TÉCNICO
LISBOA

Optimization of Approach Trajectories on Complex Force Fields Under Navigational and Operational Constraints

Application to the Mars-Phobos System

David Correia Melo

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Prof. Paulo Jorge Soares Gil

Examination Committee

Chairperson: Prof. José Fernando Alves da Silva

Supervisor: Prof. Paulo Jorge Soares Gil

Member of the Committee: Prof. José Raul Carreira Azinheira

September 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Dedicated to all those that pushed me into finishing it...

Acknowledgments

I would like to thank my parents and aunts for all their support during my whole education and all they contributed for me to have achieved this goal.

Also would like to thank João Branco and GMV for the opportunity that they gave me, to have been included in their project where I learned so much about the industry and about work ethic.

Lastly, I would like to thank professor Paulo Gil for pushing me towards finishing this thesis and for not giving up on my work.

Resumo

No nosso Sistema Solar existe o que chamamos de *corpos menores*, que são objetos com pequenas massas e formas anômalas que apresentam uma gravidade fraca e irregular, trazendo uma camada extra de complexidade ao tentar estimar as trajetórias ao seu redor. O objetivo desta tese é desenvolver um solucionador para um Problema de Valor Limite de Dois Pontos (TPBVP), que pode ser aplicado a corpos menores. O trabalho realizado aqui será baseado num pacote de software existente denominado *Small Body Navigation Analysis Tool* (SBNav), desenvolvido pela GMV. Esta ferramenta permitiu a propagação de trajetórias precisas perto de corpos menores, limitada a um Problema de Valor Inicial (IVP). Portanto, o SBNav foi adaptado para incorporar um algoritmo híbrido, uma combinação do Método de Nelder-Mead e um Algoritmo Genético. Finalmente, a ferramenta desenvolvida foi implementada com sucesso num caso de estudo, o sistema Marte-Phobos (Missão Phobos-SR).

Palavras-chave: Corpo Menor, Problema de Valor Limite de Dois Pontos, Método de Nelder-Mead, Algoritmo Genético, Missão Phobos-SR

Abstract

In our Solar System there is what we call *small bodies*, which are objects with small masses and anomalous shapes that present a weak and irregular gravity, bringing an extra layer of complexity when trying to estimate trajectories around them. The objective of this thesis is to develop a solver for a Two-Point Boundary Value Problem (TPBVP), which can be applied to small bodies. The work done here will be based upon an existing software suite called Small Body Navigation Analysis Tool (SBNav), developed by GMV. This tool allowed for the propagation of accurate trajectories near small bodies, limited to a Initial Value Problem (IVP). Ergo, SBNav was adapted to incorporate a hybrid algorithm, a combination of the Nelder-Mead Method and a Genetic Algorithm. Finally, the developed tool was successfully implemented on a case study, the Mars-Phobos system (Phobos-SR Mission).

Keywords: Small Body, Two-Point Boundary Value Problem, Nelder-Mead Method, Genetic Algorithm, Phobos-SR Mission

Contents

List of Tables

List of Figures

Nomenclature

ΔS	Constant/factor for simplex generation, representing the problem's characteristic length scale.
Δt	Transfer time from P_0 to P_T.
$f(P_0, V_0, \Delta t)$	Function representing the IVP tool of SBNav.
P_0	Starting location for the spacecraft.
P_T	Target location for the spacecraft.
V_0	Starting velocity for the spacecraft at P_0.
V_F	Final velocity given by Nelder-Mead Method if failed to converge into V_S.
V_S	Velocity solution for the spacecraft to reach P_T in Δt.
S_F	Simplex to be used by Nelder-Mead Method for convergence into V_S.

Glossary

AI	Advanced input option for TPBVP module.
ESA	European Space Agency
FASTMOPS	Fast Mission Operations Platform for Small Body GNC
INTNAV	Interplanetary Navigation Analysis Tool
IVP	Initial Value Problem
NAIF	NASA's Navigation and Ancillary Information Facility
Phobos-SR	ESA-Roscosmos Phobos Sample Return mission.
QSO	Quasi Satellite Orbit
SBNav	Small Body Navigation Analysis
TPBVP	Two-Point Boundary Value Problem

Chapter 1

Introduction

1.1 Motivation and Objectives

In our Solar System there are a multitude of astronomical objects with different characteristics and classifications. One type is what we call *small bodies*, which are objects with small masses and anomalous shapes that present a weak and irregular gravity. This brings an extra layer of complexity when trying to estimate trajectories around them and therefore, we need to develop different tools and methodologies to address this subject, given that the current methods used are not suitable.

The objective is to develop a solver for a Two-Point Boundary Value Problem (TPBVP) setting, in other words, to allow an user to specify both a start (P_0) and target end point (P_T) in space and the tool will be able to calculate the required velocity (V_S) in a predefined time interval (Δt) to reach the destination provided (Figure 1.1). On account of all the complexities regarding small bodies, classical methods such as Lambert's Problem are not applicable to this problem, hence a new approach is required.

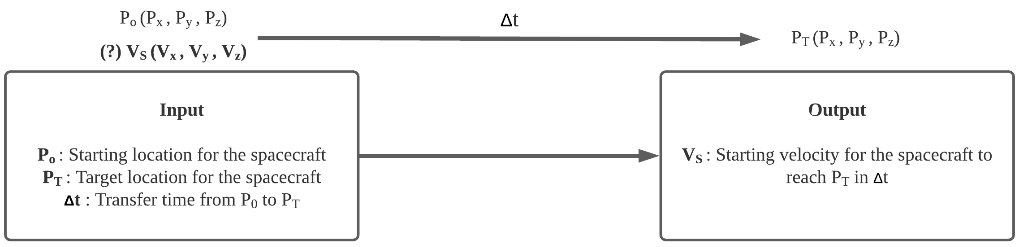

$P_o (P_x, P_y, P_z)$
(?) $V_S (V_x, V_y, V_z)$
Δt
$P_T (P_x, P_y, P_z)$
Input
P_o : Starting location for the spacecraft
P_T : Target location for the spacecraft
Δt : Transfer time from P_0 to P_T
Output
V_S : Starting velocity for the spacecraft to reach P_T in Δt

Figure 1.1: Overview of the TPBVP solver.

This thesis will be built upon an existing software suite called Small Body Navigation Analysis Tool (SBNav), which was developed under the project Fast Mission Operations Platform for Small Body GNC (FASTMOPS) by GMV, sponsored by the European Space Agency (ESA). This software is meant to perform a thorough navigation analysis around small bodies, with a focus on the system Spacecraft-Phobos. Among other functions, it allows solving Initial Value Problem (IVP) scenarios around objects with complex force fields, which is the basis for the TPBVP tool to be developed.

The SBNav tool allows for an user to provide as input a starting point in space (P_0), an initial velocity (V_0) and a time interval (Δt) and as output provides the trajectory and final location (P_F) that, for example, a spacecraft would accomplish on the proximity of a small body under the conditions provided (Figure 1.2). This is an over simplification of the features that SBNav offers but this will be addressed more in depth in a future chapter.

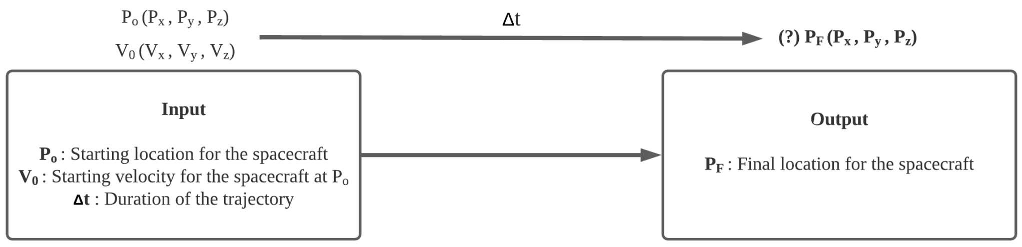

P₀ (Pₓ , P_y , P_z)
V₀ (Vₓ , V_y , V_z)
Δt
(?) P_F (Pₓ , P_y , P_z)
Input
P₀ : Starting location for the spacecraft
V₀ : Starting velocity for the spacecraft at P₀
Δt : Duration of the trajectory
Output
P_F : Final location for the spacecraft

Figure 1.2: Overview of the IVP solver from SBNav.

The case study for this problem is a mission to the Mars-Phobos system since it presents all the complex specifications that make a small body a challenge and which is why it was also used for the original SBNav. I must also state that the tool must be applicable for any small body in the Solar System, hence it needs to be a generic TPBVP solver, adding to the complexity of the problem.

As previously stated, small bodies present a vast array of inherent complexities that are caused by multiple factors: even the highest detailed images of these bodies' surface structure present a complex problem with many unknowns; their irregular shape which makes us dependent on approximated models for its density and consequently, its gravitational potential; a unique spin state which affects motion of other neighboring objects; planetary gravitational perturbations from other celestial bodies; perturbation from solar radiation; outgassing pressure in the case of comets. In other words, each system/scenario is unique and presents its own challenges [1]. These complexities and challenges are considered navigational constraints on approach trajectories for small bodies, considering their direct contribution and effect on the course of a spacecraft and its navigation.

When looking at the specific system of Mars-Phobos we will be dealing with, based on ESA requirements for FASTMOPS [2]: an approximated model for its shape and density and hence for its gravitational field; multiple gravitational perturbations from Mars, Phobos, Deimos (Mars' other moon), Jupiter, Saturn, the Sun; an estimate of Mars position in relation to Phobos; solar radiation. There are many difficulties from an orbital mechanics point of view and there are still limitations to the software suite that will be used that are still not accounted for and will be addressed further on.

The calculation of the necessary velocity to travel between two points in space in a defined time interval is the main objective of this thesis, but there is a secondary objective of determining *collision-free trajectories* with the tool developed. These are approach trajectories which guarantee that when the spacecraft reaches the target location (P_T) and continues on its current course, it would not result on a crash onto the nearest body. This type of approaches avoid the most direct path between the two defined points, requiring longer transfer durations.

Collision-free trajectories are useful from the point of view of Ground Operations, since communication with the spacecraft is not guaranteed at all times, hence placing a spacecraft in a trajectory that will not result into a collision and mission failure is advantageous. Specifically, in a scenario that a spacecraft reaches the target location (P_T), where it would start the descent and final approach to the body's surface, Ground Operations command control is released to on-board autonomy, but if final approach is aborted, then the spacecraft can safely continue on the current course, allowing for the necessary corrective actions to be performed later on. This type of scenarios are considered operational constraints on approach trajectories for small bodies, since the closer a spacecraft gets to the body's surface, the more unpredictable the region is and inputs from Ground Operations are further limited by communication speeds. Establishing collision-free trajectories are a containment for said operational constraints.

The determination of collision-free trajectories are not a boundary of the TPBVP tool/solver developed. In other words, the tool and its algorithm are not designed to find this type of trajectories in a *closed-loop*. These approaches are determined by the user by taking advantage of the ability to solve TPBVP and adjusting its inputs accordingly. However, the tool's ability of finding collision-free trajectories is testimony to its capabilities, considering the longer transfer durations required, which in turn subject the spacecraft to complex force fields for longer, increasing the difficulty of the problem. These matters will be made clear over the length of this thesis, after the complexities of the problem are made more apparent and afterwards, when examples for different trajectories and scenarios are displayed for the case-study selected.

1.2 State of the Art

The classical method for solving TPBVP and determining orbits around astronomical objects is the Lambert's Problem, but its major constraint is that one of the bodies must present an infinitesimal mass compared to the others, making it a two-body problem, which is a subject well known and studied.

In the case of small bodies, the gravity field is small and highly irregular which makes any orbit around it extremely susceptible to external perturbations. There are various literature works that study the subject, some from a more generic point of view encompassing all its varied challenges [1] while others focus on a specific one. There are works on asteroids and their different properties [3], others focus on gravity behavior under certain body shapes [4] or on how perturbations can be quantified and applied to the laws of motion [5]. One can also find numerous examples of studies more aimed towards specific missions or scenarios, for instance the asteroids Eros [6] or Itokawa [7], the moons Europa [8] or Phobos [9], and a binary asteroid system [10].

At the time of this thesis, there is no general method to solve the n-body problem, hence methods are developed in a case-by-case basis depending on particular characteristics of each individual astronomical object in study. GMV made extensive studies of the use of Quasi Satellite Orbit (QSO) on the Mars-Phobos system to place a spacecraft in a stable orbit of a small body [2]. For this case, while the spacecraft is orbiting Mars, the Phobos' gravitational pull prevents the spacecraft from drifting away, thus resulting in a ellipse-like path around Phobos (Figure 5.2). This type of orbit is sufficiently stable to allow for many months of operation in the vicinity of the moon. However, this approach is used for longer periods of motion as opposed to the short transfer trajectories that we face during TPBVP scenarios. This application of QSO is also supported by others works [11] but not applicable or in the scope of this thesis.

There are two main obstacles in this work: the physics behind orbital dynamics around a small body and the mathematical challenge of an non-linear three-dimensional boundary problem. GMV has addressed the problem of orbital dynamics when they developed the trajectory propagator present on the SBNav tool as an IVP solver, thus this thesis will focus on the mathematical problem of the TPBVP when based on the aforementioned IVP of SBNav.

1.3 The Tool

The software that will be serving as basis for this thesis and to achieve the set objective is SBNav. It was built upon another software suite called Interplanetary Navigation Analysis Tool (INTNAV) which was developed by GMV, as the name implies, for interplanetary navigation and deals with different dynamics and environment models as those required by the FASTMOPS project.

As a consequence of the complexity inherent to navigation around small bodies, a new tool was required. This tool would focus on the proximity of a spacecraft to a body of irregular shape and gravity field, and also help determine the impact of operational performance aspects and the hand-over from ground command to on-board autonomy.

The end result of SBNav is a generalist software suite that can be used or adapted to be applicable to any small body of the Solar System [2] but limited to IVP scenarios where from a starting point the tool is able to propagate the trajectory of a spacecraft in a defined time interval.

1.4 Thesis Outline

In the next chapter there is an overview of the SBNav features and its architecture with a focus on the trajectory calculation component. It is a more general perspective of the tool and its overall requirements.

Then, will proceed to review the possible methods to solve boundary value problems ranging from the more classical approaches to more complex and mathematically challenging solutions applicable to non-linear problems.

Subsequently, detailed information of the developed tool and algorithm is presented, showing how the tool is able to achieve the objective of this thesis. An overview of its architecture and different modules is provided, explaining how multiple approaches are used and combined to achieve the end result.

The developed tool is then applied to the Phobos-Mars system on different mission scenarios, estimating, when possible, any collision-free trajectories for said scenarios.

Finally, the discussion of the results obtained and the usability of the TPBVP tool created as the generalist utility that is designed to be.

Chapter 2

SBNav

2.1 Software Overview

A tool for navigation analysis requires a combination of multiple functionalities such as trajectory planning, models for different measurement systems, statistical estimation method and guidance laws for computing maneuvers. In order to achieve this, the SBNav software suite was broken into the following subsystems:

- **Trajectory Prediction**

  The capability of performing trajectory prediction based on the solution of equations of motion for the spacecraft. The product of this propagation is the spacecraft state vector and the partial derivative of this state vector with respect to the state at the initial epoch and any dynamic parameter affecting the spacecraft [12].

  The modules that allow for the functionality of this subsystem were developed in the language Fortran 77 which will also be a requirement/limitation for the TPBVP module to be developed.

  This will be the subsystem over which this work will be build upon and adapted to achieve the set objectives, hence it will be the one given more focus on this chapter. The remainder subsystems will only be briefly mentioned for a better overview of the tool and its complexity.

- **Measurement Generation**

  Models for different measurements considering all relevant parameters that contribute to the statistical errors of the model accuracy [12].

- **Parameter Estimation**

  A statistical method that provides parameter estimation and covariance analysis. A covariance analysis where the statistical random and systematic errors are properly accounted for, be it as estimated of or considered parameters [12].

- **Guidance**

  Trajectory control maneuver guidance laws for computing the trim maneuvers required to correct the trajectory and to achieve some given conditions at a point of the trajectory. Statistical estimation of maneuvers and associated execution errors [12].

- **Human-Computer Interface**

  To provide the user an interface to manage the definition and execution of a navigation strategy to be tested (to input date, execute commands and generate numerical and graphical output quantities required for mission analysis) [12]. This interface will also be used to run the new feature of TPBVP solving for the tool and it is based on the Matlab language.

- **Software Manager and Sequencer**

  In charge of managing and sequencing the execution of the different subsystems which are commanded by the user [12]. That is to say, provide a *bridge* between the Matlab interface/modules and the Fortran components.

- **Results Exploitation**

  To process the numerical data generated by the previous subsystem in a suitable way as required [12].

2.2 Requirements

In this section, there will be a focus on the trajectory definition and propagation requirements applicable to the TPBVP tool developed, given that the general requirements of the SBNav, as a full software suite, are outside the scope of this thesis.

2.2.1 Coordinate Systems

SBNav uses several types of reference frames; we highlight here those of particular relevance for our scenario: inertial, MEE2000, centered at either Mars or Phobos; body-fixed, centered at Mars; and synodic, a modification of the typical three-body problem reference frame. The latter is defined specifically for SBNav and it is the one used internally by the software for the spacecraft propagation [12].

Body-Fixed Coordinate System

Non-inertial (rotating with the given body). The fundamental plane is the equatorial plane of the given celestial body at the given epoch. The X-axis points along the prime meridian, the Z-axis points to the north pole and the Y-axis completes the right-handed system [12].

Synodic Coordinate System

In the Mars-Phobos case the primary body is Mars and Phobos is the secondary. The origin is the center of mass of the secondary body. The X-axis points in the direction of the center of mass of Mars, and the Z-axis in the direction of the orbital angular momentum vector, and the Y-axis completes the right-handed orthogonal reference frame. This system can be better observed in Figure 5.2 in a later chapter. The ephemerides and rotations of Phobos to assemble this reference frame follow the inputs provided by the user [12].

Inertial System

The inertial system used in SBNav is the Mean Earth equatorial barycentric system of J2000.0 (MEE2000). The direction of the coordinate axis is defined such that the XY plane coincides with the predicted mean Earth equator plane and the X-axis points towards the predicted mean vernal equinox at the epoch J2000.0 (noon of January 1st of the year 2000) [12].

2.2.2 Ephemerides and Rotational Elements

For Phobos and Mars, the rotational elements in MEE2000 shall follow NASA's Navigation and Ancillary Information Facility (NAIF) [13]. The user shall have the option to overwrite this data [12].

The celestial bodies' ephemeris will be loaded directly from files that can be modified/created by the user. The file will possess the Chebyshev coefficients of a body's trajectory divided into as many arcs as the user wishes [12].

2.2.3 Integration of the Equations of Motion

The software shall have the capability of performing trajectory prediction based on the solution of the equations of motion of the space vehicle (mass point model) with given initial conditions or final conditions [12].

Numerical integration

The numerical integration of the equations of motion is performed in a rectangular coordinate system. A fixed step size, 8th-order propagation method (Runge-Kutta) is used for the integration of the differential equations [12].

Dynamic model

Dynamic model options can be selected for each phase of the trajectory. The following sources of perturbing acceleration are considered:

- N-point masses perturbation model: acceleration due to the central body and the third body effects [12].
- Non-spherical gravitational effects: acceleration due to the non-spherical characteristic of the gravitational potential, expressed in terms of zonal and tesseral harmonics expansion. The model includes non-spherical gravitational effects of both the primary and secondary bodies [12].
- Solar radiation pressure: the force acting on a vehicle surface due to the solar radiation pressure is proportional to the effective area normal to the incident radiation and to a reflection coefficient and is inversely proportional to the square of the distance from the Sun. Solar eclipses caused by the primary's and secondary's shadows are considered [12].

Gravitational Models

The gravitational models indicated below are used for the integration of the equations of motion.

- **Phobos**

  A quadruple field gravitational harmonics (J2 and C22) model shall be considered – the default coefficients are user-configurable and the field can be up to a 50x50 field. The standard values are a 2x2 field that follows from NASA's Navigation and Ancillary Information Facility (NAIF) [12].

- **Mars**

  NASA's Navigation and Ancillary Information Facility (NAIF) model shall be used, reduced to degree 10 for Mars zonal harmonics and degree and order 10 for tesseral harmonics. Coefficients are user-configurable and the field can be up to a 50x50 field [12].

- **Other Celestial Bodies**

  For the remaining objects, the point mass models from NASA's Navigation and Ancillary Information Facility (NAIF) are considered. The user shall have the possibility to overwrite these values [12].

2.2.4 Trajectory Definition

The following indications are valid for the original SBNav (IVP tool only) for scenarios defined as an Initial Value Problem (IVP), the different inputs required by the user for the TPBVP module will be addressed in 4. Segmentation of trajectories shall be possible by starting a project with other project's final conditions but this feature is outside the scope.

Initial Value Problem

- **Initial State** - It is possible to compute the trajectory by propagation of the equations of motion, from an initial state (forward propagation) or final state (backwards propagation). The following options are available for initial state definition:
  - Direct input;
  - State and time at the end of another project;
  - State and time at the end of another project + direct input dV.
- **Reference Frame** - Initial state vector can be input in either Inertial, Synodic, Body-Fixed Reference Frame, or Orbital Elements.
- **Stop conditions** - The system has the capability to stop the integration of the spacecraft motion:
  - At a defined date. The user can define a date (with several format options) in which to stop the propagation;
  - After a defined time has passed;
  - At xz-semi-plane crossing on the positive or negative x-semi-axis side after n revolutions around the secondary have been completed. The user must define the number of revolutions to be completed before the trajectory is to be stopped at the semi-plane crossing;
  - At yz-semi-plane crossing on the positive or negative y-semi-axis side after n revolutions around the secondary have been completed. The user must define the number of revolutions to be completed before the trajectory is to be stopped at the semi-plane crossing;
  - At xy-plane on upwards or downwards crossing after n revolutions around the secondary have been completed. The user must define the number of revolutions to be completed before the trajectory is to be stopped at the plane crossing.

