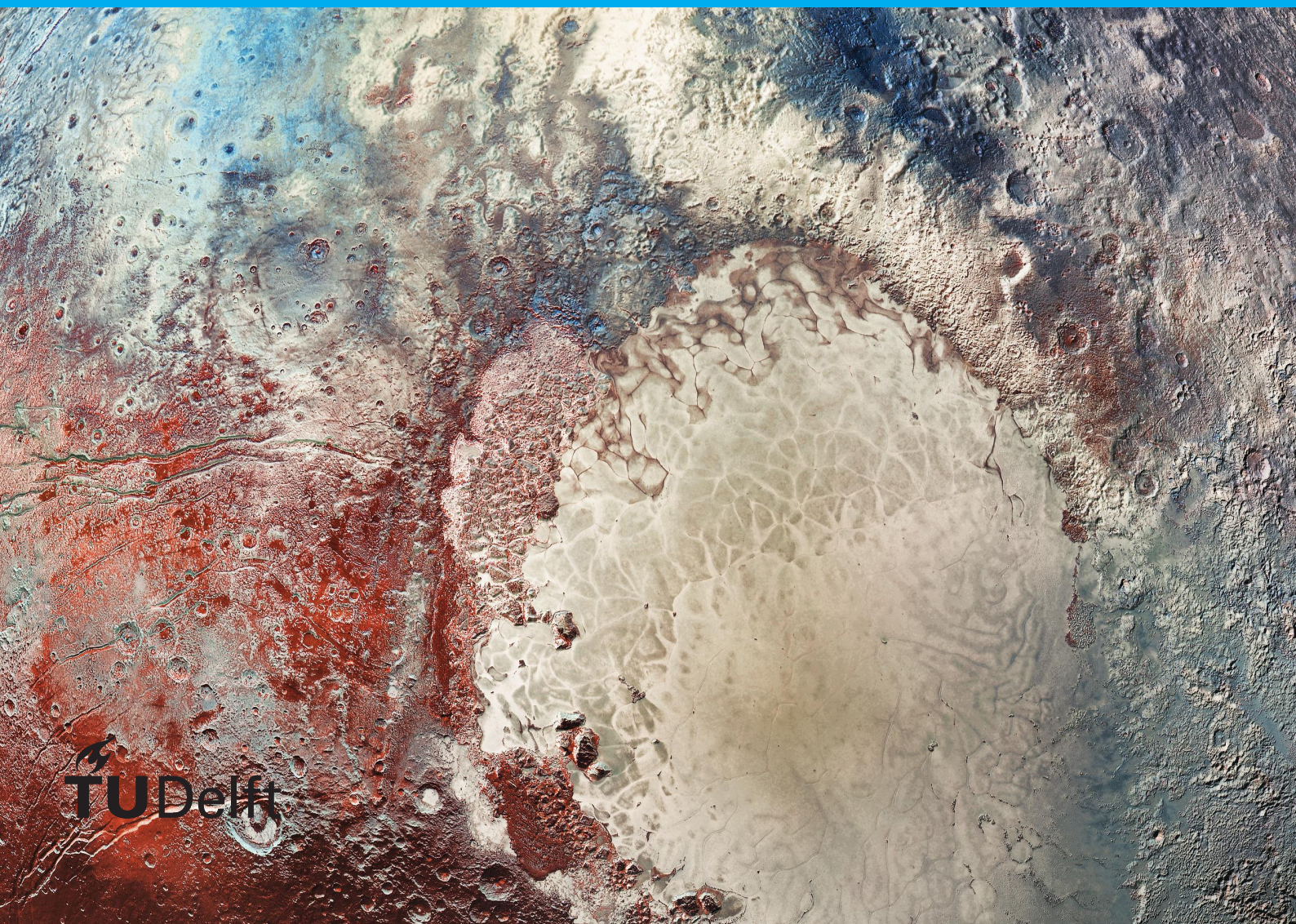
Low-Thrust Gravity Assist Trajectory Optimisation

Using Evolutionary Neurocontrol

T.A.H. Kranen



Cover image shows an enhanced color view of the surface of Pluto as captured by the New Horizons spacecraft on July 14, 2015 at a distance of 450,000 km. The image combines four images from the Long Range Reconnaissance Imager (LORRI) with blue, red and infrared images taken by the spacecraft's Ralph/Multispectral Visual Imaging Camera (MVIC). Courtesy of [57].

Low-Thrust Gravity Assist Trajectory Optimisation

Using Evolutionary Neurocontrol

by

T.A.H. Kranen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday July 18, 2019 at 14:00 AM.

Student number:	4148193	
Project duration:	January 07, 2018 – July 18, 2019	
Thesis committee:	Prof. Dr. Ir. P. N. A. M. Visser ,	TU Delft, department chair
	Ir. K. J. Cowan MBA,	TU Delft, supervisor
	Prof. Dr. B. Dachwald,	FH Aachen, supervisor
	Dr. A. Cervone,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

With pride I present my thesis on low-thrust gravity assist trajectory optimisation using the novel method of evolutionary neurocontrol. This work marks the culmination of my time at Delft University of Technology, where I have studied for both a BSc and MSc degree in Aerospace Engineering. My years at TU Delft have given me valuable insights, impeccable problem solving skills and a strong mathematical foundation. With my acquired skill set I am more than ready to tackle any upcoming challenges in both my personal and professional life.

My interest in space exploration originated at primary school, where a teacher recognised I could use a challenge and tasked me with researching the planets of our Solar system. I proudly created a mobile of the planets, ready to present it in front of the entire class, however –and I kid you not– the dog ate it the day before. I showed up at school with a mobile that was covered in bite marks and disintegrating papier-mâché planets, and proudly presented my results. From that point onwards I was hooked on the space bug, which ultimately led to this work.

The bachelor thesis project, in which we designed a mission to Jupiter's icy moon Europa, introduced me to the concepts involved in space mission analysis and phase-A studies. Mission analysis intrigued me, rendering me to enrol for an MSc programme in Space Exploration. An integral part of mission analysis in phase-A studies is the design of the trajectory, which nowadays heavily relies on numerical methods. Exposure to the GTOC[1] further increased my interest in trajectory optimisation and gravity assists, especially for intricate trajectories in the Jovian system. Intrigued by what others were achieving with artificial intelligence and machine learning, I wondered whether machine learning could be combined with trajectory design. Much to my enjoyment, I stumbled across the original work of Bernd Dachwald on the low-thrust trajectory optimisation method termed InTrance, which combines exactly what I had hoped for; artificial intelligence and interplanetary trajectory optimisation.

The original developers of InTrance, Bernd Dachwald and Andreas Ohndorf, were happy for me to continue on their work. Coincidentally, Bernd and Andreas organised a workshop on InTrance, which my thesis supervisor Kevin Cowan and I attended, rendering a host of possible research topics. Due to my interest in intricate trajectory design, such as those in the Jovian system which can perform as many as 16 gravity assist, I opted to extend InTrance with gravity assist capabilities.

Although my original intention of optimising low-thrust gravity assist trajectories to Europa could not be achieved within the scope of this work, a promising novel method has been developed. This accomplishment was no small feat, as it was uncertain whether evolutionary neurocontrol could be used to perform a gravity assist at all, and even thought to be out of scope of a MSc thesis by the original developers of InTrance. A proper baseline for low-thrust gravity assist optimisation with evolutionary neurocontrol is set, which is hopefully extended in future work to eventually allow for the optimisation of trajectories to Europa.

I would sincerely like to thank Bernd Dachwald, Andreas Ohndorf and Kevin Cowan for taking the time in guiding me through this lengthy process. Our fruitful meetings have given me clarity of the problem and guided me in the right direction. I specifically thank Bernd for our initial hours-long meetings on possible strategies, and Andreas for our weekly in-depth technical calls.

T.A.H. Kranen
Nieuwegein, June 28, 2019

[1] Global Trajectory Optimisation Competition: `https://sophia.estec.esa.int/gtoc_portal/?page_id=26`

Contents

Preface iii

List of Abbreviations ix

List of Figures xi

List of Tables xiii

1 Introduction 1

1.1 Research Motivation . . . 2
1.2 Research Framework Definition . . . 3
1.2.1 Research Questions . . . 3
1.2.2 Research Objectives . . . 3
1.3 Report Structure . . . 4

2 Gravity Assists 7

2.1 Missions with Gravity Assists . . . 7
2.1.1 New Horizons . . . 8
2.1.2 Dawn . . . 9
2.2 Dynamics and Geometry . . . 10
2.3 Effects of Gravity Assists . . . 13
2.3.1 Deflection Angle . . . 13
2.3.2 Velocity Variation . . . 13
2.3.3 Inclination Variation . . . 15
2.4 Powered Gravity Assist . . . 16
2.4.1 New Horizons Powered Gravity Assist Simulations . . . 16

3 Low-Thrust Trajectory Optimisation with Evolutionary Neurocontrol 19

3.1 The Low-Thrust Trajectory Optimisation Problem . . . 19
3.2 Traditional Trajectory Optimisation . . . 20
3.2.1 Local Trajectory Optimisation Methods . . . 20
3.2.2 Global Trajectory Optimisation Methods . . . 22
3.2.3 State-of-the-Art Trajectory Optimisation Tools . . . 22
3.3 Smart Low-Thrust Trajectory Optimisation . . . 23
3.4 Artificial Intelligence and Reinforcement Learning . . . 23
3.4.1 Markov Decision Process . . . 24
3.4.2 Solving Markov Decision Processes . . . 25
3.4.3 Trajectory Optimisation from the Perspective of Reinforcement Learning . . . 25
3.5 Artificial Neural Networks . . . 26
3.5.1 Biological Paradigm . . . 26

3.5.2 Components of Artificial Neural Networks . . . 27
3.5.3 Network of Neurons . . . 28
3.6 Evolutionary Algorithms . . . 29
3.6.1 Elements of Evolutionary Algorithms . . . 29
3.6.2 Convergence and Properties . . . 33
3.7 Neuroevolution . . . 33

4 InTrance **35**
4.1 Development History . . . 35
4.2 InTrance Architecture . . . 36
4.3 Multiphase Framework and Fitness . . . 37
4.3.1 Target State, Proximity, and Deviation . . . 37
4.3.2 Phase Transition Conditions . . . 39
4.3.3 Fitness Evaluation . . . 39
4.4 Evolutionary Algorithm . . . 40
4.4.1 Representation and Initialisation . . . 40
4.4.2 Reproduction, Crossover and Mutation . . . 41
4.5 Artificial Neural Networks . . . 41
4.5.1 Input to the Artificial Neural Networks . . . 41
4.5.2 Output Values . . . 42

5 Low-Thrust Gravity Assist Trajectory Optimisation Implementation **45**
5.1 Single Phase vs. Multiphase Gravity Assist Strategy . . . 45
5.1.1 Single Neurocontroller . . . 46
5.1.2 Multiple Neurocontrollers . . . 47
5.2 Gravity Assist Architecture . . . 47
5.3 Gravity Assist Model . . . 48
5.3.1 Analytical Gravity Assist Model . . . 49
5.4 Initial and Final States of Gravity Assist Phases . . . 49
5.4.1 Velocity Relative to SOI Entry Velocity in GA-Phase . . . 51
5.4.2 Velocity Relative to Gravity Assist Body . . . 51
5.5 Chromosome . . . 52
5.6 Fitness . . . 54
5.6.1 Algorithm . . . 56

6 Verification and Validation **57**
6.1 Analytical Gravity Assist Model . . . 57
6.2 New Horizons Validation Case . . . 59
6.2.1 Validation Data . . . 59
6.2.2 InTrance Input Parameters . . . 60
6.2.3 Optimisation Run . . . 60
6.2.4 External Integration . . . 61
6.2.5 Initial and Final Conditions at Gravity Assist . . . 62
6.3 Applicability of the Implementation . . . 63

7 Mission Analysis **65**
7.1 Low-Thrust New Horizons . . . 65
7.1.1 Simulation & Mission Defining Input Parameters . . . 65
7.1.2 Results . . . 66
7.1.3 Compared to High-Thrust New Horizons Mission . . . 67
7.1.4 Compared to Literature . . . 69
7.1.5 Improvement due to Gravity Assist . . . 70
7.1.6 Influence of Thrust During Gravity Assist . . . 71
7.2 Dawn . . . 72
7.2.1 Simulation & Mission Defining Input Parameters . . . 73
7.2.2 Results . . . 75
7.2.3 Compared to Dawn's Actual Trajectory . . . 76
7.2.4 Improvement due to Gravity Assist . . . 77
7.2.5 Influence of Thrust During Gravity Assist . . . 79
7.2.6 Optimisation of the Three-Phase Scenario – Earth to Ceres . . . 80

8 Conclusion and Recommendations **83**
8.1 Summary . . . 83
8.1.1 Research Motivation and Framework . . . 83
8.1.2 Approach and Results . . . 84
8.2 Analysis of Results and Performance . . . 86
8.2.1 Analysis of Performance, Implementation and Robustness . . . 86
8.2.2 Analysis of Results . . . 87
8.3 Recommendations for Further Work . . . 89
8.3.1 Fundamental Research . . . 89
8.3.2 Application Oriented . . . 90

A Reference Frames **91**
A.1 Inertial Cartesian Reference Frame . . . 91
A.2 Inertial Polar Ecliptic Reference Frame . . . 92
A.3 Orbital Elements . . . 93

B Results Dawn Simulations **97**

C InTrance Input Files **101**
C.1 Low-Thrust New Horizons with Jupiter Gravity Assist . . . 101
C.1.1 General Input Files . . . 101
C.1.2 Phase 1 Input Files . . . 102
C.1.3 Phase 2 Input Files . . . 102
C.2 Dawn with Mars Gravity Assist — Earth to Vesta . . . 103
C.2.1 General Input Files . . . 103
C.2.2 Phase 1 Input Files . . . 104
C.2.3 Phase 2 Input Files . . . 104
C.3 Dawn with Mars Gravity Assist — Vesta to Ceres . . . 105
C.3.1 General Input Files . . . 105

C.3.2 Phase 1 Input Files 106

Bibliography **107**

List of Abbreviations

AI Artificial Intelligence.
ANN Artificial Neural Network.

DC Delta Coding.
DE Differential Evolution.
DGA Diploid Genetic Algorithm.
DP Dynamic Programming.

EA Evolutionary Algorithm.
ENC Evolutionary Neurocontrol.
EoM Equations of Motion.
EP Electric Propulsion.

FB Flyby.
FPDC Floating Point Delta Coding.

GA Gravity Assist.
GP Genetic Programming.
GTOM Global Trajectory Optimisation Method.

InTrance INtelligent TRAjectory optimisation Using NeuroController Evolution.
IVS Input Variable Selection.

KBO Kuiper Belt Object.

LTOM Local Trajectory Optimisation Method.

MDP Markov Decision Process.
MET Mission Elapsed Time.
ML Machine Learning.

NC Neurocontroller.
NEAT NeuroEvolution of Augmenting Topologies.
NEP Nuclear Electric Propulsion.
NN Neural Network.

RDC Real Delta Coding.
REP Radioisotope Electric Propulsion.
RL Reinforcement Learning.
RV Rendezvous.

S/C Spacecraft.
SEP Solarelectric Propulsion.
SOI Sphere of Influence.
SP Selective Pressure.
SSS Search Space Scan.

TWEANN Topology- and Weight Evolving Artificial Neural Network.

List of Figures

2.1 Nominal flight-path of New Horizons and the planetary positions at Jupiter flyby. 8
2.2 Nominal New Horizons trajectory. 9
2.3 Dawn's interplanetary trajectory in the in-plane heliocentric frame. 10
2.4 Dawn's interplanetary trajectory in the heliocentric frame. 11
2.5 Geometry of the hyperbolic Gravity Assist (GA) section. 11
2.6 Vector diagrams of some gravity assist maneuvers. 14
2.7 Geometry of an inclination change of the Spacecraft (S/C)'s orbit due to a GA. 15
2.8 Maximum inclination change due to a GA as a function of the hyperbolic excess velocity. 16
2.9 Powered New Horizons like GA at Jupiter. 18

3.1 Traditional trajectory optimisation using local optimisation methods. 21
3.2 Smart low-thrust trajectory optimisation using a global trajectory optimisation method. 23
3.3 Illustration of a typical biological neuron and artificial neuron. 26
3.4 General form and block-diagram of a computing unit within an Artificial Neural Network (ANN). 27
3.5 Topology of a general feed-forward ANN. 28
3.6 Principle components and reproduction cycle of Evolutionary Algorithms (EAs). 30
3.7 One-at-a-time reproduction with tournament selection. 31
3.8 Some common crossover types. 32
3.9 Illustration of the generation of offspring with the crossover nodes operator. 32
3.10 Convergence of a simple GA. 33
3.11 Mapping of Neurocontroller (NC) parameters on a chromosome $\boldsymbol{\xi}$. 34

4.1 Multi-phase trajectory optimisation using evolutionary neurocontrol. 37
4.2 Composition of chromosome $\boldsymbol{\xi}$ in the multi-phase framework. 38

5.1 Single NC GA-approach within the multi-phase framework. 46
5.2 Multi-NC GA-approach within the multi-phase framework. 47
5.3 Multi-phase low-thrust gravity assist trajectory optimisation within INtelligent TRAjectory optimisation Using NeuroController Evolution (InTrance). 48
5.4 Definition of initial position of a phase directly following a GA in the XY-plane. 51
5.5 Definition of initial state in a phase directly following a GA using the entry velocity of the preceding phase. 52
5.6 Definition of initial state in a phase directly following a GA using the body velocity at Sphere of Influence (SOI) exit. 53
5.7 Proximity functions of the analytical gravity assist phase. 55
5.8 Example geometry to showcase transition conditions between a GA-phase and its succeeding phase. 55

6.1 Trajectory while performing a GA at both Mars (left) and Jupiter (right) for varying entry velocities. 58

6.2 Deviation between analytical GA-model and numerically integrated trajectory. 58
6.3 Heliocentric trajectory from an InTrance run of the validation case. 60
6.4 External RK4(5) integration of InTrance's result of the validation case. 61
6.5 Heliocentric distance and velocity of both InTrance and the externally re-integrated solution. . 62
6.6 Close-up of the gravity assist portion of the validation case in the heliocentric frame. 63

7.1 Heliocentric InTrance optimised New Horizons like low-thrust trajectory. 67
7.2 Close-up of the gravity assist portion of the resulting trajectory from InTrance for the New Horizons like low-thrust mission. 67
7.3 Initial state of the phase departing from Jupiter with velocities optimised relative to the entry velocity. 68
7.4 Heliocentric trajectory, velocity and distance of the actual New Horizons missions and a low-thrust alternative computed with InTrance. 68
7.5 Heliocentric trajectory, velocity and distance of two low-thrust NEP trajectories to Pluto generated with InTrance, one direct transfer and one including a Jupiter GA. 70
7.6 Helio- and planetocentric portions of the New Horizons trajectories computed with the analytical GA-model and numerical GA-model. 72
7.7 Dawn trajectory as optimised by InTrance in the heliocentric frame. 75
7.8 Close-up of the Mars GA portion in the planetocentric frame of the Dawn trajectory computed with InTrance. 76
7.9 Heliocentric trajectory, velocity and distance of the actual Dawn missions and the resulting trajectory from InTrance. 77
7.10 Heliocentric trajectory, velocity and distance of the Dawn trajectory computed with InTrance both with and without a Mars GA. 78
7.11 Helio- and planetocentric portions of both Dawn trajectories computed with the analytical GA-model and numerical GA-model. 79
7.12 Heliocentric trajectory, velocity and distance of the Dawn trajectory computed with InTrance both through a single optimisation of the complete mission and the concatenated simulations. 80
7.13 Close-up of the Mars GA portion in the planetocentric frame for the three-phase optimisation run. 81

A.1 The Inertial Cartesian reference frame $\mathcal{J}$. 91
A.2 The Inertial Polar Ecliptic Reference Frame $\mathcal{P}$. 92
A.3 Geometry of an elliptical orbit. 93
A.4 Geometry of a hyperbolic orbit. 93
A.5 Geometry of the orbital elements w.r.t. the ecliptic. 94

B.1 Three-dimensional view of the Dawn trajectory as computed by InTrance in the heliocentric frame. 97
B.2 Three-dimensional view of the initial state of the second phase in the heliocentric frame of the Dawn trajectory computed with InTrance. 98
B.3 Three-dimensional view of the heliocentric Dawn trajectory as computed by InTrance compared to Dawn's actual trajectory generated through SPICE. 98
B.4 Three-dimensional view of the heliocentric Dawn trajectory as computed by InTrance with a Mars GA and without a GA. 99
B.5 Three-dimensional view of the heliocentric Dawn trajectory as computed by InTrance, both the concatenated results of the two separate simulations and the single optimisation of the complete missions. 99

List of Tables

2.1 Maximum attainable ΔV of the S/C due to a GA at different celestial bodies. 15

2.2 Key GA parameters for the four New Horizons like scenarios under the influence of thrust inside the SOI. 17

4.1 Classification of the mechanism and techniques added by Ohndorf to InTrance. 36

4.2 Currently implemented InTrance objective functions. 40

4.3 NC input parameters. 43

5.1 Simulation parameter encoding onto the chromosome. 53

6.1 Validation data for a low-thrust New Horizons like trajectory. 59

6.2 Mission defining parameters used to generate an optimised New Horizons like low-thrust trajectory similar to the validation date. Values in square brackets indicate ranges. 60

6.3 Results and reference data for the New Horizons like validation simulation. 61

6.4 The SOI exit states computed using the analytical GA model and an external numerical integration plus the actual initial state of the second phase in InTrance. 63

7.1 Mission defining parameters used to generate an optimised New Horizons like low-thrust gravity assist trajectory 66

7.2 Results of the InTrance optimisation run of the New Horizons like low-thrust gravity assist trajectory. 66

7.3 Comparison of low-thrust NEP trajectories to Pluto generated by InTrance and the actual high-thrust New Horizons trajectory. 69

7.4 InTrance's optimised trajectory to Pluto with a GA at Jupiter compared to the validation literature. 69

7.5 Mission defining parameters used to generate a direct Earth-Pluto transfer. 70

7.6 Comparison of low-thrust NEP trajectories to Pluto generated by InTrance, both as a direct transfer and with a GA at Jupiter. 71

7.7 Different mission defining parameters used in the New Horizons numerical GA computation. . 71

7.8 Mission defining parameters used to re-compute the Dawn trajectory from Earth to Vesta. . . . 74

7.9 Mission defining parameters used to re-compute the Dawn trajectory from Vesta to Ceres. . . . 74

7.10 Results of the InTrance optimisation run of the Dawn trajectory including a Mars GA. 75

7.11 Comparison of InTrance results and actual Dawn mission. 78

7.12 Comparison of InTrance results for the Dawn mission, both with and without a Mars GA. 79

7.13 Comparison of Dawn trajectories computed with InTrance, both for a single optimisation of the complete mission and the patched results of the two separate optimisation runs. 81

1 Introduction

The earliest interplanetary space voyages, such as the Venera and Mariner missions, led the spacecraft to relatively close-to-Earth targets such as Venus and Mars. It was quickly realised that interplanetary travel was inherently limited by the propellant mass which could be taken on-board and by what relative velocity, or C_3, a launcher could provide, rendering the outer planets to be out of reach. M.A. Minovitch proposed a solution in 1961 he termed *Gravity Propelled Inteplantary Space Travel* [53], in which he suggested performing close flyby's of planets along the way to the final target, utilising a momentum exchange between the spacecraft and flyby planet, resulting in an increase of velocity relative to a third body such as the Sun. This method is now generally referred to as a *gravity assist* or *swinbg-by*, and although its mechanics were known by astronomers such as Laplace (1749-1827) [12] and used on the Luna 3 spacecraft in 1959 by the Russians [37], its application to interplanetary trajectory design is generally credited to Minovitch [26].

