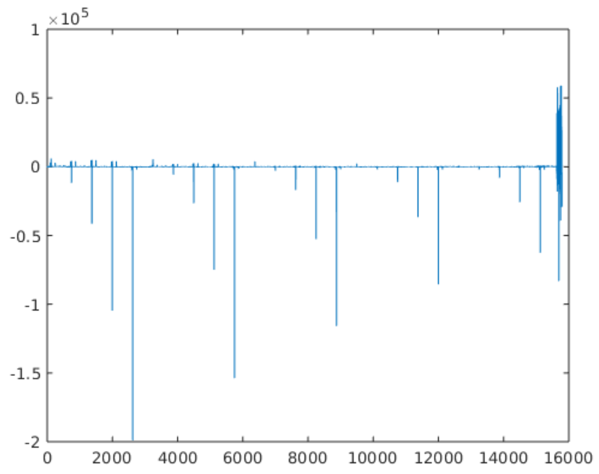

Figure 5.9: Absolute error on the ejector ramjet net data.

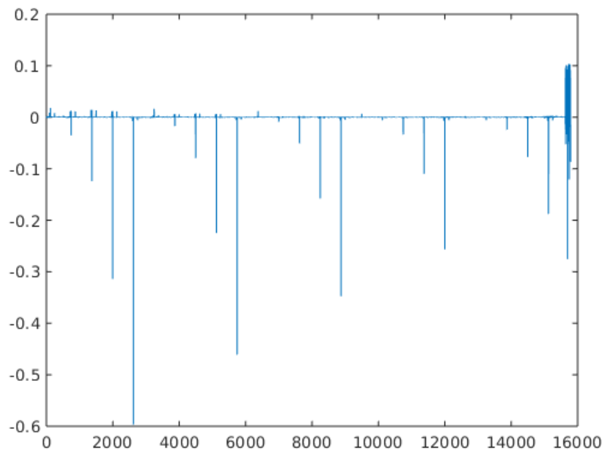

Figure 5.10: Relative error on the ejector ramjet net data.

Optimisation

The final step towards an increased accuracy of the nets consisted of optimising the net. The net was included in an optimisation routine, which first visualizes the absolute error and the relative error and then performs a multi-start research of the minimum of the net.

Once it found the minimum, the function goes back to the inputs that caused those "wrong" values of the net and prints them on a file.
HyPro runs those inputs and computes the target value for the net, which then is added to the database. A new net is trained on the updated on the new database and so on, until the error reaches acceptable values.

However, while this process proved effective in adding samples to the database to improve the predictive capability of the net, the values added to the database had to be discarded.
The reason for this is shown in Figure 5.11

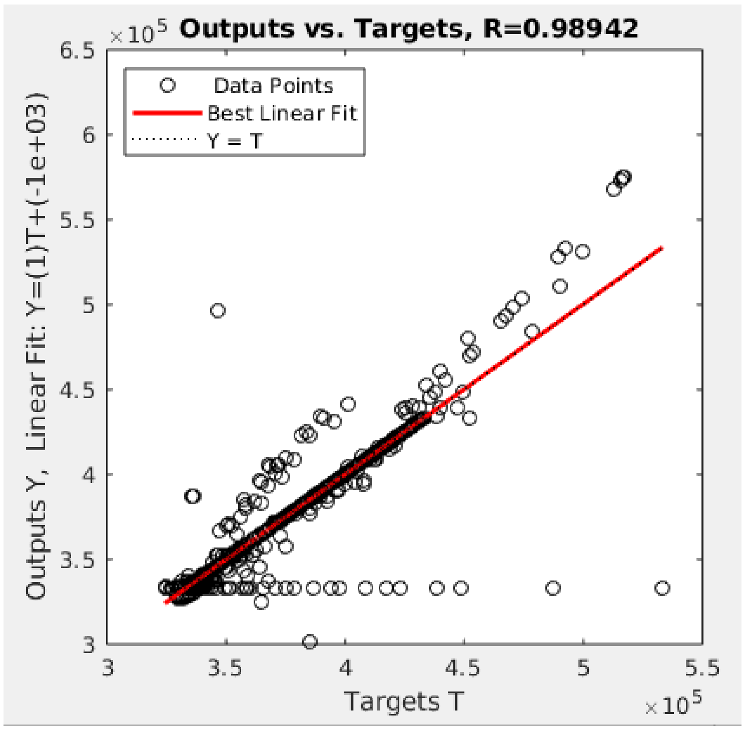

Figure 5.11: Ejector neural network fitting

We can see that the model itself has improved its convergence, but the new data are really scattered.
This means that the neural network is not "smooth", and so, in an optimisation loop based on collocation methods, where functions are approximated with polynomials, the convergence is really hard to reach.
This is the reason why it was decided to improve the training algorithm, not the database, and use maybe less accurate models, but smoother.

Chapter 6

Trajectory Optimisation

The purpose of this chapter is to outline the software employed for the solution of the optimal control problem, its main features and the way it was set for the present work.

6.1 GPOPS

GPOPS is a general-purpose MATLAB® software program for solving multiple phase optimal control problems using variable-order Gaussian quadrature collocation methods [10]. The method employed is a Legendre-Gauss-Radau quadrature orthogonal collocation algorithm and the optimal control problem is transcribed to a large sparse nonlinear programming problem (NLP).

GPOPS can solve a problem whose formulation is [10]:

$$J = \sum_{p=1}^{P} \left[\Phi^{(p)} \left(\mathbf{x}^{(p)}(t_0), t_0^{(p)}, \mathbf{x}^{(p)}(t_f), t_f^{(p)}; \mathbf{q}^{(p)} \right) + \int_{t_0^{(p)}}^{t_f^{(p)}} \mathcal{L}^{(p)} \left(\mathbf{x}^{(p)}(t), \mathbf{u}^{(p)}(t), t; \mathbf{q}^{(p)} \right) \mathrm{dt} \right] \tag{6.1}$$

with the dynamic constraints

$$\dot{\mathbf{x}}^{(p)} = \mathbf{f}^{(p)}(\mathbf{x}^{(p)}, \mathbf{u}^{(p)}, t; \mathbf{q}^{(p)}), \qquad (p = 1, ..., P), \tag{6.2}$$

the inequality path constraints

$$\mathbf{C}_{min}^{(p)} \leq \mathbf{C}^{(p)}(\mathbf{x}^{(p)}(t_0), t_0^{(p)}, \mathbf{x}^{(p)}(t_f), t_f^{(p)}; \mathbf{q}^{(p)}) \leq \mathbf{C}_{max}^{(p)}, \qquad (p = 1, ..., P), \tag{6.3}$$

the boundary conditions

$$\phi_{min}^{(p)} \leq \phi^{(p)}(\mathbf{x}^{(p)}(t_0), t_0^{(p)}, \mathbf{x}^{(p)}(t_f), t_f^{(p)}; \mathbf{q}^{(p)}) \leq \phi_{max}^{(p)} \tag{6.4}$$

and the linkage constraints

$$\mathbf{L}_{min}^{(s)} \leq \mathbf{L}^{(s)}(\mathbf{x}^{(p_l^s)}(t_f), t_f^{(p_l^s)}; \mathbf{q}^{(p_l^s)}, \mathbf{x}^{(p_r^s)}(t_0), t_0^{(p_r^s)}; \mathbf{q}^{(p_r^s)}) \leq \mathbf{L}_{max}^{(s)} \tag{6.5}$$

with $p_l, p_r \in [1, ..., P]$ e $s = [1, ..., L]$ where $\mathbf{x}^{(p)}(t) \in \mathbb{R}^{n_x^{(p)}}$, $\mathbf{u}^{(p)}(t) \in \mathbb{R}^{n_u^{(p)}}$, $\mathbf{q}^{(p)} \in \mathbb{R}^{n_q^{(p)}}$, and $t \in \mathbb{R}$ are, respectively, the state, control, static parameters, and time in phase $p = [1, ..., P]$, L is the number of pairs of phases to be linked, and $(p_l^{(s)}, p_r^{(s)}) \in [1, ..., P]$, s=1,...,L, are the "left" and "right" phase numbers, respectively. The phases *need to be consecutive.* If they are not, any two phases may be linked only if the independent variable does not change direction (i.e. in every phase linked, the independent variable goes in the same direction).