Chapter 3

Boundary Value Problem Literature Review and Approaches

3.1 Two Point Boundary Value Problem

This chapter will present each approach used to create a TPBVP solver following the same order they were researched, method by method, section by section, and explain why certain option were favored while others were discarded.

Mathematically speaking, when a problem presents boundary conditions which are defined at different values of the independent variable this creates a boundary value problem. In other words, when boundary conditions are specified at more than one point, typically, some of the conditions will be specified at a starting point P_0 and the remainder at a final/target point P_T. [14]

Considering all the difficulties inherent to estimating orbits around small bodies, we also know that we will be working with a highly unstable environment from the point of view of astrodynamics, which translates into a non-linear multidimensional boundary value problem mathematically and this will be the problem for which a solution must be devised.

3.1.1 Lambert's Problem

Lambert's problem is the classic solution to the boundary value problem for interplanetary travel, where one of the bodies is infinitesimally smaller and the other two can be taken as two point masses, hence it can be approximated into a two-body problem. Currently, the method is still well regarded [15], but unfortunately it is not applicable to our scenario, an N-body problem. However, it can be used for an initial guess of the TPBVP solver developed. As a result of experimentation with different trial scenarios, it is found that Lambert's problem provides an acceptable initial guess which can facilitate convergence for short transfer trajectories between two established points in the TPBVP, which are not exposed to the highly irregular gravitational field for long.

3.2 Quasi-Newton Methods

As it has been stated previously, SBNav presents an IVP solver, which was tested and validated by GMV via the FASTMOPS project, for the problem of small bodies [16] which will be adapted into a TPBVP solver, this is classically called a *shooting method* in the field of numerical analysis [14, 17] and involves finding the IVP solution that will satisfy the boundary conditions. In other words, one tries multiple IVP solutions trying to find the one that will fulfill the set boundary conditions, which mathematically translates into finding a root for the function $F(.)$ which represents the difference between the desired boundary value and the one obtained by the IVP. If applying this concept to our sample problem, we have

$$F(V_0) = f(P_0, V_0, \Delta t) - P_T \tag{3.1}$$

where P_0 is the starting location for the spacecraft; P_T is the target location for the spacecraft; Δt is the transfer time from P_0 to P_T; V_0 is a starting velocity for the spacecraft at P_0; and $f(P_0, V_0, \Delta t)$ is the function representing the IVP tool of SBNav.

Now that we defined the Equation 3.1, we need to find a root-finding method that can find a V_0 that will satisfy $F(V_0) = 0$, which takes us to the classic iterative method of Newton's Method [14]. This method relies on the derivative of the shooting function $f(p, v, t)$ to estimate the next iteration when starting from an initial guess V_0. Since the problem at hand is multidimensional the result would be a Jacobian matrix, that because of the non-linearity of the problem at hand, presents us with an ill-conditioned matrix that invalidates this method completely or discourages one from attempting it. Other variations of the method were analyzed [18, 19], but were also limited by the same reason or by the presence of multiple local minima which are also particular for our case study. This new issue will be better addressed further on, since at the moment the ill-conditioned Jacobian is the main obstacle.

Based on [14] it was decided to avoid the classical Newton-Method and continue on to the family of the Quasi-Newton methods, since these are designed for problems where the Jacobian or Hessian matrices are unavailable or require extensive computing power to be calculated at each iteration, which seems to fit with the current obstacle of an ill-conditioned Jacobian when selecting a valid root-finding method. From this family, a very robust and powerful method is the Broyden's Method [14] which is mentioned in multiple works regarding Quasi-Newton methods applied to nonlinear systems [20–26], thus it takes us to the next section.

3.2.1 Broyden's Method

From the Quasi-Newton group of methods available, there is one subgroup called *secant methods* which obtain an approximate Jacobian by applying the secant method [14, 27] in one dimension, which is a variation of Newton's Method [14]. The first and best of these methods [14] seems to be Broyden's Method [28] which the literature seems to agree into being a valid option for multidimensional and nonlinear systems [14, 29–33], hence the TPBVP tool started development with this as the root-finding method.

Unfortunately, the strength of this method is also its weakness, granted that it can converge faster with less cost, computationally speaking, compared to similar methods, it is still dependent and limited by an approximated Jacobian and its properties. For instance, if it becomes singular or nearly singular or the problem is ill-conditioned, then this method becomes incapable of converging. For our specific problem, we are trying to find the three-dimensional velocity vector which allows the spacecraft to reach the desired target location, but the Z component of this vector is in different orders of magnitude compared with the other ones, which makes it ill-conditioned and consequently, will cause the Jacobian to be nearly-singular, invalidating the method completely.

The limitation of the method is a known quantity and there have been some strategies that were developed to mitigate or bypass this factor of facing a singular or nearly-singular Jacobian [14], such as starting from the identity matrix as initial Jacobian for the iteration process, supposedly neutralizing the problem that the Z component causes. However, it was observed after multiple tries, that the iterative process would get blocked sooner or later by a nearly-singular Jacobian, which would be generated by the mathematical nature of the problem.

More attempts to try bypassing the issue of an ill-conditioned Jacobian could have been attempted, but while reviewing more of the literature it was also clearer that a root-finding method based in Newton's Method or its variations which are designed to converge based on the derivative of the problem do not behave well in solution spaces with multiple local minima for they would get *stuck* trying to converge into one indefinitely with no safeguard that would allow it to jump out of this area and look for a new minima. With this conclusion and the introduction of the idea of local minima we are taken into the field of minimization or maximization of functions, also called *optimization mathematics*.

3.3 Optimization Mathematics

In the minimization or maximization of functions the idea is similar as in root-finding methods. For in this methodology, one has a function with N independent variables for which the correct values of the latter need to be determined in order to achieve a maximum or a minimum. The terminology is slightly different and the concept of local vs global minima is also introduced. If a value is found that represents the absolute highest or lowest value of the function in its domain then this would be called a global extremum (maximum or minimum point), while other points that are found to be the highest or lowest value in a finite space of the function's range are called local extremum.

Our function to be minimized was already defined on Equation 3.1 and as was previously mentioned, our function's particular physics problem of motion around small bodies presents the characteristic of numerous local minima in its range, which increase in number for longer transfer trajectories, causing longer periods where the spacecraft is subject to an unstable area from the perspective of orbital dynamics. For our case study, there is no need to find a global extremum, considering these are extremely hard to find [14], therefore local minima near zero will be acceptable (exact specification for *near zero* will be defined later on Chapter 5), unfortunately these minima are also rare.

According to [14], there are two main methods applicable to a multidimensional case that do not need the function's gradient: Nelder-Mead Simplex Method [34] and Powell's Method. Since it takes time to adapt SBNav and the TPBVP tool to include a new method and there was a time constraint in having access to GMV tools and documentation, it was decided to select one of these two to continue. The Nelder-Mead Method was chosen due to being extremely robust, although at a cost of also being extremely slow for higher dimension problems [14]. Nevertheless, considering our problem's moderate dimension and the fact that Fortran, a very powerful programming language [35], is used for the computing process, it is not expected that the known weaknesses of this method will manifest for our particular problem.

3.3.1 Nelder-Mead Simplex Method

A simplex is the geometrical figure consisting, in N dimensions, of $N+1$ points or vertices and all their interconnecting line segments and polygonal faces [14], which for our problem of three dimensions would be a tetrahedron. To generate this geometrical form, one would take the initial guess of the problem as P_0 (it can assume any of the vertices' positions) and from there, generate the other N points/vertices based on

$$P_i = P_0 + \Delta S e_i \tag{3.2}$$

where the e_i's are N unit vectors and ΔS is a constant/factor that represents the problem's characteristic length scale, which will affect how *spread* apart are the points and how each new point and iteration will be generated, in other words, it will affect how further away one goes on each iteration starting from the initial guess, depending on the type of operation [36].

The Nelder-Mead Method will then evaluate the function at each of the $N+1$ points and replace the worst one with different set of operations. Below is a quote regarding the iterative process from [14], seeing that from all the literature, it presents the best explanation for the process that the method follows:

> ”The downhill simplex method now takes a series of steps, most steps just moving the point of the simplex where the function is largest (“highest point”) through the opposite face of the simplex to a lower point. These steps are called reflections, and they are constructed to conserve the volume of the simplex (and hence maintain its nondegeneracy). When it can do so, the method expands the simplex in one or another direction to take larger steps. When it reaches a “valley floor,” the method contracts itself in the transverse direction and tries to ooze down the valley. If there is a situation where the simplex is trying to “pass through the eye of a needle,” it contracts itself in all directions, pulling itself in around its lowest (best) point.” [37]

For our specific problem, the method will be considering as its termination criteria the difference stated in Equation 3.1, taking into account an acceptable tolerance (to be specified by the user) for which the spacecraft can be deemed *close enough* to the target point, guaranteeing a near zero result. That is to say, the objective is for a spacecraft to reach the target coordinates P_T, though in practice there is a margin of error inherent to such scenario, which the simplex method tries to minimize. Namely, the goal is to reach an end position P that is in an acceptable range, to be defined by the user, of the target P_T. The first version of the function for simplex generation (GENPLEX1) was developed to adopt a factor ΔS input by the user, although it assumes a user's prior knowledge of the mathematical problem and/or of Nelder-Mead Method.

After integrating the method into the tool, it was the first time that convergence was reached, specifically the first occurrence of the tool finding a velocity V_S that guarantees the spacecraft will reach P_T in the defined Δt. This solution was found quite fast I must add (one to two minutes), however when testing the method versus scenarios with longer and longer transfer trajectories it started showing a higher sensibility to the initial guess provided. For cases where the initial guess would be highly inaccurate (which most of them are), the method would keep *sinking* into multiple local minima neighborhoods and would never converge, or if fortunate, it would converge after thirty minutes or more, if enough iterations were allowed.

To better understand the irregular and complex solution space that the problem presents, a number of 3D figures were plotted of the latter (Figure 3.1 to Figure 3.4). These plots are based on an IVP tool fully developed in Matlab by GMV (for the purpose of this thesis, it shall be named *GMVPropagator*), further adapted to incorporate a TPBVP solver. However, this tool's IVP solver is based on a propagator of the 4th-order propagation method (Runge-Kutta) with simple approaches for the remainder perturbations/complexities. These approaches were established by GMV to be sufficient for the purpose of plots and other illustrative objectives, though not acceptable for the more rigorous standards of FAST-MOPS (Chapter 2). Finally, note that this GMVPropagator has a much lower precision than SBNav and is considerably slower.

The GMVPropagator can solve TPBVPs based on the Matlab function *fminsearch*, which is the equivalent of the Nelder-Mead Method. This tool is used as a substitute of the more robust and faster SBNav as a result of the latter's limitations. SBNav Matlab and Fortran *symbiosis architecture* limits its adaptability, hence it is only designed to output one solution of the problem. Therefore, GMVPropagator is used for the purpose of plots and simple representations of the problem characteristics. Particularly, the representation of a solution space of possible velocities that a spacecraft can assume and the resulting achieved position. For the purpose of a 3D representation of a 4D problem, the V_Z component was locked and replaced by $F(V)$ on the Z-axis. The following plots, Figures 3.1 to 3.4, are focused around a known solution V_S for a specific scenario, which can be found at the center of the XY-plane. There, *valleys* of local minima are observable in the neighborhood of the solution V_S, causing the Nelder-Mead Method to keep attempting to converge indefinitely into said minima, not allowing for guaranteed convergence into V_S.

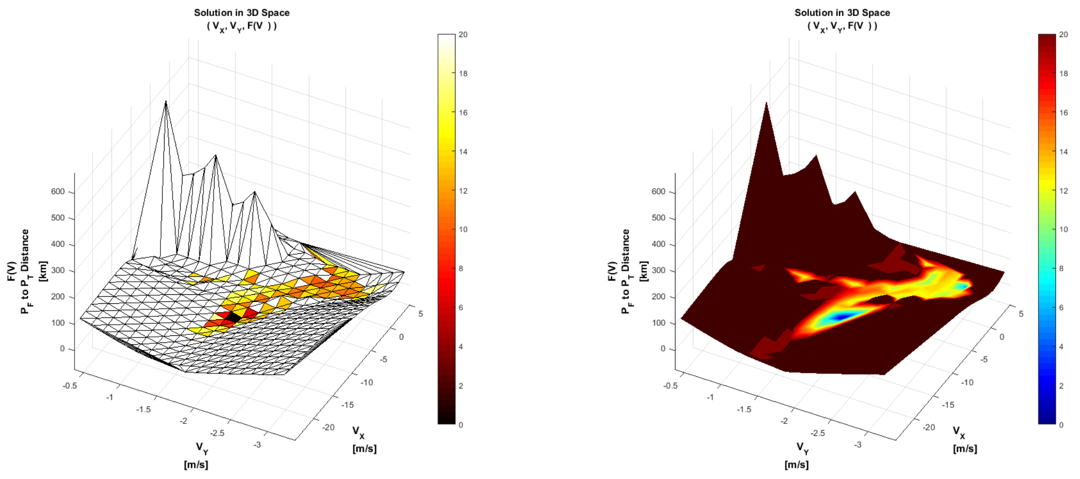


Figure 3.1: Zoomed in solution space surrounding V_S for Scenario 5, Subsection 5.3.5.

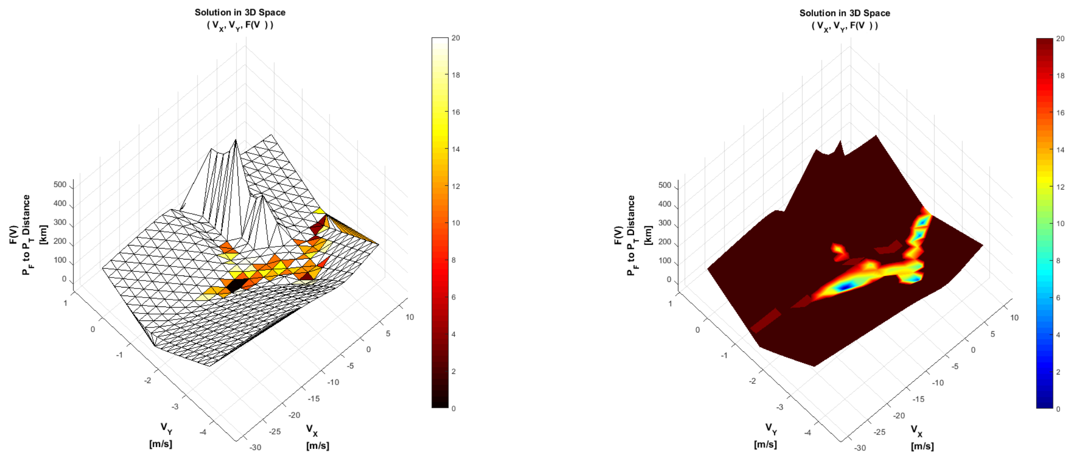


Figure 3.2: Zoomed out solution space surrounding V_S for Scenario 5, Subsection 5.3.5.

From the examples of Figure 3.1 to Figure 3.4, the highly irregular solution space is discernible, even when locking the V_Z component, thus making the observed space a somewhat optimistic version of its real complexity. Ergo, the circumstances led to a search for some type of safeguard or alternative implementation of the method, where some of the literature provided possible workarounds [38–43]. The most promising one would be making the factor ΔS an adaptive one, that could react in a way to the convergence of the problem on each iteration, making the iteration itself adaptive to the current simplex and solution environment.

The second version of the function for simplex generation (GENPLEX2) was developed to alter V_0 by a fixed interval and select the best four variations, producing a simplex (Subsection 4.3.8). Unfortunately, this modification of the method is not guaranteed to work with any scenario or problem. It would probably be possible to implement this change for the problem of Mars-Phobos system and its dynamic, but this would go against the concept of a generic TPBVP tool.

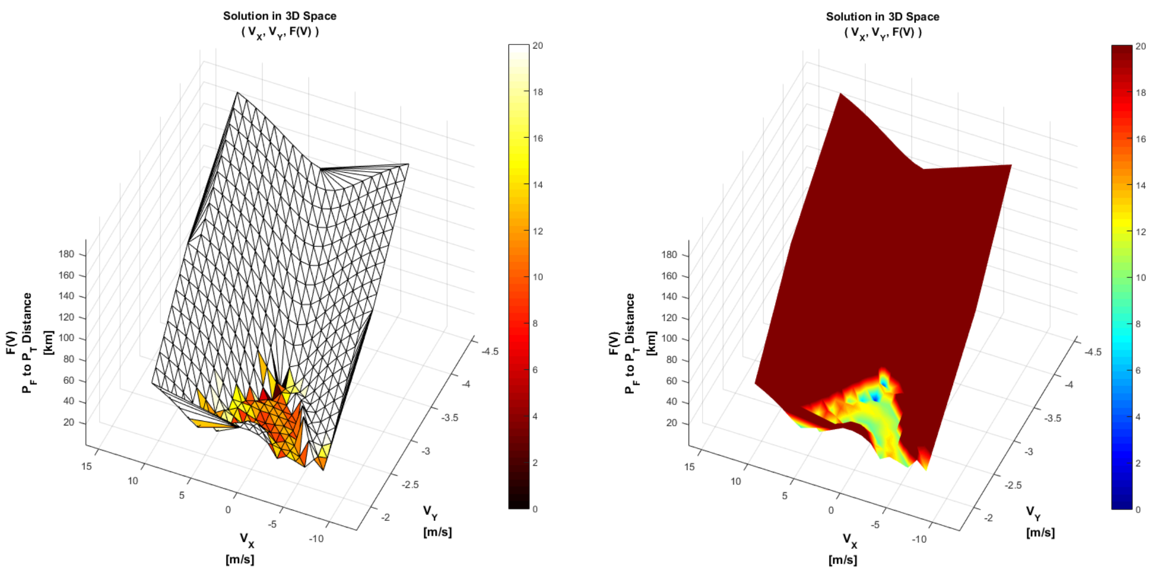


Figure 3.3: Zoomed in solution space surrounding V_S for Scenario 6, Subsection 5.3.6.

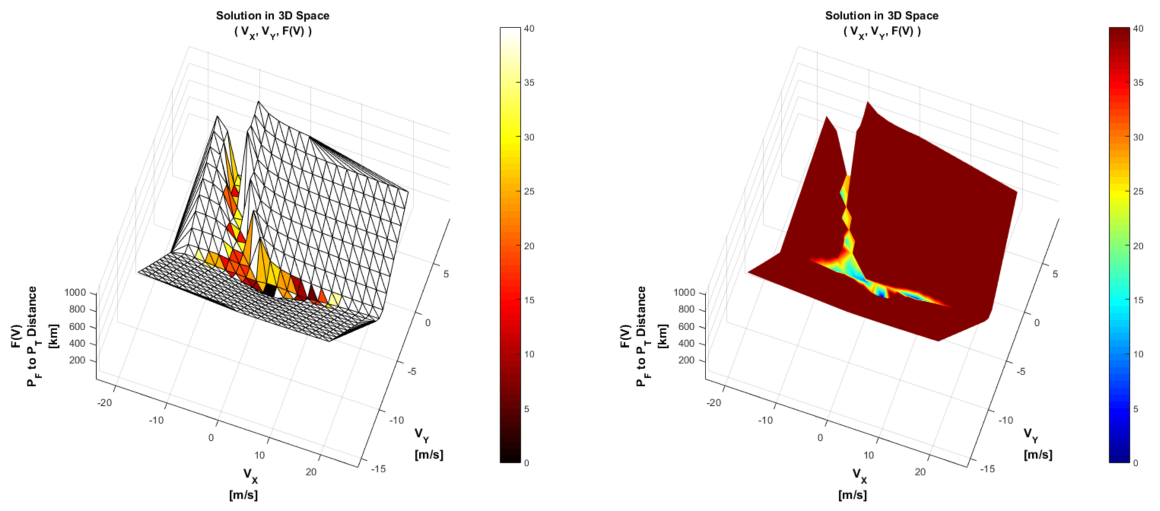


Figure 3.4: Zoomed out solution space surrounding V_S for Scenario 6, Subsection 5.3.6.

Faced with the current obstacle, a different approach was taken, which involved running the Nelder-Mead Method (via GENPLEX2 - Subsection 4.3.8) and observing each iteration individually and understand how it behaved with longer trajectories. Considering there were some solutions available for longer trajectories (obtained by longer iterations and knowledge of the problem at hand), these solutions were observed and specific safeguards to guarantee convergence were implemented (addressed in detail in Chapter 4). After a considerable amount of time and multiple modifications, the tool was working with a much higher degree of efficiency, though now showing multiple similarities to a *genetic algorithm*.

3.4 Genetic Algorithm

A genetic algorithm is a heuristic which is shared among multiple fields of science [44–46]. Although it might be defined with different terminology, depending on a specific field, there is nevertheless a general definition. A genetic algorithm is an approach based on biological evolution, in a way that relies on biologically inspired operators to generate solutions for optimization or search problems. These operators are commonly known as *selection*, *crossover* and *mutation*.

The process that the algorithm follows is relatively simple, starting from a group of individuals, also known as *generation*, which suffers a mutation and/or crossover of certain properties, followed by a selection of certain individuals that will create a new generation. This iterative process can continue indefinitely until a generation is found holding the desired solution.

The possible representation of what a mutation, crossover or selection entails varies greatly between different cases. For the work developed in this thesis, one can see a generation as different vectors representing velocities (individuals); mutation as changing one component of each vector by a certain amount; crossover as exchanging different components between vectors; selection as choosing from the generated velocities the ones that provide the best result. This is not to be taken as an exact representation of the developed genetic algorithm, but as an example focused in our specific case.