The combination of gravity assists with classical chemical high-thrust propulsion has given rise to many highly successful missions such as the Voyagers, Pioneers, and Cassini-Huygens. More recently the GA technique has successfully been applied in the New Horizons mission to Pluto and the Juno mission to Jupiter. However, the propellant of these missions still took up a large chunk of the mass and ΔV budgets, limiting the amount of payload that could be taken on-board. Gravity assist trajectories are furthermore constrained to very strict launch windows as the planets need to form favourable configurations.

With all major bodies within our Solar System explored from atleast a single flyby, modern interplanetary missions are increasingly becoming more demanding. The focus has shifted to exploring bodies in more detail, for which the spacecraft usually has to enter an orbit around it. Missions are also becoming more intricate by having multiple targets, such as was the case in the Dawn mission, which orbited both Vesta and Ceres. More intricate missions generally require a larger ΔV budget and hence a larger propellant mass, to which an upper limit is set by the maximum attainable characteristic launch energy C_3. As an example, New Horizons was launched with a C_3 of 164 km^2/s^2, which can be seen as a practical upper limit [36].

In order to accommodate the ΔV budget needed for modern interplanetary missions, more and more is relied upon low-thrust propulsion methods. Low-thrust propulsion is characterised by (very) low thrust force levels; long continuous thrust arcs, often lasting months at a time; and a high specific impulse. Low-thrust propulsion can generally be divided in two groups; Electric Propulsion (EP) and internal-reaction mass free propulsion. The former requires expelled mass to be accelerated electrically, for which the power is usually either delivered by photovoltaic-based or nuclearelectric power generation systems. The electric power is used to accelerate the expelled mass with a much higher exit velocity and much lower mass flow than compared to classical chemical high-thrust propulsion. Contrary, internal-reaction mass free propulsion methods do not require any mass to be expelled, and hence do not require any propellant. These methods use the (electro-)magnetic environment to their benefit with methods such as laser propulsion, electrodynamic tethers and solar/magnetic sails.

One of the more prominent applications of low-thrust propulsion can be found in the Dawn mission, in which both Vesta and Ceres are orbited with a single spacecraft, making use of three xenon ion-drives capable of outputting a maximum of 92mN [63]. Dawn was the first to orbit two extraterrestial bodies; previous multi-target missions, such as in the Voyager programme, were limited to flybys. Placing a spacecraft in orbit

around both Vesta and Ceres would not have been possible with conventional chemical high-thrust propulsion, and hence without low-thrust the mission would have had to be carried out by two separate probes. Both high-thrust probes would have been heavier than its low-thrust counterpart; requiring more propellant and a larger launch vehicle. Russell et al. [65] determined that each of the two high-thrust probes would have cost some $750m, compared to a total cost for the low-thrust counterpart of well under $500m, saving NASA over $1bn due to the application of low-thrust propulsion.

Low-thrust has shown to be a viable alternative to conventional chemical high-thrust propulsion, although often requiring (much) longer flight times due to the low applied thrust force and therefore small acceleration. Furthermore, Solarelectric Propulsion (SEP) and solar sails cannot be relied upon for interplanetary mission much beyond Mars' orbit, and propellant mass for Nuclear Electric Propulsion (NEP) systems quickly increases for missions to the outer planets. Therefore, low-thrust is nowadays often combined with gravity assists, which combined can deliver the required ΔV budget for intricate missions and simultaneously decrease transfer time. The Dawn missions was the first to combine low-thrust with a Mars GA, but would have been possible without; the Mars GA was solely added to increase technical margins [63].

On the contrary, BepiColombo is a low-thrust mission actively using multiple GAs to reach its final orbit about Mercury. BepiColombo makes use of an Earth GA to deflect the spacecraft towards Venus, followed by two consecutive Venus GAs to reduce the perihelion to nearly Mercury's distance, using hardly any propellant. A total of six Mercury GAs are then performed to slow down the spacecraft, followed by four more final thrust arcs to reduce the velocity enough to be weakly captured by Mercury. Benkhoff et al. [7] list the reasons for employing EP due to it providing large flexibility to trajectory design within the launcher capability, spacecraft mass and flight duration, at a very low propellant cost. EP furthermore gives flexibility in choosing the trajectory and launch window, thereby providing robustness with regards to mission operations. BepiColombo was launched in 2018 with a launch mass of 4100 kg, of which 2250 kg of payload and about 32% of propellant.

1.1. Research Motivation

Traditional high-thrust trajectory optimisation is relatively straightforward compared to its low-thrust counterpart; thrust is only applied at a few instances and each operation is modelled as an instantaneous burn. These relatively simple trajectories were traditionally designed by a team of astrodynamics experts by hand, followed by simple numerical integration. Contrary, low-thrust can be applied for months at a time, where at each time step the direction and magnitude can be varied, resulting in a much larger problem dimensions. With the inclusion of gravity assists, the optimisation of the trajectory becomes even more difficult. Traditional trajectory optimisation tools heavily rely on astrodynamics expertise to generate an initial guess to a local optimisation scheme. Global optimisation tools exist, but usually require heavy modification for each new mission scenario and are not capable of optimising *any* trajectory. There is a clear need for a *smart* global low-thrust trajectory optimisation tool, capable of optimising low-thrust trajectories from only a broad description of the mission.

One such *smart* global low-thrust trajectory optimisation method, termed InTrance, was developed by Dachwald [17] in 2004. Originally only capable of optimising single-target heliocentric low-thrust trajectories, it was later extended by Ohndorf [59] to allow for optimisation of multi-target and planetocentric low-thrust trajectories. InTrance makes use of a novel method termed evolutionary neurocontrol, tackling the problem from the perspective of reinforcement learning and artificial intelligence. Evolutionary neurocontrol combines biologically inspired Artificial Neural Networks (ANNs) and Evolutionary Algorithms (EAs). The ANN acts as an agent, indicating at each time step whether the thruster should be engaged, with what magnitude, and in what direction. The internal parameters of the ANN and initial conditions of the trajectory are optimised by an EA. The trajectory then results from numerical integration from the initial conditions onward while applying the thrust as indicated by the ANN at each step. InTrance then only requires a broad description of the mission, such as windows for the launch date, launch C_3 and desired arrival dates plus target bodies and target orbits (captured, flyby, rendezvous, etc.), and can find (near-)global optimal trajectories without relying on astrodynamics expertise or an initial guess.

With low-thrust missions becoming more intricate, often relying on one or multiple GAs, the need for a *smart* low-thrust optimisation tool is extended to the need for a *smart* low-thrust *gravity assist* trajectory optimisation tool. It was the goal of this work to develop such a tool, which was found in extending InTrance with GA capabilities. GAs have been implemented as intermediate targets, allowing a NC to learn the benefits of

performing a GA while another NC is trained to reach its next target. The EA then has to find both favourable entry and exit conditions into and out of the SOI such that the next target is reached optimally, while generating a steering strategy (internal ANN parameters) which steer the spacecraft to perform that GA and reach its next target.

The extended version of InTrance can then be used for fundamental GA research such as determining the efficiency increase due to the inclusion of GAs with respect to transfer time and propellant mass and to investigate the effects of a powered vs. unpowered GA. The extended version will furthermore be used to (re-)calculate optimal trajectories for two prominent missions; a low-thrust adaptation of New Horizons, performing a GA at Jupiter; and the previously mentioned Dawn mission, performing a GA at Mars.

1.2. Research Framework Definition

This research aims to supply the needed *smart* low-thrust gravity assist trajectory optimisation software, tackling the problem from the novel standpoint of Reinforcement Learning (RL). The software tool InTrance plays a vital role in the remainder of this work and will be extended to allow for the optimisation of both time- and propellent-optimal low-thrust gravity assist trajectories. In order to fulfil this aim, a set of research questions and objectives were devised, which are described below.

1.2.1. Research Questions

Providing the answers to the set of research questions will give the required knowledge to achieve the objectives of this research. In aiding to develop a *smart* low-thrust gravity assist optimisation tool, the main research questions is defined as:

Research Question: *To what extent can neurocontrol be used to extend the smart low-thrust trajectory optimisation tool InTrance to optimise low-thrust gravity assist trajectories either time- or propellant optimal?*

This main research question is further broken down into sub-questions:

Sub-question 1: How can the optimisation of gravity assists be implemented in InTrance?

a. Should a gravity assist sequence be optimised by InTrance or provided externally?
b. Should a gravity assist be optimised by a neurocontroller or by a dedicated (local) optimisation scheme?
c. Should a gravity assist occur mid-phase or be a target in itself and therefore have a separate neurocontroller?
d. How can a neurocontroller be trained to perform a gravity assist?
e. To what extent can the gravity assist optimisation problem be simplified by using an analytical formulation and how does that impact the performance?
f. How should (sub-)fitness functions be defined such that they drive a neurocontroller to perform a gravity assist?

Sub-question 2: To what extent can gravity assists be optimised with such a method and how do they affect the overall trajectory?

a. Can gravity assists be performed both at bodies with small and bodies with large spheres of influence?
b. What is the effect of powered vs. unpowered gravity assists?
c. What is the efficiency increase from including a gravity assist?
d. Is it possible to optimise trajectories in which multiple gravity assists are performed?

1.2.2. Research Objectives

The main aim of the research is to supply in the need for a *smart* low-thrust gravity assist optimisation tool, which was described in Section 1.1. This main objective is firstly broken down into smaller sub-objectives which need to be achieved first, resulting in the achievement of the overall objective. Once a validated

smart low-thrust gravity assist optimisation tool has been developed, it will be used to achieve a set of sub-objectives related to preliminary trajectory design.

The overarching objective is formally defined as:

Research Objective: *To develop and use a smart low-thrust trajectory optimisation tool using evolutionary neurocontrol by extending InTrance, capable of robustly optimising preliminary low-thrust gravity assist trajectories in both transfer time and propellant usage which should only be dependent on basic mission defining inputs.*

Which can be further broken down into the following sub-objectives:

Sub-objective 1: Develop a gravity assist implementation strategy.

Sub-objective 2: Formulate (sub-)fitness functions that drive optimal gravity assists.

Sub-objective 3: Assess the applicability of an analytical formulation of the gravity assist.

Sub-objective 4: Validate the implementation by comparison with reference cases from literature.

Once such a *smart* low-thrust gravity assist optimisation method has been developed it can be used to achieve the following objectives in order of importance:

Sub-objective 4: Use the extended version of InTrance to perform a gravity assist at a body with a large sphere of influence such as Jupiter.

Sub-objective 5: Optimise a low-thrust version of the New Horizons mission to Pluto using a Jupiter gravity assist

Sub-objective 6: Use the extended version of InTrance to perform a gravity assist at a body with a small sphere of influence such as Mars.

Sub-objective 7: Optimise a Dawn-like mission which rendezvouses with Vesta and Ceres, making use of a Mars gravity assist.

Sub-objective 8: Assess the performance improvement from the inclusion of a gravity assist.

Sub-objective 9: Asses the impact of a powered gravity assist.

Sub-objective 10: Asses whether multi-GA trajectories can be optimised with the developed tool.

The focus of this thesis is on the development and validation of such a *smart* low-thrust gravity assist optimisation tool, capable of answering the above research questions and achieving the research objectives. Within its intended use as a preliminary mission analysis tool, high-fidelity solutions are not required. The developed tool should however be able to provide a (near-)global optimal solution, which can be an input to existing local optimisation schemes in the detailed design phase.

1.3. Report Structure

Prior to discussing the implementation of gravity assists in InTrance to develop a *smart* global low-thrust gravity assist trajectory optimisation tool, certain concepts have to be introduced. The dynamics and effects of a gravity assist are firstly described in Chapter 2. Analytical formulations are derived for the governing parameters of a gravity assists, together with minimum and maximum values for the deflection angle, velocity variation, and inclination change. The focus then shifts to the powered gravity assist, in which it is allowed to apply thrust while inside the SOI. Four different thrust scenarios will be presented, showcasing the effect of thrust during the GA on New Horizon's nominal trajectory.

Chapter 3 formally defines the low-thrust trajectory optimisation problem, firstly transforming it to the realm of optimal control problems. With this definition, a brief description of traditional low-thrust trajectory optimisation methods is given, indicating their main drawbacks which then leads to the definition of a *smart* low-thrust trajectory optimisation method. Since this work solves the problem from the perspective of reinforcement learning, a brief introduction in artificial intelligence and reinforcement learning is presented, followed by a formal definition of the Markov Decision Process (MDP) and an introduction on how to solve

them. The low-thrust trajectory optimisation problem can then formally be transformed to the perspective of reinforcement learning, after which the two main components of evolutionary neurocontrol –the ANN and EA– are introduced.

With an understanding of the dynamics and governing methods used within evolutionary neurocontrol, the computing software InTrance is described in Section 4. This chapter starts with a brief overview of the development history, followed by a description of the overall architecture and components, with a main focus on the definitions of fitness and optimality.

The implementation of gravity assists within InTrance is described in Chapter 5, starting with a rationale and description of the employed neurocontroller strategy. The architecture of the implemented strategy in the larger architecture of InTrance is then discussed, followed by a description of its components, the initial states and the fitness evaluation.

The implementation is verified and validated in Chapter 6. The focus lies on the validation of the optimisation of gravity assists, as verification of most components, such as the correct working of the evolutionary algorithm, has been performed by Dachwald [17] and Ohndorf [59] in their versions of InTrance. The applicability of the analytical gravity assist model is is a new component, and hence verified by comparison with a numerically integrated trajectory. From this effort, the analytical GA-model is deemed accurate enough for non-extreme gravity assists within the intended use of InTrance as a preliminary design tool. The overall implementation is then validated by optimising a low-thrust New Horizon's like mission constrained to similar conditions as found in literature, and then comparing the resulting trajectory with those found in literature. The physical correctness of the resulting trajectory is furthermore verified by an external numerical integration from the initial conditions at launch while applying the thrust history as found by InTrance.

Chapter 7 demonstrates InTrance's capabilities as a preliminary design low-thrust gravity assists trajectory optimisation tool through the optimisation of the trajectories of both a low-thrust New Horizons like mission to Pluto and the re-optimisation of the Dawn mission to Ceres and Vesta. The resulting trajectories will be compared to the actual trajectories of those missions retrieved through SPICE, to results found in literature, to a trajectory excluding the gravity assists and to a trajectory in which a powered GA is allowed. The last two comparisons allow to quantify the efficiency gain due to a GA and the efficiency gain due to a powered GA, respectively.

Chapter 8 presents a summary of this report and an analysis on the implemented method and obtained results. It serves as the conclusions to this work, highlighting the answers to the research questions and achievement of objectives. It lastly provides recommendations for future research.

The appendices lastly contain a description of the reference frames used within this work, enlarged three-dimensional heliocentric plots of the generated Dawn trajectories, and selected input parameters used in the InTrance simulations.

2

Gravity Assists

It was quickly realised that chemical propulsion was inherently limited in the amount of fuel that could be taken onboard and therefore limited in the amount of orbital energy (ΔV) that could be produced, rendering exploration of the outer planets next to impossible. M.A. Minovitch proposed a new method in 1961 he termed *Gravity Propelled Interplanetary Space Travel* [53]; making use of conventional chemical propulsion and combining it with close flyby's of the planets, resulting in a momentum exchange and an increase in velocity of the spacecraft, a method now known as *gravity assists* (GA's) or *swing-by's*.

The first S/C that exploited the gravity field of a celestial body to increase its velocity was Pioneer 11, launched in 1973, performing a gravity assist maneuver at Jupiter on its way to Saturn. Since then, many missions have used the concept, among others, the highly-successful Galileo mission to Jupiter launched in 1989, the Cassini mission to Saturn launched in 1997 and the New Horizons mission to Pluto launched in 2006. These missions primarily used gravity assists to increase their orbital energy, as a direct transfer to their target planet would not have been possible with conventional high-thrust chemical propulsion. Gravity assists can also be used to change the inclination, as was achieved in the Ulysses mission to the Sun, launched in 1990. Ulysses used a gravity assist at Jupiter to achieve a 78 degree inclination change, relative to the ecliptic, enabling exploration of the South and North polar regions of the Sun. Without Jupiter's gravity assists, the high out-of-ecliptic inclination would not have been possible [77].

A gravity assist can be regarded as an elastic collision of a spacecraft with a celestial body. It is analogous to the elastic collision between a baseball and a bat. With respect to the bat, the baseball has the same velocity when coming from the pitcher as when leaving the park, however, relative to any other observer, the baseball has increased its speed by picking up the velocity of the bat in the interaction. A gravity assist works in much the same way, in the frame of the GA-body, the incoming and outgoing hyperbolic excess velocities $\mathbf{v}_\infty$ are the same, and a gravity assist then only provides a reorientation of the $\mathbf{v}_\infty$ vector. However, in the frame of an outside observer, such as the Sun, the S/C has gained some additional velocity due to its interaction with the gravity field of the GA-body. When a S/C passes behind a GA-body, the velocity relative to some outside observer will increase; the GA-body then 'drags' the spacecraft along the planet's gravitational field. When a S/C passes in front of a GA-body, the velocity relative to some outside observer will decrease; as the GA-body is moving towards the point of closest approach, the S/C travels further into the GA-body's gravitational field, and loses some energy in getting out [47].

Prior to introducing the dynamics and geometry of a GA in Section 2.2, two GA performing missions – New Horizons and Dawn– are introduced in Section 2.1. These two mission will be used throughout this work and are chosen because of their distinct GA types. Section 2.3 describes the effects of unpowered gravity assists and derives equations for the maximum generated ΔV, inclination variation and deflection angle. The effects of thrusting while performing a GA are explored in Section 2.4 through a New Horizons like simulation.

2.1. Missions with Gravity Assists

Two missions will be central in this work; the New Horizons mission to Pluto and the double-asteroid rendezvous mission Dawn to Ceres and Vesta. The trajectories of both will be re-computed with the developed

method as a low-thrust mission in Chapter 7 and will serve as examples to showcase effects of GAs in the remainder of this work.

2.1.1. New Horizons

The New Horizons mission was NASA's first mission in its flagship New Frontiers Programme, the medium class of principal investigator-led projects, and had a total cost of approximately $700 million over the period 2001-2016. New Horizons was selected for development in November 2001 following a competitive selection after a NASA Announcement of Opportunity; it was the first mission to the Pluto system and intended to complete the reconnaissance of the then prevailing classical planets. New Horizons achieved the highest launch ΔV relative to Earth of 16.2 km/s, setting the S/C on a direct solar escape trajectory.

Mission Description and Objectives

While S/C have visited each of the terrestrial and Jovian planets, the high-thrust New Horizons mission was the first to explore icy dwarf planets that dominate the outer portion of our solar system. Its nominal mission would direct New Horizons past Jupiter to perform a flyby of the Pluto-Charon system. If the mission would be extended, the S/C could continue to investigate Kuiper Belt Objects (KBOs). The orbits of the planets and Pluto, together with the nominal flight-path of New Horizons are shown in Figure 2.1.

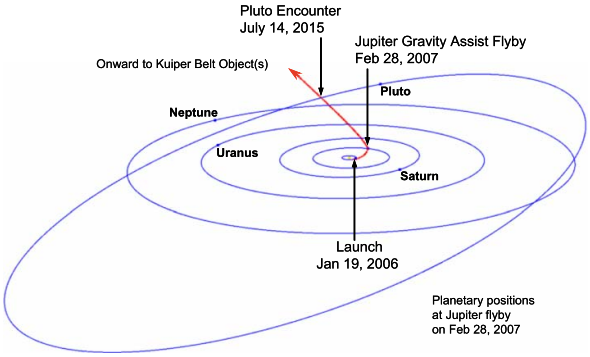


Figure 2.1: Nominal flight-path of New Horizons and the planetary positions at Jupiter flyby [36].

Pluto and Charon form a double planet system, with the barycentre of the system exterior to Pluto. Despite their proximity, the two bodies are vastly different. Pluto is red in colour, has a surface covered in volatiles which supports a seasonal atmosphere, and has a large variable albedo. Contrary, Charon is neutral in colour, has a water-ice dominated surface and no atmosphere has been observed [85]. At the time of launch, little was known about the other two satellites in the system, Nix and Hydra, which where discovered only a year prior to launch.