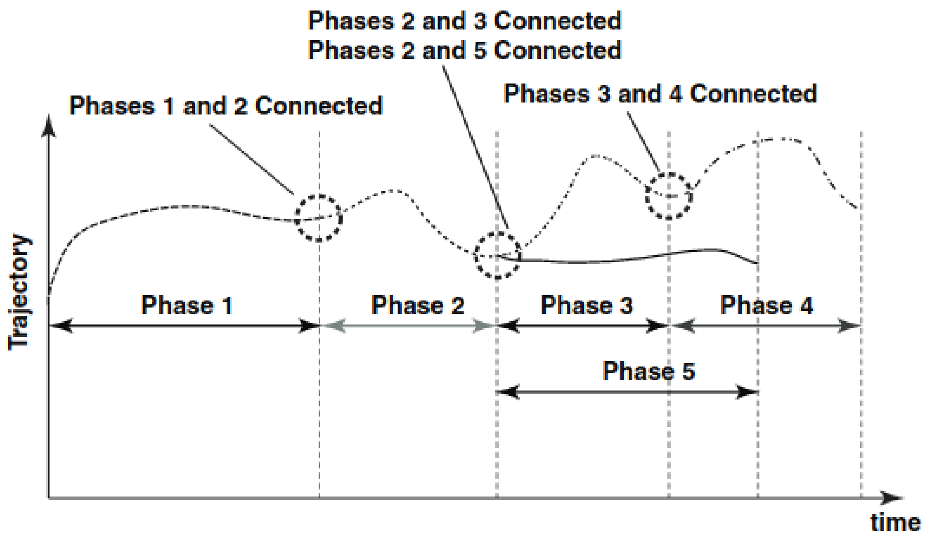

Figure 6.1: Schematic of linkage for multiple-phase optimal control problem [10]

The method employed by GPOPS is a Radau Pseudospectral Method (RPM), which is an orthogonal collocation method. It is a Gaussian quadrature implicit integration scheme and converges faster for smooth problems (here is the reason why the models of the engine had to be improved in this sense).

As the user manual reports [27] GPOPS is organized as follows: in each phase of the problem, the user has to provide functions that define

1. the cost functional;
2. the differential equations and the path constraint1;
3. the boundary conditions;
4. the constraints at the interface of consecutive phases.

It is also necessary to specify the lower and upper limits of the following quantities

1. initial and terminal time of the phase;
2. the state at the beginning, during and at the end of the phase;
3. the control;
4. the static parameters;

5. the path constraints;
6. the boundary conditions;
7. the phase duration;
8. the linkage constraints.

6.1.1 Setting

In the case study developed in this work, the state variables are the altitude h, the velocity v, the fly path angle γ, the heading angle χ, the latitude, the longitude and the mass of the vehicle.
The control variables are angle of attack α and the throttle δ_T.
The path constraint for each phase is the Mach value of speed of the vehicle, which has to lie in a specific range, equal to the range of training of the surrogate model of the engine in use. It is worth reminding that every phase corresponds to an engine configuration, so there necessary must be a range of speeds in which the engine can and must work, and never go below or over it. Only $phase4$ has no Mach limits.
The boundary conditions are summarized in the next chapter, the phase duration is from 50 to 1000 s for $phase1$, from 50 to 1500 s for $phase2$, from 10 to 1000 s for $phase3$, from 50 to 1000 s for $phase4$.
The linkage settings are the default set for the launch-ascent trajectory optimisation.

The user has to give an initial guess of the solution. Here is for the different phases:

	min	**MAX**
t	0 s	400 s
h	1000 m	15000 m
v	400 m/s	800 m/s
γ	2°	2°
χ	0	0
λ	0	0
θ	0	0
mass	300000 kg	270000 kg
α	0	0
δ_T	2	2

Table 6.1: Phase 1 guess solution.

	min	**MAX**
t	400 s	600 s
h	9500 m	21000 m
v	800 m/s	1200 m/s
γ	0	0
χ	0	0
λ	0	0
θ	0	$2°$
mass	300000 kg	270000 kg
α	0	0
δ_T	2	2

Table 6.2: Phase 2 guess solution.

	min	**MAX**
t	600 s	700 s
h	15500 m	39500 m
v	1200 m/s	1600 m/s
γ	0	0
χ	0	0
λ	0	0
θ	$4°$	$6°$
mass	240000 kg	210000 kg
α	0	0
δ_T	2	2

Table 6.3: Phase 3 guess solution.

	min	**MAX**
t	700 s	800 s
h	39500 m	100000 m
v	1600 m/s	7850 m/s
γ	0	0
χ	0	0
λ	0	0
θ	$6°$	$8°$
mass	210000 kg	180000 kg
α	0	0
δ_T	2	2

Table 6.4: Phase 4 guess solution.

6.2 System Models

Dynamic model

The vehicle in object performs a multi-phase ascent mission .
The dynamics of the vehicle is governed by the following equations:

$$\begin{aligned}
\dot{h} &= v\sin\gamma, && (6.6)\\
\dot{v} &= \frac{T\cos\alpha - D}{m} - g\sin\gamma + \omega_e^2(R_E+h)\cos\lambda(\sin\gamma\cos\lambda - \cos\gamma\sin\chi\sin\lambda), && (6.7)\\
\dot{\gamma} &= \frac{T\sin\alpha + L}{mv}\cos\mu - \left(\frac{g}{v} - \frac{v}{R_E+h}\right)\cos\gamma + 2\omega_E\cos\chi\cos\lambda && (6.8)\\
&+ \omega_E^2\left(\frac{R_E+h}{v}\right)\cos\lambda(\sin\chi\sin\gamma\sin\lambda + \cos\gamma\cos\lambda), &&\\
\dot{\chi} &= \frac{L}{mv\cos\gamma}\sin\mu - \left(\frac{v}{R_E+h}\right)\cos\gamma\cos\chi\tan\lambda && (6.9)\\
&+ 2\omega_E(\sin\chi\cos\lambda\tan\gamma - \sin\lambda) - \omega_e^2\left(\frac{R_E+h}{v\cos\gamma}\right)\cos\lambda\sin\gamma\cos\chi, &&\\
\dot{\lambda} &= \frac{v\cos\gamma\sin\chi}{R_E+h}, && (6.10)\\
\dot{\theta} &= \frac{v\cos\gamma\cos\chi}{(R_E+h)\cos\lambda}. && (6.11)
\end{aligned}$$

where h is the altitude above sea level, v is the absolute velocity and γ is the flight path angle, χ is the heading angle and μ is the bank angle, λ is the latitude, θ is the longitude, m is the mass of the vehicle, T is the thrust given by the engine, L is the lift force, D is the drag force, R_E is the mean Earth radius, ω_E is the rotational velocity of the Earth, g_0 is the acceleration of gravity and α is the angle of attack (AoA) of the vehicle. For simplicity, no offset between the thrust vector and the velocity vector was considered, and thus (α=α_{eng}).