From the previous Subsection 3.3.1, a hybrid algorithm including the Nelder-Mead Method and some characteristics of a genetic algorithm was created. The latter had mutation and selection operators, but not crossover. Based on literature [47, 48], the crossover operator was implemented to provide more *genetic diversity* to each generation. Note that the genetic algorithm created was fully developed during the course of this thesis, hence its operators are particular to this work. The operations of mutation/selection/crossover were established based on a heuristics born from experience with the problem and analysis of the solution space and simplex convergence behavior. The finalized hybrid algorithm relies on the genetic algorithm to filter out the majority of undesired local minima and start towards the solution of the problem via an optimized simplex (GENPLEX3); when close to the latter, will then use the Nelder-Mead Method for a faster convergence, making use of a more efficient computational approach. The detailed architecture of such tool and how the genetic algorithm is defined will be addressed on the next chapter.

Chapter 4

TPBVP Module Specification

4.1 Hybrid Algorithm

In this section is presented the basic concept of the hybrid algorithm developed, based on the previous chapter. As per Figure 4.1, the following process is followed: an initial guess V_0 is given; based on how good or bad the initial guess is, a class is defined, which will affect the order of mutation of the genetic algorithm; the genetic algorithm is defined by three phases (blue boxes on Figure 4.1 - M stands for Mutation, S for Selection and C for Crossover), where all three components of the velocity vector are altered, X in Phase 1, Y in Phase 2 and Z in Phase 3, generating new guesses/individuals; from the new individuals, the best four are selected, generating a simplex S_F; this simplex is then submitted to the Nelder-Mead Method; if the Simplex Method converges than the algorithm has successfully achieved its goal; otherwise, the local minima, for which the Simplex Method got stuck and unable to converge, is taken and replaces the original initial guess V_0, starting an iterative process. The following sections will focus on the implementation of this concept into the SBNav tool and a more detailed overview of the module architecture.

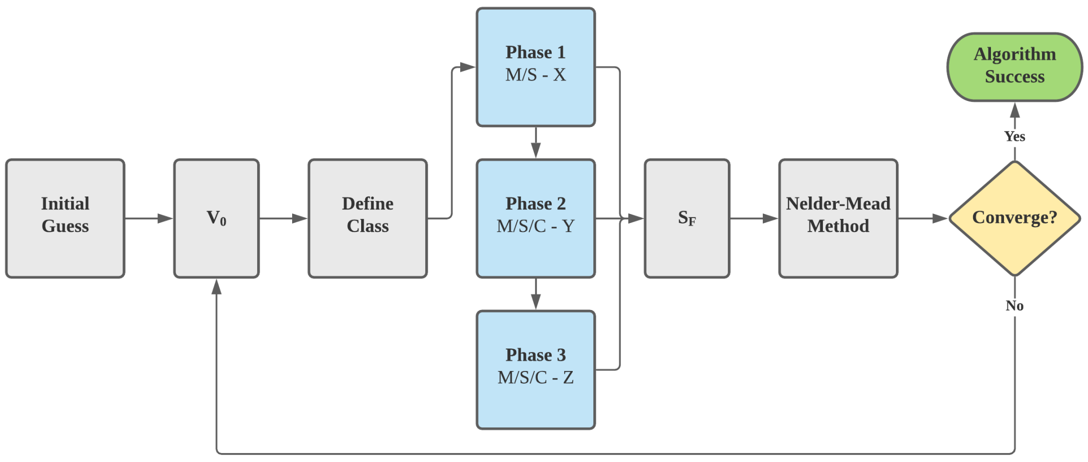


Figure 4.1: Basic overview of the hybrid algorithm developed.

4.2 User Input

The SBNav inputs can be broken down into three categories: scenario characteristics, with the possible options listed in Section 2.2 and then detailed for our specific scenario in Section 5.2; TPBVP inputs; and lastly, the advanced inputs (identifiable by the *-AI* add-on), found in different sections of the tool with specific purposes - editing them is not recommended. The present section will focus on the inputs regarding the boundary value problem itself. Next, the possible options will be listed and described. An overview can be found on Table 4.1.

Table 4.1: User input options regarding TPBVP tool.

TPBVP User Input	**Options**	**Detailed On**
Initial Guess	Lambert's Problem Solver	Subsection 4.3.1
	Sphere of Guesses	Subsection 4.3.1
	User Input	Subsection 4.3.1
Generate Simplex	GENPLEX1	Section 4.2
	GENPLEX2	Subsection 4.3.8
	GENPLEX3	Subsection 4.3.3 to 4.3.5
Initial State	Initial state vector	Section 4.2
Target State	Target state vector	Section 4.2
Initial State Reference Frame	Synodic Reference Frame	Section 2.2.1
Stopping Conditions	Stop after a given time	Section 4.2
Transfer Duration	Time from Initial to Target state	Section 4.2
Error Tolerance	Minimum tolerance accepted	Section 4.2

Initial Guess

The user can choose from three options to obtain V_0: Lambert's Problem solver, shown to be somewhat a good estimate for short transfers ($\Delta t < 6\ h$ for Mars-Phobos system); sphere of guesses, where the best estimate is chosen from multiple guess samples; user input. Detailed description on Subsection 4.3.1.

Generate Simplex

Once more the user can choose from three options, now to generate the simplex S_F which will be used by the Nelder-Mead Method: GENPLEX1, simplex generated based on user provided factor OFACT(-AI) (ΔS = OFACT), not the recommended option since it depends highly on a good V_0; GENPLEX2, simplex generated based on altering V_0 by a fixed interval and select the best four variations (detailed description found on Subsection 4.3.8), not suggested as the core simplex generator, although is used in certain sections of the TPBVP solver; GENPLEX3, simplex generated based on adaptive factor ΔS (detailed description found on Subsection 4.3.3, 4.3.4 and 4.3.5), the recommended option with the most robust approach.

Initial State

User must provide a vector P_0(1x6), where P_0(1,1:3)[km] are the spacecraft's initial position coordinates on the chosen reference frame in the *Initial State Reference Frame* option. The remaining components P_0(1,4:6)[km/s] can be used to give a V_0 input by the user, if so chosen in *Initial Guess* option, otherwise they will be ignored by the tool.

Target State

User must provide a vector P_T(1x6), where P_T(1,1:3)[km] are the spacecraft's target position coordinates on the chosen reference frame in *Initial State Reference Frame* option. The remaining components P_0(1,4:6)[km/s] are ignored by the tool, their existence is only justified by having uniform sized vectors during the tool's operations.

Initial State Reference Frame

The only available option for the user as a reference frame is the Synodic option, detailed in Section 2.2.1.

Stopping Conditions

The only available option for the user as a stopping condition is after a user specified Δt [Days] in *Transfer Duration* option, the spacecraft must reach P_T.

Transfer Duration

Option where the user must define a time Δt [Days] for the spacecraft to go from initial position P_0 to target position P_T.

Error Tolerance

The user must specify a minimum tolerance TLMIN [km] where $F(V) \leq TLMIN \approx 0$. The default value is one millimeter (0.000001 km).

4.3 Module Architecture

This section will focus on the overall module architecture, detailing each of its segments and processes independently. These segments, or steps, are defined by a blue box on the architecture overview on Figure 4.2, also marked with a [.]. Strictly speaking, the module will start from an initial guess V_0 and from there try to converge into a solution to the problem V_S. To elaborate on this, the process starts from a initial guess V_0, then proceeds into the GENPLEX3 phases: in phase 1, the V_X component goes through mutation and selection; in phase 2, the V_Y component goes through mutation, selection and crossover; lastly, in phase 3, the V_Z component goes through mutation, selection and crossover. After gathering the best candidates from the three phases, four are selected to generate a simplex S_F, which is then submitted to the Nelder-Mead Method. If the method converges into a problem solution V_S then the algorithm is successful, otherwise will instead produce a local minima V_F. From failed convergence, V_0 is then updated with the previously obtained V_F, starting a new iteration. There can only be four iterations of the process described by default, after which will be considered an algorithm failure. The user can choose to modify it, by editing the input variable NRSTR(-AI) into another integer greater than zero, by default is set to 5 (0 $<$ number of iterations $<$ NRSTR).

Note two unique features of this architecture. The first, regarding the event where the initial estimate V_0 is considered *good enough* to skip GENPLEX3's phases, taking advantage of GENPLEX2's faster approach to generate the simplex S_F. The second, about the tool's ability of reversing the order by which the X-Y-Z components are mutated in GENPLEX3's phases, when enabling a specific flag IMUTA(-AI). For $IMUTA = 13$, the process takes the path described in the previous paragraph: Phase 1 alters V_X, Phase 2 alters V_Y and Phase 3 alters V_Z. For $IMUTA = 31$, the process takes the inverted path: Phase 1 alters V_Z, Phase 2 alters V_Y and Phase 3 alters V_X. This particular feature was useful during the development stages of the tool, since it allowed to study a possible advantage, by allowing V_X to suffer a greater range of mutations if it came last. In the end, this aspect proved to be ineffective with no obvious gains, hence it is not used or mentioned in the remainder of this work.

Considering simplex properties, there are two variables: HPLEX(-AI) and NIMAX(-AI). HPLEX defines the acceptable interval range for which the solution $F(S_F)$ can fluctuate, while NIMAX specifies the maximum amount of iterations that the Nelder-Mead Method can execute. If either is modified to possess a reduced range/value, it can possibly make simplex convergence faster. However, the tool is highly efficient and such gain has shown to be irrelevant. By default, $NIMAX = 1000$ and $HPLEX = [0\ 1000]$ [km].

Lastly, two variables were created that address the calculation of collision-free trajectories, previously mentioned on Section 1.3. The latter are ISAFE(-AI) and TSAFE(-AI). The first, ISAFE, is a flag to enable the tool ($ISAFE = 1$) to verify if a converged solution V_S provides a collision-free trajectory. This check is performed through maintaining the propagation of the spacecraft's trajectory after it reaches P_T, extending it by a TSAFE amount of time. By default $TSAFE = 3$ [Days].

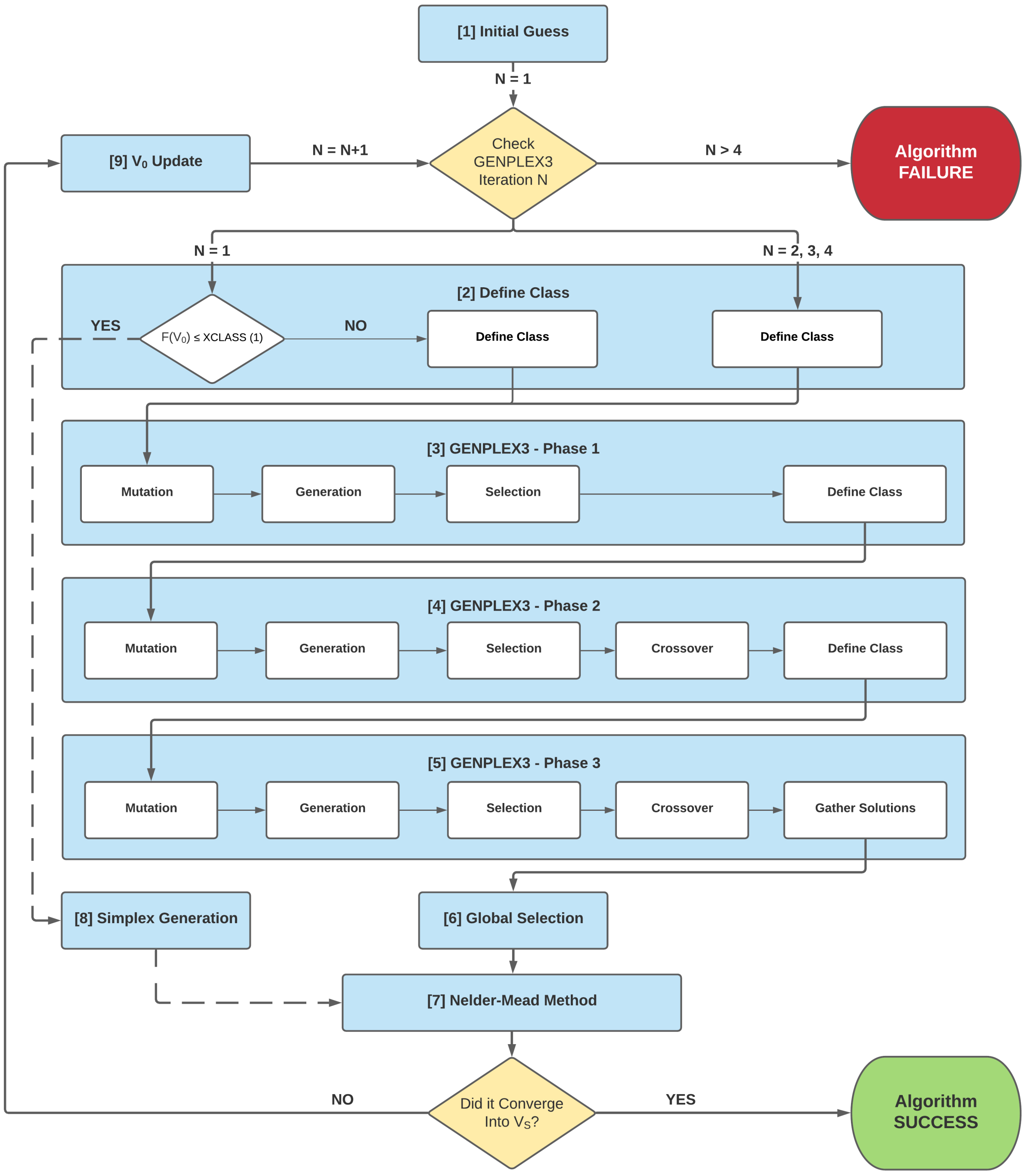


Figure 4.2: Basic overview of the TPBVP module.

4.3.1 Initial Guess

Based on the users input, this segment can take three possible paths. This section will focus on the option *Sphere of Guesses*, based on the fact that the remaining options have been addressed previously and/or are self-explanatory. This option relies on choosing the best guess from a generated sample group, being the recommended option for the user. Note that, *User Input Guess* can be given in either spherical ($ISPHI = 1$) or Cartesian ($ISPHI = 0$) coordinates (Figure 4.3), defined by the ISPHI(-AI) flag. For the generation of the sample group, a certain amount of advanced inputs are required: XGEN1(-AI), XGEN2(-AI) and XGEN3(-AI). They have the following default values in a spherical coordinate system:

$$XGEN1 = \begin{bmatrix} 0.001 & 0.02 & 20 \end{bmatrix}$$

$$XGEN2 = \begin{bmatrix} 0 & 360 & 20 \end{bmatrix}$$

$$XGEN3 = \begin{bmatrix} -90 & 90 & 20 \end{bmatrix}$$

Note: XGEN1/2/3 can be given in either either spherical ($ISPHC = 1$) or Cartesian ($ISPHC = 0$) coordinates, defined by the ISPHC(-AI) flag, same as ISPHI flag, but for advanced inputs.

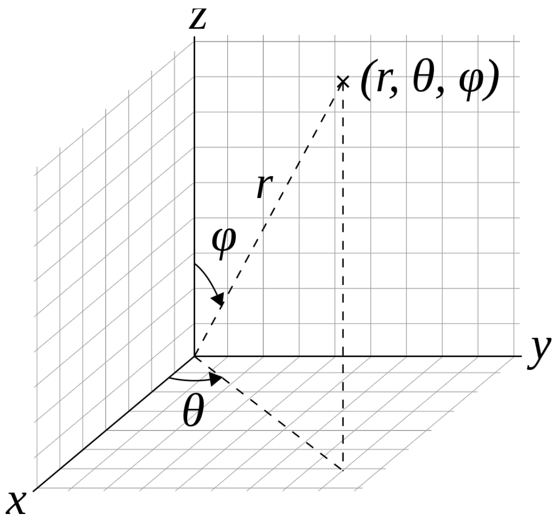


Figure 4.3: Spherical coordinate system.

Looking at XGEN1, its first and second components define the interval for which the velocity module $|V|$ in a spherical coordinate system (Figure 4.3) can vary. The third component defines the number of elements that can be evenly arranged in the previous interval, thus we have the sample [0.001:0.001:0.02] [km/s]. The next two parameters follow the same logic, XGEN2 regarding the azimuth angle θ, while XGEN3 refers to the polar angle φ. To summarize, we have then two new sample groups: one defined by XGEN2, [0:18:360] [°]; another defined by XGEN3, [-90:9:90] [°]. Finally, there are three sample groups, each with 20 elements, that when joined to allow for all possible combinations between them, generates a sphere of guesses of 8000 elements (20x20x20). From these elements, the best is selected to be the initial guess V_0. Note that going forward, when mentioning a *best* vector (V) or a matrix (M), it refers to the vector with the lower F(V), while in the case of a matrix, represents the one with lowest F(M) average.

The user can opt for a custom XGEN1/2/3 by enabling the flag ICUST(-AI), where $ICUST = 1$ requires a new set of XGEN1/2/3 vectors, whilst $ICUST = 0$ is the default and favored option. The overall flow of this segment is represented on Figure 4.4. It outputs a guess V_0 and its $F(V_0)$ into the next segment, *Define Class* (Subsection 4.3.2).

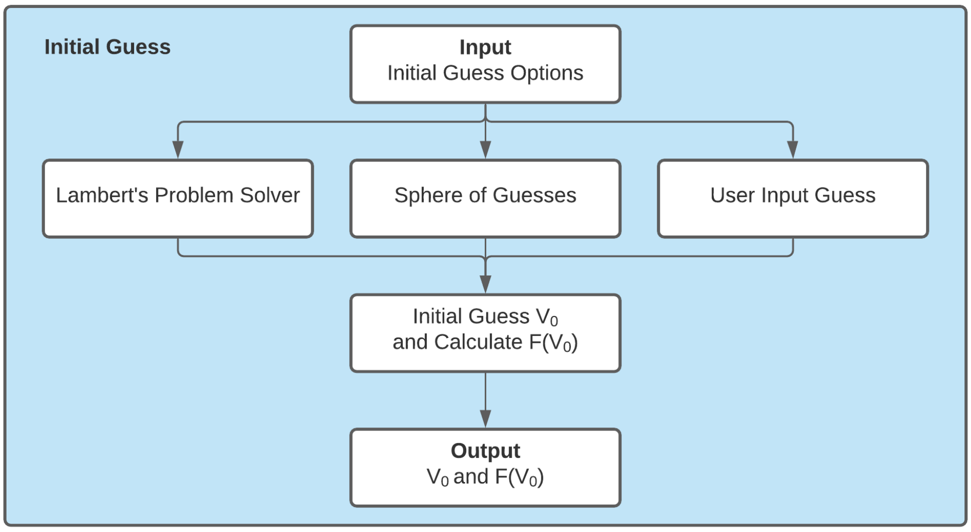


Figure 4.4: Diagram for [1] Initial Guess segment.

Going forward, when a velocity vector V or a matrix M are generated, the corresponding F(V) and F(M) are always calculated, allowing for comparison and analysis of the results. This rule is applicable for all the individuals produced during the mutation/selection/crossover operations of the genetic algorithm.

4.3.2 Define Class

This segment is mainly responsible for checking how accurate is the initial guess V_0 via comparing $F(V_0)$ against a predefined set of intervals. There are a total of five interval ranges, specified by the limits set on the variable XCLASS(-AI)(1x5). The number of classes is fixed ($NCLASS = 1:5$), but the interval limits can be changed if the user desires it. However, default values are provided that have been tested and proven to be efficient. The process of attribution of a class adopts the following logic, considering the default $XCLASS = [10\ 20\ 50\ 100\ 200]$ [km]:

$$\begin{aligned} F(V_0) \leq XCLASS(2) &\longrightarrow NCLASS = 1 \\ XCLASS(2) < F(V_0) \leq XCLASS(3) &\longrightarrow NCLASS = 2 \\ XCLASS(3) < F(V_0) \leq XCLASS(4) &\longrightarrow NCLASS = 3 \\ XCLASS(4) < F(V_0) \leq XCLASS(5) &\longrightarrow NCLASS = 4 \\ XCLASS(5) < F(V_0) &\longrightarrow NCLASS = 5 \end{aligned}$$

There is a particular situation where, on the condition of being the first iteration of the TPBVP module, if $F(V_0) \leq XCLASS(1)$, the decision is made to bypass most of the module into a later segment. This segment is the *Generate Segment* (Subsection 4.3.8). Nonetheless, this is a rare occurrence, in a more likely scenario NCLASS will be updated and output towards the next segment, *GENPLEX3 - Phase 1* (Subsection 4.3.3). The class determined here is used later on to influence the mutation operation. An overview of this segment can be found on Figure 4.5.