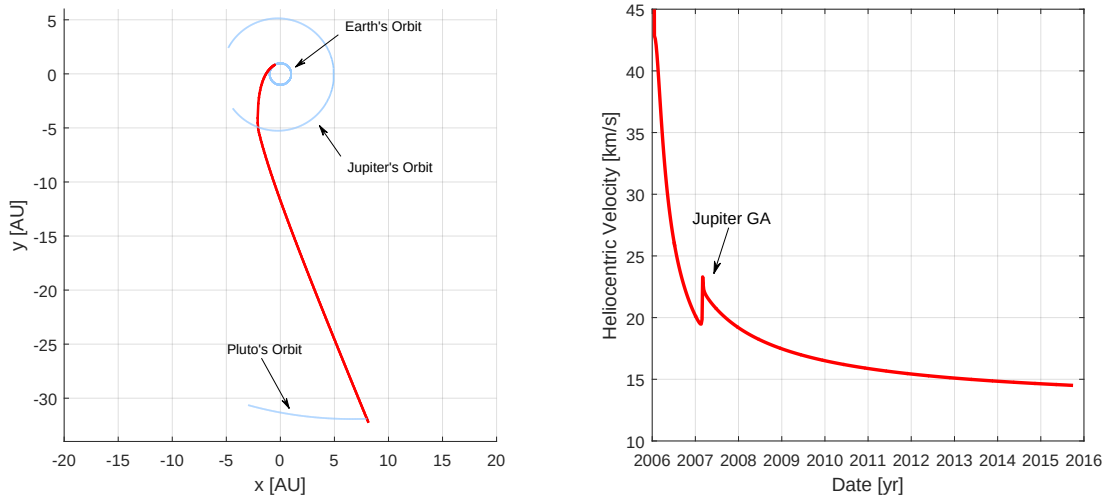
The 478 kg (wet mass) New Horizons spacecraft carries seven scientific instruments, including imagers, spectrometers, radio science, a plasma and particles suite, and a dust counter. New Horizons was designed to broaden the understanding of the Pluto system, such as its origin, processes operating on the surface, the volatile transport cycle, and the energetics and chemistry of the atmosphere [85]. The observations will furthermore extend the knowledge of other bodies which are; formed by giant impact (Earth-moon), formed in the outer solar system (comets and icy dwarf planets), with surfaces in vapour-pressure equilibrium (Triton and Mars), and other bodies with N_2:CH_4 atmospheres (Titan, Triton, early Earth). New Horizons will study the surface composition of Pluto, take measurements of its atmosphere, and characterise its topography, surface temperatures and solar wind interaction.

Trajectory

New Horizons was launched on January 19, 2006 from Cape Canaveral on a Jupiter gravity assist trajectory toward the Pluto system with a maximum launch C_3 of 164 km^2/s^2 on an Atlas V. It performed a GA at Jupiter with a closest approach distance of 32.2 $R_{\text{♃}}$ (2.3 million km) on February 28, 2007. After 9.5 years, New Horizons encounters Pluto on July 14, 2015, at a distance of 5.2 $R_{\text{♇}}$ (12.4 thousand km) during its 9-month

observation window of the Pluto system. The mission was designed as a ballistic flight from Earth to Pluto, and all energy and encounter geometry were computed and specified in the launch targets [36].

The heliocentric trajectory followed by New Horizons is shown in Figure 2.2(a), which is computed through NASA's SPICE [1] toolkit with official New Horizons kernels.[1] Figure 2.2(b) shows the heliocentric velocity over time, clearly indicating the effect of its Jupiter GA. Due to the planetary geometry (see Figure 2.1), there was hardly any need for an out-of-plane manoeuvre and most of the GA's effect is concentrated in increasing the heliocentric velocity and deflecting the in-plane direction.



(a) Heliocentric trajectory

(b) Velocity over time

Figure 2.2: Nominal New Horizons trajectory.

2.1.2. Dawn

NASA's Dawn mission is the first to orbit a main belt asteroid and the first to orbit two extraterrestrial bodies. With the Dawn mission being part of the small mission programme Discovery, it was quickly realised that within the associated cost cap only flyby's of Ceres and Vesta could be performed. However, much more can be learned with an orbiter than with a flyby mission, which rendered the Dawn team to investigate alternatives. They found a viable alternative in using low-thrust propulsion in the form of ion engines, which had been demonstrated with the Deep Space 1 mission in the late '90s. The use of low-thrust propulsion allowed to orbit both asteroids with a single craft well within the budget of Discovery class missions [65].

Mission Description and Objectives

Ceres, Vesta and Pallas are the three largest minor planets, and are intact survivors from the earliest period of the formation of the solar system. Visiting these bodies would greatly increase our understanding of the conditions and processes acting at the solar system's earliest epoch. Pallas' high inclination makes a mission difficult, even with low-thrust propulsion, so the Dawn mission focusses on orbiting both Ceres and Vesta.

The 747 kg (dry mass, +425 kg propellant) Dawn S/C carries three scientific instruments; a pair of framing cameras, a visible and infrared mapping spectrometer, and a gamma ray and neutron detector. Gravitational field measurements are furthermore possible through its radiometric tracking system [65]. Dawn's objectives are designed to provide insight into the conditions and processes that have acted upon Ceres and Vesta from their formation to the present, its level 1 requirements are described by Rayman [63] as:

1. Determination of the bulk density of Vesta and Ceres to better than 1%;
2. Determination of the spin axis orientation of Vesta and Ceres to better than 0.5%;
3. Determination of the gravity fields of Vesta and Ceres;
4. Optical mapping of the surface of Vesta and Ceres;

[1] New Horizons kernels available at: `https://naif.jpl.nasa.gov/naif/data_archived.html`

5. Creation of a topographical map of the surface of Vesta and Ceres;
6. Measure and map the abundances of major rock-forming elements and the elements H, K, Th, and U of the surface of Vesta and Ceres;
7. Obtain spectral frames of the surfaces of Vesta (≥ 10 000 frames) and Ceres (≥ 8 000 frames).

Trajectory

Dawn was launched on September 27, 2007 from Cape Canaveral toward its first target Mars on a Delta 2925H. Although direct trajectories towards Vesta are possible, it was chosen to include a GA at Mars to increase technical margins [63]. The heliocentric in-plane trajectory is shown in Figure 2.3, in which the portions indicated in black denote when the spacecraft is not thrusting. Comparing this trajectory to the one of New Horizons (Figure 2.2), it is clear they are distinctly different. The Dawn case makes use of characteristic long low-thrust arcs to slowly increase its semi-major axis over multiple rotations about the Sun, whereas New Horizons was launched directly into a ballistic trajectory to Jupiter.

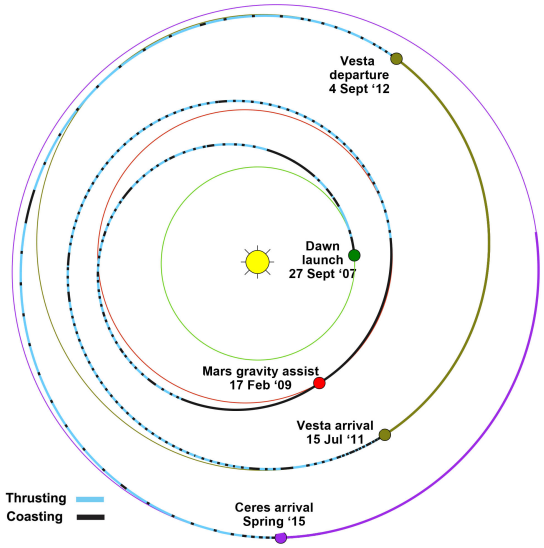


Figure 2.3: Dawn's interplanetary trajectory in the in-plane heliocentric frame. [11]

Dawn performed its gravity assist at Mars in February 2009 at a closest approach distance of $1.15R_{♂}$ (3.8 thousand km). Some 2.5 years later Dawn arrives at Vesta and enters an orbit around it, departing towards Ceres some 1.5 years later. Dawn arrived in the spring of 2015 at its final nominal mission target Ceres.

The heliocentric trajectory is shown from a different perspective in Figure 2.4, generated with SPICE, indicating the effect of the Mars GA is mostly concentrated in an out of plane manoeuvre; altering the inclination to be more in line with those of Ceres and Vesta. This is again contrary to the New Horizons mission, which made use of a GA to predominantly increase its heliocentric velocity.

2.2. Dynamics and Geometry

The dynamics of the many-body problem can be used to describe the motion of a S/C while performing a gravity assist, both the version that is relative to the GA-body or relative to an inertial reference frame (barycenter), with the gravitational attraction of the GA-body modelled as an additional perturbing acceleration, as was described in the Literature Review [43] performed prior to this work. However, in the fashion of the *patched conics approximation*, an unpowered GA is often modelled as a two-body problem from the instance of entering until exiting the SOI. In that case, the S/C follows a hyperbolic trajectory in a non-rotating inertial frame centred at the GA-body, for which analytical solutions exist. The equations relating to conic sections used within this chapter which are not derived are from Wakker [76], unless stated otherwise.

The geometry of an unpowered and unperturbed GA is shown in Figure 2.5, in which v_∞^+ and v_∞^- are respectively the incoming and outgoing hyperbolic excess velocities, θ the true anomaly, a the semi-major axis, e the

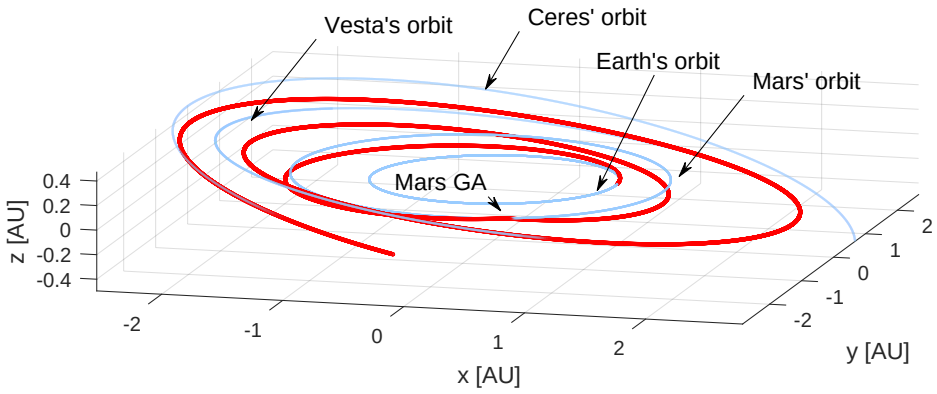


Figure 2.4: Dawn's interplanetary trajectory in the heliocentric frame.

eccentricity, b the semi-minor axis, r_p the point of closest approach and δ the turn or deflection angle.

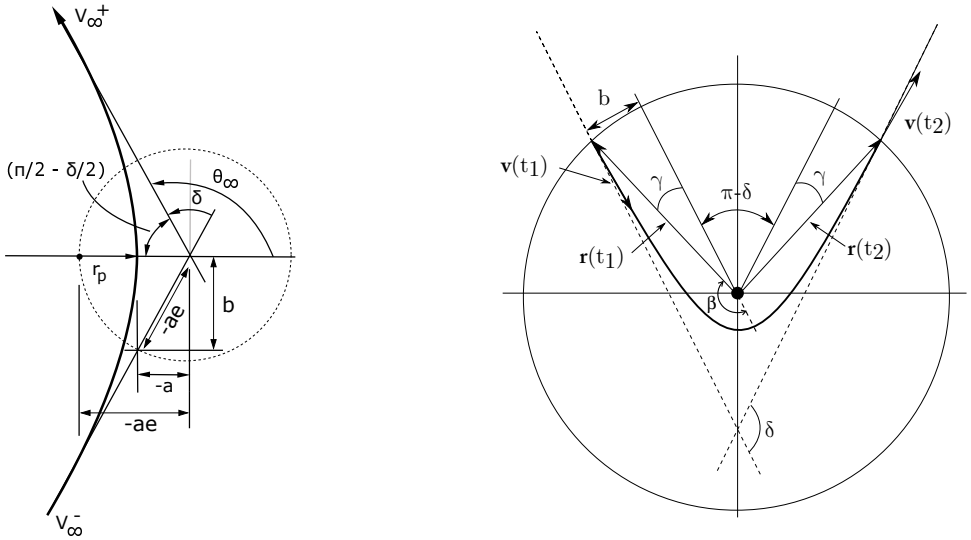


Figure 2.5: Geometry of the hyperbolic GA section. [5]

Realising that a GA results in a rotation of the $\mathbf{v}_\infty$ vector around the angular momentum axis in a non-rotating inertial reference frame centred at the GA-body, the exit state out of the SOI can be determined from the entry state into the SOI. Firstly, the position and velocity of the S/C with respect to the GA-body at epoch t ($\mathbf{r}(t)$, $\mathbf{v}(t)$) can be computed from the heliocentric states of the GA-body ($\mathbf{R}_b(t), \mathbf{V}_b(t)$) and S/C ($\mathbf{R}(t), \mathbf{V}(t)$) as

$$\mathbf{r}(t) = \mathbf{R}(t) - \mathbf{R}_b(t) \tag{2.1}$$

$$\mathbf{v}(t) = \mathbf{V}(t) - \mathbf{V}_b(t) \tag{2.2}$$

One of the most profound effects of GAs is the rotation of the position and velocity vectors around the angular momentum axis $\mathbf{c} = (c_1, c_2, c_3)^T = \mathbf{r}(t) \times \mathbf{v}(t)$. The corresponding rotation matrix for a rotation of angle α is given by [5]

$$R(\alpha) = \begin{bmatrix} \cos\alpha & \frac{c_3}{c}\sin\alpha & -\frac{c_2}{c}\sin\alpha \\ -\frac{c_3}{c}\sin\alpha & \cos\alpha & \frac{c_1}{c}\sin\alpha \\ \frac{c_2}{c}\sin\alpha & -\frac{c_1}{c}\sin\alpha & \cos\alpha \end{bmatrix}. \tag{2.3}$$

The rotation angle of the position vector α is comprised of the angles δ and γ, see Figure 2.5, and is given by

$$\alpha = \pi - \delta + 2\gamma. \tag{2.4}$$

The velocity vector is rotated over the deflection angle δ, as shown in Figure 2.5. The velocity and position vector of the S/C with respect to the GA-body at the end of the GA (epoch t_2), as a function of the velocity and position at the beginning of the GA (epoch t_1), are then given by

$$\mathbf{r}(t_2) = R(\alpha)\mathbf{r}(t_1) = R(\pi - \delta + 2\gamma)\mathbf{r}(t_1), \tag{2.5}$$

$$\mathbf{v}(t_2) = R(\delta)\mathbf{v}(t_1), \tag{2.6}$$

and converted to heliocentric frame as

$$\mathbf{R}(t_2) = \mathbf{R}_b(t_2) + R(\pi - \delta + 2\gamma)\mathbf{r}(t_1), \tag{2.7}$$

$$\mathbf{V}(t_2) = \mathbf{V}_b(t_2) + R(\pi - \delta)\mathbf{v}(t_1). \tag{2.8}$$

Rotation Angles γ and δ

The incoming position vector $\boldsymbol{r}$ is rotated over an angle α through equation 2.4, for which the determination of the intermediate angles δ and γ is described below. The deflection angle can be computed by firstly considering the true anomaly of the incoming asymptotes θ_∞ (Figure 2.5), which is found by letting r tend to infinity as

$$\cos\theta_\infty = \lim_{r\to\infty}\left\{\frac{1}{e}\left[\frac{a\left(1-e^2\right)}{r} - 1\right]\right\} = -\frac{1}{e}, \tag{2.9}$$

which can be rewritten as

$$\theta_\infty = \cos^{-1}\left(-\frac{1}{e}\right). \tag{2.10}$$

From the geometry in Figure 2.5, the true anomaly of the asymptotes can also be written as

$$\theta_\infty = \frac{\pi}{2} + \frac{\delta}{2}, \tag{2.11}$$

such that after substituting equation 2.11 in equation 2.10, the deflection angle is found as

$$\frac{1}{e} = \sin\frac{\delta}{2} \tag{2.12}$$

$$\Rightarrow \delta = 2\sin^{-1}\left(\frac{1}{e}\right). \tag{2.13}$$

To evaluate this function, an equation for the eccentricity is needed. The semi-major axis for any hyperbola can be expressed as

$$b = a\sqrt{1-e^2}, \tag{2.14}$$

re-arranging and substituting $a = -\mu/v_\infty^2$ gives

$$e = \sqrt{1 + b^2\frac{v_\infty^4}{\mu^2}}. \tag{2.15}$$

Lastly, from the geometry in Figure 2.5 and setting $|\mathbf{r}(t_1)| = |\mathbf{r}(t_2)| = R_{\text{SOI}}$, the following equations can be derived:

$$\sin\gamma = \frac{b}{R_{\text{SOI}}} \qquad ; \qquad b = \frac{\mathbf{r}(t_1)\wedge\mathbf{v}(t_1)}{v(t_1)}, \tag{2.16}$$

in which $b = |\mathbf{B}|$ is the B-plane aiming point distance, also known as collision parameter or target point.

Equation 2.15 is then used to evaluate the eccentricity with $r = R_{\text{SOI}}$. The eccentricity can then be used to calculate the deflection angle with equation 2.13. The deflection angle, together with equation 2.16, gives all angles necessary to determine the rotation angle α through equation 2.4.

Flight Time within SOI

The time spent within the SOI can be determined from Kepler's equation as

$$n(t - T_p) = e_h \sinh H - H, \tag{2.17}$$

where $n^2 a^3 = \mu$ and T_p is the epoch at periapsis. Using the symmetry of the hyperbolic orbit, $t = T_p - \Delta t/2$ at the entry point, the time spent within the SOI is given by

$$\Delta t = -2\sqrt{\frac{a^3}{\mu}} \left(e \sinh H - H\right), \tag{2.18}$$

in which the hyperbolic anomaly H is given by

$$\cosh H = \left(\frac{r}{a} - 1\right)\frac{1}{e}, \tag{2.19}$$

where $r = R_{\mathrm{SOI}}$.

2.3. Effects of Gravity Assists

GAs are mostly used to increase the magnitude and/or alter the direction of the heliocentric S/C velocity vector, or to change the inclination. This section shall briefly describe some of the effects caused by a GA and derive maximum values for some of them. Equations within this section are either derived from basic equations for conical sections given by Wakker [76] or stem from Barrabes et al. [5].

2.3.1. Deflection Angle

The deflection angle is one of the major characteristics of a GA and will determine the extent of the achieved ΔV and direction of the S/C after the GA. The minimum deflection angle of 0° occurs when there is no attraction from the GA-body ($\mu = 0$ or $r \to \infty$) or when the incoming velocity tends to infinity. The maximum deflection angle is found by considering the eccentricity, which solely determines the deflection angle according to equation 2.13. The eccentricity is a function of the incoming hyperbolic velocity v_∞^2 and closest approach distance r_p as

$$r_p = a(1 - e) \tag{2.20}$$

$$\Rightarrow \quad e = 1 + r_p a = 1 + r_p \frac{V_\infty^2}{\mu}. \tag{2.21}$$

Substituting this formulation of the eccentricity in equation 2.13, gives the deflection angle as

$$\delta = 2\sin^{-1}\left(\frac{1}{1 + r_p v_\infty^2/\mu}\right), \tag{2.22}$$

which shows that performing a GA with a lower closest approach results in a higher deflection angle. The limit case occurs when the closest approach distance is equal to the radius of the GA-body R_b, such that

$$\delta_{max} = \sin^{-1}\left(\frac{1}{1 + R_b v_\infty^2/\mu}\right). \tag{2.23}$$

2.3.2. Velocity Variation

Besides deflecting the trajectory, a GA usually affects the velocity of the S/C. Gravity assists can supply both a positive and negative ΔV, depending on whether the GA-body is approached from the front or behind. The former is useful in reaching targets which are further away with less propellant or simply to get there faster, whereas the latter is useful in slowing a S/C down. Multiple GAs of Jupiter's moons are for instance used to slow down a S/C enough to enter into a Europa orbit without requiring excessive propellant use [74].

Not all bodies within the solar system are good candidates for a gravity assists, which will be shown by deriving an equation for the maximum velocity variation, which is shown to be a function of GA-body properties

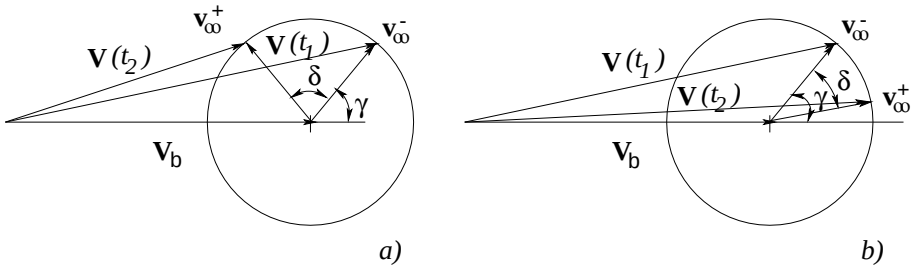


Figure 2.6: Vector diagrams of some gravity assist maneuvers [5].

only. First off, consider the vector diagrams in Figure 2.6, where $\mathbf{V}(t_1)$ is the incoming heliocentric velocity of the S/C towards the GA-body, $\mathbf{V}(t_2)$ is the outgoing heliocentric velocity after the GA, $\mathbf{V}_b$ is the heliocentric velocity of the GA-body, and $|\mathbf{v}_\infty^-| = |\mathbf{v}_\infty^+| = v_\infty$ is both the incoming and outgoing hyperbolic excess velocity of the S/C.

When the S/C passes in front of the GA-body, the angles γ and δ add up, resulting in a decrease of the heliocentric velocity of the S/C, see Figure 2.6a. When the S/C passes behind the GA-body, the GA will produce an increase in heliocentric velocity of the S/C as shown in Figure 2.6b. The geometry furthermore shows that the minimum outgoing velocity occurs when $\delta + \gamma = \pi$. Similarly, the maximum value is attained when $\gamma - \delta = 0$. In both cases, the S/C's heliocentric velocity after the GA will be collinear to the GA-body's velocity vector.

From the geometry in Figure 2.6, the velocity increment due to the GA ($\Delta\mathbf{V} = \mathbf{V}(t_2) - \mathbf{V}(t_1)$) is described by

$$\Delta V = |\Delta\mathbf{V}| = 2v_\infty \sin\frac{\delta}{2}, \tag{2.24}$$

substituting equation 2.22 and re-writing gives

$$\Delta V = \frac{2v_\infty}{1 + r_p v_\infty^2/\mu}. \tag{2.25}$$

This result shows that the maximum ΔV for a given v_∞ is attained when the closest approach distance r_p is at its minimum, i.e., equal to the radius of the GA-body R_b. The v_∞ that will give the maximum velocity variation can be computed from

$$\left.\frac{\partial \Delta V}{\partial v_\infty}\right|_{r_p = R_b} = 0, \tag{2.26}$$

so that the maximum ΔV is found at a value of

$$v_{\infty,\text{opt}} = \left(\frac{\mu}{R_b}\right)^{1/2}, \tag{2.27}$$

that is, the maximum ΔV is attained when the magnitude of the hyperbolic excess velocity is equal to the circular velocity of a S/C in a circular orbit around the GA-body with radius $r = R_b$.