In each of the four phases of the ascent mission, the value of the state variables must fall in the following intervals:

	min	MAX
h	1000 m	120000
v	400 m/s	8000 m/s
γ	-60 rad	60 rad
χ	-10 rad	10 rad
λ	-1 rad	1 rad
θ	0 rad	20 rad
mass	15000 kg	300000 kg

Table 6.5: State variables range of values

In each phase, as previously mentioned, Mach and altitude must be in a specific range of values, which are the boundaries of the data set used to train the surrogate models.

	1	2	3	4
hmin [m]	1000	9500	15500	1000
hMAX [m]	15000	21000	39500	120000
Mmin	0	2	4	/
MMAX	2	5.5	10	/

Table 6.6: Mach and altitude bounds for the four phases.

.

This makes it possible to create four distinct phases of ascent for the four engines, in the conditions they realistically can operate.

6.2.1 Aerodynamic Model

To calculate the aerodynamic forces acting on the vehicle, a simple algebraic model was employed, as a function of the angle of attack α, the Mach number and the dynamic pressure. The reference surface A_{ref} is 300 m^2. The drag and lift coefficients are computed with two ANNs and are then used to calculate the actual lift and drag forces.

$$L = C_L\, A_{ref}\, p_{dyn}, \tag{6.12}$$
$$D = C_D\, A_{ref}\, p_{dyn}. \tag{6.13}$$

6.2.2 Atmospheric Model

The atmospheric model is a simple algebraic model which computes the speed of sound a and the air density ρ as a function of the altitude h.

6.2.3 Propulsion Model

As the vehicle is propelled by an hybrid engine, this function calls two surrogate models for each phase, one for the computation of thrust, the second for the computation of mass flow.
These two quantities are then scaled by δ_T, which is the throttle, whose range of values is [0, 8] and corresponds to the number of engines employed.

It goes without saying, if the δ_T assumes a fractional value, then the closest integer quantity of engines must be considered in use for that phase of the mission.

The last phase does not call any model and the thrust is computed using the basic rocket equation

$$F = (C\,B - A_e\,p), \tag{6.14}$$

$$M_f = \frac{T}{g_0\,I_{sp}}. \tag{6.15}$$

where the specific impulse I_{sp} has an average value of $460s$, C and B are two constants and are, respectively, equal to 700000 N and 1. A_e is the external cross sectional area of the rocket and it is assumed equal to 2.5 m^2, which is a standard value for a rocket nozzle, while p is the atmospheric pressure.

Chapter 7

MDO

The purpose of this chapter is to outline the structure of the Multidisciplinary Design Optimisation and the interconnection between outer loop (engine performance optimisation) and inner loop (trajectory optimisation).

7.1 MP_AIDEA

The optimizer that was used in this work is a Multi Population Adaptive Inflationary Differential Evolution Algorithm (MP_AIDEA).
Inflationary differential evolution algorithm combines basic differential evolution with some of the restart and local search mechanisms of Monotonic Basin Hopping (MBH)[11].

The Differential Evolution (DE) algorithm is a population based algorithm to solve global optimisation over continuous domain. The peculiarity of this software is that it adapts the search during the evolutionary process. At the beginning of the process the parents for the mutation are distant, so the DE performs a global exploratory search and the perturbations are large. As the process advances, the search becomes local and the perturbations become smaller and smaller.

IDEA is a combination of Differential Evolution and and Monotonic Basin Hopping, the combination being an evolutionary process with an improved restarting procedure in the neighborhood of a local minimum and a global restart in the whole search space.

MP-AIDEA algorithm basic operation can be described as follows [11]: n_{pop} Differential Evolutions run in parallel (where n_{pop} is the number of populations) and the populations contract within a sphere of a given radius.
The relative position of the best individual of the population $\mathbf{x}_{best,m}$ with respect to the local minima, $\mathbf{x}_{LM}$ is evaluated. If the best individual of a population m is not close to any already detected local minimum, then a local search starts to find one. The local restart bubble has radius δ_{local}.

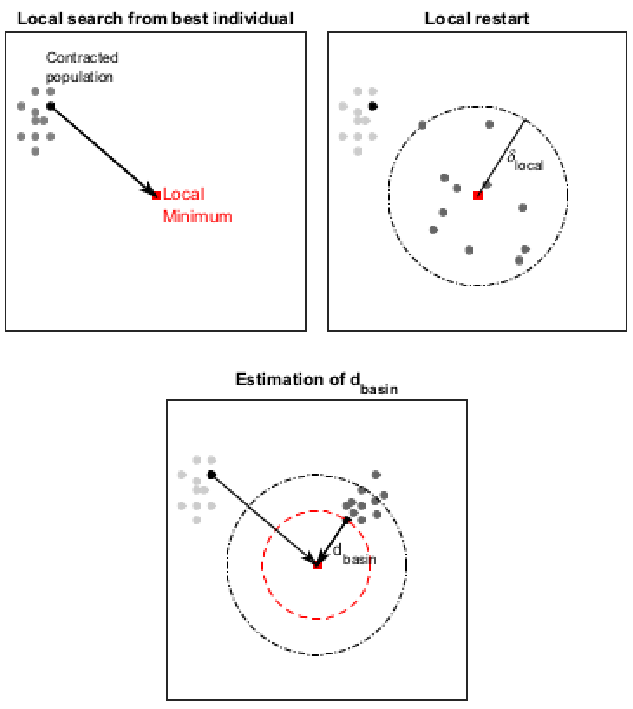

Figure 7.1: Identification of the basin of attraction of local minimum $\mathbf{x}_{LM}$ [11]

The probability distribution associated to δ_{local} is updated right before starting a new global or local search. Once the population is restarted, the process starts again from the initialisation of the main parameters and proceeds until the number of maximum function evaluations $n_{feval,max}$ is reached.

7.2 Bi-level optimisation

The optimisation is set by coupling the adaptive inflationary differential evolution algorithm with a direct collocation method for optimal control problems (performed by GPOPS).

This hybrid algorithm works integrating the engine design, in this specific case the geometry of the flow path through specific sections of the engine, and optimal control into the same optimisation process.

The parameters of the engine we seek to optimise are the sensitive areas of the four modes. Their values must all be in the range used to train the surrogate models and fall in an interval which physically allows a correct operation of the whole propulsion system.

The following table summarises the sensitive areas and their boundary values.
It is important to notice that the sensitive areas in common between different configurations are considered just once, as the structure of the engine is only one for

Area $[ft^2]$	min	MAX
Throat	3.5	6.75
Primary	9	11.25
Rocket Outlet	2	3.01
End Diffuser	19.0	22.5
Intake	22.5	27.0
Nozzle	40	44.3
Pre-Intake	22.5	27.0

Table 7.1: Engine optimisation parameters

all the four configurations. So the Throat, which is an optimisation parameter for both the Ejector Ramjet and the Ramjet, is the same area. That is why the vector of the optimisation parameters contains only seven elements.

The optimisation procedure is structured as follows [28]: an outer loop optimises the engine performances (function of the above mentioned sensitive areas) and an inner loop optimises the trajectory. The interconnection between these two loops lies in the objective function driving the whole optimisation, which makes them part of a single process.

In short, the outer loop that works on the engine and its parameters takes into consideration the objective and the constraints of the values of the performance indices coming from the inner loop, which is in charge of the optimal control.

This algorithm searches for engine flowpath geometries that maximize the final mass of the vehicle (i.e. maximising the payload of the mission).

Once a vector of areas is defined by the outer loop, the optimal control problem is solved, then the mean and variance of the performance indices just obtained are the objective and constraint of the outer loop, until convergence.