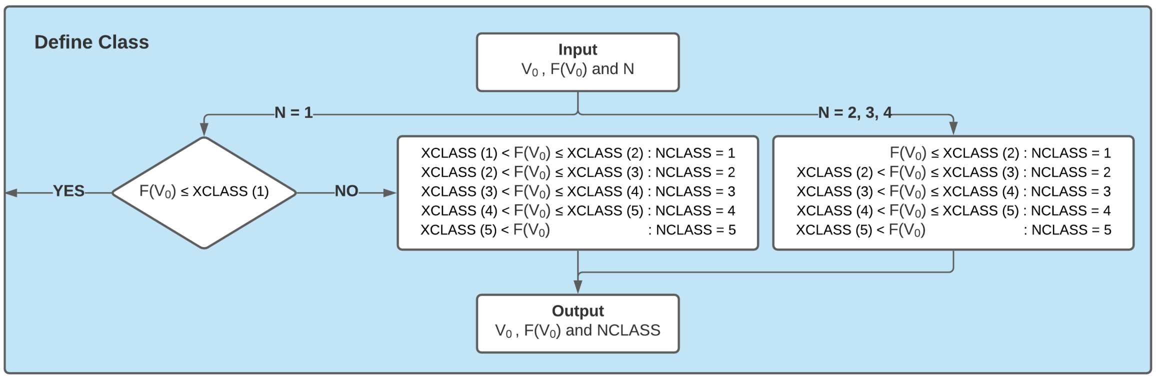

Define Class
Input
V₀ , F(V₀) and N
N = 1
N = 2, 3, 4
YES
F(V₀) ≤ XCLASS (1)
NO
XCLASS (1) < F(V₀) ≤ XCLASS (2) : NCLASS = 1
XCLASS (2) < F(V₀) ≤ XCLASS (3) : NCLASS = 2
XCLASS (3) < F(V₀) ≤ XCLASS (4) : NCLASS = 3
XCLASS (4) < F(V₀) ≤ XCLASS (5) : NCLASS = 4
XCLASS (5) < F(V₀) : NCLASS = 5
F(V₀) ≤ XCLASS (2) : NCLASS = 1
XCLASS (2) < F(V₀) ≤ XCLASS (3) : NCLASS = 2
XCLASS (3) < F(V₀) ≤ XCLASS (4) : NCLASS = 3
XCLASS (4) < F(V₀) ≤ XCLASS (5) : NCLASS = 4
XCLASS (5) < F(V₀) : NCLASS = 5
Output
V₀ , F(V₀) and NCLASS

Figure 4.5: Diagram for [2] Define Class segment.

4.3.3 GENPLEX3 - Phase 1

This segment represents the beginning of the genetic algorithm integrated in this tool. The process starts by mutating the initial guess V_0 first component, thus creating the matrix XT(12x3). The mutation of any velocity vector V follows the equation

$$V_{Mutated}(i) = V(i) \pm [\sqrt{n_{n=1:6}} \times FACTi_{i=1,2,3}(NCLASS) \times V(i)] \tag{4.1}$$

where i is the vector component, n is a constant that can assume the values in the interval [1:1:6] and lastly, $FACTi$ is a vector (1x5) that expresses a mutation factor based on the NCLASS determined in a previous segment. Equation 4.1 was developed based on extensive tests and different mutation strategies, for which the present approach was found to be the best candidate. This method, called *quadratic factorization*, showed a good degree of mutation of an individual, providing an acceptable *jump* out of a local minima area and at the same time, keeping close to the desired solution neighborhood.

In view of Equation 4.1, there is the *plus side* of the mutation and the *minus side*, each taking six different n values. Consequently, the outcome based on Equation 4.1 is a matrix XT1(12x3):

$$V_0 \xrightarrow{\text{Mutation}} XT1 = \begin{bmatrix} V_0(1) = V_0(1) + [\sqrt{6} \times FACT1(NCLASS) \times V_0(1)] & V_0(2) & V_0(3) \\ V_0(1) = V_0(1) + [\sqrt{5} \times FACT1(NCLASS) \times V_0(1)] & V_0(2) & V_0(3) \\ V_0(1) = V_0(1) + [\sqrt{4} \times FACT1(NCLASS) \times V_0(1)] & V_0(2) & V_0(3) \\ V_0(1) = V_0(1) + [\sqrt{3} \times FACT1(NCLASS) \times V_0(1)] & V_0(2) & V_0(3) \\ V_0(1) = V_0(1) + [\sqrt{2} \times FACT1(NCLASS) \times V_0(1)] & V_0(2) & V_0(3) \\ V_0(1) = V_0(1) + [\sqrt{1} \times FACT1(NCLASS) \times V_0(1)] & V_0(2) & V_0(3) \\ V_0(1) = V_0(1) - [\sqrt{1} \times FACT1(NCLASS) \times V_0(1)] & V_0(2) & V_0(3) \\ \dots & V_0(2) & V_0(3) \\ V_0(1) = V_0(1) - [\sqrt{6} \times FACT1(NCLASS) \times V_0(1)] & V_0(2) & V_0(3) \end{bmatrix} \tag{4.2}$$

Moving forward, when mentioning the *plus side* of a matrix, it refers to the rows (1:6) for a (12x3) matrix. The *minus side*, refers to the remaining (7:12) rows. Also, the variable $FACTi$ is generated for each component/phase of GENPLEX3 in the same fashion, based on a variable XFACT(-AI). However, XFACT also suffers adjustments made by another variable XPLUS(-AI), depending on each iteration N of the tool. Refer to Equation 4.3 to 4.8.

$$N = 1,2,3,4: \quad FACTi(1) = XFACT(i) \times (1 + XPLUS) \tag{4.3}$$

$$N = 2 \bigwedge i = 1: \quad FACT1(1) = XFACT(1) \times (1 + 2 \times XPLUS) \tag{4.4}$$

$$N = 3 \bigwedge i = 2: \quad FACT2(1) = XFACT(2) \times (1 + 3 \times XPLUS) \tag{4.5}$$

$$N = 4 \bigwedge i = 3: \quad FACT3(1) = XFACT(3) \times (1 + 4 \times XPLUS) \tag{4.6}$$

$$FACTi = \begin{bmatrix} FACTi(1) & FACTi(1) \times 2 & FACTi(2) \times 2 & FACTi(3) \times 2 & FACTi(4) \times 2 \end{bmatrix} \tag{4.7}$$

$$FACT1 = \begin{bmatrix} FACT1(1) & FACT1(1) \times 2 & FACT1(2) \times 2 & FACT1(3) \times 2 & FACT1(4) \times 2 \end{bmatrix} \tag{4.8}$$

The variable XFACT(1x3) can be altered by the user, although not recommended. The default values are $XFACT = [0.025\ 0.5\ 0.25]$ and $XPLUS = 0.25$. The previous values are given in spherical coordinates (Figure 4.3), enabled by the flag ISPHC ($ISPHC = 1$ for spherical coordinates). In short, XFACT(1) refers to either the V_X component or $|V|$; XFACT(2) refers to the V_Y component or the azimuth angle θ; and XFACT(3) refers to the component V_Z or the polar angle φ.

Next, after the mutation operation, the selection process begins. The matrix XT is divided into the plus and minus sides and the worst individual (individual with the highest F(.)) from each side is removed, generating two XA (5X3) matrices. From the five individuals, the four that present the best standard deviation of F(.) are selected into a (4x3) matrix. Finally, the two (4x3) matrices are joined, generating the final generation V_1. Based on the best individual of V_1, a new NCLASS is defined, as per the process shown on Subsection 4.3.2. Figure 4.6 shows the mutation and selection steps, whereas Figure 4.7 provides an overview of the entire segment. Figure 4.7 can be found on a more eligible size on Appendix A Figure A.1.

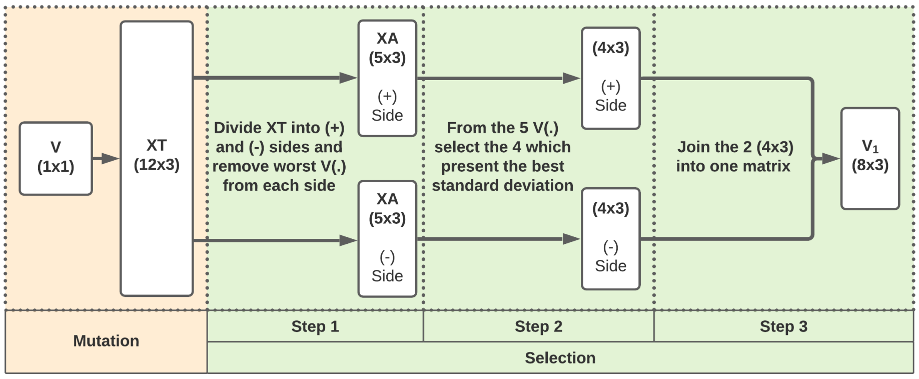
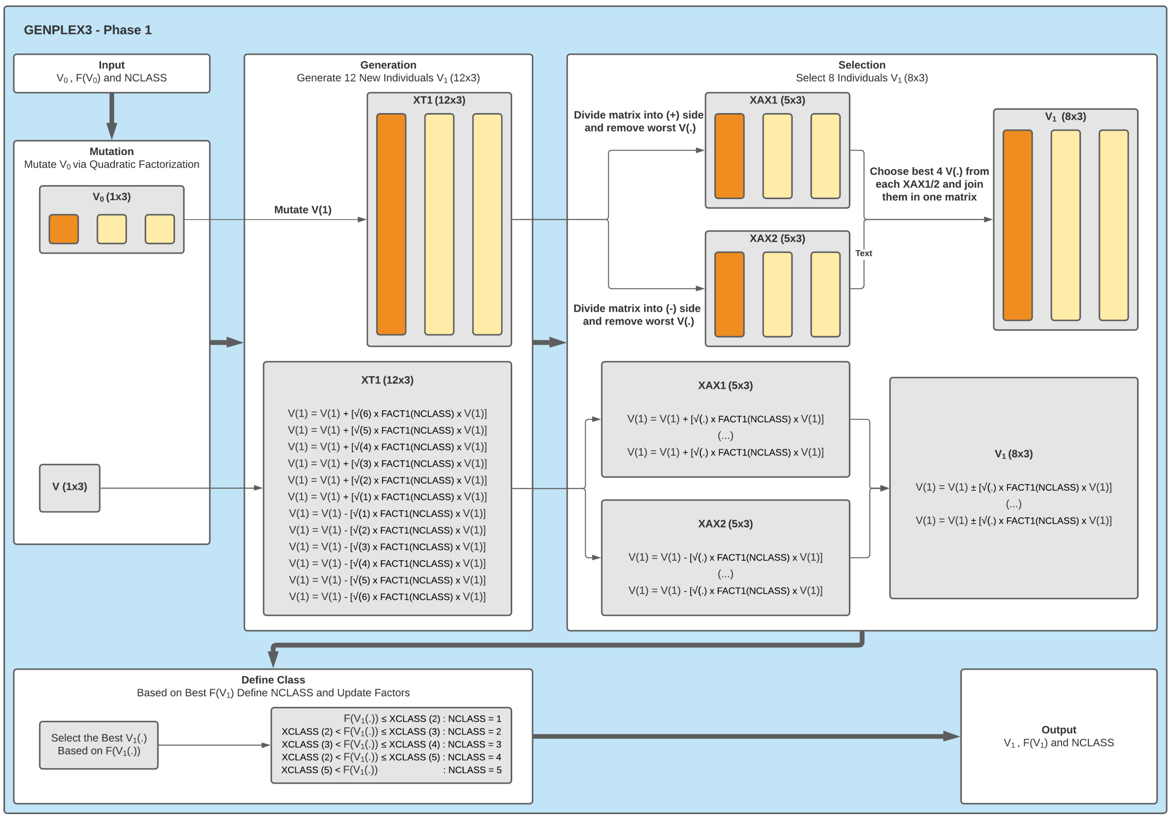

V (1x1)
XT (12x3)
XA (5x3) (+) Side
XA (5x3) (-) Side
Divide XT into (+) and (-) sides and remove worst V(.) from each side
From the 5 V(.) select the 4 which present the best standard deviation
(4x3) (+) Side
(4x3) (-) Side
Join the 2 (4x3) into one matrix
V₁ (8x3)
Mutation
Step 1
Step 2
Step 3
Selection

Figure 4.6: Diagram for mutation and selection steps of [3] GENPLEX3 - Phase 1 segment.



GENPLEX3 - Phase 1
Input
V₀ , F(V₀) and NCLASS
Mutation
Mutate V₀ via Quadratic Factorization
V₀ (1x3)
V (1x3)
Generation
Generate 12 New Individuals V₁ (12x3)
XT1 (12x3)
Mutate V(1)
XT1 (12x3)
V(1) = V(1) + [√(6) x FACT1(NCLASS) x V(1)]
V(1) = V(1) + [√(5) x FACT1(NCLASS) x V(1)]
V(1) = V(1) + [√(4) x FACT1(NCLASS) x V(1)]
V(1) = V(1) + [√(3) x FACT1(NCLASS) x V(1)]
V(1) = V(1) + [√(2) x FACT1(NCLASS) x V(1)]
V(1) = V(1) + [√(1) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(1) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(2) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(3) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(4) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(5) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(6) x FACT1(NCLASS) x V(1)]
Selection
Select 8 Individuals V₁ (8x3)
Divide matrix into (+) side and remove worst V(.)
XAX1 (5x3)
XAX2 (5x3)
Divide matrix into (-) side and remove worst V(.)
Choose best 4 V(.) from each XAX1/2 and join them in one matrix
Text
V₁ (8x3)
XAX1 (5x3)
V(1) = V(1) + [√(.) x FACT1(NCLASS) x V(1)]
(...)
V(1) = V(1) + [√(.) x FACT1(NCLASS) x V(1)]
XAX2 (5x3)
V(1) = V(1) - [√(.) x FACT1(NCLASS) x V(1)]
(...)
V(1) = V(1) - [√(.) x FACT1(NCLASS) x V(1)]
V₁ (8x3)
V(1) = V(1) ± [√(.) x FACT1(NCLASS) x V(1)]
(...)
V(1) = V(1) ± [√(.) x FACT1(NCLASS) x V(1)]
Define Class
Based on Best F(V₁) Define NCLASS and Update Factors
Select the Best V₁(.) Based on F(V₁(.))
F(V₁(.)) ≤ XCLASS (2) : NCLASS = 1
XCLASS (2) < F(V₁(.)) ≤ XCLASS (3) : NCLASS = 2
XCLASS (3) < F(V₁(.)) ≤ XCLASS (4) : NCLASS = 3
XCLASS (2) < F(V₁(.)) ≤ XCLASS (5) : NCLASS = 4
XCLASS (5) < F(V₁(.)) : NCLASS = 5
Output
V₁ , F(V₁) and NCLASS

Figure 4.7: Diagram for [3] GENPLEX3 - Phase 1 segment.

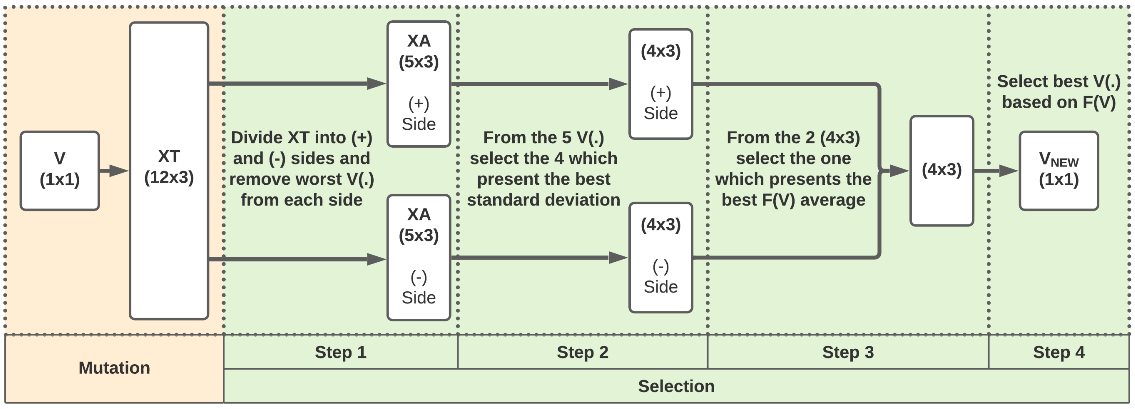
Take note that the crossover operation is not performed in this segment, considering the limited genetic diversity. In other words, the generation created in this segment is too small to present a meaningful gain by suffering a crossover of its properties.

4.3.4 GENPLEX3 - Phase 2

Phase 2 shares multiple similarities with Phase 1. In summary, it is Phase 1 applied to the second component of each individual multiplied by eight. The same process that V_0 underwent in the previous sections is followed for each of V_1's eight individuals, besides two modifications: modified selection and addition of crossover.

Regarding the selection process, for from Step 3 of Figure 4.6, this phase's process slightly changes. Rather than joining both (4x3) matrices, it instead chooses the best one (best F(V) average), after which it selects the best individual from the latter. The new flow of the selection operation can be observed in Figure 4.8. The mutation of the second component still follows Equation 4.1, now for each of of V_1's individuals:

$$V_{Mutated}(2) = V_1(2) \pm [\sqrt{n_{n=1:6}} \times FACT2(NCLASS) \times V_1(2)] \tag{4.9}$$



V (1x1)
XT (12x3)
Divide XT into (+) and (-) sides and remove worst V(.) from each side
XA (5x3) (+) Side
XA (5x3) (-) Side
From the 5 V(.) select the 4 which present the best standard deviation
(4x3) (+) Side
(4x3) (-) Side
From the 2 (4x3) select the one which presents the best F(V) average
(4x3)
Select best V(.) based on F(V)
VNEW (1x1)
Mutation
Step 1
Step 2
Step 3
Step 4
Selection

Figure 4.8: Diagram for [4] GENPLEX3 - Phase 2 segment.

In this segment, crossover is first introduced. Taking the result of mutation and selection, a matrix (8x3), this operation follows a simple rule to double the amount of individuals: it takes an individual's mutated component and transfers it to the next row's individual (4.10). Take into consideration that all the remaining elements of the (8x3) matrix are different, consequently all the new individuals will be unique. Applying this crossover operation to this phase's mutated component, produces the final (16x3) matrix as the second generation V_2.

$$(8x3) \xrightarrow{\text{Crossover}} (16x3) = \begin{bmatrix} \cdot & V(1,2) & \cdot \\ \vdots & \vdots & \vdots \\ \cdot & V(8,2) & \cdot \\ \cdot & V(9,2) = V(8,2) & \cdot \\ \cdot & V(10,2) = V(1,2) & \cdot \\ \cdot & V(11,2) = V(2,2) & \cdot \\ \vdots & \vdots & \vdots \\ \cdot & V(15,2) = V(6,2) & \cdot \\ \cdot & V(16,2) = V(7,2) & \cdot \end{bmatrix} \tag{4.10}$$

Finally, a new NCLASS is established and we can continue to Phase 3, Subsection 4.3.5. Figure 4.9 provides an overview of the entire segment. Figure 4.9 can be found on a more eligible size on Appendix A Figure A.2.

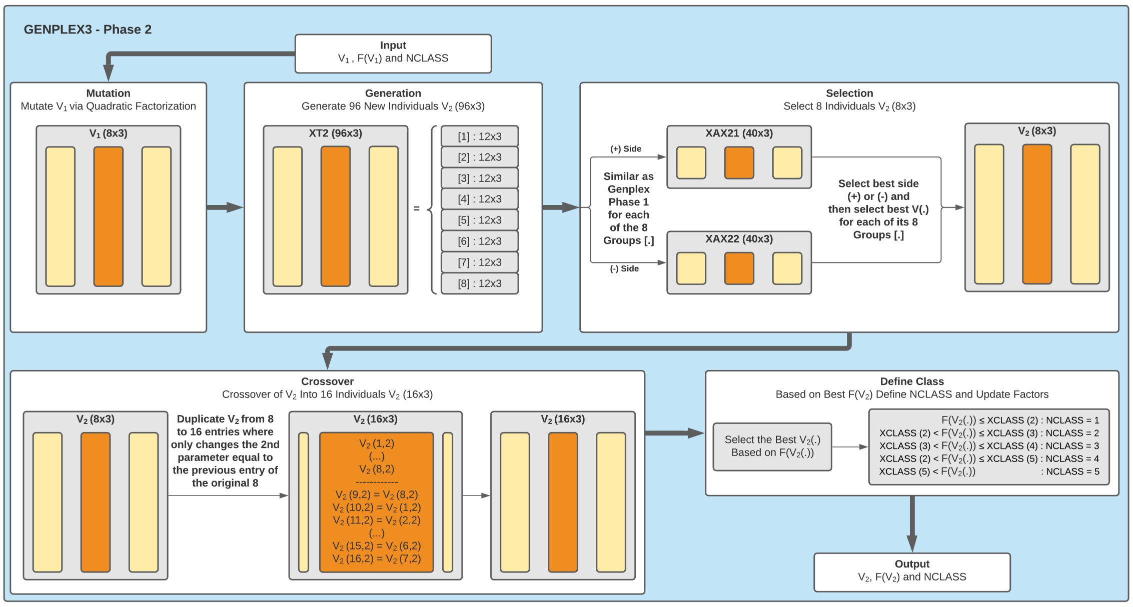

GENPLEX3 - Phase 2
Input
V_1, $F(V_1)$ and NCLASS
Mutation
Mutate V_1 via Quadratic Factorization
V_1 (8x3)
Generation
Generate 96 New Individuals V_2 (96x3)
XT2 (96x3)
[1] : 12x3
[2] : 12x3
[3] : 12x3
[4] : 12x3
[5] : 12x3
[6] : 12x3
[7] : 12x3
[8] : 12x3
Selection
Select 8 Individuals V_2 (8x3)
(+) Side
(-) Side
Similar as Genplex Phase 1 for each of the 8 Groups [.]
XAX21 (40x3)
XAX22 (40x3)
Select best side (+) or (-) and then select best V(.) for each of its 8 Groups [.]
V_2 (8x3)
Crossover
Crossover of V_2 Into 16 Individuals V_2 (16x3)
V_2 (8x3)
Duplicate V_2 from 8 to 16 entries where only changes the 2nd parameter equal to the previous entry of the original 8
V_2 (16x3)
V_2 (1,2)
(...)
V_2 (8,2)
V_2 (9,2) = V_2 (8,2)
V_2 (10,2) = V_2 (1,2)
V_2 (11,2) = V_2 (2,2)
(...)
V_2 (15,2) = V_2 (6,2)
V_2 (16,2) = V_2 (7,2)
V_2 (16x3)
Define Class
Based on Best $F(V_2)$ Define NCLASS and Update Factors
Select the Best $V_2(.)$ Based on $F(V_2(.))$
$F(V_2(.)) \leq$ XCLASS (2) : NCLASS = 1
XCLASS (2) < $F(V_2(.)) \leq$ XCLASS (3) : NCLASS = 2
XCLASS (3) < $F(V_2(.)) \leq$ XCLASS (4) : NCLASS = 3
XCLASS (2) < $F(V_2(.)) \leq$ XCLASS (5) : NCLASS = 4
XCLASS (5) < $F(V_2(.))$: NCLASS = 5
Output
V_2, $F(V_2)$ and NCLASS

Figure 4.9: Diagram for [4] GENPLEX3 - Phase 2 segment.