Substituting the result of equation 2.27 in equation 2.25 with $r_p = R_b$ gives

$$\Delta V_{\max} = \frac{2v_{\infty,\text{opt}}}{1 + R_b/\mu \cdot v_{\infty,\text{opt}}^2} = \frac{2v_{\infty,\text{opt}}}{1 + R_b/\mu \cdot \frac{\mu}{R_b}} = \frac{2v_{\infty,\text{opt}}}{1+1} = v_{\infty,\text{opt}} \tag{2.28}$$

$$= \left(\frac{\mu}{R_b}\right)^{1/2}. \tag{2.29}$$

Table 2.1 shows the maximum attainable ΔV for a S/C performing a GA at the major planets, Sun and Moon, determined from equation 2.29, with planetary data from [46]. Jupiter has by far the highest potential velocity gain of the planets, which partly explains why many missions to the outer planets and beyond –such as New Horizons and the Voyagers– have performed GA's at Jupiter.

Table 2.1: Maximum attainable ΔV of the S/C due to a GA at different celestial bodies.

Body	r_b [km]	μ [km^3/s^2]	$v_{\infty,\text{opt}} = \Delta V_{\max}$ [km/s]
Sun	$6.96 \cdot 10^5$	$132712.4 \cdot 10^6$	436.67
Mercury	2440	22032	3.00
Venus	6051.8	324859	7.32
Earth	6371	398600	7.91
Moon	1737	4905	1.68
Mars	3389.9	42828	3.55
Jupiter	71492	$126686.5 \cdot 10^3$	42.10
Saturn	60268	$379311.9 \cdot 10^2$	25.09
Uranus	25559	$579393.9 \cdot 10^1$	15.06
Neptune	24766	$683652.9 \cdot 10^1$	16.62

2.3.3. Inclination Variation

Gravity assists can also be used to significantly alter the inclination of the trajectory. This is for instance used to reach an orbit over the poles of the Sun, as was the case in the Ulysses mission. When the S/C approaches the GA-body from within the body's orbital plane, the hyperbolic excess velocity vector can be rotated to any location on a sphere, as shown in Figure 2.7.

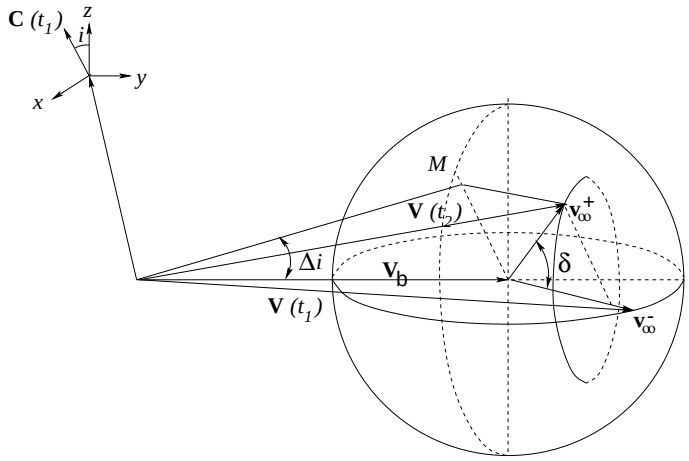


Figure 2.7: Geometry of an inclination change of the S/C's orbit due to a GA [5].

From this geometry, it follows that the change in inclination Δi as a result of a GA is described by:

$$\sin \Delta i = \frac{v_\infty \sin \delta}{V_b}, \tag{2.30}$$

in which, from Section 2.3.1, the deflection angle δ is given by

$$\sin \frac{\delta}{2} = \frac{1}{1 + r_p v_\infty^2 / \mu}. \tag{2.31}$$

It can then be derived that for a fixed value of v_∞, the maximum increment in inclination is given by

$$\sin \Delta i = \frac{v_\infty}{V_b}, \tag{2.32}$$

which is plotted in Figure 2.8 for the planets and Pluto. The potentialities in altering the inclination of the terrestrial planets is much lower than those of the outer planets. The inner planets can still be used to generate large inclination changes, but will require multiple GAs, whereas a single pass of one of the outer planets usually suffices.

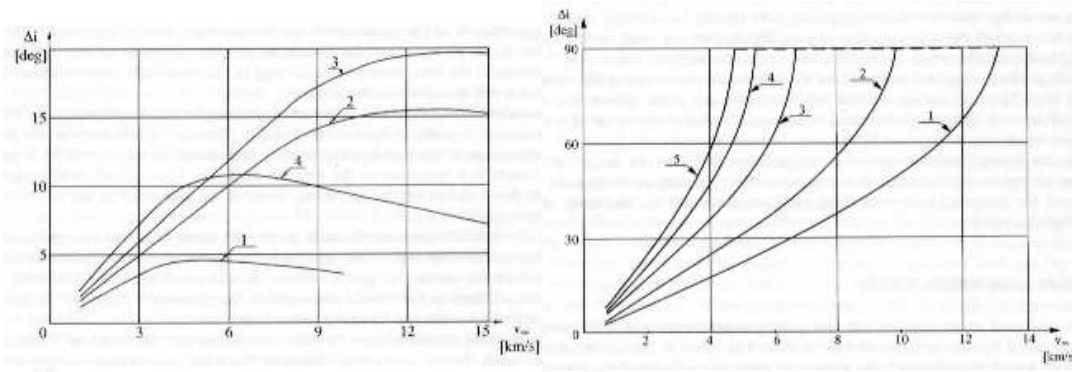


Figure 2.8: Maximum inclination change due to a GA as a function of the hyperbolic excess velocity. Left figure: 1 = Mercury, 2 = Venus, 3 = Earth, 4 = Mars. Right figure: 1 = Jupiter, 2 = Saturn, 3 = Uranus, 4 = Neptune, 5 = Pluto. Courtesy of [5].

2.4. Powered Gravity Assist

All analytical equations derived so far relating to GAs are only valid for an unpowered and unperturbed S/C in the vicinity of the GA-body and are a result of modelling the problem as a two-body problem. However, thrusting within the SOI can have a significant effect, as this section will demonstrate. Within this section the focus lies on the effect of thrust during a GA on the resulting ΔV and other GA parameters; the effect of a powered GA on trajectory design and overall mission objectives shall be touched upon in Chapter 7.

There are two reasons why thrusting while inside the SOI can greatly affect the resulting ΔV and other GA-related parameters. The first reason being that the closest approach distance r_p can be altered, which according to the result of Section 2.3 has a big impact on the generated ΔV and other parameters. Although the equations derived in Section 2.3 are no longer valid due to the application of thrust, the qualitative results still hold. The second reason why thrusting during the GA can have a significant impact is due to the ΔV generated by that thrust force and the Oberth Effect [58], which states that an impulse applied inside a gravitational well is a more efficient way to gain kinetic energy then if applied outside of that gravitational well. It essentially states that a burn should be applied when the S/C is at its maximum velocity, i.e. at its periapsis. The effect occurs since the propellant has more usable energy due to its kinetic energy in addition to its chemical potential energy, such that the S/C is able to employ that kinetic energy to generate more mechanical energy. Since the S/C only remains near the periapsis for a short time, a large impulsive shot has the most effect, which is limited to high-thrust missions. Such an impulsive shot at the periapsis does not affect the closest approach distance, which means the increase in ΔV from that shot can readily be evaluated.

Low-thrust has to be applied over a longer duration to have much of an effect, and therefore will always alter the closest approach distance. One could devise a simulation which would alter the entry conditions into the SOI such that a comparison can be made between a powered and unpowered GA with both the same closest approach distance. However, both would enter the SOI with different hyperbolic excess velocities, meaning the two cannot be directly compared. It is assumed that in low-thrust application the effect of thrust on the GA exit parameters is mostly caused by a change in closest approach distance, for which simulations follow.

2.4.1. New Horizons Powered Gravity Assist Simulations

A New Horizons like scenario will be presented which will show the effects of continuously applying tangential low-thrust at different parts of the trajectory within the SOI. The initial entry conditions into the SOI are those of the actual mission, generated through SPICE [1] from official NASA kernels[(2)]. The initial state will be propagated while under the influence of the thrust and gravitational pull of the GA-body, after which the resulting trajectories are compared to the baseline trajectory without thrust.

Four scenarios are computed numerically with a simple RK4 integrator with a step-size of h = 100s from the initial conditions a few days prior to entering the SOI with a total integration time of 3 months (15 Jan - 15 Apr 2007). The helio- and planetocentric trajectories, together with the heliocentric velocity and planetocentric distance of these scenarios are shown in Figure 2.9. Key parameters are tabulated in Table 2.2. The four

[(2)] Kernels available at: `https://naif.jpl.nasa.gov/naif/data_archived.html`

computed scenarios are:

1. **Baseline**. The baseline trajectory is the nominal trajectory flown by New Horizons in which no thrust was applied during the GA-manoeuvre and is shown in blue in Figure 2.9. The S/C spent a total of 59.2 days inside the SOI and approached Jupiter at a minimum distance of $r_p = 32.23$ $R_{♃}$ with closest approach velocity of $V_p = 21.2$ km/s. The deflection angle of the resulting GA is $\delta = 15.9°$ which resulted in a ΔV of 5.14 km/s.

2. **From entry to periapsis**. The trajectory shown in red is the result of applying a tangential thrust-force of 1 N from the point of entry upto the point of closest approach. This thrust profile significantly lowers the closest approach distance and therefore greatly increases the resulting ΔV and deflection angle. The resulting flight time within the SOI is 52.6 days (-11% w.r.t. baseline) with a total thrust-time of 27.5 days. The closest approach occurs at a distance of $r_p = 2.67$ $R_{♃}$ (-92%) with a velocity of $V_p = 42.2$ km/s (+99%). The ΔV due to the thrust and resulting GA is 8.58 km/s (+67%) and the trajectory is deflected over 74.6° (+369%).

3. **While altitude < $\frac{1}{2}R_{SOI}$**. Applying a tangential thrust force of 1 N from halfway into, to halfway out of the SOI, results in the trajectory shown in magenta. In this trajectory, the thrust is applied while closest to the periapsis and therefore makes use of the Oberth effect most effectively. Since thrusting is applied before the periapsis is reached, the point of closest approach will be altered and therefore the effects of the GA are also altered. With respect to the baseline of no thrust, the time within the SOI is decreased by 6% to 55.3 days, of which thrust is applied continuously for 27.3 days. The trajectory has a closest approach at 23.8 $R_{♃}$ (-26%) where the closest approach velocity is $V_p = 23.3$ km/s (+10%). The resulting ΔV is 5.89 km/s (+15%) and the deflection angle is 23.1° (+45%).

4. **From periapsis to exit.** The trajectory shown in green is the effect of applying a tangential thrust force of 1 N from the point of closest approach onward, until exiting the SOI. This thrust profile does not alter the point of closest approach which renders the ΔV due to the GA to be broadly equal to that of the baseline. The increase in ΔV and deflection angle is mainly caused by the effect of the thruster, and not because of altering the GA geometry. The flight-time within the SOI is 57.2 days (-3%), during which the thruster is engaged for 27.6 days. The closest approach distance is similar to the baseline case at $r_p = 32.23$ $R_{♃}$ (+0%) where the closest approach velocity is $V_p = 21.2$ km/s (+0%). The ΔV has increased to 5.44 km/s (+6%) as a direct result from expelling mass through is thruster. Similarly, the deflection angle has increased to 20.6°, an increase of +30% from the baseline case.

Table 2.2: Key GA parameters for the four New Horizons like scenarios under the influence of thrust inside the SOI.

Scenario	T_{flight} (days)	T_{thrust} (days)	r_p ($R_{♃}$)	V_p (km/s)	ΔV (km/s)	δ (deg)
1. Baseline	59.2	-	32.2	21.2	5.1	15.9
2. From entry to periapsis	52.6	27.5	2.7	42.2	8.6	74.6
3. While altitude < $\frac{1}{2}R_{\text{SOI}}$	55.3	27.3	23.8	23.3	5.9	23.1
4. From periapsis to exit	57.2	27.6	32.2	21.2	5.4	20.6

The three presented scenarios have about the same thrust-time with the same thrust-force, but do show tremendous differences amongst each other. These simulations have shown that the effects of thrusting while performing a GA are mostly caused by lowering the closest approach distance, rather than by the direct ΔV generated by the thruster and the Oberth effect as in high-thrust missions. It is clear that thrusting during a GA alters the trajectory in such a way that the analytical equation derived in the previous sections cannot be used and one will have to rely on numerical methods. It should be noted that the SOI of Jupiter is much larger than those of the terrestrial planets. Due to their smaller size, the maximum thrust-time within the SOI is much shorter and therefore the effects of thrust will be smaller.

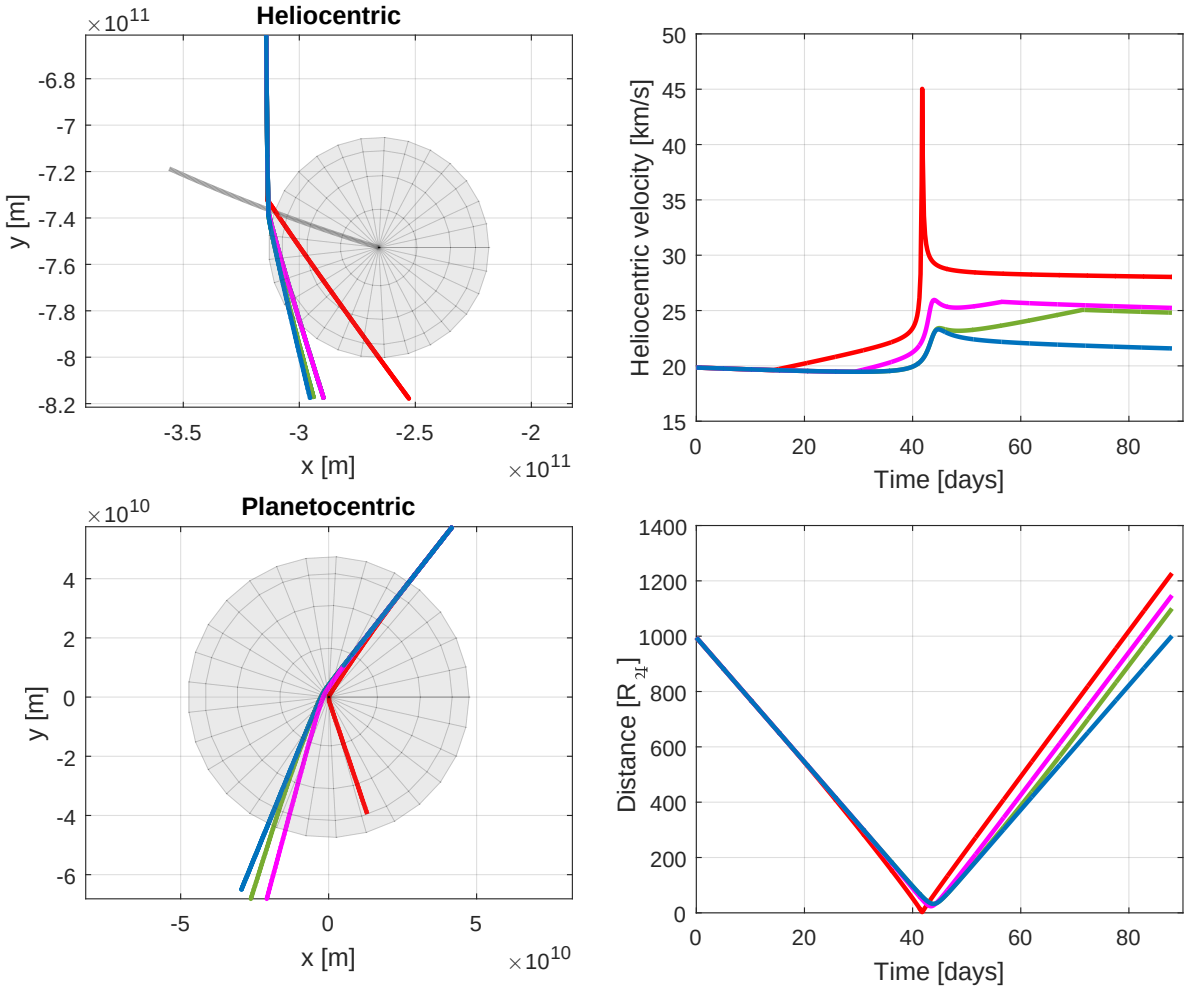


Figure 2.9: Powered New Horizons like GA at Jupiter, continuous thrust of $T = 1$ N applied tangential to the trajectory. Grey spheres denote the SOI of Jupiter, grey line denotes properties of the orbit of Jupiter around the Sun. Blue lines relate to the unpowered baseline, red lines are the result of thrusting from entering the SOI until reaching the periapsis, magenta when the thrust is applied from halfway into to halfway out of the SOI and green when the thrust is applied from the point of closest approach until exiting the SOI.

3

Low-Thrust Trajectory Optimisation with Evolutionary Neurocontrol

This chapter will formally define the low-thrust trajectory optimisation problem. The problem is conventionally solved from an optimal control standpoint, which will briefly be introduced in Section 3.1. This will help in determining optimality and in the formulation of objective and fitness functions. An important distinction in classical methods is that of local and global trajectory optimisation methods, and a high-level description is given of each in Section 3.2. Due to the drawbacks of these classical method, a high-level description of a *smart* low-thrust trajectory optimisation method is presented in Section 3.3; indicating what requirements such a method should fulfil. The remainder of this chapter describes elements which will be used to create such a *smart* method, tackling the problem from the perspective of RL, a sub-field of Machine Learning (ML). To this end, Section 3.4 formally introduces the concepts of Artificial Intelligence (AI), ML and in particular RL. With an understanding of RL, the low-thrust trajectory optimisation problem is tackled from the perspective of RL in Section 3.4.3. The focus shall then shift to a specific RL method termed *neuroevolution* or *evolutionary neurocontrol*, which combines ANNs (Section 3.5) with EAs (Section 3.6) to solve optimal control problems. This method lies at the heart of the low-thrust trajectory optimisation tool termed InTrance (Chapter 4), which will be adapted in Chapter 5 to create a *smart* low-thrust trajectory optimisation tool capable of optimising low-thrust gravity assist trajectories.

3.1. The Low-Thrust Trajectory Optimisation Problem

The trajectory of a S/C is defined as an image of some time interval $[t_0, t_f]$ in some six-dimensional S/C state space $\{\mathbf{x}\} \subset \mathbb{R}^6$. The state at each time instance t can be described with any suitable coordinate system, with Cartesian elements being the most common. The state $\mathbf{x}$ at some instance t is then described by a vector $\mathbf{x} = [\mathbf{r}\ \mathbf{v}]^T = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$. The low-thrust trajectory optimisation problem then consists of finding a set of states that describe the complete trajectory from some initial condition to some target, while abiding to the constraints of and on the system, and maximising some performance criteria.

The low-thrust trajectory optimisation problem can be transformed to the perspective of optimal control theory. The goal is then to find an optimal S/C control function $\mathcal{U}$ which maps some input domain onto a control vector $\mathbf{u} \in \mathbb{R}^{n_u}$, thereby completely defining the magnitude and direction of the S/C's thrust force $\mathbf{T}$, while abiding to the constraints of and on the system, and minimising some cost function J. The trajectory is then found by (numerically) integrating the Equations of Motion (EoM) from the initial conditions while applying the control vector as $\dot{\mathbf{x}}_{\mathrm{SC}}(t) = G(\mathbf{x}_{\mathrm{SC}}(t), \mathbf{u}(t))$, $G : \mathbb{R}^{6+n_u} \mapsto \mathbb{R}^6$. The optimal trajectory is then a combination of optimal initial conditions $\mathbf{x}^*(t_0)$ plus optimal control vector $\mathbf{u}^*(t)$.

Finding a continuous control vector $\mathbf{u}^*(t)$ from a continuous control function $\mathcal{U}^*$ in a per definition infinite solution space is often not possible. Discretising the domain $[t_0, t_f]$ ($\underline{t}_0 \leq t_0 = \bar{t}_0, ..., \bar{t}_f = t_f \leq \bar{t}_\tau \leq \bar{t}_f$) into τ intervals is usually more practical and efficient. This reduces the function parameter space to $n_u\tau$ and solving the problem turns into a problem of finding the optimal S/C control vector history $\mathbf{u}^*[\bar{t}] \in \mathbb{R}^{n_u\tau}$. The optimal trajectory $\mathbf{x}^*_{\mathrm{SC}}[t] = \mathbf{x}^*_{\mathrm{SC}}[\bar{t}_0, \bar{t}_f]$ is then found by applying $\mathbf{u}^*[\bar{t}]$ and numerically integrating the EoM from the

initial conditions onward.