The optimal control problem is called from the general optimiser through the function `optimise_mpaidea` together with the vector of areas and respective lower and upper boundaries, the optimal control problem is solved and returns the objective and constraints for the next engine optimisation and so on.

```
[x,fval,exitflag,output]=optimise_mpaidea(fitnessfcn,LB,UB,options)
```
(7.1)

The output `x` is the optimal design solution found, the function `fitnessfcn` contains the objective function calling the optimal control.

Chapter 8

Results

The purpose of this chapter is outlining the results obtained from the optimisation process described in detail in the former chapter.

The main purpose of this thesis was designing a process for a bi-level optimisation, trying out a strategy which could improve the preliminary design of an hybrid propulsion system.

In the following sections two different solutions are analysed and commented.

8.1 First solution

8.1.1 Altitude

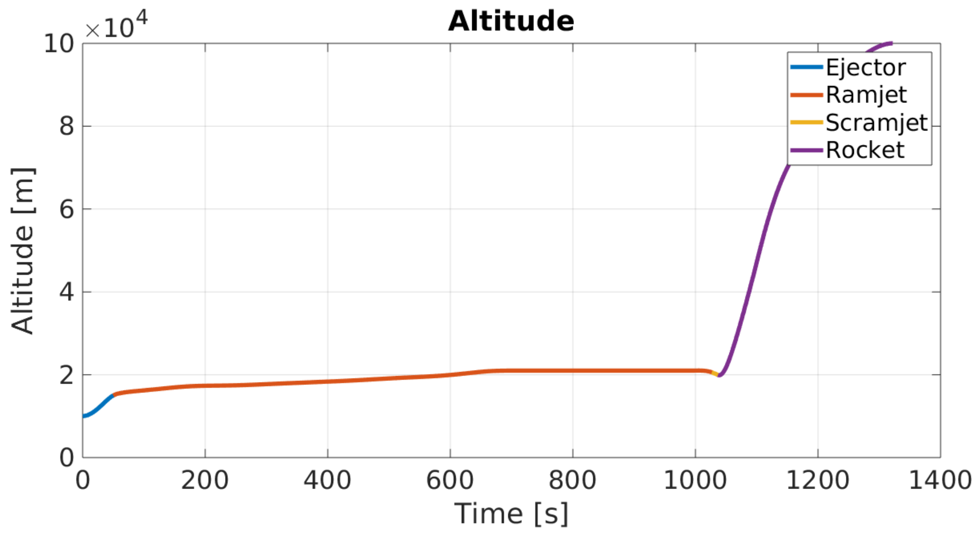

Figure 8.1: First solution - Altitude.

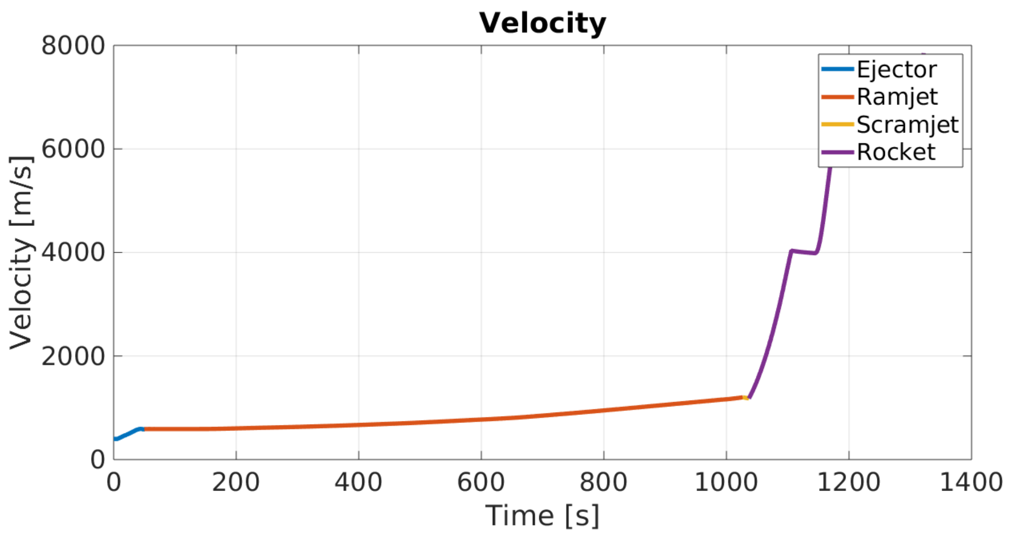

Figure 8.2: First solution - Velocity.

8.1.2 Velocity

8.1.3 Mass

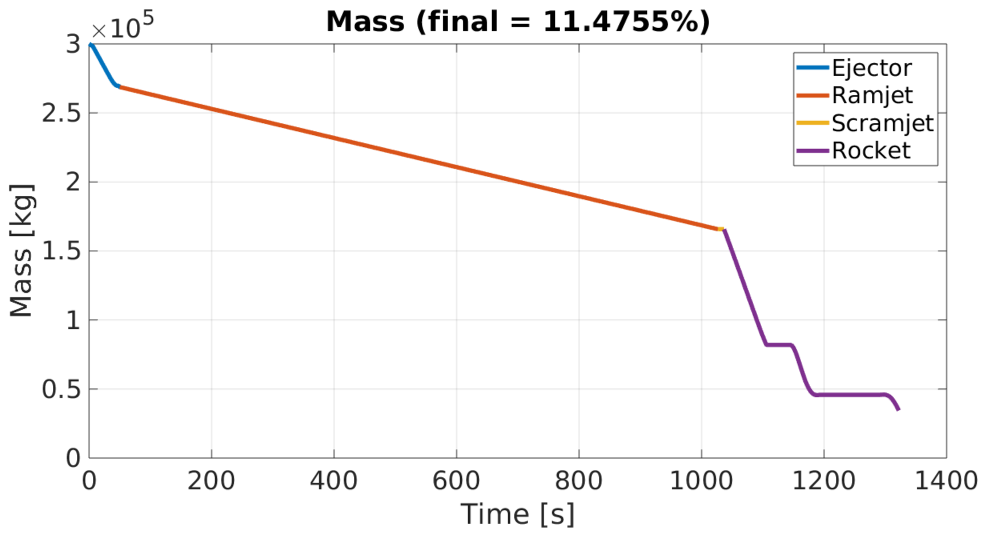

Figure 8.3: First solution - Mass.

8.1.4 Angle of Attack

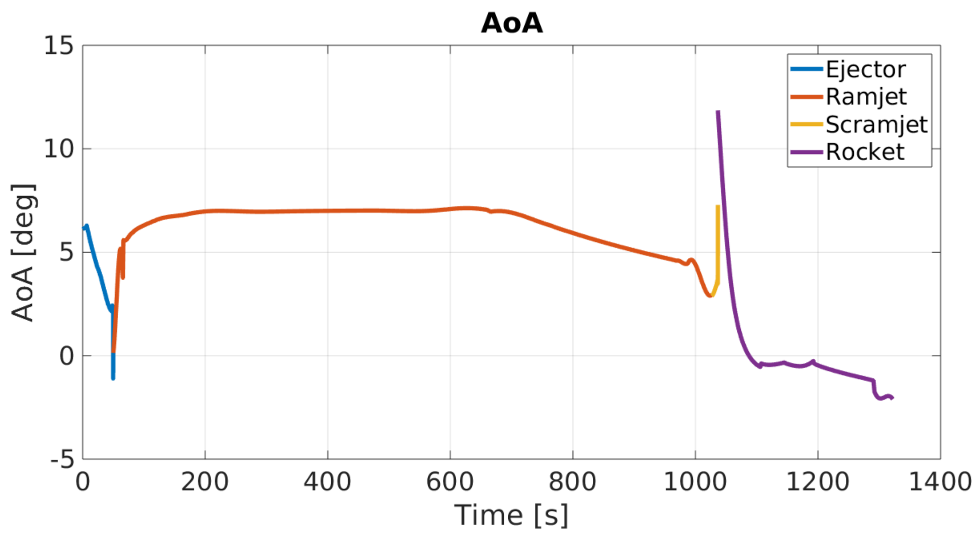

Figure 8.4: First solution - Angle of Attack.