4.3.5 GENPLEX3 - Phase 3

This phase shares all the same operations as Phase 2, now applied to sixteen individuals. Mutation (Equation 4.11) now produces a (192x3) generation, selection defines a (16x3) matrix which crossover turns into a (32x3). Finally, all the products of each phase are joined into a final V_T(56x3) matrix. Figure 4.10 provides an overview of the entire segment (Appendix A Figure A.3).

$$V_{Mutated}(3) = V_2(3) \pm [\sqrt{n_{n=1:6}} \times FACT3(NCLASS) \times V_2(3)] \tag{4.11}$$

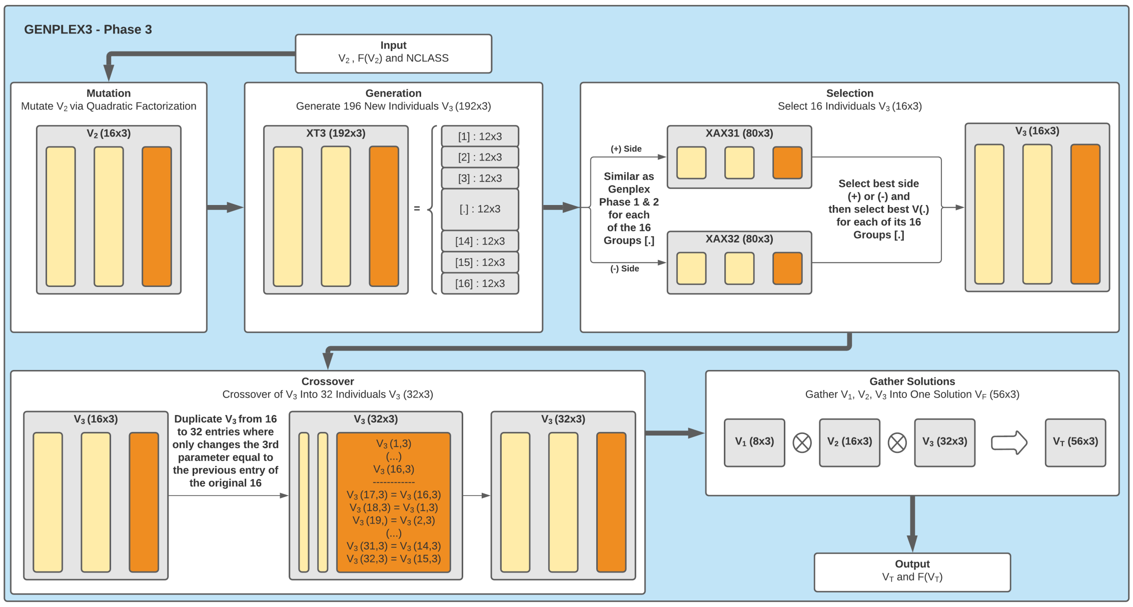

GENPLEX3 - Phase 3
Input
V_2, $F(V_2)$ and NCLASS
Mutation
Mutate V_2 via Quadratic Factorization
V_2 (16x3)
Generation
Generate 196 New Individuals V_3 (192x3)
XT3 (192x3)
[1] : 12x3
[2] : 12x3
[3] : 12x3
[.] : 12x3
[14] : 12x3
[15] : 12x3
[16] : 12x3
Selection
Select 16 Individuals V_3 (16x3)
(+) Side
(-) Side
Similar as Genplex Phase 1 & 2 for each of the 16 Groups [.]
XAX31 (80x3)
XAX32 (80x3)
Select best side (+) or (-) and then select best V(.) for each of its 16 Groups [.]
V_3 (16x3)
Crossover
Crossover of V_3 Into 32 Individuals V_3 (32x3)
V_3 (16x3)
Duplicate V_3 from 16 to 32 entries where only changes the 3rd parameter equal to the previous entry of the original 16
V_3 (32x3)
V_3 (1,3)
(...)
V_3 (16,3)
V_3 (17,3) = V_3 (16,3)
V_3 (18,3) = V_3 (1,3)
V_3 (19,) = V_3 (2,3)
(...)
V_3 (31,3) = V_3 (14,3)
V_3 (32,3) = V_3 (15,3)
V_3 (32x3)
Gather Solutions
Gather V_1, V_2, V_3 Into One Solution V_F (56x3)
V_1 (8x3)
V_2 (16x3)
V_3 (32x3)
V_T (56x3)
Output
V_T and $F(V_T)$

Figure 4.10: Diagram for [5] GENPLEX3 - Phase 3 segment.

4.3.6 Global Selection

In this segment, the four best individuals of V_T are selected, which generates the final simplex S_F that can be submitted to the Nelder-Mead Method.

4.3.7 Nelder-Mead Method

In the present segment, the process takes a simplex S_F and tries to converge into a solution V_S by the use of the Nelder-Mead Method, described in detail in Subsection 3.3.1. If convergence succeeds, then the algorithm has finished its task, output V_S, otherwise it means that it converged into a local minima V_F.

4.3.8 Simplex Generation

This segment is responsible for generating a simplex based on the developed GENPLEX2 function. This version alters a vector V by a fixed interval (-20:2:20)[%] on each component, hence is a linear variation of said vector. From this step, a (20x3) matrix is created, with twenty new candidates, from which the best four are selected to create a simplex S_F. Figure 4.11 shows an overview of the described process.

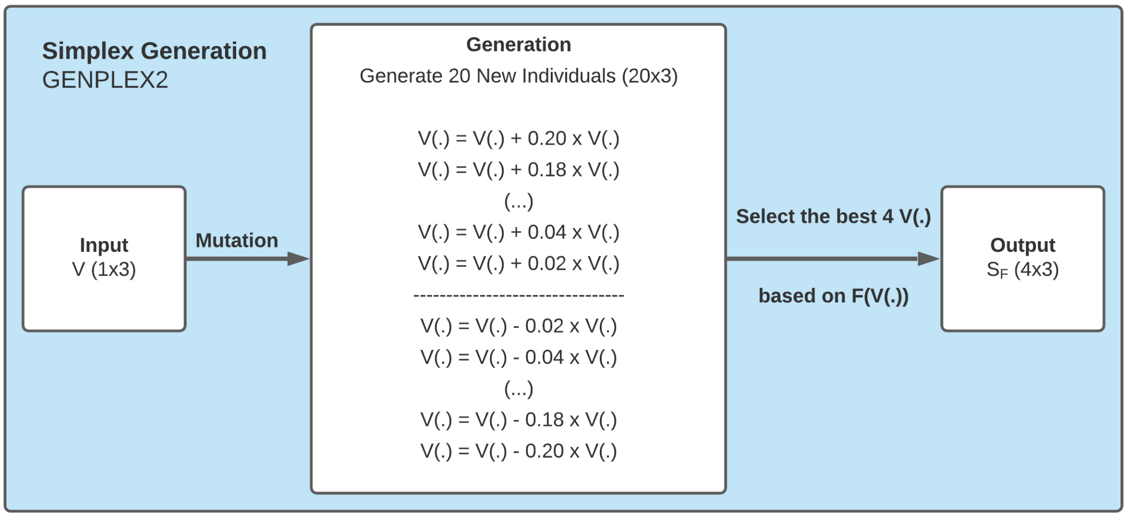

Simplex Generation
GENPLEX2
Input
V (1x3)
Mutation
Generation
Generate 20 New Individuals (20x3)
V(.) = V(.) + 0.20 x V(.)
V(.) = V(.) + 0.18 x V(.)
(...)
V(.) = V(.) + 0.04 x V(.)
V(.) = V(.) + 0.02 x V(.)
V(.) = V(.) - 0.02 x V(.)
V(.) = V(.) - 0.04 x V(.)
(...)
V(.) = V(.) - 0.18 x V(.)
V(.) = V(.) - 0.20 x V(.)
Select the best 4 V(.)
based on F(V(.))
Output
SF (4x3)

Figure 4.11: Diagram for [8] Simplex Generation / GENPLEX2 segment.

4.3.9 V_0 Update

This segment only applies in the event that the Nelder-Mead Method fails to converge into a solution V_S, instead getting *stuck* in a local minima V_F. In that event, then a new iteration of the tool is required, starting by updating the initial guess of the new iteration with the local minima V_F found.

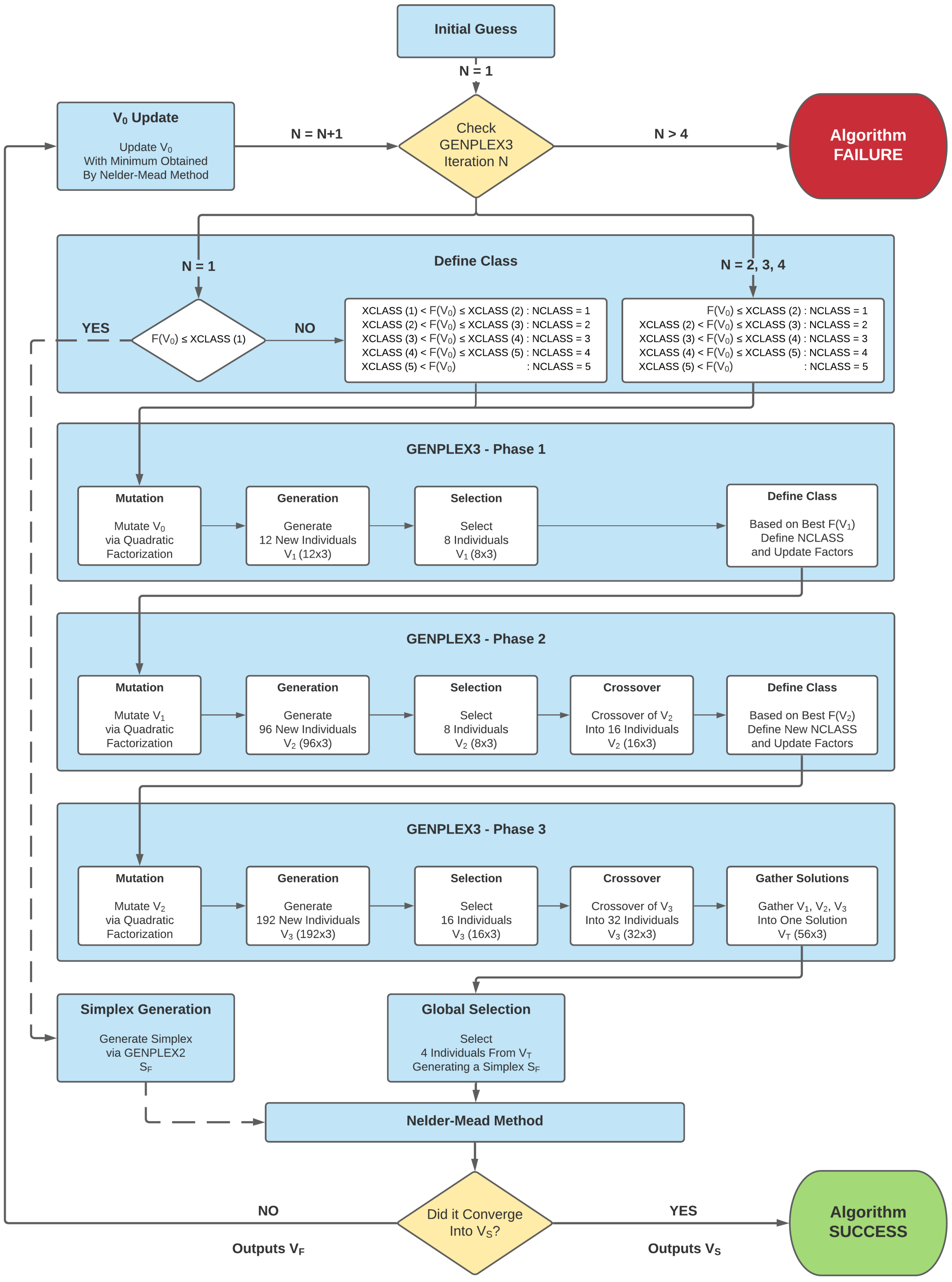
4.4 Module Advanced Overview

This section is just a brief summary of the information established in the previous sections and subsections. In Table 4.2 are listed all the advanced input (AI) variables that the user can alter, although not recommended. Figure 4.12 has a more detailed overview of the TPBVP module when compared to Figure 4.2.

Table 4.2: Advanced input options overview.

AI Option	Short Description	Type	Detailed On
ISPHI	Flag to enable either spherical or Cartesian options.	Binary	Section 4.2
OFACT	Factor ΔS to generate a simplex.	Float	Section 4.2
NRSTR	Maximum number of iterations for TPBVP algorithm.	Integer	Section 4.3
IMUTA	Order by which vector components are mutated.	Integer	Section 4.3
HPLEX	Interval for which $F(S_F)$ may vary.	Vector	Section 4.3
NIMAX	Maximum number of iterations for Nelder-Mead Method.	Integer	Section 4.3
ISAFE	Flag to enable check for a collision-free trajectory.	Binary	Section 4.3
TSAFE	Time interval accepted for a collision-free trajectory.	Float	Section 4.3
ISPHC	Same as ISPHI, but for advanced inputs.	Binary	Section 4.2
ICUST	Flag to enable custom sphere of guesses.	Binary	Subsection 4.3.1
XGEN1 XGEN2 XGEN3	Interval range for $V_X/\lvert V\rvert$ of an initial guess. Interval range for V_Y/θ of an initial guess. Interval range for V_Z/φ of an initial guess.	Vector	Subsection 4.3.1
XCLASS	Defines the interval range for each class.	Vector	Subsection 4.3.2
XPLUS	Factor to adjust the mutation factor XFACT.	Float	Subsection 4.3.3
XFACT	Defines the mutation factor for each velocity component.	Vector	Subsection 4.3.3

As a final note, must point out that the created hybrid algorithm was designed by aiming for a greater range of genetic diversity. The ever changing mutation factors, the classes that determine how *aggressive* the next mutation will be, the crossover based only in interchanging elements, all these different approaches are intended to generate new individuals in very diverse ways. A myriad of combinations was tested and by analyzing each generation for different scenarios, the current process was found to be competent in finding a solution V_S when other known methods failed.



Initial Guess
N = 1
Check GENPLEX3 Iteration N
N > 4
Algorithm FAILURE
V₀ Update
Update V₀ With Minimum Obtained By Nelder-Mead Method
N = N+1
Define Class
N = 1
N = 2, 3, 4
YES
F(V₀) ≤ XCLASS (1)
NO
XCLASS (1) < F(V₀) ≤ XCLASS (2) : NCLASS = 1
XCLASS (2) < F(V₀) ≤ XCLASS (3) : NCLASS = 2
XCLASS (3) < F(V₀) ≤ XCLASS (4) : NCLASS = 3
XCLASS (4) < F(V₀) ≤ XCLASS (5) : NCLASS = 4
XCLASS (5) < F(V₀) : NCLASS = 5
F(V₀) ≤ XCLASS (2) : NCLASS = 1
XCLASS (2) < F(V₀) ≤ XCLASS (3) : NCLASS = 2
XCLASS (3) < F(V₀) ≤ XCLASS (4) : NCLASS = 3
XCLASS (4) < F(V₀) ≤ XCLASS (5) : NCLASS = 4
XCLASS (5) < F(V₀) : NCLASS = 5
GENPLEX3 - Phase 1
Mutation: Mutate V₀ via Quadratic Factorization
Generation: Generate 12 New Individuals V₁ (12x3)
Selection: Select 8 Individuals V₁ (8x3)
Define Class: Based on Best F(V₁) Define NCLASS and Update Factors
GENPLEX3 - Phase 2
Mutation: Mutate V₁ via Quadratic Factorization
Generation: Generate 96 New Individuals V₂ (96x3)
Selection: Select 8 Individuals V₂ (8x3)
Crossover: Crossover of V₂ Into 16 Individuals V₂ (16x3)
Define Class: Based on Best F(V₂) Define New NCLASS and Update Factors
GENPLEX3 - Phase 3
Mutation: Mutate V₂ via Quadratic Factorization
Generation: Generate 192 New Individuals V₃ (192x3)
Selection: Select 16 Individuals V₃ (16x3)
Crossover: Crossover of V₃ Into 32 Individuals V₃ (32x3)
Gather Solutions: Gather V₁, V₂, V₃ Into One Solution V_T (56x3)
Simplex Generation: Generate Simplex via GENPLEX2 S_F
Global Selection: Select 4 Individuals From V_T Generating a Simplex S_F
Nelder-Mead Method
Did it Converge Into V_S?
NO
Outputs V_F
YES
Outputs V_S
Algorithm SUCCESS

Figure 4.12: Advanced overview of the TPBVP module.

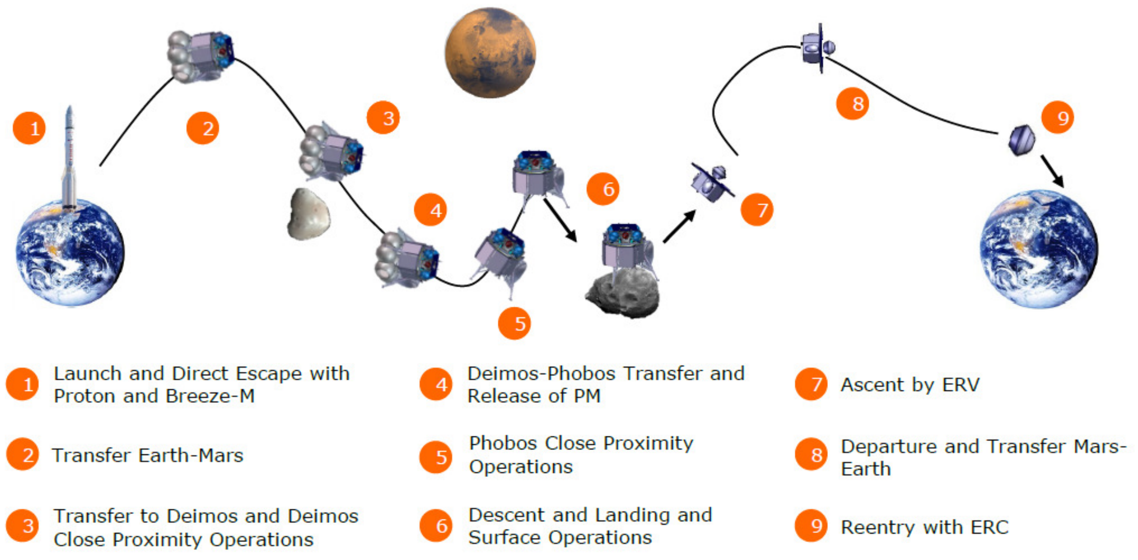
Chapter 5

Phobos-SR Mission

5.1 Mission Overview

The joint ESA-Roscosmos Phobos Sample Return mission (Phobos-SR) was a candidate of the Mars Robotic Exploration Preparation (MREP) Programme [16]. Its main objective is to acquire and return a sample from the Mars' moon Phobos, after a scientific characterization phase of the moon and of the landing site. On Figure 5.1 we have the mission overview, from which this chapter will focus on phase (5) *Phobos Close Proximity Operations*. Specifically, on the approach trajectories from an established orbit to the gate position, where the spacecraft would start phase (6) *Descent and Landing and Surface Operations*.

The Phobos-SR mission is the case study for the developed TPBVP tool. In the following sections, the hybrid algorithm is used to calculate velocities that can achieve a number of distinct trajectories, under the assumptions set by GMV for the purpose of this work. All other subjects regarding each phase or details specific to the mission are outside the scope of this chapter and thesis.


1
2
3
4
5
6
7
8
9
1 Launch and Direct Escape with Proton and Breeze-M
2 Transfer Earth-Mars
3 Transfer to Deimos and Deimos Close Proximity Operations
4 Deimos-Phobos Transfer and Release of PM
5 Phobos Close Proximity Operations
6 Descent and Landing and Surface Operations
7 Ascent by ERV
8 Departure and Transfer Mars-Earth
9 Reentry with ERC
Figure 5.1: Phobos-SR mission overview.

5.2 Assumptions

The following assumptions regarding the orbital dynamics and its complexities were established by ESA and GMV for the purpose of the FASTMOPS project [2]. The requirements referring to collision-free trajectories were specifically set for this thesis by GMV alone. The next subsections will address the specified assumptions on a matter-of-fact basis, since they were set independently from this work and are just meant to provide the reader with some background information to the case study at hand.

This case study's subject is a boundary value problem under a three-body problem scenario. The bodies would be a spacecraft, Mars (primary body) and its moon Phobos (secondary body), under a Synodic reference frame. Figure 5.2 represents an example of said reference system, while also displaying a QSO orbit [11]. The approach trajectory to be calculated is departing from a QSO orbit in this mission scenario, although such QSO orbit is not defined or taken into account by the tool/algorithm, considering it has no influence on the TPBVP solver.