Two elemental formulations of the low-thrust trajectory optimisation problem are the Rendezvous (RV) problem and the Flyby (FB) problem. A successful RV requires to match its velocity and position to that of the target at some time instance, whereas a S/C in a FB problem only requires to match the position. The target is generally a body such as a planet or asteroid but can be any of the following: a celestial body, the orbit of a celestial body, a free orbit described by an (incomplete) set of orbital elements, a fixed state vector, or a fixed state in space. In the realm of optimal control, the rendezvous and flyby problem boil down to finding the optimal control vector $\mathbf{u}^*$ that steers the S/C optimally from the initial conditions. The formal expressions, in the case of discrete time, are described by Ohndorf [59, p. 32] as:

> **Discrete RV problem from the perspective of optimal control theory:**
> Find a spacecraft control vector history $\mathbf{u}[\bar{t}]$, with $\bar{t} \in \left[\bar{t}_0, ..., \bar{t}_{f-1}\right]$, which forces the state $\mathbf{X}_{\mathrm{SC}}(t) = \left(\mathbf{r}_{\mathrm{SC}}^T, \dot{\mathbf{r}}_{\mathrm{SC}}(t)^T\right)^T$ of the spacecraft from its initial value $\mathbf{x}_{\mathrm{SC}}(\bar{t}_0)$ to the state $\mathbf{x}_T(\bar{t})$ of the target body, along a trajectory that obeys the dynamic constraint $\dot{\mathbf{x}}_{\mathrm{SC}}(\bar{t}) = G(\mathbf{x}_{\mathrm{SC}}(t), \mathbf{u}(t))$ and the terminal constraint $\mathbf{x}_{\mathrm{SC}}(\bar{t}_f) = \mathbf{x}_T(\bar{t}_f)$, and at the same time minimises a specific cost function J for that transfer.
>
> **Discrete FB problem from the perspective of optimal control theory:**
> Find a spacecraft control vector history $\mathbf{u}[\bar{t}]$, with $\bar{t} \in \left[\bar{t}_0, ..., \bar{t}_{f-1}\right]$, which forces the position $\mathbf{r}_{\mathrm{SC}}(t)$ of the spacecraft from its initial value $\mathbf{r}_{\mathrm{SC}}(\bar{t}_0)$ to the position $\mathbf{r}_T(\bar{t})$ of the target, along a trajectory that obeys the dynamic constraint $\dot{\mathbf{x}}_{\mathrm{SC}}(\bar{t}) = G(\mathbf{x}_{\mathrm{SC}}(t), \mathbf{u}(t))$ and the terminal constraint $\mathbf{r}_{\mathrm{SC}}(\bar{t}_f) = \mathbf{r}_T(\bar{t}_f)$, and at the same time minimises a specific cost function J for that transfer.

Within the realm of optimal control problems, the optimality of a trajectory is determined through minimisation of cost functions. Common examples of cost functions within the low-thrust trajectory optimisation problem are to minimise transfer time

$$J_T = \int_{t_0}^{t_f} dt = t_f - t_0 = \Delta t, \tag{3.1}$$

or to minimise the required propellant for a given transfer time and launch mass

$$J_{m_p} = \int_{t_0}^{t_f} \dot{m}_p dt = m_p(t_f) - m_p(t_0) = \Delta m_p. \tag{3.2}$$

However, both are examples of single-objective optimisation, whereas within the low-thrust trajectory optimisation problem one is usually interested in optimising both propellant usage and flight time. Such multi-objective optimisation can be performed by Pareto optimisation, or more simply by introducing constraints on one parameter and solely making the other parameter subject to optimisation.

3.2. Traditional Trajectory Optimisation

Satellites have been flying through interplanetary space since the 1960's with the Venera missions to Venus. Initially, trajectories were often based on an initial guess by an astrodynamics expert. Nowadays, more sophisticated methods are available to optimise trajectories, and their general characteristics will be discussed in this section. Traditional trajectory optimisation usually transforms the problem of finding a 6-dimensional state representation of the trajectory over time into finding a 3-dimensional control vector history, which when combined with an initial state and a numerical integrator, results in the trajectory. Solving for the optimal control vector is no trivial tasks, and in general, no analytical methods exist that can tackle the problem. In high-thrust optimisation problems, one usually looks to optimise a handful of instantaneous burns along the trajectory. Low-thrust complicates the problem further, as one is no longer looking at a handful of burns but at very long thrust arcs, often lasting for months at a time, vastly increasing the problem's dimensions.

In general, there are two traditional classes of trajectory optimisation methods for both high- and low-thrust trajectories, these are either termed *local* or *global* methods, which are described next.

3.2.1. Local Trajectory Optimisation Methods

Local Trajectory Optimisation Methods (LTOMs) are build on an extensive mathematical background based on the calculus of variation, employing so called optimal control methods. LTOMs can be divided in direct

and indirect methods; where nonlinear programming is an example of a direct method, and gradient methods like hill climbing and neighboring extremal methods are examples of indirect methods. The theoretical basis behind these methods is not required for a qualitative assestment of LTOMs, and therefore will not be elaborated on. For a comprehensive survey of both direct and indirect methods, the reader is referred to Betts [9]. The common denominator in all LTOMs is their need for an initial guess, often of the control vector history $\mathbf{u}[\bar{t}]$, and are therefore not suited for autonomous trajectory design. LTOMs find the most optimal steering strategy that is in the vicinity of that initial guess, and boil down to finding a better solution than the initial guess, rather than finding the absolute *best* solution.

A block diagram of conventional LTOMs is shown in Figure 3.1. The first step is the setup of the initial conditions, arrival conditions and a discretisation. Setup of these values is done by the user, and requires expert knowledge in the fields of mission design, astrodynamics and optimal control theory. A simulation is then performed during which the local steering laws change the orbital elements, until the S/C's terminal state is close enough to the target state or body. Local steering laws are a set of equations that give the locally optimal thrust direction that changes some specific osculating orbital element of the S/C with a maximum rate. Lagrange's planetary equations in Gauss' form (see [76]) are often used as local steering laws, as they describe the rate of change of a body's osculating elements due to some acceleration or force.

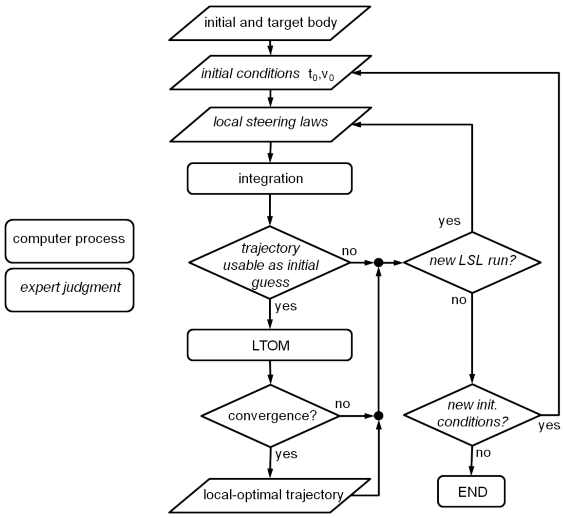


Figure 3.1: Traditional trajectory optimisation using local optimisation methods. Courtesy of [59].

LTOMs are, despite their need of constant supervision, quite popular and have been used in many trajectory optimisation problems. A major advantage is that they have a proven mathematical foundation, and their behaviour is well understood and deterministic. The main upside of LTOMs is their accuracy; the resulting trajectory usually matches exactly to the initial and desired final conditions. However, they naturally also have their drawbacks. The major disadvantage is the need for an initial guess, which either requires an expert in the fields of mission design, astrodynamics and control theory; or expertise in intricate shape-based methods. Furthermore, when LTOMs converge to a solution, it is usually a local optimum, not the desired global optimum. Iterative variation of the initial guess to improve a solution is difficult, as similar inputs can produce very dissimilar results. Additionally, local steering laws are needed and often cannot be described analytically, necessitating additional effort to approximate them numerically. Lastly, the computational effort of LTOMs can be substantial, as many transcribe the trajectory problem into a system of linear equations, often resulting in matrices with dimensions reaching up to many thousands.

Hence, if high accuracy is of the utmost importance, one should rely on LTOMs. However, they can only find the global optimal solution if it is sufficiently close to the initial guess. Therefore, LTOMs are often initialised with an initial guess generated by Global Trajectory Optimisation Methods (GTOMs), which are the subject of the next section.

3.2.2. Global Trajectory Optimisation Methods

GTOMs are different from LTOMs in that they search for the true global optimum and do not require an initial guess. Some common global methods in trajectory optimisation are; Dynamic Programming (DP), branch and bound algorithms, differential evolution, particle swam optimisation, genetic programming and simulated annealing. There is no GTOM method superior to all other methods for all trajectory optimisation problems, as the optimal method is highly problem dependent. Again, the mathematics of every algorithm is not necessary to discuss their common behaviour. For additional information, Alemany [2] gives an extensive survey of GTOMs applicable to the low-thrust trajectory optimisation problem for an asteroid rendezvous mission.

The major advantage of GTOMs is their global search behaviour and independence from an initial guess. The search space of a GTOM is not constrained to the vicinity of an initial guess, and if a global optimum exist, a GTOM is likely to find it when properly coded and given sufficient resources. Due to their independence of an initial guess, GTOMs can often be initialised at random, without the need for expert knowledge. Another major advantage of GTOMs is their ability to optimise both the initial conditions of the trajectory, and the optimal control history $\mathbf{u}$. Especially when incorporating gravity assists; optimisation of initial conditions of the S/C becomes important. Gravity assists require the planets to form favourable configurations, which can more easily be found with an optimisation of a launch window, as opposed to a fixed launch date. However, GTOMs also have their drawbacks. The first being their lower accuracy as compared to LTOMs, however, this is not necessarily a drawback for preliminary mission design studies. Furthermore, since GTOMs are heuristic optimisation methods, and their behaviour is therefore non-deterministic, they could potentially require multiple runs to converge to a solution.

It is clear that for intricate problems a GTOM is preferable over an LTOM, where the solution of the GTOM can be used later on as an initial guess for an LTOM to generate a higher fidelity solution. For preliminary design though, a GTOM often suffices in terms of required accuracy. The problem with conventional GTOMs is that they are often problem specific, and no single tool exists that can determine optimal trajectories for all types of missions, especially not for low-thrust gravity assist trajectories.

3.2.3. State-of-the-Art Trajectory Optimisation Tools

Methods that are commonly used in the industry are ESA's PaGMO, NASA's Mystic and independent open-source PSOPT. PaGMO [39] is developed by ESA ESTEC and is a library for massively parallel optimisation and provides a unified interface to optimisation algorithms. Although very powerful, it is not equipped to optimise a large set of trajectories, rather, it provides the building blocks to create a tool capable of optimising a specific trajectory.

NASA's Mystic [78] is a tool combining both global and local methods and is fully capable of optimising many types of interplanetary low-thrust trajectories and can autonomously find gravity assists if beneficial. It is still dependent on an initial guess through a 'guesstool' GUI, requires heavy modification for new missions, and is highly dependent on astrodynamics and mission design expertise. This tool was used to design the Dawn and Jupiter Icy Moon Orbiter missions. Mystic is only available to US companies/citizens due to ITAR regulations.

PSOPT [6] is often used by nations involved in spaceflight lacking a dedicated tool for trajectory optimisation and (university) researchers. It is, for instance, used by the Brazilian space agency to determine optimal trajectories to an asteroid [81]. PSOPT is an open-source optimal control software package that uses direct collocation methods. It is a general tool capable of solving many types of control problems, with no specific focus on trajectory optimisation, and therefore has to be adapted for each new mission, which is no trivial task.

A commonly used method for the design of low-thrust gravity assist trajectories is to combine a software tool termed GALLOP with STOUR-LTGA [15, 25, 51]. STOUR-LTGA was developed by Petropoulos [61] and is capable of making broad searches in the set of low-thrust gravity assist trajectories with simple analytical shape-based methods. STOUR-LTGA generates thousands of candidate solutions, from which the user should select a single one to serve as an initial guess for GALLOP. GALLOP [51] then uses a direct local optimisation method developed by Sims and Flanagan [68] to find an optimal solution which is close to the initial guess. All in all, before a trajectory results from GALLOP, a total of at least three software tools has to be used, all which rely heavily on astrodynamics expertise [50].

3.3. Smart Low-Thrust Trajectory Optimisation

A truly *smart* GTOM should be able to function independent from knowledge of low-thrust trajectory optimisation experts, both during the optimisation process and at its initialisation. Such a method should furthermore not require any input different than a description of the mission, i.e. which targets it should visit and intervals for the initial conditions. It should furthermore be able to optimise a broad set of trajectories to varying targets without the need of re-writing its code for each new mission. The general concept of such a method was sketched by Dachwald [17] and is shown in Figure 3.2.

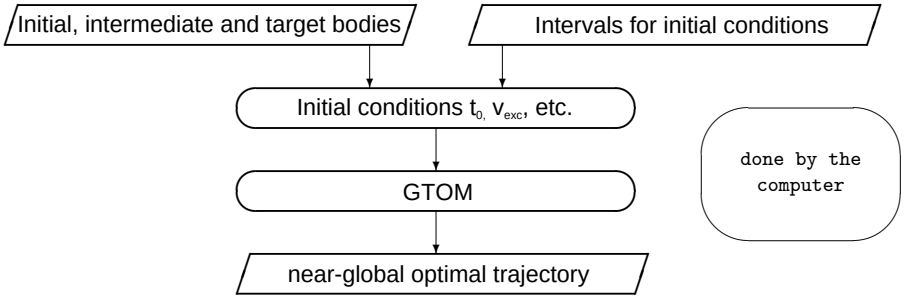


Figure 3.2: Smart low-thrust trajectory optimisation using a global trajectory optimisation method. Courtesy of [17].

Dachwald [17] created such a *smart* low-thrust trajectory optimisation method he termed InTrance, which was later extended by Ohndorf [59] to include multi-phase missions. This method is described in Chapter 4 and will serve as the basis to create a tool capable of optimising low-thrust gravity assist trajectories as described in Chapter 5. InTrance makes use of a method from the field of RL, for which the general characteristics are described in the remainder of this chapter.

3.4. Artificial Intelligence and Reinforcement Learning

Artificial Intelligence (AI), as its name suggests, is an artificial tool that aims to mimic the capabilities that distinguish man from machine, i.e. the ability to learn, think, generalise, solve problems, and adapt to its environment. A comprehensive definition of AI was devised by Shapiro [67] as:

> "Artificial Intelligence is a field of science and engineering concerned with the computational understanding of what is commonly called intelligent behaviour, and with the creation of artefacts that exhibit such behaviour."

An essential part of AI is Machine Learning (ML); i.e. the ability for a machine to learn from –an adapt itself to– its environment. A common denominator in the definition of ML is, according to De Jong [23], its ability to make structural changes to themselves over time with the intention of improving their performance on given tasks evaluated by the problem environment, to discover and subsequently exploit interesting patterns/concepts, or to improve the consistency and generality of internal knowledge structures. ML tasks are generally divided into three categories; (1) Supervised Learning (SL), (2) Unsupervised Learning (UL), and (3) Reinforcement Learning (RL).

Within the first category, supervised learning, the algorithm is presented with example inputs and their desired outputs, in which the goal is to learn a general rule that maps inputs to outputs. The second category, unsupervised learning, is given data as its input, but there is no accompanying output, leaving the machine to find a structure within the data. Unsupervised learning can also be the goal in itself, for instance, in discovering hidden patterns or as a means to learn a specific feature. The last category, reinforcement learning, is most often used when solving control problems; it allows the application of ML to problems for which no data is available. Within RL, a program interacts with the environment in which it must perform a certain goal, for which feedback is provided through the use of an objective or fitness function. The learning system in ML is termed an agent, and optimal behaviour of that agent is defined by an associative mapping from a situation domain $\mathcal{X}$ onto an actions domain $\mathcal{A}$ trough $S : \mathcal{X} \mapsto \mathcal{A}$. Evaluation of the reaction of the environment then results in a scalar fitness value J, which 'measures' the quality of the agent's action. Reinforcement learning can further be subdivided in immediate or delayed reinforcement learning, where in the former the

environment reacts immediately after each of the agent's actions, and in the latter only a single fitness J is returned for all actions during a certain time step. The optimal strategy then is the strategy that maximises the sum of positive reinforcements and minimises the sum of negative reinforcements.

With respect to the intended use, RL boils down to solving a Markov Decision Process (MDP), indicating which action to perform for a certain state of the system, while maximising some reward. The concept of MDPs will be the subject of the following section.

3.4.1. Markov Decision Process

The control of a S/C can be described by a Markov Decision Process (MDP), that is, at each time step the S/C has a certain state and a decision has to be made on what action to perform to eventually reach a target in minimum time, with minimal fuel consumption, or any other metric. The environment is then modelled as a set of states, where actions can be performed to control the system's state in such a way that a performance criterion is maximised.

A multitude of problems can be modeled as MDPs, such as (stochastic) planning problems, learning (robot) control and game playing problems. Instead of creating a planning in which every foreseeable action and outcome is documented, MDPs deal with policies that map states onto actions in such a way that the expected outcomes will have the intended effects. Principally, good or optimal policies for problems modeled as MDPs can be computed with either Dynamic Programming (DP) or RL, where RL deals with the more general case in which no (prior) knowledge about the MDP is available. DP can be used when problems have a relatively small number of states and when the underlying random structure is relatively simple [34]. Delayed RL can then be described as a class of algorithms in the field of ML that aims at allowing an agent to learn how to behave in an environment, where (scalar) feedback is only provided after numerous actions have been completed. The remainder of this section will consist of formal definitions of states, actions, transition between states, and rewards, after which the MDP can formally be defined. The following definitions are from Wiering [80]:

States
A state is a unique characterization of all information that is important in describing the state of the modelled problem. As an example, in chess, the state is given by a complete configuration of the board, that is, all the positions and types of both black and white pieces. The complete set of all possible states is then the finite set $S = \{s^1, ..., s^N\}$, where the size of the state space is $|S| = N$.

Actions
Actions are used to control the system state, the set of actions that can be applied to a particular state $s \in S$ is denoted by $A(s)$, with $A(s) \subseteq A$. The complete set of actions is then given by $A = \{a^1, ...a^K\}$, where the size of the action space is $|A| = K$. Not all actions can be applied in all states $s \in S$, and hence a precondition function pre : $S \times A \rightarrow$ {true,false} is applied, stating whether action $a \in A$ is applicable to state $s \in S$. As an example, the action A of moving the queen in chess can only be applied when the queen is still available in the state s, hence, when it has not yet been removed from the board.

The transition function
The system makes a transition from state $s \in S$ to a new state $s' \in S$ by applying an action $a \in A$, based on a probability distribution over the entire set of possible transitions. The transition function T is defined as $T : S \times A \times S \rightarrow [0, 1]$, which states that the probability of ending up in state s' after applying action a on state s is $T(s, a, s')$. The state transitions of a MDP satisfy the Markov or memoryless property, that is, the action of the state does not conditionally depend on the previous states and actions, but only on the current state:

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ...) = P(s_{t+1}|s_t, a_t) = T(s_t, a_t, s_{t+1}). \tag{3.3}$$

Hence, the current state s gives enough information to make an optimal decision, regardless of what states were achieved and what actions were taken before.

Reward function
The reward function (positive for a reward, negative for a penalty) specifies rewards for being in a specific state or for performing some action in a particular state. Within this work, the reward function will be defined as $R : S \times A \times S \rightarrow \mathbb{R}$, giving rewards for particular transactions between states. The reward function implicitly states the goal of learning, for instance in chess; all states that resulted in a won can be given a positive reward, all states that resulted in a loss a negative reward, and a zero reward for each state that resulted in a draw, such

that the reward function gives direction to how the system (i.e. the MDP) should be controlled.

With the above definitions, the MDP is defined as tuple $\langle S, A, T, R \rangle$ where S is a finite set of states, A a finite set of actions, T a transition function defined as $T : S \times A \times S \rightarrow [0,1]$ and R a rewards function defined as $R : S \times A \times S \rightarrow \mathbb{R}$. Hence, a MDP is a discrete stochastic control process, where at each step the system is in some state s, such that the policy π decides which action a should be taken, resulting in a new state s', from which it receives a certain reward $R(s, a, s')$. The policy, or agent, π is then a computable function that outputs what action $a \in A(s)$ should be taken given state $s \in S$, and hence acts as a mapping from inputs to outputs, formally defined as $\pi : S \rightarrow A$.

3.4.2. Solving Markov Decision Processes

Solving a MDP is equivalent to computing the optimal policy π^*. There are two classes of algorithms, based on whether the algorithm is model-based or model-free, that can solve a MDP. Model-based algorithms are categorized under the umbrella term Dynamic Programming (DP), in which the basic assumption is that a model of the MDP is known a priori such that it can be used to compute value functions and policies. Model-free algorithms fall in the class of RL, they do not rely on the availability of a perfect model, but rather interact with an environment, such that a simulation of a policy generates samples of state transitions and rewards, which are then used to estimate state-action value functions.

Within the remainder of this work, only RL will be considered. The reason is twofold, first off, DP requires an a-priory model, which is tedious and difficult to obtain for the low-thrust trajectory optimisation problem. Furthermore, the mentioned DP algorithms require iterations over the full state-space and are therefore at a major disadvantage due to the curse of dimensionality. RL methods can still find near-optimal solutions without the need for a model, thereby rendering a much more efficient process for large scale problems [35].

There are roughly four categories of direct RL methods; (1) direct policy-search, (2) Temporal Difference (TD) learning, (3) brute forcing, and (4) Monte Carlo simulations. The latter three are not used within this work and the reader is referred to Wiering [80] and the Literature Review [43] for more information. Direct policy-search searches directly in the policy space, such that the problem becomes a case of stochastic optimisation. The main type of algorithms within this class are the Evolutionary Algorithms (EAs), which when used in conjunction with an Artificial Neural Network (ANN) as the agent, results in the field of neuroevolution.

Direct policy-search is by no means limited to EAs, and can be any global or local optimisation method. However, evolutionary methods have proven to be a particularly popular way to search policy space, rendering that a rich collection of algorithms and results are available in the RL literature. Furthermore, EAs often outperform other global methods in RL tasks [43]. Local methods will not be considered since they often find sub-optimal solutions and can get trapped in local minima. The determination of parameters (weights/thresholds) in ANNs with a predetermined structure, such as fixed feed-forwards multilayer perceptrons, requires the solution of a non-linear optimisation problem. Such problems are usually NP-complete and the chance of finding the optimal solution using gradient-based methods is minimal for large dimensions [82].

3.4.3. Trajectory Optimisation from the Perspective of Reinforcement Learning

The low-thrust trajectory optimisation problem can be modelled as a Markov Decision Process (MDP). The task is then to solve the MDP, hence, to determine the optimal policy π^*. The policy, combined with an agent, results in a steering strategy $\mathcal{S}$.