8.1.5 Throttle

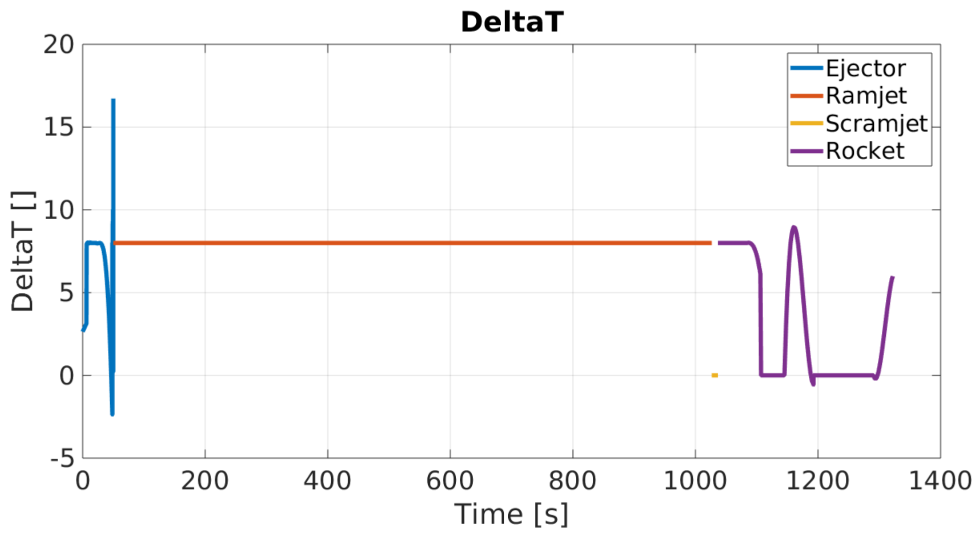

Figure 8.5: First solution - Throttle.

8.1.6 Fly Path Angle

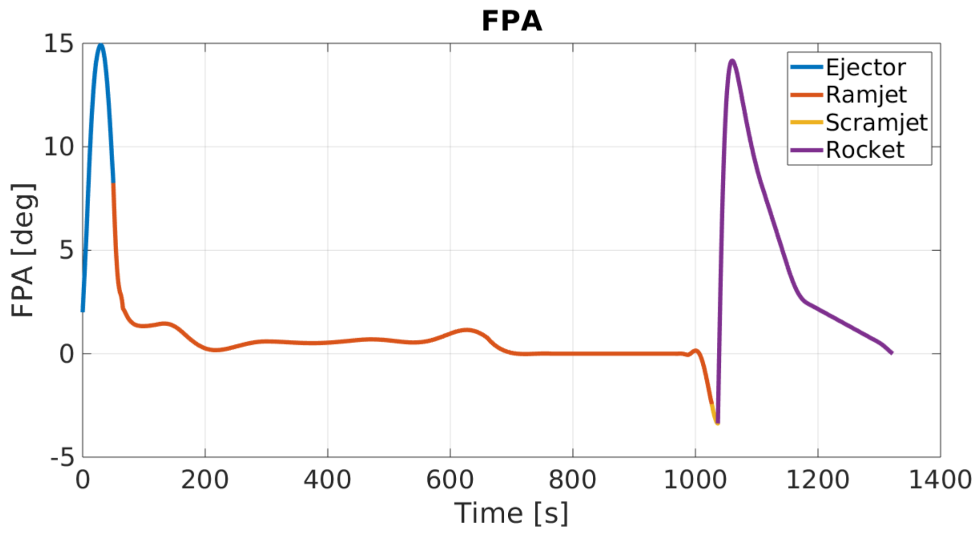

Figure 8.6: First solution - Fly path angle.

8.1.7 Longitude and Latitude

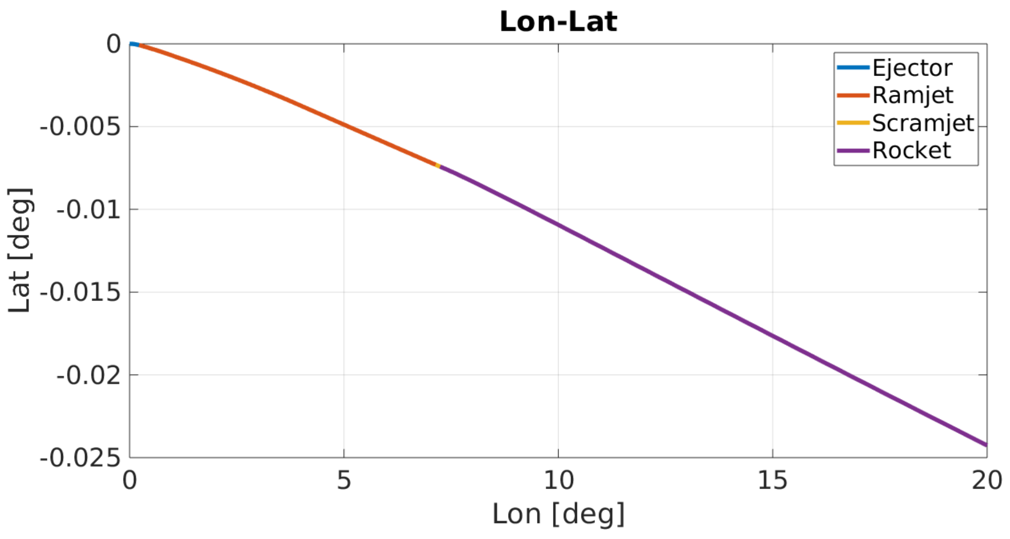

Figure 8.7: First solution - Longitude and Latitude.

8.1.8 Design Cross Sectional Areas

Area $[ft^2]$	
Throat	4.87986962083807
Primary	10.2707352117151
Rocket Outlet	2.79041521047196
End Diffuser	19.5110741384452
Intake	22.8780058135268
Nozzle	42.955526246486
Pre-Intake	26.9512859927499

Table 8.1: First solution - Optimal areas

8.1.9 Comments

The final mass of this solution is

$$m_f = 34426\,\text{kg} \tag{8.1}$$

with a final payload ratio

$$\text{PayloadRatio} = 11.4755\,\% \tag{8.2}$$

The mass variation from start to finish is due to the fuel consumption, as fuel takes up the majority of the take-off weight of the spacecraft. The behaviour of the fly path angle, the angle of attack and the throttle have to be considered carefully: in fact, for example, the throttle curve seems to go beyond the limit of 8 that has been set for this variable in the trajectory optmisation. This, of course, is not true, it is a representation problem of the data resulting from the routine, which leads to a certain elongation of the curve which - it is worth repeating - does not correspond to the real trend of the control variable.
The same thing can be said for the other control variables, the angle of attack α and the fly path angle γ.
It is to be noted that the use of the scramjet configuration is relegated to a handful of seconds: actually, GPOPS did not use this mode at all and a constraint had to be set to force the algorithm to use it for at least ten seconds. In fact, it would rather use the ramjet until the very last Mach value permitted (around 5.5), as the velocity plot Fig.8.2 shows, and then switch to rocket.