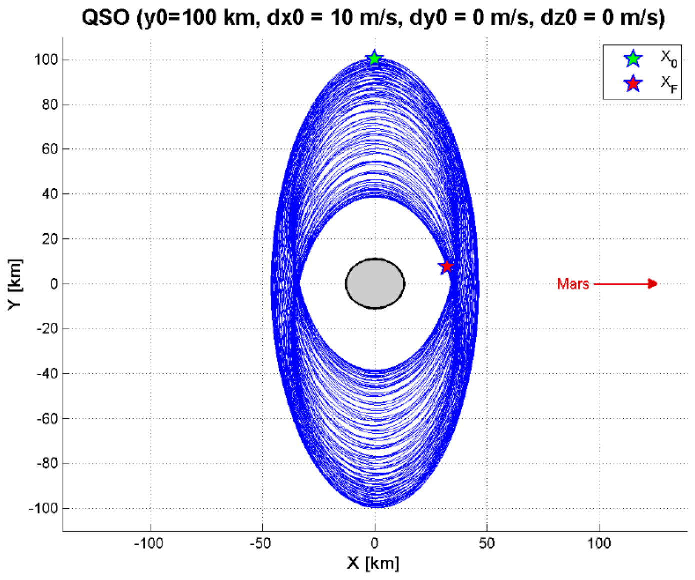


Figure 5.2: Synodic coordinate system observed on a QSO scenario [16].

The approach trajectory is assumed to suffer perturbations from solar radiation, multiple third bodies and a non-spherical gravity from the primary and secondary bodies. Constraints are also defined for the mission scenario: specifically the minimum and maximum altitudes from the secondary body, to prevent crashing into the latter or departing the area of operations; a position error tolerance, to determine if the spacecraft arrived at the target location; finally, the amount of time the spacecraft can be left under an uncorrected course, while not crashing into Phobos, qualifying it as a collision-free trajectory. On Table 5.1 are listed all the assumptions and considerations taken for the scenarios that will be tested in this chapter.

Table 5.1: Assumptions overview for the Phobos-SR Mission.

Assumptions for Phobos-SR Mission	
Problem Statement	**Input**
Type of Problem	Boundary Value Problem
Primary Body	Mars
Secondary Body	Phobos
Coordinate System	Synodic Reference Frame
Initial Epoch [MJD 2000]	2025: 9:21 13:06:31:7 (9395.5462)
Initial Mass [kg]	1500
Perturbations Considered	
Atmospheric Drag	No
Solar Radiation	Yes
Reflective Coefficient:	1.5
Cross Sectional Area [m^2]:	5.0
Third Body	Yes
Third Bodies:	Deimos; Mars; Sun; Earth; Jupiter; Saturn
Non-Spherical Gravity	Yes
Number of Zonals (Secondary):	2
Number of Zonals (Primary):	10
Number of Tesserals (Secondary):	2
Number of Tesserals (Primary):	10
Secondary Ellipsoid Semi-Axes [km]	[13.05 11.10 9.30]
Tolerances	
Maximum Altitude to Secondary [km]	1000
Minimum Altitude to Secondary [km]	5
Acceptable Position Error [km]	0.000001
Acceptable Duration for Collision-Free status [Day(s)]	3

5.3 Results

The scenarios presented in this section are representative of the possible range of approach trajectories that can be determined via the developed TPBVP tool. These all share the same assumptions presented on Table 5.1, diverging only on initial position, target position and trajectory duration. Note that in the following scenario inputs, when initial guess is marked as (U), it represents *User Input*, (S) means *Sphere of Guesses* and (L) means *Lambert's Problem*. Also, all solutions provided were obtained under default AI values. The plots presented in the following sections use X_0 as P_0 and X_F as P_T, as well the figure's unchanged title of *"... QSO ..."*, these are a limitation from the original SBNav.

Take into consideration that GMV's primary request was regarding the type of scenarios obtained/tested, for these should focus on diversity among the trajectories. Secondary to the requirement of applying the tool to different approach paths, was doing so while favoring the adoption of collision-free trajectories. In other words, the main focus was finding diverse scenarios to apply the developed tool, with the welcome addition of presenting themselves as collision-free. No other requisite was established besides the diversity of results. Hence for example, there is no investigation into trajectories with lower velocity to reach a certain gate while maintaining a collision-free characteristic.

The mission scenarios (initial and target locations) are also defined according to GMV's standards, which are based upon the original test cases used for FASTMOPS [16]. The results presented are based on departure from a stable orbit further away from Phobos (75 $\sim$ 100 km) to a gate position closer to Phobos (20 $\sim$ 50 km) where a spacecraft would start descent procedures. The ideal test scenario is departure from a position 75 km from the center of Phobos in the negative Y-axis of the defined Synodic reference frame, to an altitude of $\sim$20 km from Phobos center. The target location at $\sim$20 km of Phobos can be either in positive or negative side of X or Y axis, thus different cases will be presented. The value of 50 km for target position is used for initial test cases where trajectories are less susceptible to Phobos irregular gravity, hence being easier to estimate and stabler.

The following subsections are arranged by the level of complexity required to obtain/determine them. From Subsection 5.3.1 to 5.3.3, we have trajectories further away from Phobos, which allows for an acceptable initial guess via Lambert's Problem. Next, the majority of the results are obtained by user input, from experience with the mission scenario and its environment. This heuristic was necessary to obtain the more exotic type of trajectories.

5.3.1 Scenario 1

Table 5.2: Scenario 1 summary.

Input	
Initial Position P_0 [km]	(0, -100.0, 0)
Target Position P_T [km]	(-50.0, 0, 0)
Trajectory Duration Δt [Day(s)]	2 / 24
Initial Guess V_0 [km/s]	(L)
Collision-Free Trajectory	Yes

Output	
Initial Velocity V_X [km/s]	-0.01080271
Initial Velocity V_Y [km/s]	-0.00108816
Initial Velocity V_Z [km/s]	-0.00000540
Time to Convergence [s]	24
Simplex Iterations	186

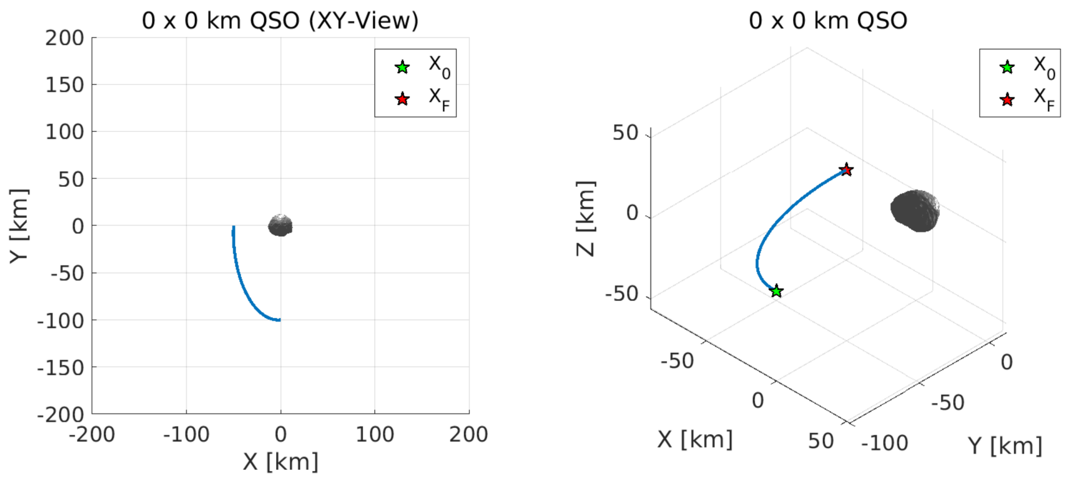


Figure 5.3: Scenario 1 XY-plane and 3D view.

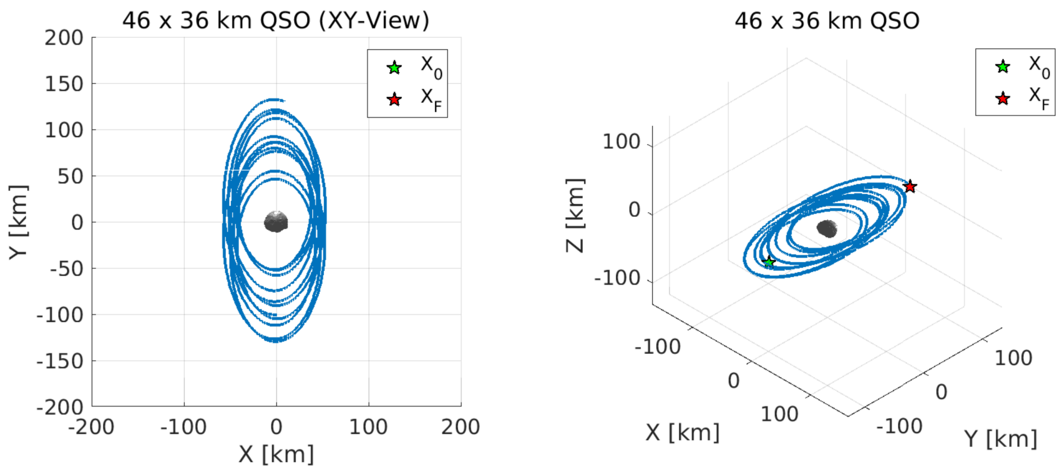


Figure 5.4: Scenario 1 XY-plane and 3D view, three days after reaching P_T.

5.3.2 Scenario 2

Table 5.3: Scenario 2 summary.

Input	
Initial Position P_0 [km]	(0, -100.0, 0)
Target Position P_T [km]	(0, 50.0, 0)
Trajectory Duration Δt [Day(s)]	4 / 24
Initial Guess V_0 [km/s]	(L)
Collision-Free Trajectory	No

Output	
Initial Velocity V_X [km/s]	-0.00776879
Initial Velocity V_Y [km/s]	-0.00151588
Initial Velocity V_Z [km/s]	0.00012144
Time to Convergence [s]	50
Simplex Iterations	271

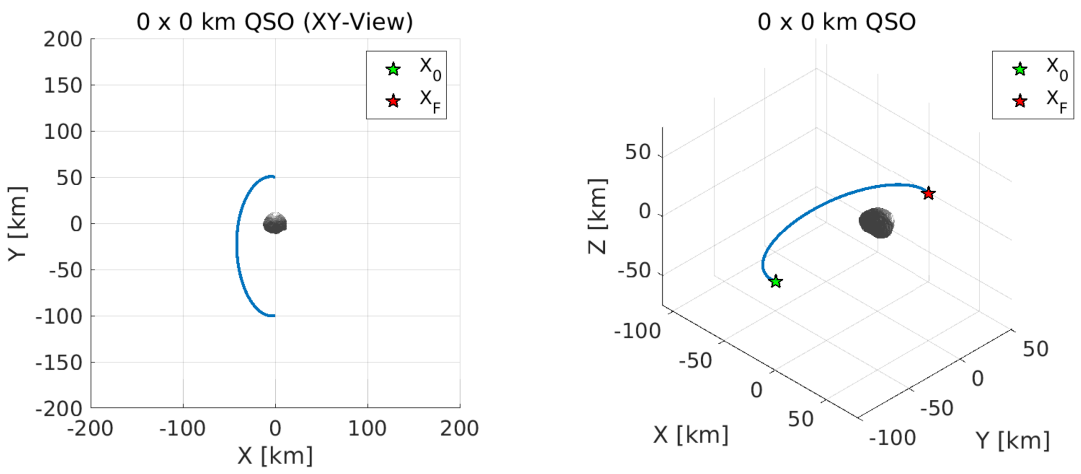


Figure 5.5: Scenario 2 XY-plane and 3D view.

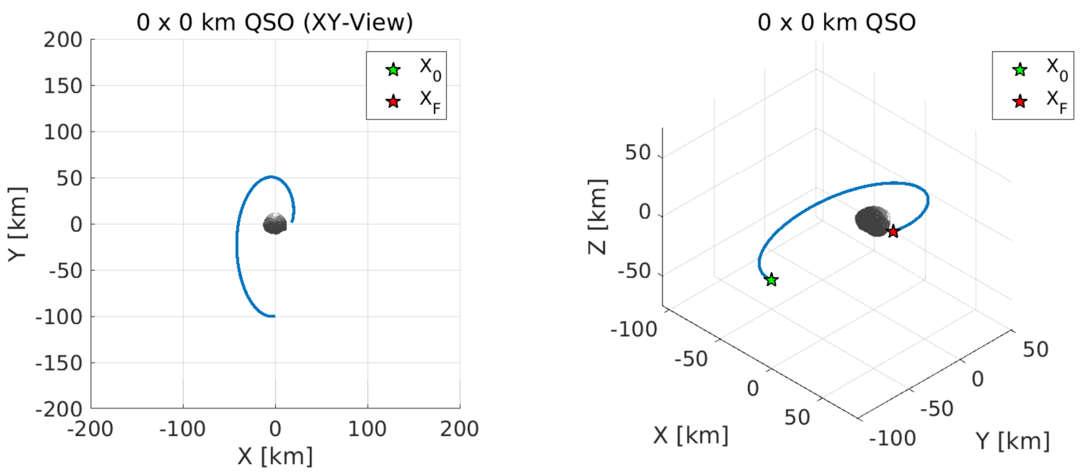


Figure 5.6: Scenario 2 XY-plane and 3D view, three days after reaching P_T.

5.3.3 Scenario 3

Table 5.4: Scenario 3 summary.

Input	
Initial Position P_0 [km]	(0, -100.0, 0)
Target Position P_T [km]	(50.0, 0, 0)
Trajectory Duration Δt [Day(s)]	6 / 24
Initial Guess V_0 [km/s]	(L)
Collision-Free Trajectory	Yes

Output	
Initial Velocity V_X [km/s]	-0.01376686
Initial Velocity V_Y [km/s]	-0.00093636
Initial Velocity V_Z [km/s]	0.00003518
Time to Convergence [s]	287
Simplex Iterations	1442

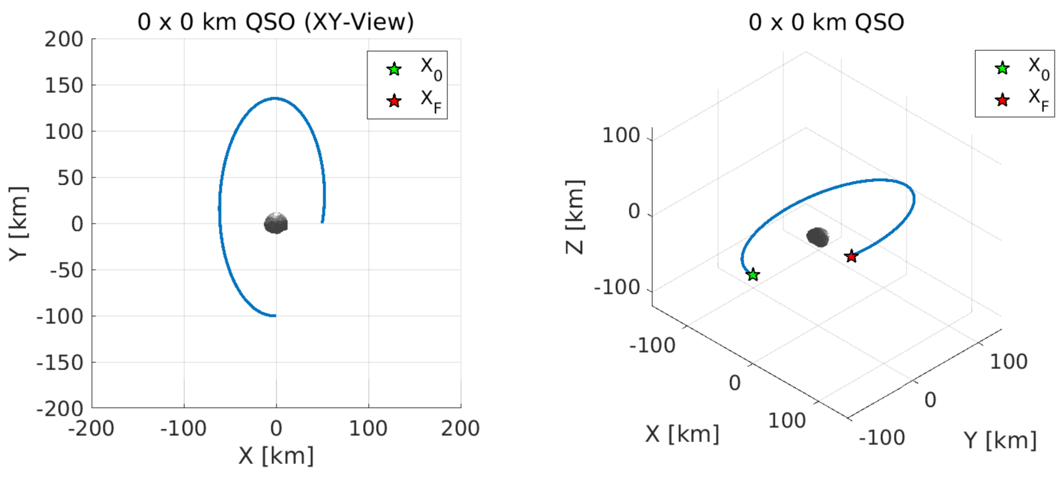


Figure 5.7: Scenario 3 XY-plane and 3D view.

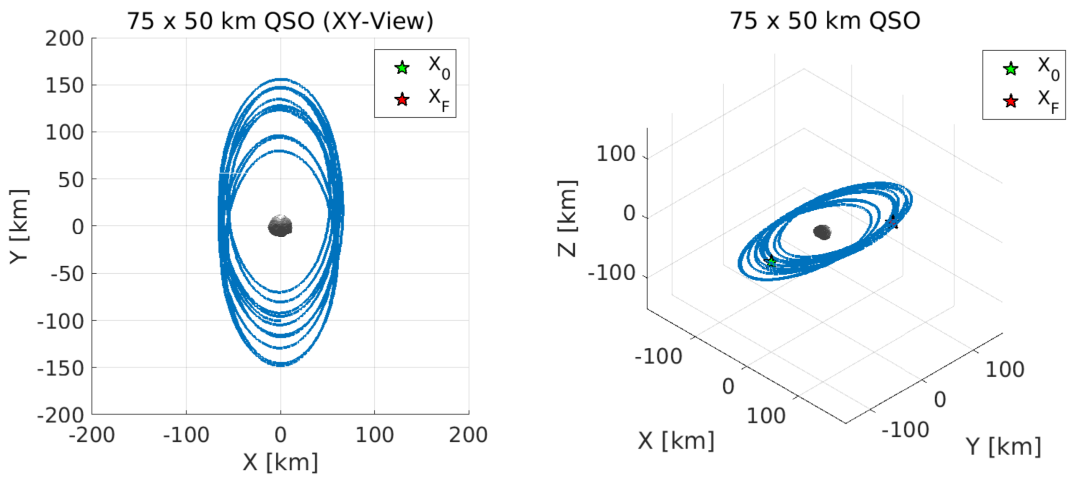


Figure 5.8: Scenario 3 XY-plane and 3D view, three days after reaching P_T.

5.3.4 Scenario 4

Table 5.5: Scenario 4 summary.

Input	
Initial Position P_0 [km]	(0, -100.0, 0)
Target Position P_T [km]	(0, -20.0, 0)
Trajectory Duration Δt [Day(s)]	7 / 24
Initial Guess V_0 [km/s]	(S)
Collision-Free Trajectory	Yes

Output	
Initial Velocity V_X [km/s]	0.00503333
Initial Velocity V_Y [km/s]	-0.00179442
Initial Velocity V_Z [km/s]	-0.00009215
Time to Convergence [s]	586
Simplex Iterations	1397

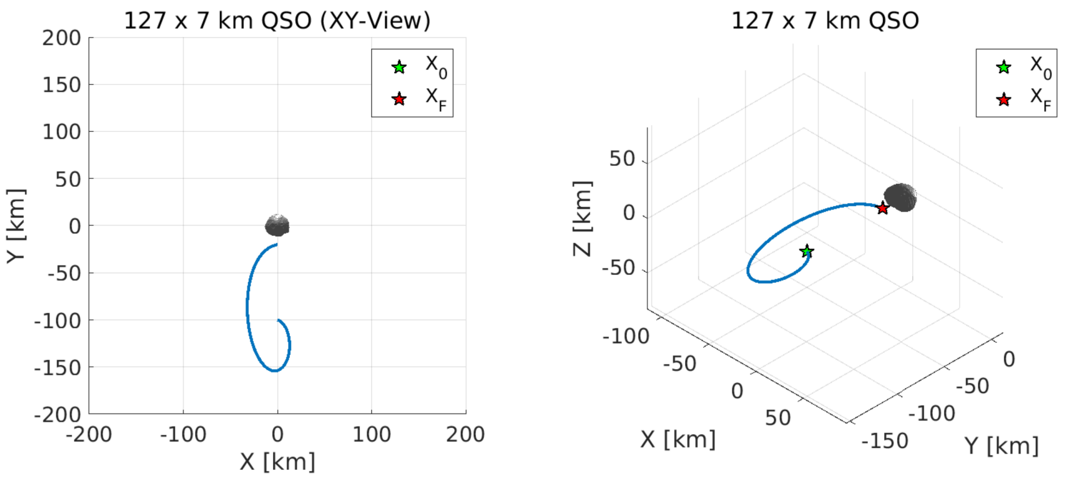


Figure 5.9: Scenario 4 XY-plane and 3D view.

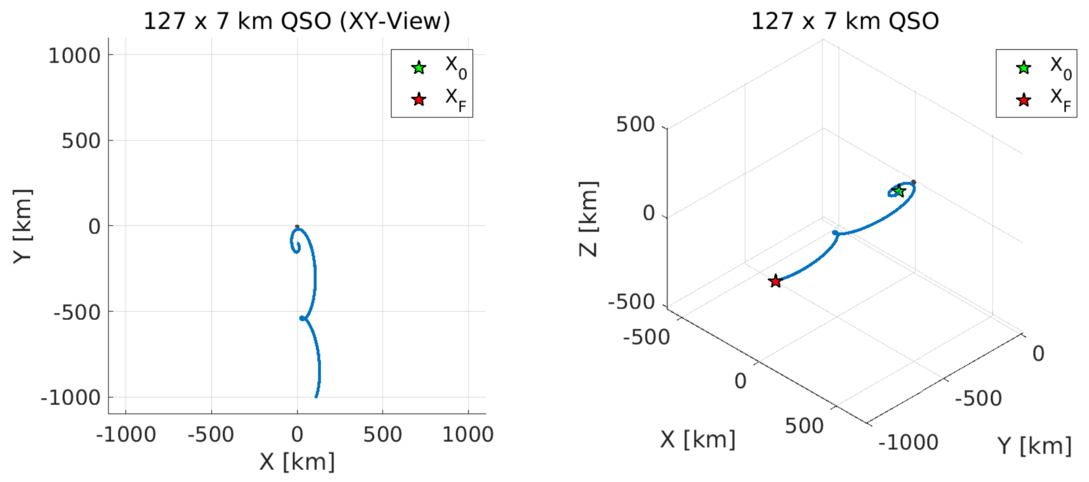


Figure 5.10: Scenario 4 XY-plane and 3D view, three days after reaching P_T.

5.3.5 Scenario 5

Table 5.6: Scenario 5 summary.