The RV and FB problem are readily transformed to the domain of RL as:

> **Discrete RV problem from the perspective of reinforcement learning:**
> Find a spacecraft steering policy $\boldsymbol{\pi}^*$, which forces the state $\mathbf{X}_{\text{SC}}(t) = \left(\mathbf{r}_{\text{SC}}^T, \dot{\mathbf{r}}_{\text{SC}}(t)^T\right)^T$ of the spacecraft from its initial value $\mathbf{x}_{\text{SC}}(\bar{t}_0)$ to the state $\mathbf{x}_T(\bar{t})$ of the target body, along a trajectory that obeys the dynamic constraint $\dot{\mathbf{x}}_{\text{SC}}(\bar{t}) = G(\mathbf{x}_{\text{SC}}(t), \mathbf{u}(t))$ and the terminal constraint $\mathbf{x}_{\text{SC}}(\bar{t}_f) = \mathbf{x}_T(\bar{t}_f)$, and at the same time minimises a specific cost function J for that transfer.
>
> **Discrete FB problem from the perspective of reinforcement learning:**
> Find a spacecraft steering policy $\boldsymbol{\pi}^*$, which forces the position $\mathbf{r}_{\text{SC}}(t)$ of the spacecraft from its initial value $\mathbf{r}_{\text{SC}}(\bar{t}_0)$ to the position $\mathbf{r}_T(\bar{t})$ of the target, along a trajectory that obeys the dynamic constraint $\dot{\mathbf{x}}_{\text{SC}}(\bar{t}) = G(\mathbf{x}_{\text{SC}}(t), \mathbf{u}(t))$ and the terminal constraint $\mathbf{r}_{\text{SC}}(\bar{t}_f) = \mathbf{r}_T(\bar{t}_f)$, and at the same time minimises a specific cost function J for that transfer.

3.5. Artificial Neural Networks

Artificial Neural Networks (ANNs) are technologies counterpart to the biological Neural Network (NN). An ANN tries to simulate the workings of a NN, making it possible to tackle numerous types of problems under varying conditions. An amazing biological example of the capability of a NN can be seen in the housefly, with an estimated amount of 100 000 neurons, its NN is capable of controlling the housefly's intricate flight path, guiding the fly on its quest to find food and a suitable mate, to eventually produce offspring; arguably the main goal of the housefly. The controller within the housefly does not rely on intricate calculus of variations, but has adapted –and improved– itself due to Darwin's [22] principle of evolution. Nature and time have taken care of the optimisation of the neurocontroller trough both recombination and mutations of the fly's genetic material, and trough natural selection, i.e. through "survival of the fittest".

The analogy to nature in ANNs can be taken a step further by also employing an optimisation scheme that works in much the same way as the evolution of the housefly, collectively termed EAs, used to optimise the effectiveness of the NN. Combining ANNs with EAs results in the field of research that is then termed neuroevolution, or evolutionary neurocontrol when applied to control problems. ANNs have been applied in many fields of research, primarily due to their ability to approximate complex environments without any a priori knowledge about the underlying function. Hence, ANNs have opened a complete new realm of possibilities, allowing research in topics in which the underlying theory is unknown or simply not fully understood.

This section will describe the basics of ANNs, most prominently being the components which make up the network. Section 3.6 will detail the workings of EAs such that neuroevolution can formally be defined in Section 3.7. The reader is reffered to the Literature Review [43] conducted prior to this work for a more comprehensive overview of the history, inner workings and learning algorithms of ANNs.

3.5.1. Biological Paradigm

Artificial Neural Networks (ANNs) are an attempt at modeling the information processes of nervous systems. Contrary to cellular automata, NNs have a hierarchical multilayered structure; information is not only transmitted to its direct neighbors but also to more distant units [64]. It achieves this by interconnecting one neuron with many other neurons, thereby forming a network which is able to process and store information, generalise, learn, and make decisions, based on both experience and the current environment. Biological neurons exist in many different shapes and forms, depending on its task and the specie. However, common features in most types of neurons are the cell body, dendrites, synapses, and the axon. An illustration is depicted in Figure 3.3. Although the exact workings of the neuron are not completely understood, the general consensus is that a neurons' electrical output is dependent on the level of exciting and inhibitive inputs. If the excitement exceeds the sum of inhibitive inputs, the neuron fires an electrical signal along its output extension. Inhibitive inputs are received by other neurons and connect to the cell body trough extensions called dendrites. If the neuron fires an electrical signal, it travels through its output extension, termed axon, and either connects directly to muscles and organs, or branches of and connects to dendrites of other neurons via synapses.

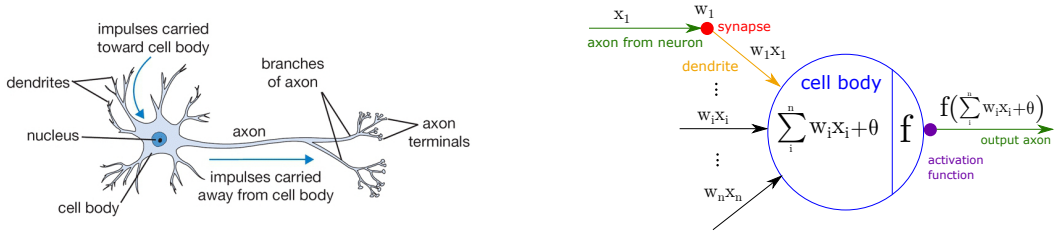


Figure 3.3: Illustration of a typical biological neuron (left) and artificial neuron (right), both from [41].

An artificial neuron (also termed unit), as depicted on the right side of Figure 3.3, mimics much of the basic functions of a biological neuron. In an artificial neuron, the output of one neuron is connected to many other neurons, receiving the output of the former neurons as input, similar to the axon in a biological neuron. The artificial axon then connects to a synapse, where the input is multiplied by some weight w. The product of the two then travels along to the cell body, much in the same way as the dendrite in a biological neuron. Within the cell body, the inputs are summed and compared to some threshold θ. It is then run trough an activation

function f, which determines how large the output of that neuron is, which again travels over an axon and connects to a new neuron.

3.5.2. Components of Artificial Neural Networks

ANNs are constructed by connecting single neurons (also termed units). The flow of information within a unit and internal workings are discussed first. A vital part of the unit is the activation function, which determines how and if a neuron should fire, which is discussed next.

Single Neuron

A neuron can be generalised to the form as shown on the left in Figure 3.4. It receives inputs x_i with $i = 1, ..., n$, which are multiplied by their respective weights w_i, then run through an integration function g, and lastly through an activation function f. The integration function is usually taken as the addition function. Possible activation functions are described below, but the sigmoid and step-functions are most common. The output of the unit is then given by $f\left(g\left(x_1, x_2, ..., x_n\right)\right)$.

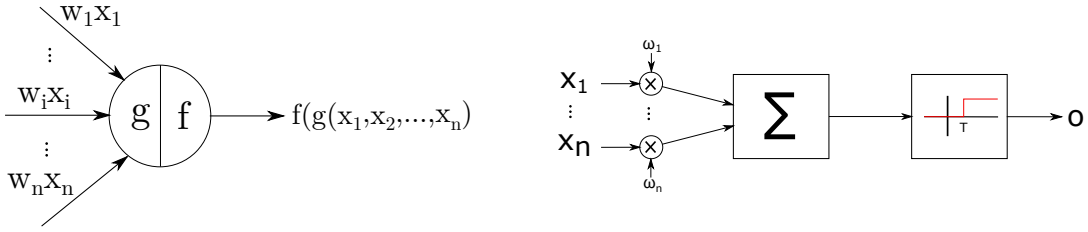


Figure 3.4: General form (left, [64]) and block-diagram (right) of a computing unit within an ANN.

A block diagram of this sequence is shown on the right side of Figure 3.4. The input is a vector $\mathbf{x}$, each element x_i $(i = 1, .., n)$ of that vector is multiplied by a certain weight ω_i (denoted by $\otimes$ in Figure 3.4). The sum of these products is then taken as $\sum_{i=1}^{n} \mathbf{x} \cdot \boldsymbol{\omega}$ ($\boxed{\Sigma}$ in Figure 3.4), of which the output is run through an activation function. This particular example makes use of the step-function, returning a 1 only if the summed input is larger than some threshold value T. The output of the network will then be a vector function f of the inputs $\mathbf{x}$ and the weights $\boldsymbol{\omega}$, which can be denoted as $\mathbf{o} = f\left(\mathbf{x}, \boldsymbol{\omega}\right)$.

Activation Functions

The activation function of a unit effectively controls the output of a unit. Its role is usually to introduce non-linearity in the mapping so that any complex function can be approximated. Without an activation function, an ANN would only be able to deal with linearly separable data. In theory, the step-function is a good activation function; a neuron either fires or it does not. However, since it is not differentiable at 0, which is necessary in gradient based optimisation methods, other functions are devised. Furthermore, the binary step-function would require a large number of neurons when approximating mappings that are not linearly separable. Hence, a quest arose to find alternative activation functions, some based on biological neurons, others on statistics, and some simply on intuition.

Some desirable properties in activation functions are:

- **Nonlinearity.** When multiple layers use linear activation functions, all those layers can be replaced by a single layer. When the activation functions are non-linear, a three-layer model can be proven to be a universal function approximator [16].
- **Continuously differentiable.** A necessity for gradient-based optimisation methods.
- **Monotonic.** The error surface associated with a single layer is guaranteed to be convex if the activation function is monotonic.
- **Smooth.** A necessity for gradient-based optimisation methods. Smooth functions with a monotonic derivative have been shown to generalise better in specific cases [30].
- **Approximates identity near origin.** Activation functions that approximate identity ($f(x) = x$) near the origin can be initialised with small random weights [66].

The logistic sigmoid activation function $s(x)$ is the most common used activation function in NN literature [31]. The sigmoid is one of the most popular activation functions, as it is a smooth representation of the step-function, and is described by

$$s(x) = \frac{1}{1+e^x}. \tag{3.4}$$

Although the logistic sigmoid is often used, it has its limitations. The sigmoid does not have a steady state at 0, a property that is desirable within activation functions from an optimisation standpoint [45]. Furthermore, due to asymptotes at $s(x) = 0$ and $s(x) = 1$, the sigmoid is known to saturate and kill gradients. Hence, when the input is either small or large, the gradient will be almost zero, limiting the signal flow through a neuron and thereby limiting the efficiency of the weight updates. Lastly, the outputs of sigmoids are not zero-centered, which leads to undesired zig-zagging dynamics in gradient updates [41]. Other activation functions are available with their own upsides and limitations, the reader is referred to the Literature Review [43] for a comprehensive overview.

3.5.3. Network of Neurons

Single neurons are only capable of solving simple threshold logic. Single units can be used for classification purposes, but only when the inputs are linearly separable. Combining multiple neurons results in a network, and increases the computing capacity. Feed-forward networks only allow edges to run from a particular neuron to a neuron in the next layer; there are no backwards connections or skips of layers. Recurrent networks, in which a neuron in layer j's output can be fed into a neuron in layer $p < j$, have its own advantages and disadvantages (see [64] for a discussion), but are not used within this work.

The topology of a general feed-forward ANN is shown in Figure 3.5. The layers of the network can be split in (1) the input layer, which is the very first layer in which the n inputs are given to the network, (2) the output layer, which is the last layer and generates the m output of the entire network, and (3) the hidden layers, which are all layers in between the input and output layers. Hidden layers are necessary to determine more complex mappings. Within this model, connections can only run from left to right, connecting a neuron with all neurons within the next layer. The input and output layers are always a single layer, whereas the number of hidden layers can be any positive integer number. Each layer can contain a different amount of units; the number of input sites within the input layer is always equal to the number of inputs, and the number of output units in the final layer is always equal to the number of outputs of the network.

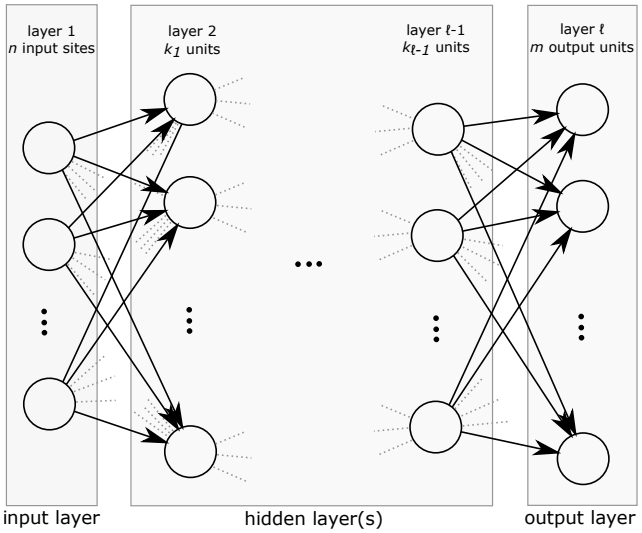


Figure 3.5: Topology of a general feed-forward ANN.

It can be proven that the general learning problem for networks of threshold functions is NP-complete [64]. This follows from Kolmogorov's theorem, which states that any continuous function of n arguments can always be represented using a finite composition of functions of a single argument, in addition. It implicitly

states that addition is the only function needed to represent continuous functions with any number of arguments. A modern version of Kolmogorov's theorem (proof in [69]) is given by

Kolmogorov's Theorem. *Let $f : [0,1]^n \rightarrow [0,1]$ be a continuous function. There exist functions of one argument g and ϕ_q for $q = 1,...,2n+1$ and constants λ_p, for $p = 1,...,n$ such that*

$$f(x_1, x_2, ..., x_n) = \sum_{q=1}^{2n+1} g\left(\sum_{p=1}^{n} \lambda_p \phi_q(x_p)\right).$$

In the perspective of ANNs, Kolmogorov's theorem states that any continuous function of n variables can be represented by a finite network of functions of a single argument, where addition is used as the only function of several arguments. Accepting units capable of computing integral powers of the input, polynomial approximation of a given function can be used, resulting that any real continuous function can be approximated with arbitrary precision using a finite number of computing units [64].

3.6. Evolutionary Algorithms

Evolutionary Algorithms (EAs) is an umbrella term for a family of population-based metaheuristic or stochastic optimisation algorithms inspired by biological evolution. Best known algorithms within this class include Genetic Algorithms (GAs) [40], Differential Evolution (DE) [72], Evolutionary Programming (EP) [28] and Genetic Programming (GP) [42]. All of these methods share the same enabling principles but focus on different problem classes. Common features in all are the simulation of evolution of individual structures through reproduction, inheritance, and selection mechanisms, dependent on their perceived performance defined with respect to an environment (survival of the fittest). This section will describe the basic mechanism and principles of EAs, the reader is referred to De Jong [24] for a more comprehensive overview.

3.6.1. Elements of Evolutionary Algorithms

The key element in EAs is a population Ξ^t that is comprised of numerous individuals $\boldsymbol{\xi}^t_{k\in 1,...,q}$, also termed chromosomes or strings, where the superscript t denotes the time step or generation within the simulated evolution. Each individual within a population is a potential solution to the given optimisation problem. Analogous to nature, the relatively good solutions within the population reproduce, whereas the relatively bad solutions go extinct. In order to determine which solutions are relatively good, a performance measure J is introduced. This performance measure is similar to the cost function in optimal control theory, and is also termed fitness or objective function. A selection scheme then determines which individuals (parents), with a probability according to their fitness value $J(\boldsymbol{\xi}^t_k)$, are to reproduce and create offspring into a newly created population Ξ^{t+1}. Offspring is reproduced by recombination of the 'genetic' material of the parents, which is possibly subject to mutation. In order to further avoid premature convergence, immigration of new individuals within the population is often beneficial. After some reproductive cycles, all individuals should converge to a single solution, which is in best case the global optimal solution $\boldsymbol{\xi}^*$ to the given problem.

EAs mostly have four essential components; (1) a population Ξ, containing the candidate solutions; (2) a mechanism to select candidates for reproduction; (3) the reproduction itself, containing both the mutation and recombination operators; and (4) an evaluation of the fitness of a solution. These four components are illustrated in Figure 3.6, in which the arrows show the flow of the reproduction cycle of a population individual ξ_i. Parent individuals are denoted by ξ_p and children individuals by ξ_c. ξ_{in} are either individuals that initialise the population or outside immigration individuals, and ξ_d are extinct individuals removed from the population.

Initialisation and Representation

The population is usually initialised as a pool of randomly generated individuals within the domain $\mathcal{S}$. In general, the accuracy of an EA is dependent on the population size, and a few trial-and-error runs usually suffice. Alternatively, a commonly used rule of thumb in selecting a population size is based on the number of optimisation parameters and length of the chromosome as $n_{pop} = n_{parm} \times 2^{\ell}$ [32].

An important distinction between EAs is whether an individual is represented by a bit string or by a real-valued number vector. Using real-valued coding, a chromosome corresponds to a vector of real numbers,

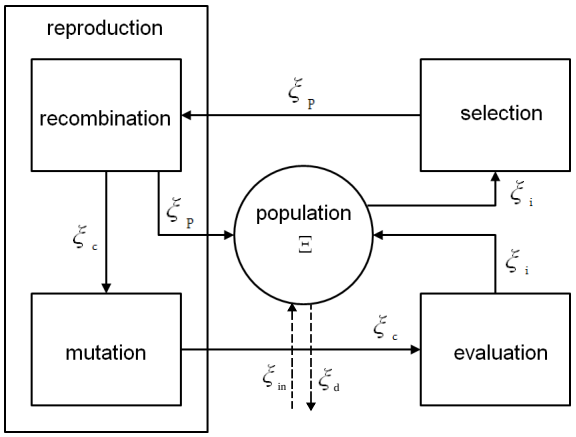


Figure 3.6: Principle components and reproduction cycle of EAs. Adapted from [59].

an allele to a real number value, and the loci to the index; $\boldsymbol{\xi} = \langle r_1, r_2, ..., r_\ell \rangle$. The binary representation was assumed superior for a long time, as it decomposes the optimisation problem into the largest number of smallest building blocks. However, it was found that EAs using the binary representation often suffer from lack of precision and inability to operate in the presence of nontrivial constraints [52]. These problems result in high-dimensional high-precision numerical problems, as is often the case in training ANNs. As an example, consider an ANN with three layers with 24 input nodes, 30 hidden nodes and 3 output nodes, the total number of internal parameters is then 908. If the domain of all of them is [-2,2] and a precision of 8 digits after the decimal is desired, the total length of the binary vector is $\ell = 908 \cdot 29 = 26332$. For such problem dimensions, EAs do not perform optimal [52]. Contrary, when using a real-valued number representation, the length of the vector for each individual is only $\ell = 908$ and the precision is the machine precision [17].

Adaptive parameter encoding schemes, such as Delta Coding (DC), optimise the representation of an individual [79]. DC uses an EA that restarts multiple times with a bit-string representation; the first run is used to find an interim or partial solution $\mathbf{h}$, and subsequent runs decode the genes as a distance (delta value) from the last interim solution, such that $\boldsymbol{\xi}_k = \mathbf{h} + \boldsymbol{\delta}_k$. Each restart forms a new hypercube with the interim solution at its origin and the search space can be extended or contracted by altering the resolution of the delta values. A restart (re-initialisation of population) occurs when the Hamming distance (number of differing bits) between the best and worst individual of the population ($\sum_{i=1}^{\ell} |b_{i,1} - b_{i,q}|$) is larger than 1. Usually, the hypercube is extended by one bit if the new partial solution is identical to the old one, i.e. if the best $\boldsymbol{\delta}$-chromosome is a zero-string. Otherwise, the hypercube is contracted by one bit. The reduction mechanism allows the algorithm to focus on promising search subspaces, whereas the expansion mechanism allows the algorithm to explore previously overlooked areas of the search space [48]. DC has outperformed both a standard GA and a mutation-driven stochastic hill-climbing algorithm on a suite of standard EA test functions. DC used fewer trials than the other methods for most test functions, and was the only method to consistently find the global optimum for all test functions [48]. The idea of DC has also been extended to real-valued strings, termed Floating Point Delta Coding (FPDC), the reader is referred to [21] for more information.

Selection and Diversity

Two important issues in any EA are Selective Pressure (SP) and population diversity. They are mutually-influencing; an increase in selective pressure results in a decrease in population diversity, and vice versa. SP is defined as the expected number of individuals of the next generation, whereas the population diversity is a measure of the different genome material in a population. There exist a multitude of selection schemes, of which some will be discussed here. The selection of a scheme highly depends on the dimensions and type of optimisation problem. The first and simplest method is termed elitist selection, in which the q individuals with the highest fitness, in a pool with both the parents and children, are selected to populate the next generation. A major problem with elitist selection is premature convergence, and hence is not the recommended approach. So-called superindividuals (individuals with a much higher fitness than other individuals in the population) have a much larger probability of creating offspring, thereby limiting the diversity within a population, and thereby possibly resulting in premature convergence to a local minimum. Carefully cho-

sen, problem-dependent, fitness functions or fitness scaling functions can partly alleviate this problem, but are not recommended as they dependent on a user decision, which makes the method less robust and less diverse.

There are a multitude of selection schemes that do not require scaling of the fitness functions, but most of them are dependent on an individuals relative fitness –its rank[(1)]–, therefore requiring an ordering of the individuals, which is computationally expensive. Another selection scheme, also based on ranking but more computationally efficient, is tournament selection. Within tournament selection, a single individual is selected to progress to the next generation by choosing the individual with the highest fitness from some random chosen subset of the entire population. Note that if the subset is equal to the entire population, the method is simply a relative elitist selection scheme. A common subset size is $\mu = 2$ [4], resulting in a so-called binary tournament selection scheme. The tournament is then repeated a total of q times, such that all parents that create offspring are selected. With tournament selection, the selection probabilities are given by [4]

$$p_i(\mu) = \frac{(q-i+1)^\mu - (q-i)^\mu}{q^\mu} \qquad \Rightarrow \qquad p_1(2) = \frac{2q-1}{q^2}, \tag{3.5}$$

so that for a binary tournament ($\mu = 2$)

$$SP = p_1 \cdot 1 = \frac{2q-1}{q} = \text{const.} \quad \text{with} \quad \lim_{q\to\infty} SP = 2. \tag{3.6}$$

Hence, the SP remains constant throughout the search process, and the best individual receives –on average– about two copies in the next generation. Since selection probability is independent of the absolute fitness, tournament selection does not require fitness scaling. A major advantage of tournament selection is that each tournament can be performed with respect to a different optimisation objective, thereby allowing multi-objective optimisation without explicitly weighing the objectives; such a scheme prefers individuals that perform resonably well with respect to all objectives. In the words of Dachwald [17], "Its like a duel between two cowboy gunslingers. To survive a duel, one must draw fast *and* aim accurately".