The reason for this choice is probably connected to the similarities of ramjet and scramjet and the fact that the first turns out to be more efficient in that specific range of altitude and Mach.

However, this peculiar behaviour needs to be further looked into and the cause of the problem might be even searched in the nature of the models which "level" the features of the engines and are not, for the simple reason of their being models, 100 % accurate on the real operation performances of the engines.

In the next section the optimal solution(i.e. the best optimal solution obtained so far) is displayed.

8.2 Best solution

8.2.1 Altitude

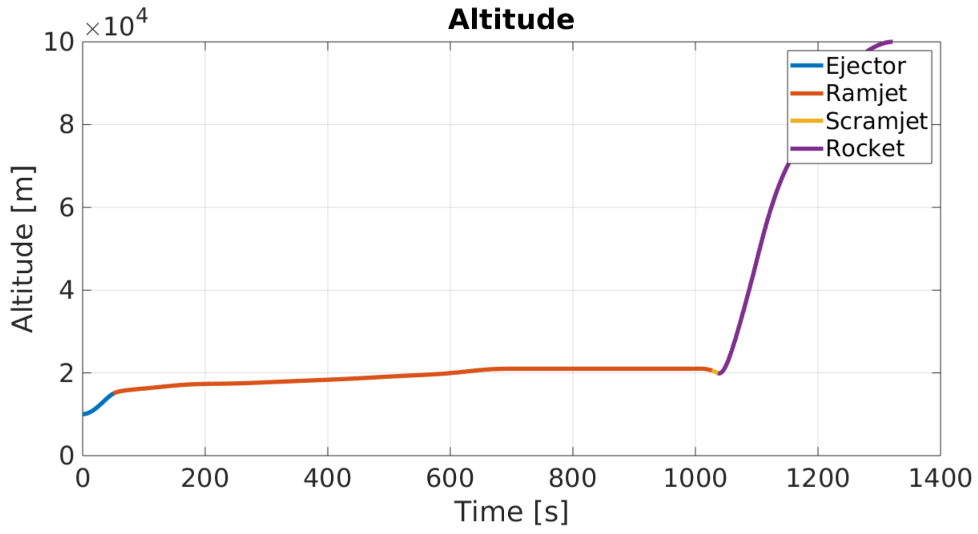

Figure 8.8: Best solution - Altitude.

8.2.2 Velocity

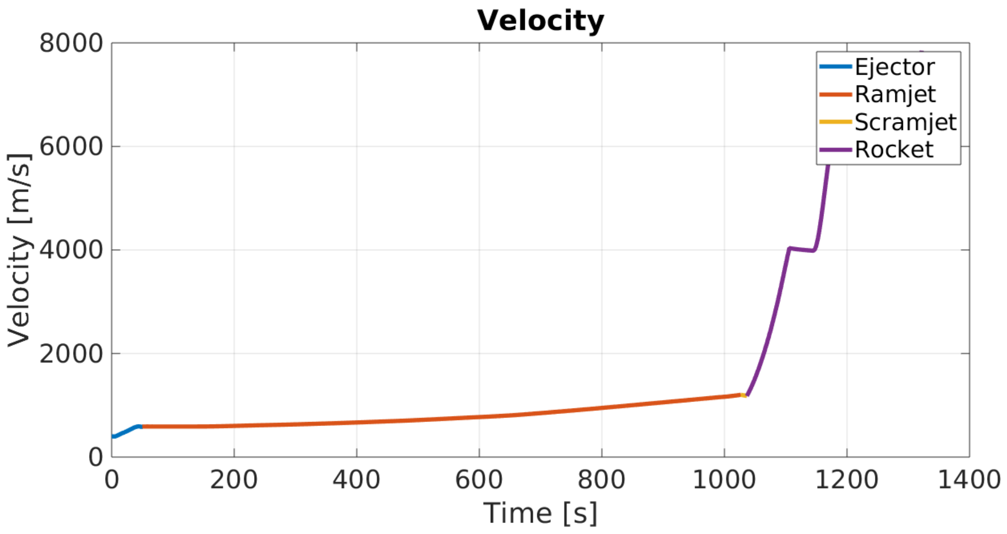

Figure 8.9: Best solution - Velocity.

8.2.3 Mass

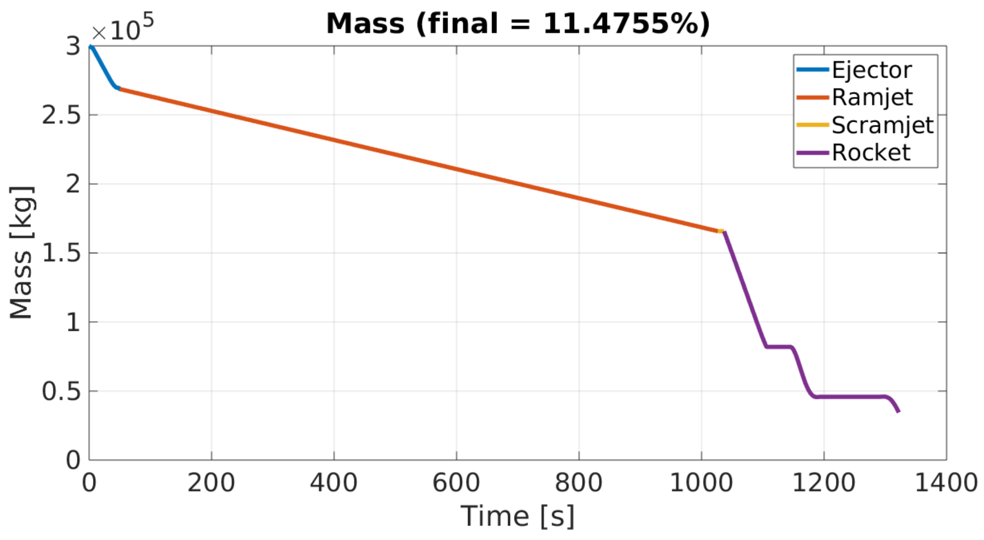

Figure 8.10: Best solution - Mass.

8.2.4 Angle of Attack

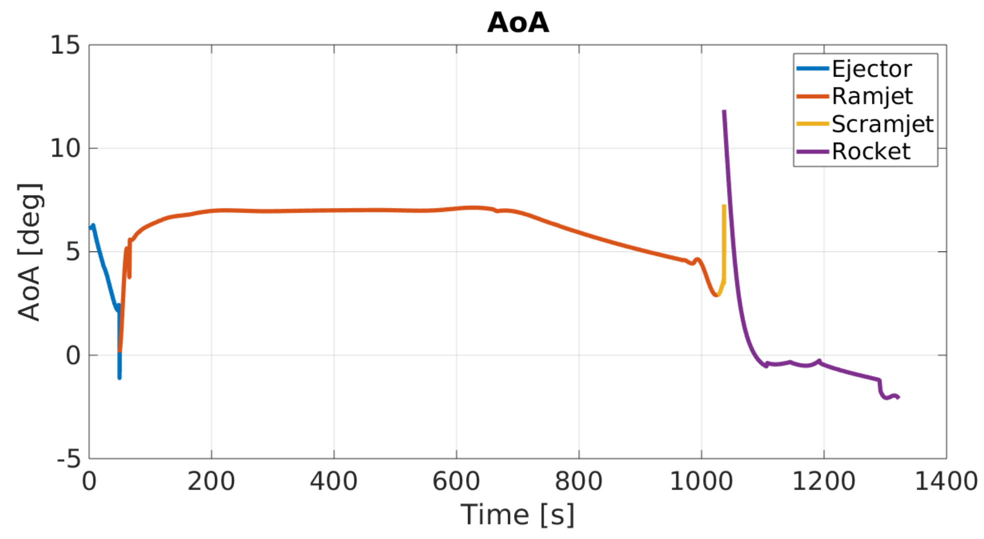

Figure 8.11: Best solution - Angle of Attack.