Input	
Initial Position P_0 [km]	(0, -75.0, 0)
Target Position P_T [km]	(0, 21.1, 0)
Trajectory Duration Δt [Day(s)]	7 / 24
Initial Guess V_0 [km/s]	(U) (-0.003, 0, 0)
Collision-Free Trajectory	Yes

Output	
Initial Velocity V_X [km/s]	-0.00853550
Initial Velocity V_Y [km/s]	-0.00219707
Initial Velocity V_Z [km/s]	-0.00010377
Time to Convergence [s]	257
Simplex Iterations	1243

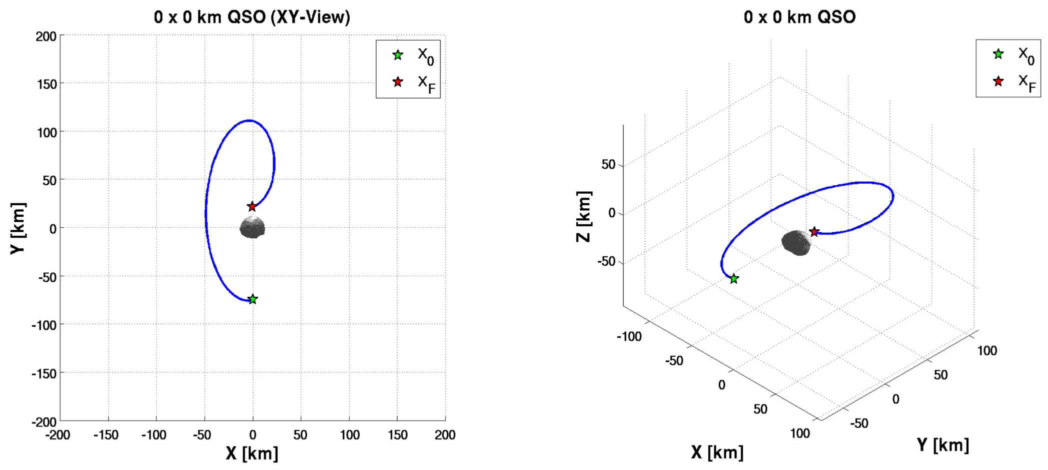


Figure 5.11: Scenario 1 XY-plane and 3D view.

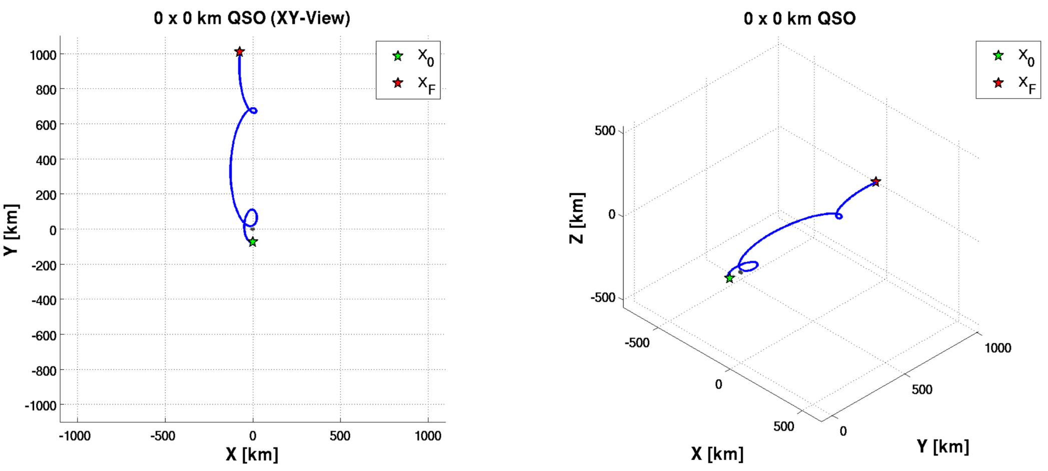


Figure 5.12: Scenario 1 XY-plane and 3D view, three days after reaching P_T.

5.3.6 Scenario 6

Table 5.7: Scenario 6 summary.

Input	
Initial Position P_0 [km]	(0, -75.0, 0)
Target Position P_T [km]	(0, 20.0, 0)
Trajectory Duration Δt [Day(s)]	7 / 24
Initial Guess V_0 [km/s]	(U) (0.0007, -0.0035, 0)
Collision-Free Trajectory	Yes

Output	
Initial Velocity V_X [km/s]	0.00169723
Initial Velocity V_Y [km/s]	-0.00311805
Initial Velocity V_Z [km/s]	0.00011270
Time to Convergence [s]	132
Simplex Iterations	707

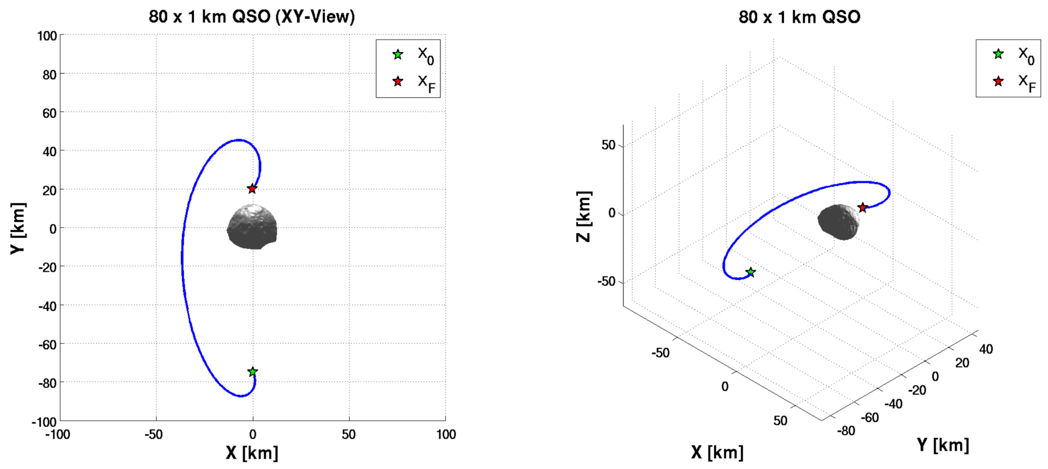


Figure 5.13: Scenario 1 XY-plane and 3D view.

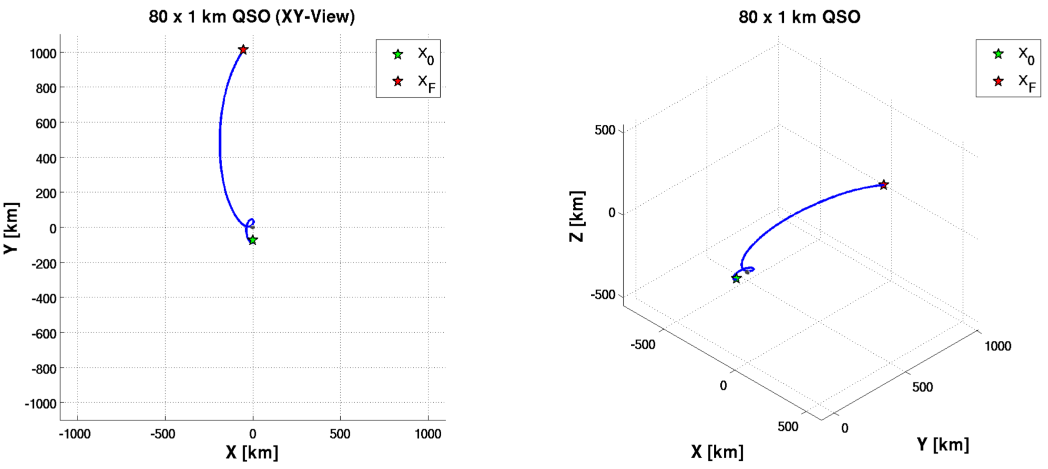


Figure 5.14: Scenario 1 XY-plane and 3D view, three days after reaching P_T.

5.3.7 Scenario 7

Table 5.8: Scenario 7 summary.

Input	
Initial Position P_0 [km]	(0, 100.0, 0)
Target Position P_T [km]	(0, -20.0, 0)
Trajectory Duration Δt [Day(s)]	12.6 / 24
Initial Guess V_0 [km/s]	(U) (0.003, 0, 0)
Collision-Free Trajectory	Yes

Output	
Initial Velocity V_X [km/s]	0.01003698
Initial Velocity V_Y [km/s]	0.00128617
Initial Velocity V_Z [km/s]	0.00011656
Time to Convergence [s]	174
Simplex Iterations	1261

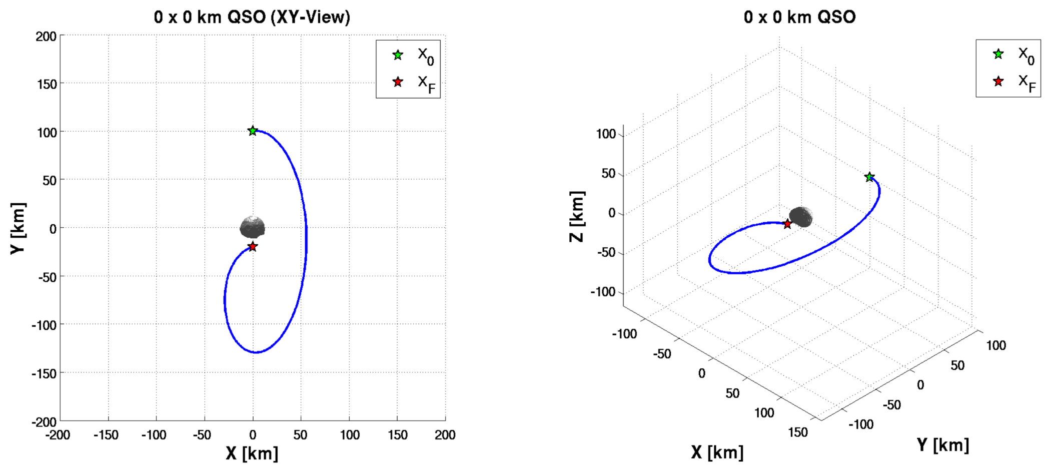


Figure 5.15: Scenario 7 XY-plane and 3D view.

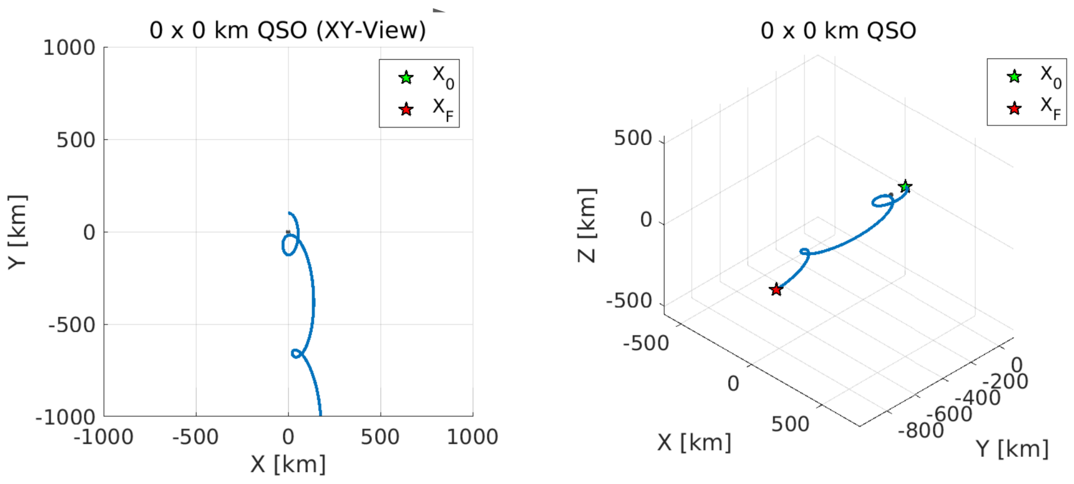


Figure 5.16: Scenario 7 XY-plane and 3D view, three days after reaching P_T.

5.3.8 Scenario 8

Table 5.9: Scenario 8 summary.

Input	
Initial Position P_0 [km]	(0, -100.0, 0)
Target Position P_T [km]	(0, 20.0, 0)
Trajectory Duration Δt [Day(s)]	6.4 / 24
Initial Guess V_0 [km/s]	(U) (0.0007, -0.0035, 0)
Collision-Free Trajectory	Yes

Output	
Initial Velocity V_X [km/s]	-0.00433855
Initial Velocity V_Y [km/s]	-0.00682030
Initial Velocity V_Z [km/s]	0.00002138
Time to Convergence [s]	123
Simplex Iterations	752

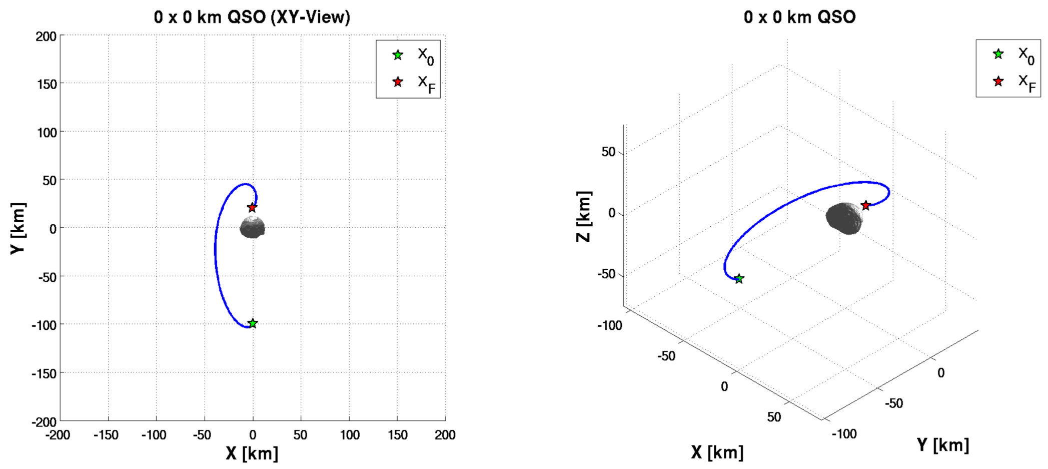


Figure 5.17: Scenario 8 XY-plane and 3D view.

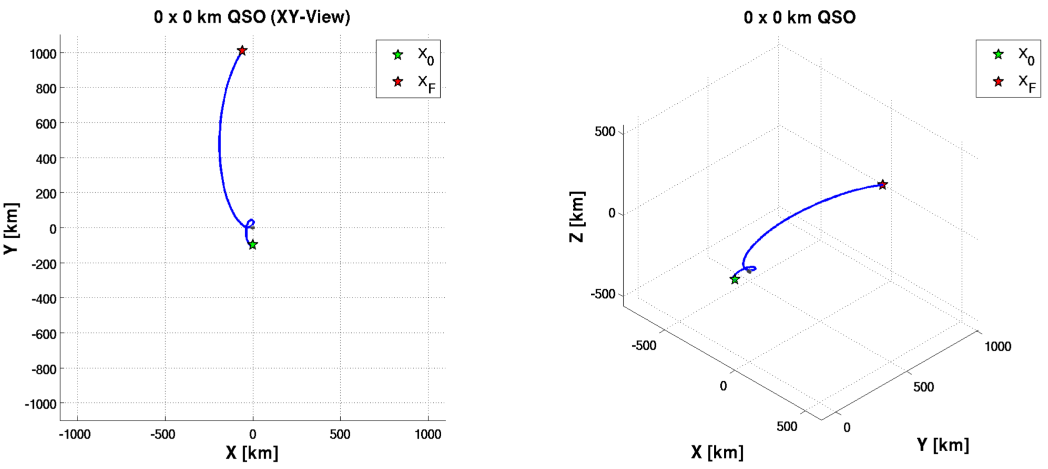


Figure 5.18: Scenario 8 XY-plane and 3D view, three days after reaching P_T.

5.4 Discussion of Results

In the first scenarios, Scenario 1 to Scenario 3, the tool was able to converge fairly quickly with the use of Lambert's Problem estimate. However, it can be observed that the amount of iterations required increases with the transfer duration. This is also true for other cases, but Lambert's Problem showed to be only advantageous for short transfers and only when those presented a target location further away from the secondary body, Phobos. Note that these scenarios display a gate position (target position) at a coordinate 50 km away from the center of Phobos, versus the 20 km option used by the remaining tests. Also important to note that from all the scenarios, only those traveling to a higher gate position (Scenario 1 and Scenario 3), are able to recover to a QSO as collision-free trajectory. The others all move away from Phobos.

Multiple attempts were made to determine a collision-free trajectory from 75 km (negative Y-axis) to the X-axis (positive or negative), but unsuccessfully. The trajectories obtained were similar to Scenario 1 and Scenario 3, hence were not presented here as separate scenarios. Observing Scenario 4 to Scenario 8, there is a distinct form that the trajectory path presents to achieve a collision-free trajectory, which entails performing a kind of *knot* to avoid a collision path. This type of trajectory seems unattainable for a target position on the X-axis, since from numerous trials it was observed that the spacecraft would be pulled strongly towards Phobos, even if attempted a longer transfer for a broader trajectory arch as in Scenario 7. Various attempts were required to obtain the data here displayed, since exhaustive test of different transfer times and target positions had to be investigated, until different trajectory types were obtained.

From Scenario 4 onwards it can be concluded that a gate closer to Phobos requires a more precise approach to each scenario. With enough time and familiarity with the problem at hand it was possible to determine different trajectories. Seven out of the eight cases presented possess the collision-free trajectory quality, thus showing how effective the tool is at finding this particular type of path. Although collision-free trajectories were not found for negative Y-axis to X-axis, the tool provided a means to perform this verification, providing information if such path would be possible. The aim of this section was to establish multiple test cases where it showed the tool's ability to find a solution for even the more unusual problem statement, which it has.

Chapter 6

Conclusions

6.1 Achievements

The main achievement of this work is the fulfillment of both the primary and secondary objectives set for this thesis by GMV. The development of the TPBVP tool followed multiple paths, taking into consideration a variety of approaches under distinct methodologies. The techniques used were able to cope with the difficulties of the problem. This tool proved to be successful at finding the desired solution in a highly unstable environment.

The secondary objective of being able to find collision-free trajectories under different scenarios and constrains, was a complete success. However, it required an heuristic approach which was developed while engaging the specific system of Mars-Phobos.

The final product of this thesis is a tool designed to be adaptable and generic, for future implementation within other software suites. The software was developed with multiple inputs, thus providing an advanced user with more knowledge of a specific problem to take full advantage of SBNav and its new TPBVP module.

6.2 Future Work

At the time of this thesis development, GMV was using SBNav for a new case study, the paired Didymos asteroids. The opportunity to apply the TPBVP tool to a new scenario would give more opportunities to analyze its generic ability to solve TPBVP.

Finally, the more interesting path to take this work forward would be to remove one of the boundaries of the TPBVP: the trajectory transfer duration. Such problem would add the transfer time as one of the variables to be optimized. If looking through Nelder-Mead Method, it would create a simplex of five dimensions, adding more complexity to the problem. However, defining a new convergence/termination criteria for the simplex method would be the real challenge, since it is now defined by different types of variables. Alas, the basic architecture of SBNav and the original targets of the FASTMOPS project limits this option.

Bibliography

[1] D. J. Scheeres. *Orbital Motion in Strongly Perturbed Environments*. Springer, 2012.

[2] J. Branco, L. Guerreiro, and F. Cabral. *FASTMOPS TN1 - Assumptions Consolidation*. GMV Innovating Solutions, 2015.

[3] W. F. B. Jr., A. Cellino, P. Paolicchi, and R. P. Binzel. *Asteroids III*. The University of Arizona Press, 2002.

[4] G. Balmino. Gravitational potential harmonics from the shape of an homogeneous body. *Celestial Mechanics and Dynamical Astronomy*, 60:331–364, 1994.

[5] D. Boccaletti and G. Pucacco. *Theory of Orbits*, volume Perturbative and Geometrical Methods. Springer-Verlag Berlin Heidelberg, 1999.

[6] A. Konopliv, J. Miller, W. Owen, D. Yeomans, J. Giorgini, R. Garmier, and J.-P. Barriot. A global solution for the gravity field, rotation, landmarks, and ephemeris of Eros. *Icarus*, 160:289–299, December 2002.

[7] T. Kubota, M. Otsuki, T. Hashimoto, H. Y. N. Bando, M. Uo, K. Shirakawa, and J. Kawaguchi. Touchdown dynamics for sampling in Hayabusa mission. In *Astrodynamics Specialist Conference*, number 6539 in Paper AIAA 2006. American Institute of Aeronautics and Astronautics, 2006.

[8] M. Lara and R. Russell. Computation of a science orbit about Europa. *Journal of Guidance, Control, and Dynamics*, 30(1):259–263, 2007.

[9] V. Shishov. Determination of spacecraft and phobos parameters of motion in the Phobos-Grunt project. *Solar System Research*, 42(4):319–328, 2008.

[10] J. Bellerose. The restricted full three body problem: Applications to binary asteroid exploration. Phd Thesis, The University of Michigan, 2008.

[11] P. J. S. Gil and J. Schwartz. Simulations of quasi-satellite orbits around phobos. *Journal of Guidance, Control, and Dynamics*, 33(3), May-June 2010.

[12] J. Branco, L. Guerreiro, and F. Cabral. *FASTMOPS TN2 - Software Requirement and Justification Document*. GMV Innovating Solutions, 2014.

[13] NASA. The navigation and ancillary information facility (NAIF), 2016. URL `https://naif.jpl.nasa.gov/naif/naif/`.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.

[15] D. Izzo. Revisiting Lambert's Problem. *Springer Science+Business Media Dordrecht*, 2014.

[16] J. Branco. *FASTMOPS Final Report*. GMV Innovating Solutions, 2016.

[17] B. Adam and M. H. Hashim. Shooting method in solving boundary value problem. *IJRRAS*, 2014.

[18] D. Bachrathy. *Multi-Dimensional Bisection Method User's Method*. Budapest University of Technology and Economics, Department of Applied Mechanics, 2014.

[19] W. J. Gilbert. Newton's Method for multiple roots. *Pergamon Press*, 1994.

[20] J. L. Martin. A Quasi-Newton method for solving small nonlinear systems of algebraic equations. *Massachusetts Institute of Technology*, 2013.

[21] R. F. King. A secant method for multiple roots. *BIT Numerical Mathematics*, 1977.

[22] J. Paloschi and J. Perkins. An implementation of quasi-newton methods for solving sets of nonlinear equations. *Pergamon Press*, 1987.