Recombination, Crossover and Mutation

The basic recombination scheme within the realm of EAs is the generational reproduction scheme. Within the generational reproduction scheme, q individuals are selected from the population pool of both parents and children, and copied to the new population Ξ^{t+1}. The scheme is computationally expensive, as most of the runtime is spent for copying strings without progressing the search [17]. An alternative method is termed one-at-a-time or steady-state reproduction, which only lets one reproduction take place at each time step, and is often combined with tournament selection, as shown in Figure 3.7. Two tournaments are performed, which results in two winners and two losers. Within the new population, the two losing individuals are replaced by the offspring of the two winning individuals, and the other individuals remain the same. Hence, only two individuals have to be replaced to form a new population, whereas generational reproduction requires copying of q individuals from a population of $2q$ (parents+children) into a new population.

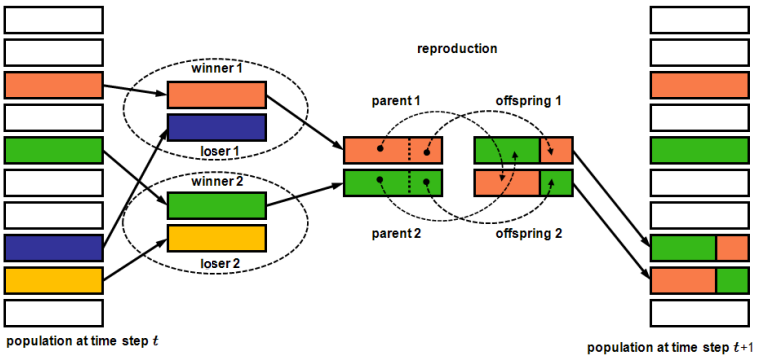
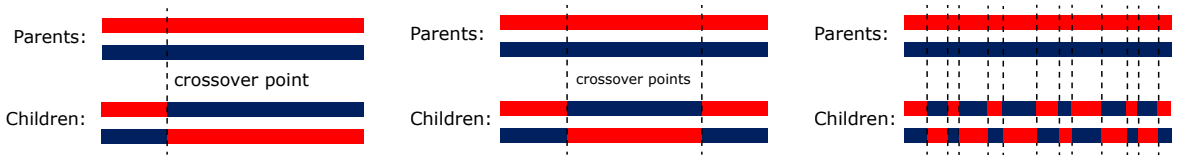


Figure 3.7: One-at-a-time reproduction with tournament selection. Courtesy of [59].

[(1)] The best individual has a rank of (population size – 1) and the worst individual a rank of zero.

A multitude of crossover types exist, of which some common ones are (1) one-point crossover, (2) two-point crossover, (3) uniform crossover, and (4) arithmetic crossover. The first three types can be applied to both bit strings and real-valued strings, the 4^{th} type is limited to real-valued strings. A special crossover type is termed (5) node crossover, derived from the uniform crossover type, and is derived for the training of ANNs with an EA [59]. Within one-point crossover (Figure 3.8a), a single crossover loci on both parents' chromosome is selected at random, all data behind that loci is then swapped between the chromosomes of the parents. The downside is that not all possible schema (building blocks) can be represented, furthermore, schemas with long defining lengths are likely to be destroyed under one-point crossover, and the endpoints of the loci are always switched, thereby treating some loci preferentially [27]. Two-point crossover (Figure 3.8b) is similar, but now the data within two loci of both parents is swapped. The method alleviates the problem of treating the endpoint preferentially, and is less likely to disrupt schemas with large defining lengths. However, it still cannot combine all schemas [54]. Uniform crossover (Figure 3.8c) evaluates each loci in the parent string for exchange with a probability p, often $p = 0.5$. In that case, children have approximately 50% of the genes from the first parent and 50% of genes from the second parent. Empirical evidence suggests that uniform crossover is a more exploratory approach then the more traditional approaches that maintain longer schemata. Hence, the uniform crossover results in a more complete search of the search space while maintaining good exchange of information [38]. The uniform crossover operator does not have any positional bias; any schema contained at different positions in the parents can potentially be recombined in the offspring. However, the lack of positional bias can prevent coadapted alles from ever forming in the population, since the uniform crossover operator can be highly disruptive of any schema [54].



(a) One-point crossover. (b) Two-point crossover. (c) Uniform crossover.

Figure 3.8: Some common crossover types.

Arithmetic crossover is only applicable to real-valued chromosomes, and is the only crossover type mentioned here that actually changes the value, and not just the location. The offspring $\boldsymbol{\xi}_{c,i}$ is a linear combination of its parents ($\boldsymbol{\xi}_{p,i}$), determined as

$$\boldsymbol{\xi}_{c,1} = \alpha \boldsymbol{\xi}_{p,1} + (1-\alpha)\,\boldsymbol{\xi}_{p,2} \tag{3.7}$$

$$\boldsymbol{\xi}_{c,2} = (1-\alpha)\,\boldsymbol{\xi}_{p,1} + \alpha \boldsymbol{\xi}_{p,2}, \tag{3.8}$$

in which α is a random weighting factor, chosen before each crossover operation.

Crossover nodes (Figure 3.9) is designed specifically for the training of ANNs. Within this method, it is decided for each coded neuron in the first offspring (with a probability of 50%) which parent contributes its coded parameters for that neuron. The second offspring then receives the coded neuron from the other parent. A promising feature of this method is that the parameters of a single neuron are not torn apart. Crossover nodes has been tested on a complex sonar image classification problem by Montana [55], and was found to outperform the backpropagation algorithm (see the Literature Review [43]). However, it is difficult to asses how the method compares to more conventional crossover operators, as the performance is highly problem-dependent and fundamentally different.

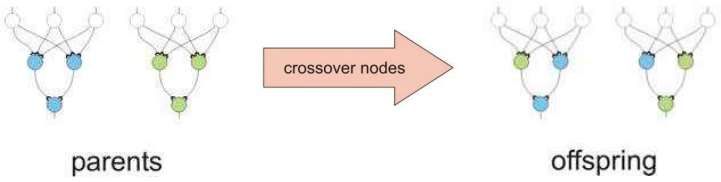


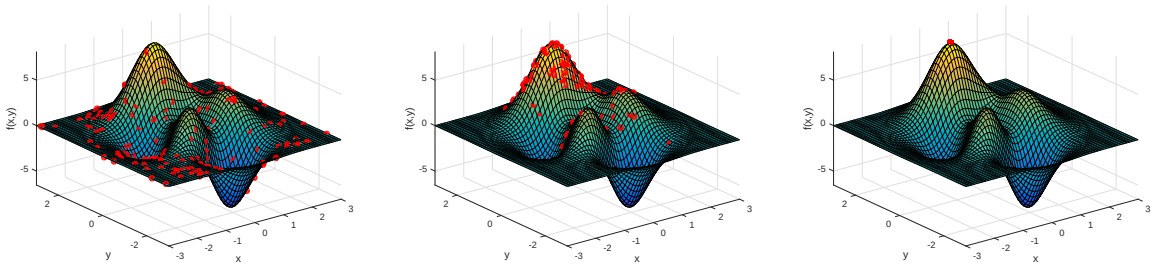
Figure 3.9: Illustration of the generation of offspring with the crossover nodes operator. Figure from [17].

It is difficult to prefer some crossover type over another, as the success or failure depends in a complicated fashion on the particular fitness function, encoding and other details of the EA [54]. Extensive literature is available which quantifies aspects of different crossover types, such as the positional bias, the disruption potential, the ability to create different schemas in one step, and so on. However, these do not give a definitive answer to which operator should be preferred. As an example, some problems work better with highly disruptive schemas, whereas other benefit from coevolution of alleles. A multitude of literature is available that compare the performance of different operators on a suite of test functions, but results are often conflicting, indicating the problem-dependency [54]. In general, it is accepted that two-point and uniform crossover outperform one-point crossover, but it might be beneficial to alternate between different types of operators to receive some of the benefits of each.

Mutation introduces genetic diversity into the population pool and can help the algorithm to move away from a local optimum. In bit strings, the mutation operator determines for each loci with probability p_m whether its allele should be mutated, that is, wheter a 1 will be mutated to a 0, and vice versa. Uniform mutation is most common and determines for each locus whether that allele should mutate, resulting in the generation of many random numbers for potentially only a few mutations, making it a computationally intensive method. According to Whitely [79], mutation is not necessary when using DC.

3.6.2. Convergence and Properties

At first sight, the process of recombination and mutation seems to be completely random and therefore one might think that the search process within EAs is completely undirected. However, with the inclusion of a selection method that prefers fitter individuals, the search becomes directed and an active force of improvement is present [3]. Compared to gradient based methods, the main advantages of EAs is their good global search behaviour, their blindness, their problem independence, and their robustness. These advantages are not exclusive to EAs, and to some respect are also shared with other global optimisation methods. EAs initially evaluate the target function to be optimised at randomly selected points of the domain. From an evaluation of their fitness, a new generation is formed, and gradually the points in the population approach the global optimum, as shown in Figure 3.10. This results in a good global search behaviour within multi-modal environments, since the optimum is relatively independent of the initial conditions. Contrary to gradient based methods, EAs achieve this without any additional information about the gradient of the function at the evaluated points –they are blind– and the function itself need not be differentiable or even continuous. This makes EAs especially suited for delayed reinforcement learning problems, where auxiliary information is not available. Furthermore, EAs can be applied to a wide range of optimisation problems, since every problem whose parameters can be encoded on a string can be optimised, thereby making it a very robust method.



(a) Generation: 1 (b) Generation: 5 (c) Generation: 39

Figure 3.10: Convergence of a simple GA with a population size of $q = 200$ over the generations. Mutation probability is set at $p_m = 0.001$, tolerance at $\epsilon = 10^{-9}$, and the bit string lengths at $\ell_1 = 18$ and $\ell = 17$. Underlying function: $f(x,y) = 3(1-x)^2 \exp(-(x^2) - (y+1)^2) - 10(x/5 - x^3 - y^5)\exp(-x^2 - y^2) - 1/3\exp(-(x+1)^2 - y^2)$.

3.7. Neuroevolution

The coupling between ANNs and evolutionary methods is termed neuroevolution [84], and is often referred to as evolutionary neurocontrol when applied to control problems. Within the realm of neuroevolution, it is the task of an evolutionary algorithm to optimise the internal parameters of the ANN in such a way that

it provides a mapping from a state of the agent to an optimal action. The Literature Review [43] conducted prior to this work found that neuroevolution is one of the most popular and successful direct policy-search RL methods for complex control problems. Evolutionary RL is by no means limited to ANN representations, but is by far the most common [80], and is the representation that will be used within this work.

In the context of neuroevolution, low-thrust trajectory optimisation becomes a problem of finding the optimal weights of an ANN that acts as a policy π. Those weights are determined by an EA, such that the entire problem boils down to finding an optimal chromosome $\boldsymbol{\xi}^*$, see Figure 3.11. A trajectory is then formed by sampling the ANN at each control step, telling the agent what action to take. With the current state plus the action (a thrust force or acceleration), the EoMs can be integrated, resulting in the next state. This process is then repeated from the initial time $\bar{t}_0$ to the final time $\bar{t}_f$, resulting in a complete trajectory. Upon reaching the final target, the trajectory can be evaluated and a fitness J is determined. The process is repeated until finding the (near) global optimal trajectory. A major upside of neuroevolution is that the initial state and launch date can also be optimised by including them as optimisation parameters onto the chromosome.

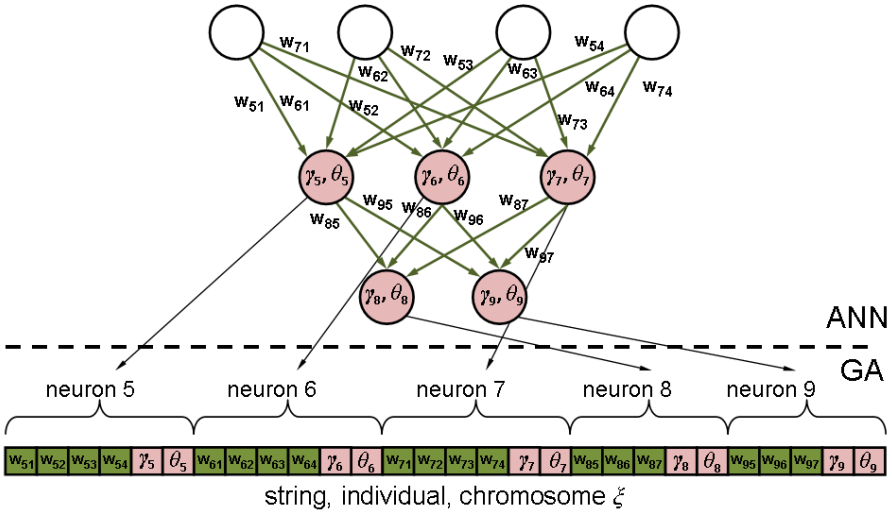


Figure 3.11: Mapping of NC parameters on a chromosome $\boldsymbol{\xi}$. Here, the parameters of the NC are the connection weights w_{ij}, the bias or threshold of each unit θ_i, and the reciprocal of the temperature coefficient γ_i used in the sigmoid activation function of each unit. Note that the first layer consists of input nodes, and hence has no tunable parameters. Courtesy of [59].

4

InTrance

SMART-1 was a very successful mission carried out by ESA and demonstrated the *in situ* application of low-thrust electric propulsion. It reached a Selenocentric orbit by following a preset thrust profile with a maximum of 70 mN generated by a single ion engine. Although following a moderately simple trajectory, its steering command history was developed with significant effort by a team of flight dynamic experts [60]. Local trajectory optimisation methods are plagued by their need for an initial guess and existing global methods are often far from robust and require heavy modification for every new mission, as detailed in Section 3.2. For future application and greater onboard autonomy there is a need for easy-to-use and robust techniques to find (near-)global optimal steering strategies for low-thrust trajectories.

A promising method to overcome the drawbacks of traditional trajectory optimisation methods was developed by Dachwald [17] which he termed *INtelligent TRAjectory optimisation Using NeuroController Evolution (InTrance)*. InTrance tackled the low-thrust trajectory optimisation problem from the perspective of reinforcement learning through Evolutionary Neurocontrol (ENC). InTrance is a smart GTOM according to the definition of Section 3.3; it is capable of autonomously finding the (near-)global optimal trajectory, without the need of an expert in the field of mission design and astrodynamics, and optimises both the initial conditions and trajectory with no dependence on an initial guess.

InTrance is adapted and extended as described in Chapter 5 to allow for the optimisation of low-thrust gravity assist trajectories. To this end, this chapter will contain a high level description of the governing algorithms in InTrance, and furthermore discuss essential elements such as the specific EA, the size of, and inputs and outputs to, the ANN, how the multi-phase framework is incorporated, how the trajectory is integrated, and how the fitness is evaluated. For a more thorough description, the reader is referred to the original work by Dachwald [17] and the revised version by Ohndorf [59].

4.1. Development History

InTrance was developed by Dachwald [17] in 2004 with the intention of devloping a *smart* preliminary design tool capable of solving the single-phase low-thrust trajectory optimisation problem for interplanetary heliocentric trajectories from a reinforcement learning perspective. Dachwald implemented conventional neuroevolution, as described in Section 3.7, utilising a custom evolutionary algorithm for the training of the Artificial Neural Network (ANN). Dachwald used InTrance to recalculate a variety of trajectories utilising solar sails, SEP and NEP, and often found superior results over some reference cases [17–20].

Ohndorf [59] extended and revised the original version of InTrance to accommodate the optimisation of both non-heliocentric and heliocentric multi-phase low-thrust trajectories, such that each phase is governed by a separate neurocontroller. Ohndorf classified his enhancements in three groups; type A contains mechanisms that are essential to accommodate the multi-phase framework; type B contains mechanisms that allow refined mission analysis through trajectory integration in a more realistic environment; and type C comprises mechanisms that increase the robustness of the optimisation. The updates incorporated by Ohndorf can then be summarised as shown in Table 4.1, and the reader is referred to [59] for a thorough description of each.

Table 4.1: Classification of the mechanism and techniques added by Ohndorf to InTrance. Courtesy of [59, p. 66].

Element/mechanism	Type A	Type B	Type C
Non-heliocentric simulation	x	n/a	n/a
Planetary shadows	n/a	x	n/a
Third-body perturbation	x	x	n/a
Excess energy optimisation	n/a	x	n/a
Dynamic control step size	n/a	n/a	x
Parameter range adaptation	n/a	n/a	x
Search space scan	n/a	n/a	x
Hypercube size control	n/a	n/a	x
Variable boundary constraints	n/a	n/a	x

Ohndorf used the revised version of InTrance to simulate three transfers; (1) a two-phase transfer from an Earth-bound orbit into an orbit about the Moon, (2) a multiple asteroid rendezvous mission, similar to Dawn but without the GA, and (3) a solar system escape mission to the heliosphere bow shock using SEP, Radioisotope Electric Propulsion (REP) and a Jupiter GA. It should be noted that Ohndorf's version of InTrance lacks the generic support for gravity assists, but that they are implicitly possible.

Carnelli [13] worked on extending Dachwald's initial version of InTrance to incorporate GAs within the single-phase framework, thereby relying on a single Neurocontroller (NC); hence optimisating the weights of a single ANN that is to solve the entire trajectory and autonomously detect GAs. Carnelli's initial hypothesis was that since a single NC was found capable of performing solar photonic assists by Dachwald [17], it should also be able to optimise gravity assist manoeuvres. However, this method was deemed unsuccessful after many iterations and many adaptations to both the environment (artificially increasing the SOI) and the genetic algorithm. Individuals suffered from premature death, and the method did not converge to a solution in most cases. Even when convergence was achieved, it usually was a sub-optimal local optimum; such that non-GA performing trajectories still achieved a higher fitness. Carnelli eventually relied on a local gradient based method to optimise GAs, which is only applicable to single GA trajectories and no longer implemented in the multi-phase version of InTrance.

4.2. InTrance Architecture

The general functional principle of ENC within InTrance is shown in Figure 4.1. The method consists of three loops: an outer EA optimisation loop, a middle phase loop, and an inner trajectory integration loop. Within InTrance, a chromosome $\boldsymbol{\xi}$ encodes both a policy $\boldsymbol{\pi}$ (i.e. the internal parameters of the ANN) and the initial simulation conditions, which are usually comprised of; the initial state $\mathbf{x}_{\mathrm{SC},0}$, usually depending on the launch date t_0; and the initial propellant mass $m_{p,0}$, if the spacecraft uses a propellant-depend propulsion method. This ensures that both an optimal steering strategy and optimal initial conditions are found.

The outer loop takes care of the reproduction of individuals and the fitness evaluation of generated trajectories. These individuals are then fed into the middle loop, in which the chromosome is decoded for a specific flight phase. The inner loop initialises the ANN with the parameters $\boldsymbol{\pi}_j$. The state of the S/C $\mathbf{x}(t_i)$ and the state of the target $\mathbf{x}_T$ are then supplied to the ANN (NC); which returns a control $\mathbf{u}(\bar{t}_i)$. With the control and state specified, the EoMs are integrated[(1),(2)] and a check is performed whether the termination conditions are met. If not, the resulting new state is again supplied to the NC, together with the state of the target at that time, which again results in a control $\mathbf{u}(\bar{t}_{i+1})$, which is used to propagate to the next S/C state $\mathbf{x}(\bar{t}_{i+1})$.

The process repeats until a termination condition is met, i.e. on fulfilment of the accuracy constraints, on violation of boundary constraints, or after a preset maximum integration/mission time. Once the termination conditions are met, the middle loop performs a check whether there is an additional phase. If so, it returns to the inner integration loop to integrate the next phase of the trajectory. After all phases have been integrated, the resulting trajectory $\mathbf{x}_{\mathrm{SC}}[t]$ is evaluated and a new population is formed. This process repeats until the outer loop achieves population convergence, i.e. no $\boldsymbol{\xi}_i$ with a higher fitness could be generated.

(1) The control step size and integration step size are not necessarily the same.

(2) InTrance uses either the RKF4(5) or RKDP8(7) adaptive step-size integrators. RKF4(5) is used for lower-precision calculations and RKDP8(7) for high-fidelity simulations.

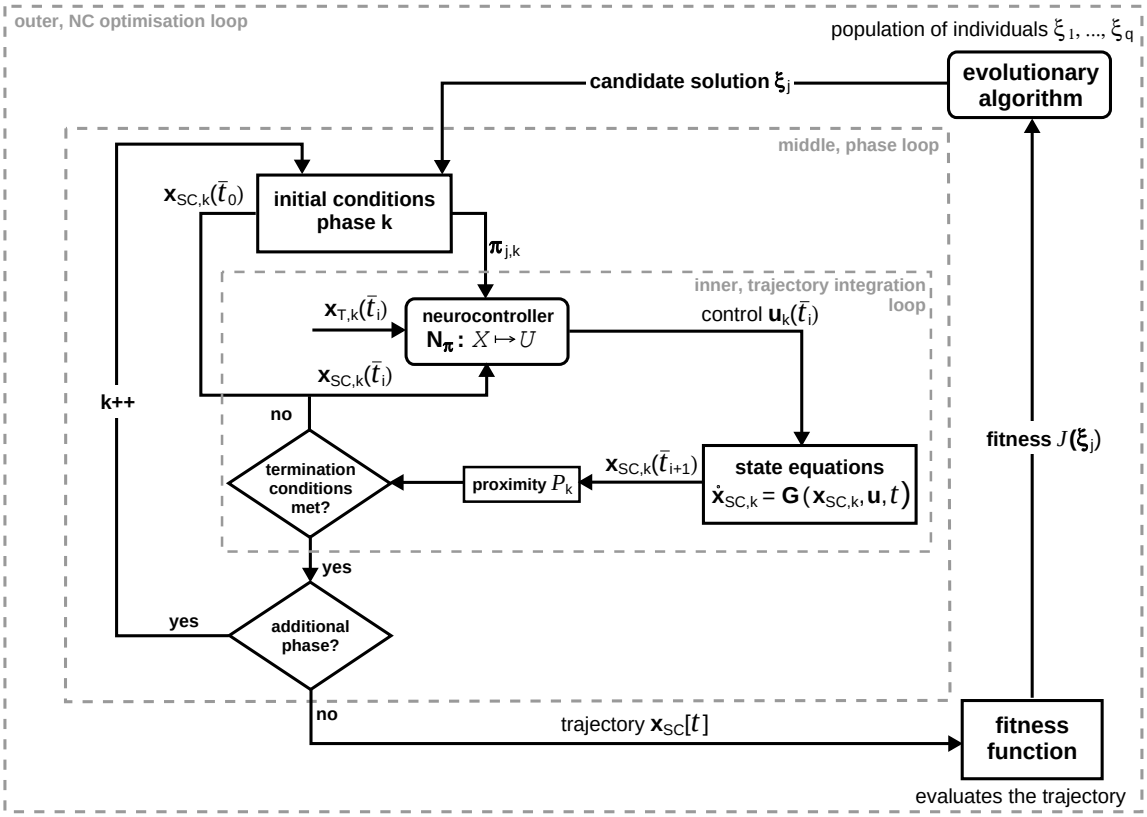


Figure 4.1: Multi-phase trajectory optimisation using evolutionary neurocontrol. Figure adapted from [59].