8.2.5 Throttle

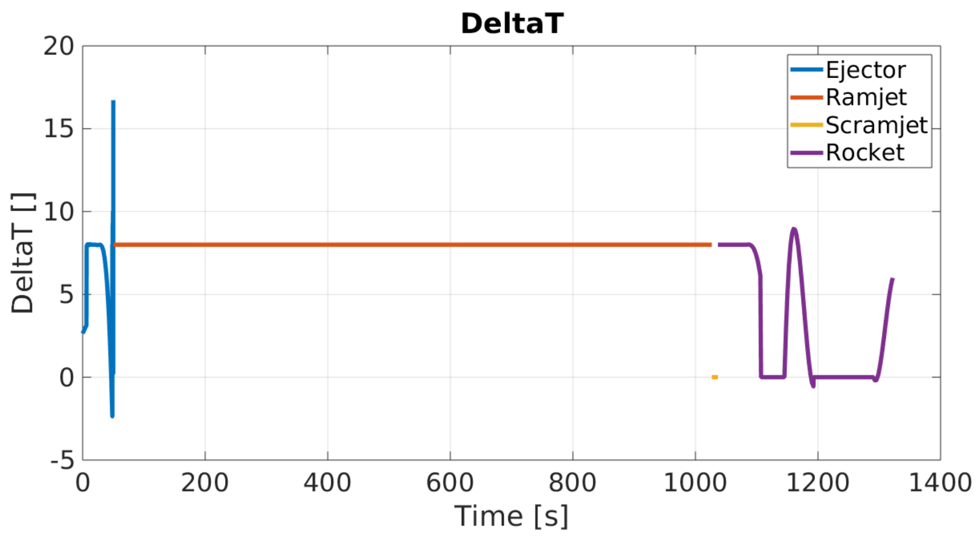

Figure 8.12: Best solution - Throttle.

8.2.6 Fly Path Angle

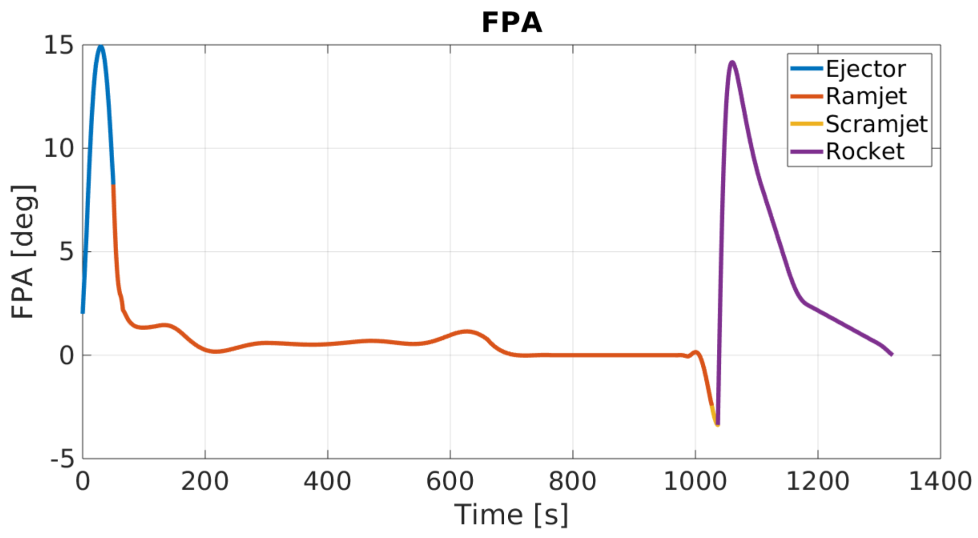

Figure 8.13: Best solution - Fly path angle.

8.2.7 Longitude and Latitude

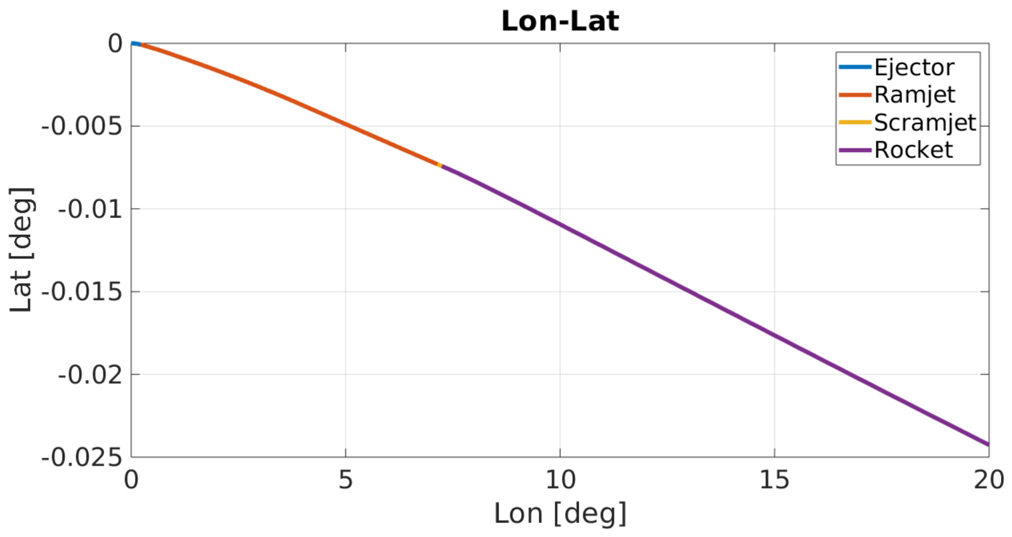

Figure 8.14: Best solution - Longitude and Latitude.

8.3 Design Cross Sectional Areas

Area $[ft^2]$	
Throat	4.14111378532018
Primary	9.1040239709584
Rocket Outlet	2.0370053551228
End Diffuser	22.3150969792183
Intake	24.9349549612673
Nozzle	40.6689583796651
Pre-Intake	23.6953392788937

Table 8.2: Best solution - Optimal areas

8.3.1 Comments

The final mass is

$$m_f = 42042\,\text{kg} \tag{8.3}$$

with a final payload ratio

$$\text{PayloadRatio} = 14.0139\,\% \tag{8.4}$$

This is the best optimal solution we were able to get.
We can notice the same elongations of the curves of the throttle, the flypath angle, the angle of attack we already saw in the former solution.
Again, the scramjet was used for a really short time and the speed curve increases steeply from the transition from ramjet to rocket.
The pattern of the two solutions is quite similar, the main difference lies in the higher value of the final mass and the best Payload Ratio the vehicle turns out having.

The reason why those specific areas are optimal for that given trajectory is a matter that should be further looked into. In fact, it is not as straightforward as one may think, as the optimality does not depend just on the engine performance, but there is the whole optimal control solution that comes into it and moves the needle of optimality towards values of cross sectional areas that would not be optimal otherwise.
This specific issues, together with others discussed in the next chapter, will be the focus of further research and improvement of the algorithm developed so far.