[23] J. Klement. On using Quasi-Newton algorithms of the Broyden class for model-to-test correlation. *Tesat-Spacecom GmbH & Co*, 2014.

[24] J. Carlsson and J. R. Cary. The hypersecant Jacobian approximation for quasi-Newton solves of sparse nonlinear systems. *Elsevier*, 2009.

[25] C. Remani. Numerical methods for solving systems of nonlinear equations. *Math 4301 (Honour's Seminar)*, 2013.

[26] R. P. LeBoeuf. *Root-Finding Methods in Two and Three Dimensions*. University of Massachusetts, December 2009.

[27] K. A. Kopecky. *Root-Finding Methods*. CiteSeerx, 2007.

[28] C. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19(95):577–593, 1965.

[29] K. Muhammad, M. Mamat, and M. Y. Waziri. A Broyden's-like Method for solving systems of nonlinear equations. *World Applied Sciences Journal*, 21:168–173, 2013.

[30] E. Kvaalen. A faster Broyden method. *BIT Numerical Mathematics*, 31:369–372, 1991.

[31] A. Griewank. Broyden updating, the good and the bad! *Documenta Mathematica*, 2012.

[32] D. M. Gay and R. B. Schnabel. Solving systems of non-linear equations by Broyden's method with projected updates. *Working Paper Series*, 169, March 1977.

[33] M. Mamat, K. Muhammad, and M. Y. Waziri. Trapezoidal broyden's method for solving systems of nonlinear equations. *Applied Mathematical Sciences*, 8(6):251–260, 2014.

[34] J. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4): 308–313, January 1965.

[35] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, and M. Metcalf. *Numerical Recipes in Fortran 90*. Cambridge University Press, 2nd edition, 1996.

[36] J. H. Mathews and K. K. Fink. *Numerical Methods Using Matlab*. Prentice-Hall Inc., 4th edition, 2004. Chapter 8. Numerical Optimization.

[37] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes The Art of Scientific Computing*, chapter 10. Minimization or Maximization of Functions, page 503. Cambridge University Press, 3rd edition, 2007.

[38] J. Lagarias, J. Reeds, M. Wright, and P. Wright. Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147, December 1998.

[39] H. P. Gavin. *The Nelder-Mead Algorithm in Two Dimensions*. Duke University, 2016. CEE 201L. Uncertainty, Design, and Optimization.

[40] F. Gao and L. Han. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51:259–277, 2012.

[41] N. Pham and B. M. Wilamowski. Improved Nelder Mead's simplex method and applications. *Journal of Computing*, 3(3), March 2011.

[42] L. A. Yarbro and S. N. Deming. Selection and preprocessing of factors for simplex optimization. *Analytica Chimica Acta*, 73:391–398, 1974.

[43] J. C. Lagarias, B. Poonen, and M. H. Wright. Convergence of the restricted Nelder-Mead algorithm in two dimensions. *SIAM Journal on Optimization*, 22(2), April 2011.

[44] X.-S. Yang. *Nature-Inspired Optimization Algorithms*. Elsevier, 1st edition, 2014. Chapter 5 - Genetic Algorithms.

[45] X.-S. Yang, A. H. Gandomi, S. Talatahari, and A. H. Alavi. *Metaheuristics in Water, Geotechnical and Transport Engineering*. Elsevier, 1st edition, 2013.

[46] D. Mondal, A. Chakrabarti, and A. Sengupta. *Metaheuristics in Water, Geotechnical and Transport Engineering*. Academic Press, 1st edition, 2014.

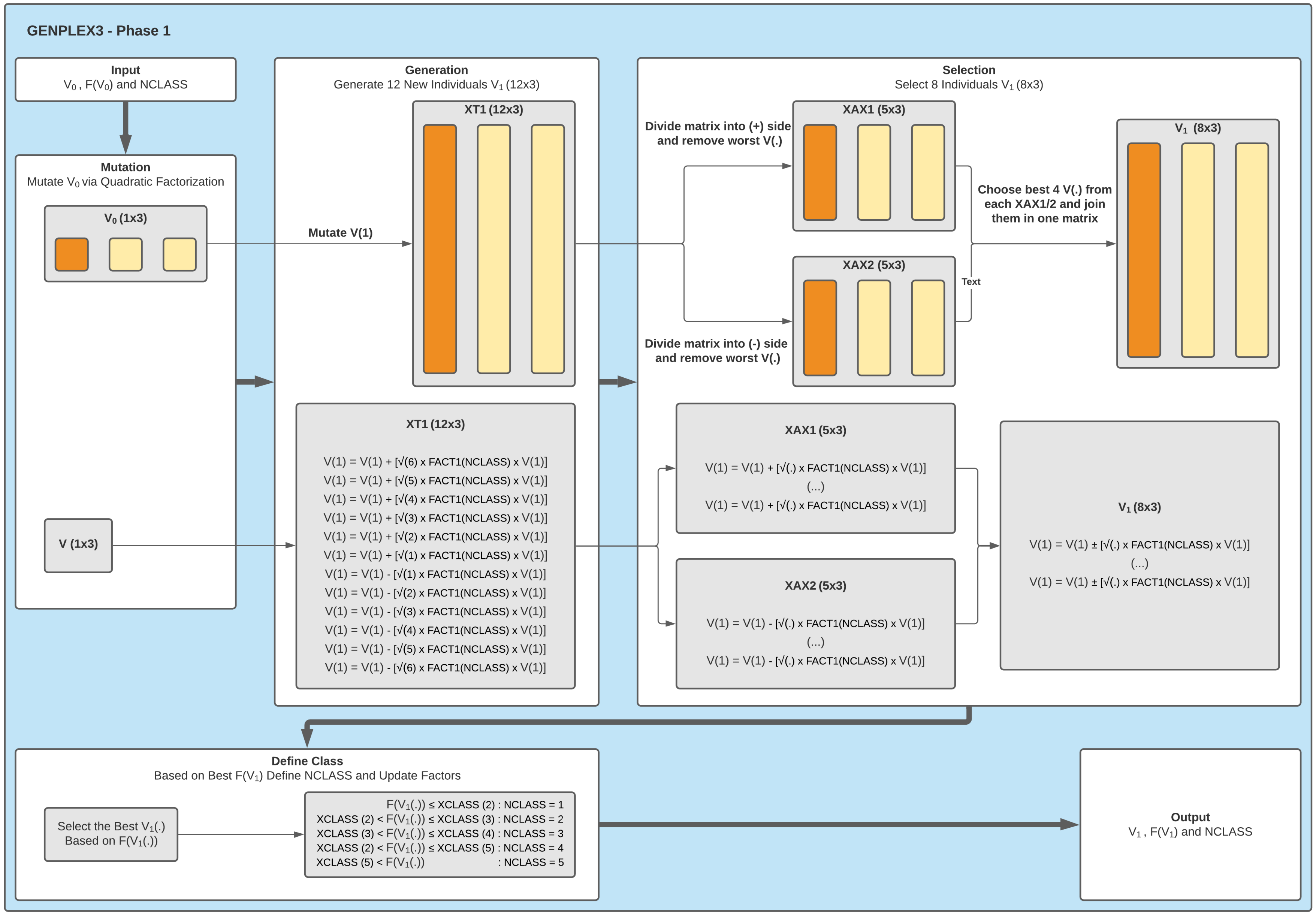
[47] T. P. Llanos, E. D. Sotto, J. P. Muñoz, and P. Rogata. Genetic algorithms in the generation of an initial guess for the optimisation of ascent trajectory with an hybrid method. In *5th International Conference on Space Launchers*, ResearchGate. GMV S.A., November 2003.

[48] P. Rogata, E. D. Sotto, M. Graziano, and F. Graziani. Guess values for interplanetary transfer through genetic algorithms. *Advances in the Astronautical Sciences*, 114, January 2003.

Appendix A

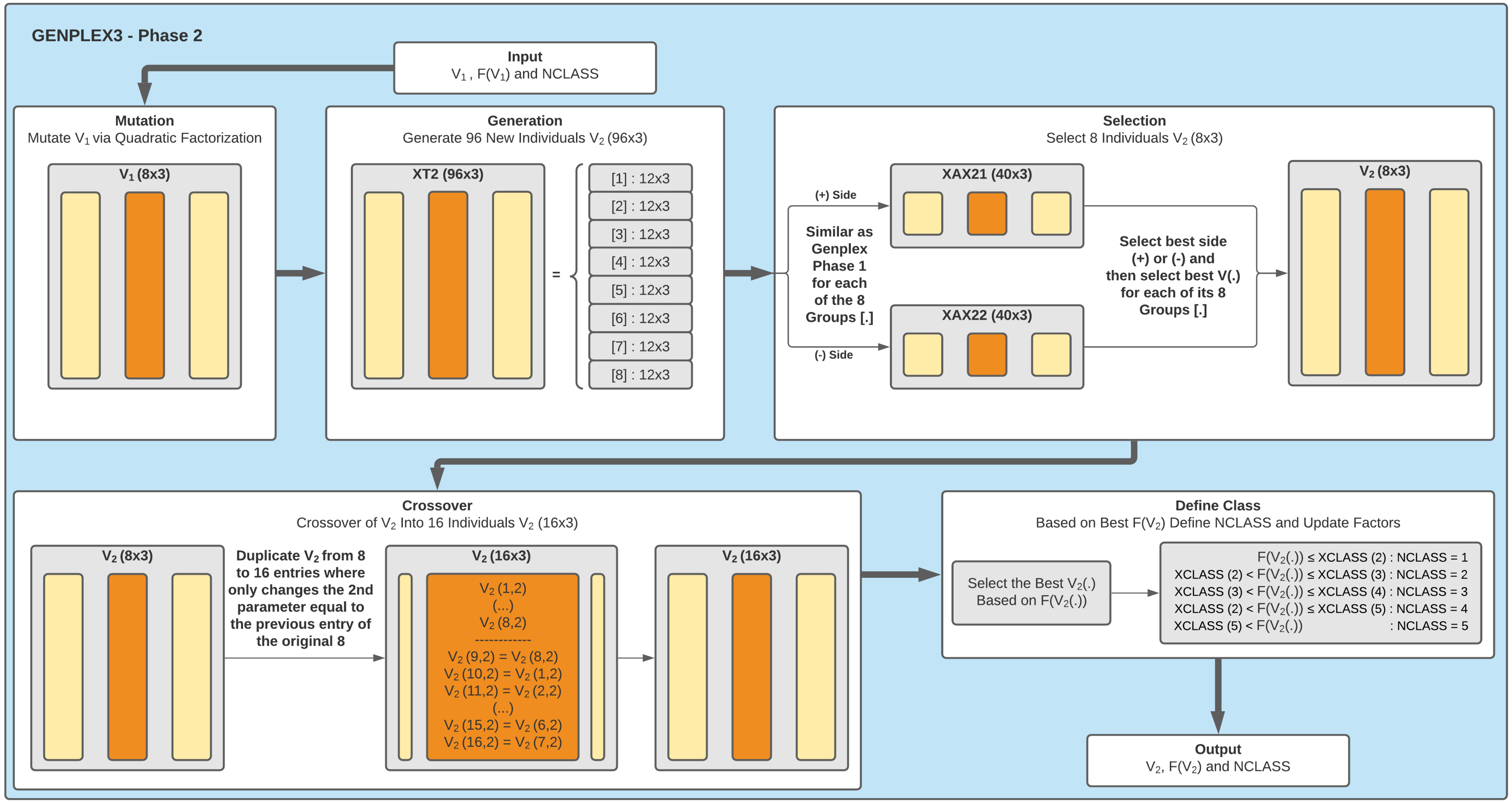
Figures

In the following pages one can found figures found along this thesis that might need a higher dimension to be eligible.



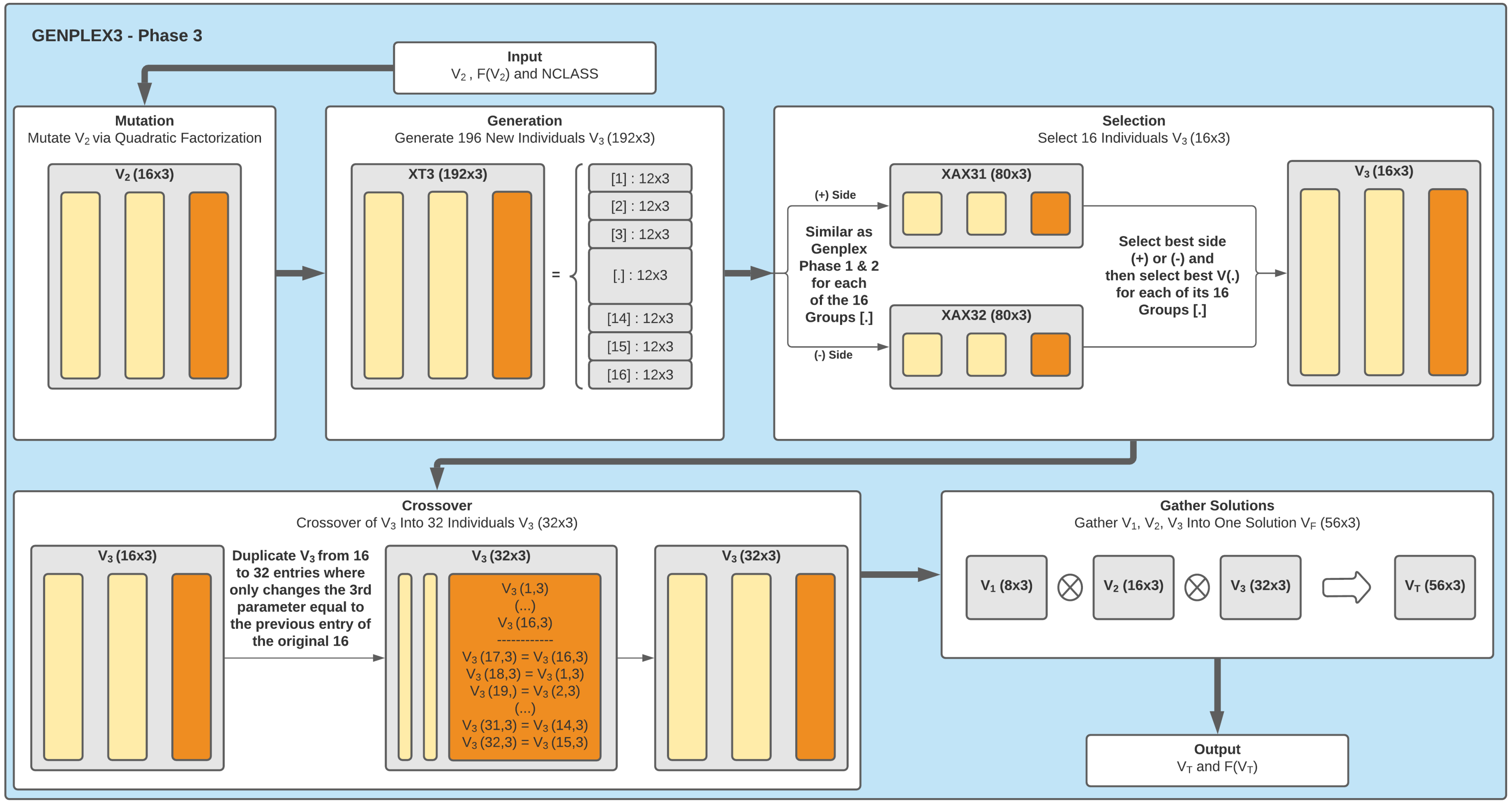
GENPLEX3 - Phase 1
Input
V_0 , $F(V_0)$ and NCLASS
Mutation
Mutate V_0 via Quadratic Factorization
V_0 (1x3)
V (1x3)
Generation
Generate 12 New Individuals V_1 (12x3)
Mutate V(1)
XT1 (12x3)
XT1 (12x3)
V(1) = V(1) + [√(6) x FACT1(NCLASS) x V(1)]
V(1) = V(1) + [√(5) x FACT1(NCLASS) x V(1)]
V(1) = V(1) + [√(4) x FACT1(NCLASS) x V(1)]
V(1) = V(1) + [√(3) x FACT1(NCLASS) x V(1)]
V(1) = V(1) + [√(2) x FACT1(NCLASS) x V(1)]
V(1) = V(1) + [√(1) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(1) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(2) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(3) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(4) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(5) x FACT1(NCLASS) x V(1)]
V(1) = V(1) - [√(6) x FACT1(NCLASS) x V(1)]
Selection
Select 8 Individuals V_1 (8x3)
Divide matrix into (+) side and remove worst V(.)
Divide matrix into (-) side and remove worst V(.)
XAX1 (5x3)
XAX2 (5x3)
Choose best 4 V(.) from each XAX1/2 and join them in one matrix
Text
V_1 (8x3)
XAX1 (5x3)
V(1) = V(1) + [√(.) x FACT1(NCLASS) x V(1)]
(...)
V(1) = V(1) + [√(.) x FACT1(NCLASS) x V(1)]
XAX2 (5x3)
V(1) = V(1) - [√(.) x FACT1(NCLASS) x V(1)]
(...)
V(1) = V(1) - [√(.) x FACT1(NCLASS) x V(1)]
V_1 (8x3)
V(1) = V(1) ± [√(.) x FACT1(NCLASS) x V(1)]
(...)
V(1) = V(1) ± [√(.) x FACT1(NCLASS) x V(1)]
Define Class
Based on Best $F(V_1)$ Define NCLASS and Update Factors
Select the Best V_1(.) Based on $F(V_1(.))$
$F(V_1(.)) \leq$ XCLASS (2) : NCLASS = 1
XCLASS (2) < $F(V_1(.)) \leq$ XCLASS (3) : NCLASS = 2
XCLASS (3) < $F(V_1(.)) \leq$ XCLASS (4) : NCLASS = 3
XCLASS (2) < $F(V_1(.)) \leq$ XCLASS (5) : NCLASS = 4
XCLASS (5) < $F(V_1(.))$: NCLASS = 5
Output
V_1 , $F(V_1)$ and NCLASS

Figure A.1: Diagram for [3] GENPLEX3 - Phase 1 segment.


GENPLEX3 - Phase 2
Input
V_1 , $F(V_1)$ and NCLASS
Mutation
Mutate V_1 via Quadratic Factorization
V_1 (8x3)
Generation
Generate 96 New Individuals V_2 (96x3)
XT2 (96x3)
=
[1] : 12x3
[2] : 12x3
[3] : 12x3
[4] : 12x3
[5] : 12x3
[6] : 12x3
[7] : 12x3
[8] : 12x3
Selection
Select 8 Individuals V_2 (8x3)
(+) Side
(-) Side
Similar as Genplex Phase 1 for each of the 8 Groups [.]
XAX21 (40x3)
XAX22 (40x3)
Select best side (+) or (-) and then select best V(.) for each of its 8 Groups [.]
V_2 (8x3)
Crossover
Crossover of V_2 Into 16 Individuals V_2 (16x3)
V_2 (8x3)
Duplicate V_2 from 8 to 16 entries where only changes the 2nd parameter equal to the previous entry of the original 8
V_2 (16x3)
V_2 (1,2)
(...)
V_2 (8,2)

$V_2 (9,2) = V_2 (8,2)$
$V_2 (10,2) = V_2 (1,2)$
$V_2 (11,2) = V_2 (2,2)$
(...)
$V_2 (15,2) = V_2 (6,2)$
$V_2 (16,2) = V_2 (7,2)$
V_2 (16x3)
Define Class
Based on Best $F(V_2)$ Define NCLASS and Update Factors
Select the Best $V_2(.)$ Based on $F(V_2(.))$
$F(V_2(.)) \leq$ XCLASS (2) : NCLASS = 1
XCLASS (2) $< F(V_2(.)) \leq$ XCLASS (3) : NCLASS = 2
XCLASS (3) $< F(V_2(.)) \leq$ XCLASS (4) : NCLASS = 3
XCLASS (2) $< F(V_2(.)) \leq$ XCLASS (5) : NCLASS = 4
XCLASS (5) $< F(V_2(.))$: NCLASS = 5
Output
V_2, $F(V_2)$ and NCLASS

Figure A.2: Diagram for [4] GENPLEX3 - Phase 2 segment.


GENPLEX3 - Phase 3
Input
V_2, $F(V_2)$ and NCLASS
Mutation
Mutate V_2 via Quadratic Factorization
V_2 (16x3)
Generation
Generate 196 New Individuals V_3 (192x3)
XT3 (192x3)
=
[1] : 12x3
[2] : 12x3
[3] : 12x3
[.] : 12x3
[14] : 12x3
[15] : 12x3
[16] : 12x3
Selection
Select 16 Individuals V_3 (16x3)
(+) Side
(-) Side
Similar as Genplex Phase 1 & 2 for each of the 16 Groups [.]
XAX31 (80x3)
XAX32 (80x3)
Select best side (+) or (-) and then select best V(.) for each of its 16 Groups [.]
V_3 (16x3)
Crossover
Crossover of V_3 Into 32 Individuals V_3 (32x3)
V_3 (16x3)
Duplicate V_3 from 16 to 32 entries where only changes the 3rd parameter equal to the previous entry of the original 16
V_3 (32x3)
V_3 (1,3)
(...)
V_3 (16,3)

V_3 (17,3) = V_3 (16,3)
V_3 (18,3) = V_3 (1,3)
V_3 (19,) = V_3 (2,3)
(...)
V_3 (31,3) = V_3 (14,3)
V_3 (32,3) = V_3 (15,3)
V_3 (32x3)
Gather Solutions
Gather V_1, V_2, V_3 Into One Solution V_F (56x3)
V_1 (8x3)
$\otimes$
V_2 (16x3)
$\otimes$
V_3 (32x3)
V_T (56x3)
Output
V_T and $F(V_T)$

Figure A.3: Diagram for [5] GENPLEX3 - Phase 3 segment.