Chapter 9

Conclusions and Future Developments

9.1 Conclusions

As it was mentioned more than once throughout these pages, the purpose of this work was creating a routine for a bi-level optimisation of an hybrid propulsion system coupling the performances with the trajectory.
The interface of HyPro with MATLAB and the consequent development of the algorithm in this environment was certainly successful and made software more user friendly and improved the interface and the visualisation of the output.

The sensitivity analysis on the engine configurations of the software led the way for a better understanding of the tool itself and made it possible to simplify the optimisation process, focusing the attention on specific parameters whose influence on the output was more important.
A pivotal part of this work was the creation and testing of the surrogate models, from the construction of the database to the attempts of fine tuning with varied strategies and then the improvement of the training algorithm.

However, as it naturally happens during the design process, many simplifications had to be made: the surrogate models, for example, do not include the computation of the volume (and the weight) of the engine. The mass is a parameter of the optimal control problem but there is no computation of mass linked to the design areas of the engine. This natural selection of the surrogate models to keep and train was due to many factors, including time and computational costs, not to mention the simplified optimisation routine.
Generally speaking, the results obtained from the optimisation performed are realistic and do give us directions for an improved design for the spacecraft, but a more appropriate and realistic approach should include all the uncertainties.

Our goal here was to set up a process, a operational and effective process which could be the baseline for further improvement, and that goal was reached.

9.2 Future Developments

For future applications and improvements of the optimisation suite, a deeper understanding of the parameters influencing the output of HyPro would be ideal.
In this work, we focused on the cross sectional areas of the flow through specific sections of the engine, but there are many other parameters which could be investigated: the angle of injection of the fuel in the chamber, the angle of ejection of the rocket plume, the length of the modules, which could possibly have an influence on the resistence and the frictions on the walls of the case of the engine.
Once estimated the influence of these factors on the thrust and the mass flow of the engines, then the surrogate models could be more comprehensive and representative of the operational features of the four modes.
The uncertainties should be included and their distribution should be analysed, the nature of the optimality of the solutions obtained from the algorithm should be investigated and explained, these and many others improvement can and should definitely be made.
If so, the optimisation process developed in this work can, one day, be part of a much more complex and accurate optimisation algorithm for preliminary design of hybrid propulsion and, maybe, be integrated to many other modeling tools, not just HyPro.
The routine could be tested using other optimisation methods (other transcription methods for the optimal control problem, for example) and the general optimiser used could use a different algorithm.

There is so much space for improvement, in any direction, in any of the disciplines involved in this process, and we hope that this will be a valid starting point.

Bibliography

[1] Wen Yao, Xiaoqian Chen, Wencai Luo, Michel van Tooren, and Jian Guo. Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles. *Progress in Aerospace Sciences*, 47(6):450 – 479, 2011.

[2] Robert D Braun, Arlene A Moore, and Ilan M Kroo. Collaborative approach to launch vehicle design. *Journal of spacecraft and rockets*, 34(4):478–486, 1997.

[3] TD Robinson, Michael S Eldred, Karen E Willcox, and R Haimes. Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping. *AIAA journal*, 46(11):2814–2822, 2008.

[4] Anil V Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

[5] Zeyu Dong, Mingbo Sun, Zhenguo Wang, Jian Chen, and Zun Cai. Survey on key techniques of rocket-based combined-cycle engine in ejector mode. *Acta Astronautica*, 164:51–68, 2019.

[6] Lei Shi, Guojun Zhao, Yiyan Yang, Da Gao, Fei Qin, Xianggeng Wei, and Guoqiang He. Research progress on ejector mode of rocket-based combined-cycle engines. *Progress in Aerospace Sciences*, 107:30–62, 2019.

[7] Qingchun Yang, Wen Shi, Juntao Chang, and Wen Bao. Maximum thrust for the rocket-ejector mode of the hydrogen fueled rocket-based combined cycle engine. *International Journal of Hydrogen Energy*, 40(9):3771 – 3776, 2015.

[8] Daniel P. Loucks and Eelco van Beek. *System Sensitivity and Uncertainty Analysis*, pages 331–374. Springer International Publishing, Cham, 2017.

[9] Philip G Hill and Carl R Peterson. Mechanics and thermodynamics of propulsion. *aw*, 1992.

[10] Michael A. Patterson and Anil V. Rao. Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Trans. Math. Softw.*, 41(1), October 2014.

[11] Marilena Di Carlo, Massimiliano Vasile, and Edmondo Minisci. Adaptive multi-population inflationary differential evolution. *Soft Computing*, July 2019.

[12] Alessandro Mogavero and Richard E Brown. Modular, fast model for design and optimization of hypersonic vehicle propulsion systems. *Journal of Spacecraft and Rockets*, 55(5):1261–1281, 2018.

[13] Nikolaos V Sahinidis. Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering*, 28(6-7):971–983, 2004.

[14] Karel Crombecq, Luciano De Tommasi, Dirk Gorissen, and Tom Dhaene. A novel sequential design strategy for global surrogate modeling. In *Proceedings of the 2009 winter simulation conference (WSC)*, pages 731–742. IEEE, 2009.

[15] John Eason and Selen Cremaschi. Adaptive sequential sampling for surrogate model generation with artificial neural networks. *Computers Chemical Engineering*, 68:220 – 232, 2014.

[16] RD Braun, RW Powell, RA Lepsch, DO Stanley, and IM Kroo. Comparison of two multidisciplinary optimization strategies for launch-vehicle design. *Journal of Spacecraft and Rockets*, 32(3):404–410, 1995.

[17] M Giselle Fernández-Godino, Chanyoung Park, Nam H Kim, and Raphael T Haftka. Issues in deciding whether to use multifidelity surrogates. *AIAA Journal*, 57(5):2039–2054, 2019.

[18] Fabrizio Pescetelli, Edmondo Minisci, Christie Maddock, Ian Taylor, and Richard Brown. Ascent trajectory optimisation for a single-stage-to-orbit vehicle with hybrid propulsion. In *18th AIAA/3AF International Space Planes and Hypersonic Systems and Technologies Conference*, page 5828, 2012.

[19] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[20] Alessandro Mogavero. *Toward automated design of combined cycle propulsion.* PhD thesis, University of Strathclyde, 2016.

[21] Wei Huang, Li Yan, and Jian-guo Tan. Survey on the mode transition technique in combined cycle propulsion systems. *Aerospace Science and Technology*, 39:685–691, 2014.

[22] John R. Olds. Results of a rocket-based combined-cycle ssto design using parametric mdo methods. In *SAE Technical Paper*. SAE International, 04 1994.

[23] Tian-tian Zhang, Zhen-guo Wang, Wei Huang, Jian Chen, and Ming-bo Sun. The overall layout of rocket-based combined-cycle engines: a review. *Journal of Zhejiang University-SCIENCE A*, 20(3):163–183, 2019.

[24] OLA BREVIG. A simplified, preliminary comparison between the ejector ramjet and the shrouded rocket. *Journal of Spacecraft and Rockets*, 5(4):444–448, 1968.

[25] Ronald S Fry. A century of ramjet propulsion technology evolution. *Journal of propulsion and power*, 20(1):27–58, 2004.

[26] I.A Basheer and M Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3 – 31, 2000. Neural Computting in Micrbiology.

[27] Anil V Rao, David Benson, CL Darby, B Mahon, C Francolin, M Patterson, I Sanders, and GT Huntington. User's manual for gpops version 4. x: A matlab software for solving multiple-phase optimal control problems using hp–adaptive pseudospectral methods. *University of Florida, Gainesville*, pages 1–32, 2011.

[28] Edmondo Minisci and Massimiliano Vasile. Robust design of a reentry unmanned space vehicle by multifidelity evolution control. *AIAA journal*, 51(6):1284–1295, 2013.