4.3. Multiphase Framework and Fitness

The major change in Ohndorf's extension of InTrance is the inclusion of the multi-phase framework, making it possible to optimise mission designs that consist of more than one transfer phase. This extensions allows for the optimisation of, for instance, an Earth-Mars-Earth double RV mission with the scientific operations at Mars accounted for with a stay or dwell time. This example is then modelled as a two-phase mission –the first from Earth to Mars and the second from Mars back to Earth– where each phase is controlled by its own distinct Neurocontroller (NC).

Independently optimising the two separate separate NCs –and therefore the two phases– does not necessarily result in the optimal overall trajectory. After all, there are physical constraints which must be met which are dependent on both phases. As an example, the 2nd phase can only start after the arrival date of the first phase, even when an earlier departure might result in a more optimal 2nd phase. Furthermore, considering transfer time minimisation, the time optimal independent phases might not result in a time optimal overall trajectory. After all, a slightly slower first phase might result in favourable initial conditions for the second phase, which makes the overall Mission Elapsed Time (MET) shorter. InTrance tackles this problem by co-evolving the solutions of both NCs and both phases' initial conditions by encoding them on the same single chromosome, see Figure 4.2. The optimisation then drives the S/C to reach the targets of each phase, to optimise the overall objective and to match the final conditions of phase i to the initial conditions of phase $(i+1)$.

4.3.1. Target State, Proximity, and Deviation

The target state describes the state which the S/C should attain at the end of a mission phase. Flybys or rendezvous at specific celestial bodies are the most common targets, but a target can also be an escape or capture from/by a body, rendezvousing with a fixed point in space or rendezvousing with an incomplete set of orbital elements. In the case of a single phase mission, the target state of this phase is the same as the mission target. In case of multi-phase missions, there are intermediate targets and a final mission target. In

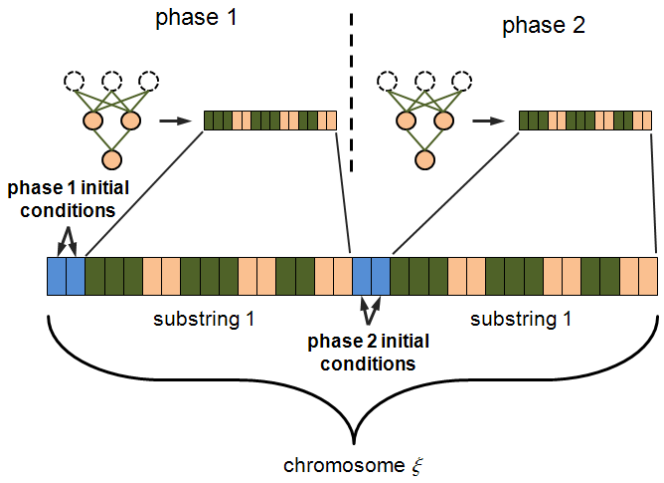


Figure 4.2: Composition of chromosome $\boldsymbol{\xi}$ in the multi-phase framework. Courtesy of [59].

the case of a double Earth-Mars-Earth rendezvous, both Mars and Earth are targets.

Proximity

As was discussed in the previous section, prior to evaluation of a given candidate solution $\boldsymbol{\xi}$, the simulation integrates the EoMs until reaching some stopping criteria. In the absence of physical constraint violations, such as flying too close to the Sun, this stopping criteria is often the maximum integration time or maximum MET. However, it is unlikely that the final spacecraft state at the stopping criteria is the best in terms of being close to the target. To this end, a measure of 'being close to the target' is generated through a proximity $P \in \mathbb{R}$ which has to be determined at each trajectory point, see Figure 4.1. The proximity P is determined through one or more so-called deviations δ_i as

$$P(\delta_i) := \begin{cases} -\sqrt{\sum \delta_i^2} & \text{if } \exists \delta_i : \delta_i < 0 \\ \min \delta_i & \text{if } \forall \delta_i : \delta_i \geq 0 \end{cases}, \tag{4.1}$$

and hence is negative if at least one criterion is negative. Alternatively, if all $\delta_i \geq 0$, then P corresponds to the smallest positive δ_i.

The computation of the deviations δ_i depends on the target type, and is generally one of three: the flyby problem, the rendezvous problem, or the capture problem. A successful flyby requires the distance between the spacecraft and the target position Δr_f at time of closest approach to be smaller or equal to some predefined maximum distance $\Delta r_{\max}$. The deviation of the position is then determined as:

$$\delta_r \in (-\infty, 1] : \delta_r := \frac{\Delta r_{\max} - \Delta r_f}{\Delta r_{\max}}. \tag{4.2}$$

Positive δ_r values denote a successful flyby, with a maximum value of 1 denoting a perfect flyby.

Analogous, the rendezvous problem must fulfil both a distance Δr_f and velocity Δv_f constraint, such that the proximity $P_{\text{RV}}(\delta_r, \delta_v)$ can be determined from equation 4.1. The determination of δ_r is equivalent to that of the flyby problem, whereas the velocity deviation is determined as

$$\delta_v \in (-\infty, 1] : \delta_v = \frac{\Delta v_{\max} - \Delta v_f}{\Delta v_{\max}}. \tag{4.3}$$

The proximity in the capture problem is essentially also determined with $P(\delta_r, \delta_v)$ and the above defined deviations δ_r, δ_v. However, the $\Delta r_{\max}$ and $\Delta v_{\max}$ are no longer arbitrarily chosen values, but should be the maximum that describe a closed orbit that resides completely within the influence region of the body around which it is captured. The maximum distance is then given as the maximum of either the Hill sphere r_H or the SOI; $\Delta r_{\max} = \max(r_H, r_{\text{SOI}})$.

4.3.2. Phase Transition Conditions

Since a mission involving multiple targets is split into multiple phases, certain transition condition constraints have to be put in place to ensure the physical validity at phase crossings. Transition conditions can stem from a mission having more than one phase, such as incorporating a dwell or stay time in a RV mission, and can also stem from the concurrent optimisation of each flight leg, as each leg is separately optimising its initial conditions. The overall fitness should take these gaps between initial and final states of consecutive stages into account; driving the optimisation to both a physically valid trajectory which reaches all targets, and optimises the overall objective (minimise flight time, minimise propellant usage, etc.).

The scalar violation ${}^{(i)}V_{\mathrm{dw}}$ counts the days by which a launch date t_0 is outside the specified launch date window $\left[t_{0,\min}\ \ t_{0,\max}\right]$ and is given by

$$ {}^{(i)}V_{\mathrm{dw}} = \begin{cases} {}^{(i)}t_{0,\min} - {}^{(i)}t_0 & \text{, if } \ {}^{(i)}t_0 < {}^{(i)}t_{0,\min} \\ 0 & \text{, if } \ {}^{(i)}t_{0,\min} \leq {}^{(i)}t_0 \leq {}^{(i)}t_{0,\max} \\ {}^{(i)}t_0 - {}^{(i)}t_{0,\max} & \text{, if } \ {}^{(i)}t_{0,\max} < {}^{(i)}t_0 \end{cases} . \tag{4.4} $$

When incorporating a dwell or stay time at a certain target, ${}^{(i)}t_{0,\max}$ is the maximum allowed stay time $\Delta t_{\mathrm{dw,max}}$ plus the arrival time of the preceding phase ${}^{(i-1)}t_{\mathrm{f}}$. The earliest allowed launch date ${}^{(i)}t_{0,\min}$ is then the minimum dwell time $\Delta t_{\mathrm{dw,min}}$ plus the arrival time of the preceding phase ${}^{(i-1)}t_{\mathrm{f}}$. If no dwell time between phases is required, for instance in the flyby case, a steady and smooth continuous time history along the entire trajectory is provided by setting ${}^{(i)}\Delta t_{\mathrm{dw,min}} = {}^{(i)}\Delta t_{\mathrm{dw,max}} = 0$ such that ${}^{(i)}t_{0,\min} = {}^{(i)}t_{0,\max} = {}^{(i-1)}t_{\mathrm{f}}$.

Due to the concurrent optimisation of the initial conditions of all phases, state and (propellant) mass gaps are inevitable between phases. The optimisation process is driven towards decreasing these gaps by including a scalar state violation ${}^{(i)}V_s$ and a scalar mass violation ${}^{(i)}V_m$ in the fitness, defined as follows:

$$ {}^{(i)}V_s = \left({}^{(i)}\boldsymbol{x}_0 - {}^{(i-1)}\boldsymbol{x}_f\right)^2 \tag{4.5} $$

$$ {}^{(i)}V_m = \left({}^{(i)}m_{\mathrm{SC},0} - {}^{(i-1)}m_{\mathrm{SC,f}}\right)^2 - {}^{(i)}\Delta_m^2 \tag{4.6} $$

These violations measure the difference between the states and mass at the end of a predecessor phase $(i-1)$ and the beginning of a successor phase i. Indices 0 and f denote initial and final conditions, respectively. The mass ${}^{(i)}\Delta_m$ accounts for any intended mass change, such as a jettisoned propulsion stage, collected sample material or docked-on spacecraft.

4.3.3. Fitness Evaluation

The overall scalar fitness $J(\boldsymbol{\xi})$ of an individual (trajectory) is a function of the proximity, the transition conditions and overall objective function. The proximity indicates how 'close' each phase approaches its target, the transition conditions indicate the physical validity in phase transitions, and the overall objective function indicates how well the trajectory performs in terms of propellant usage, transfer time, or some other metric.

Firstly an intermediate fitness function is implemented, indicating how well all phases perform in reaching their target with a single scalar value. This intermediate fitness J_P is a function of the proximities P_i of each phase and is given by

$$ J_P = \begin{cases} -\sqrt{\frac{\sum_{i=1}^{n_p} \min(0,P_i)^2}{n_p}} & \text{if } \exists i \in (0,n_p) \,:\, P_i < 0 \\ \min(P_i) & \text{if } \forall\, i \in (0,n_p) \,:\, P_i \geq 0 \end{cases} . \tag{4.7} $$

Hence, if all proximity variables $P_i \geq 0$, then all constraints of all phases have been met and the sub-fitness $J_p \geq 0$. The overall fitness $J(\boldsymbol{\xi})$ is then determined by the objective function $\mathcal{O}$ and phase transition conditions V_i. The complete list of currently implemented objectives functions in InTrance is depicted in Table 4.2, which are always subject to maximisation. The overall fitness of the complete trajectory is then determined as

$$ J(\boldsymbol{\xi}) = \begin{cases} J_P & \text{if } J_P < 0 \\ \mathcal{O} \cdot \prod_{i \in (dw,m,s)} \frac{1}{1+V_i} & \text{if } J_P \leq 0 \end{cases} . \tag{4.8} $$

Hence, once all phase-specific constraints have been achieved, any fitness improvement results from an increased value of the respective objective function $\mathcal{O}$ or from a decrease of the transition condition violations V_i, or both.

Table 4.2: Currently implemented InTrance objective functions [59].

No.	Symbol	Definition	Description
1	$\mathcal{O}_{\Delta\text{t,min}}$	$1-\frac{\sum_{i=1}^{n_p}\Delta t_i}{\sum_{i=1}^{n_p}\overline{\Delta t_i}}$	transfer time minimisation
2	$\mathcal{O}_{m_p}$	$\frac{1}{{}^{(0)}m_{p,0}-{}^{n_p}m_{p,f}}$	minimum propellant consumption
3	$\mathcal{O}_{m_{\text{SC}}\Delta t}$	$\frac{{}^{(n_p)}m_{\text{SC}}}{\Delta t}$	maximum final mass over transfer time
4	$\mathcal{O}_{m_{\text{SC}}\Delta t_m}$	$\frac{{}^{(n_p)}m_{\text{SC}}}{{}^{(n_p)}t_f-{}^{(0)}t_0}$	maximum final mass over mission duration
5	$\mathcal{O}_{\Delta t_{\text{dw}}}$	$\frac{\sum\Delta t_{\text{dw,i}}}{\sum\overline{\Delta t}_{\text{dw,i}}}$	maximum stay (dwell) time
6	$\mathcal{O}_{\Delta t_m}$	$\frac{\overline{\Delta t}_m-\Delta t_m}{\overline{\Delta t}_m}$	minimum mission duration
7	$\mathcal{O}_{t_0,\text{min}}$	$\frac{{}^{(0)}\bar{t}_0-{}^{(0)}t_0}{{}^{(0)}\bar{t}_0-{}^{(0)}\underline{t}_0}$	earliest launch date
8	$\mathcal{O}_{t_0,\text{max}}$	$\frac{{}^{(0)}t_0-{}^{(0)}\underline{t}_0}{{}^{(0)}\bar{t}_0-{}^{(0)}\underline{t}_0}$	latest launch date

4.4. Evolutionary Algorithm

InTrance makes use of an EA to optimise both the ANN parameters of each NC and the initial conditions for each phase, as described in Section 4.3. The basic elements and customisation options of EAs have been described in Section 3.6, and hence will not be repeated here. This section briefly describes the chosen initialisation method, representation, reproduction scheme, crossover and mutation operators. The reader is referred to Ohndorf [59] for a more thorough description of the EA employed within InTrance.

4.4.1. Representation and Initialisation

The chromosomes in InTrance's EA are represented by real valued strings as the distance $\boldsymbol{\delta}$ to a previous solution $\mathbf{h}$, such that $\boldsymbol{\xi}=\boldsymbol{\delta}+\mathbf{h}$, by an iterative search strategy termed Real Delta Coding (RDC). RDC is an extension of Delta Coding (DC) [79], which was described in Section 3.6.1. Within RDC, a real-valued population individual $\boldsymbol{\xi}_i$ is comprised of a $\mathbf{h}$-chromosome and a $\boldsymbol{\delta}$-chromosome

$$\boldsymbol{\xi}_i=\boldsymbol{h}+\boldsymbol{\delta}_i \Leftrightarrow \langle r_1,\ldots,r_l\rangle_i=\langle h_1,\ldots,h_l\rangle_i+\langle\delta_1,\ldots,\delta_l\rangle_i\,. \tag{4.9}$$

RDC thus limits the EA to explore a dynamically selected subspace around the current best solution $\boldsymbol{h}=\boldsymbol{\xi}^*$ instead of the entire search space.

RDC runs in cycles, called epochs, in which a dynamically selected parameter subspace –the hypercube H– around the current best solution is explored. The hypercube of the very first epoch e_0 is constructed around the null solution $\boldsymbol{h}(e_0)=\mathbf{0}$ as

$$H_0=[-\delta_{\max}(e_0),\delta_{\max}(e_0)]\in\mathbb{R} \tag{4.10}$$

such that the very first population is randomly initialised as $\Xi^{t_0}(e_0)=\left(\boldsymbol{\delta}_1^{t_0}(e_0),\ldots,\boldsymbol{\delta}_q^{t_0}(e_0)\right)$. The EA then runs until convergence of that epoch's population; that is until the relative fitness improvement over the last ν time steps is smaller than a preset limit ϵ.

The best solution $\xi_1^{t_c}(e_0)$ from this epoch becomes the partial solution $\boldsymbol{h}(e_1)$ of the next epoch e_1, whose new delta range is

$$\delta_{\max}(e_1)=\kappa\cdot\delta_{\max}(e_0), \tag{4.11}$$

in which κ is a search-space reduction mechanism. The reader is referred to Ohndorf [59] for a discussion on this hypercube size control method. The new population $\Xi^{t_0}(e_1)$ is then constructed from equation 4.9

within the updated search space hypercube

$$H_1 = [h_1(e_1) - \delta_{\max}(e_1), h_1(e_1) + \delta_{\max}(e_1)] \times \ldots \times [h_l(e_1) - \delta_{\max}(e_1), h_l(e_1) + \delta_{\max}(e_1)] \subset \mathbb{R} \tag{4.12}$$

This process continues until convergence over the epochs is achieved, i.e., until the relative improvement between two consecutive epochs is smaller than a preset limit ϵ.

InTrance can start an optimisation run either as a warmstart or coldstart. The prior starts from a user-provided solution, usually generated from an earlier run of InTrance, such that a higher fidelity solution can be generated. The second option, coldstart, requires no externally provided solution and initialises the optimisation via a heuristic parameter search called Search Space Scan (SSS). This algorithm uses niching and co-evolution to explore the entire search space to find a suitable starting point. SSS is initialised by random individuals which are evaluated and ordered according to their fitness. Each of these individuals initialise a new generation which are evaluated until convergence within its epoch. If the fitness of an individual at the end of the epoch is better than that of the one which initialised that epoch, it replaces the old one in the list of individuals. After all individuals are evaluated within the list, they are ordered and the one with the worst fitness is removed. This process repeats until only a single individual is is left, which is the result of the SSS and the starting solution $\boldsymbol{h}$ for the subsequent optimisation run.

4.4.2. Reproduction, Crossover and Mutation

As reproduction mechanism, InTrance uses the in Section 3.6.1 described one-at-a-time reproduction with tournament selection. The selection scheme is a binary tournament scheme, hence, $\mu = 2$. Four crossover types have been implemented in InTrance; (1) one-point crossover, (2) uniform crossover, (3) arithmetic crossover and (4) loci crossover. All of these crossover operators have been described in Section 3.6.1. A crossover type is chosen at random during each reproduction, with the probability for each initially set at $p = 0.25$. Once more than 100 successful reproductions have taken place, crossover types are determined dynamically. The probability for each type is then given by the number of times that operator has been applied, divided by the total amount of reproductions. This ensures that operator types that have been successful in the past are used more often in later stages of the optimisation.

InTrance uses a more computationally friendly method derived from uniform mutation, termed fast uniform mutation, as its mutation operator. In conventional uniform mutation, each allele at each loci of the chromosome has a chance p_m of mutating, and consequently a random number generator has to be initiated for each loci. Contrary, in fast uniform mutation, it is decided (with probability p_m) for the entire chromosome whether an allele of a singly loci is to mutate. If it is decided that mutation is to occur, a random loci of that chromosome Ξ_i is chosen and its allele δ_{ij} is replaced with a new $\delta_{\text{ij}} \in [-\delta_{\max}, +\delta_{\max}]$.

4.5. Artificial Neural Networks

The Artificial Neural Network (ANN) within InTrance is an a priori specified fixed feed-forward network. The number of hidden layers, and number of nodes within each, can freely be specified in InTrance's input files. The size of the input and output layers are fixed, with one node for each input/output parameter. The inputs to the ANN are described in Section 4.5.1, and Section 4.5.2 describes how to decode the ANN output to result in the required control data.

4.5.1. Input to the Artificial Neural Networks

The ANN in InTrance receives the complete set of inputs as shown in Table 4.3. The input to the network is a combination of normalised forms of:

- the current Cartesian S/C state $\mathbf{x}_C = \left[\begin{array}{cccccc} x & y & z & v_x & v_y & v_z \end{array}\right]$ — nodes 1-6;
- the current control step size[3] h — node 7;
- the current polar S/C state $\mathbf{x}_P = \left[\begin{array}{cccccc} r & \varphi & \vartheta & \dot{r} & \dot{\varphi} & \dot{\vartheta} \end{array}\right]$ — nodes 12-15;
- the current Cartesian state of the target $\mathbf{x}_{T_C} = \left[\begin{array}{cccccc} x_T & y_T & z_T & v_{x_T} & v_{y_T} & v_{z_T} \end{array}\right]$ — nodes 16-21
- the current relative state of the S/C w.r.t. the target $\mathbf{x}_{T_C} - \mathbf{x}_C$ — nodes 22-27; and
- the currently available propellant mass m_p — node 28

[3] The control step size is dynamically adapted by a dedicated algorithm, for more details the reader is referred to [59].

The use of the sigmoid and/or tanh activation functions within InTrance requires input data to be normalised to prevent saturation of neurons. Normalisation of positions and distances is achieved by scaling them with the norm of the current S/C distance w.r.t. the central body, defined as:

$$\hat{r} = r_{\mathrm{SC}}. \tag{4.13}$$

Units of time are scaled to the time norm $\hat{t}$, which is derived from the circular velocity of a body in an orbit with radius $\hat{r}$ as:

$$\hat{t} = \frac{\hat{r}}{v_{\mathrm{circ}}} = \hat{r}\sqrt{\frac{\hat{r}}{\mu}}, \tag{4.14}$$

such that all velocities can be scaled by the velocity norm $\hat{v}$, defined as

$$\hat{v} = \frac{\hat{r}}{\hat{t}} = v_{\mathrm{circ}}(r_{\mathrm{SC}}). \tag{4.15}$$

Lastly, the propellant mass is normalised to the initial propellant mass $m_{p,0}$.

To overcome the discontinuous nature of angular values, angular inputs are provided to the ANN by both the sine and cosine of those angles. Although the ANN could be trained to overcome sudden jumps in angular inputs –while the actual change is small– it would be rather time consuming. Providing the network with the sine and cosine values provides a smooth input domain, essentially helping the ANN so that it can spend all its resources on modelling the control vector. Since the sine and cosine have a domain of (-1,1), there is no need for normalisation.

4.5.2. Output Values

The output of the NC is a vector $\mathbf{d} \in \mathbb{R}^{n_d}$, with $n_d = 3$ for solar sailing missions and $n_d = 6$ for propellant-dependent propulsion systems. This output vector can be decoded to result in the spacecraft control vector $\mathbf{u}$, which is comprised of the local-optimal thrust direction vector $\mathbf{e}_f$ and an additional throttle factor χ for propellant-dependent propulsion systems. The reader is referred to Ohndorf [59] for a complete description on the encoding scheme of the output parameters.

Table 4.3: NC input parameters [59].

Input node	Variable	NC Input	Norm	Frame
1	Spacecraft position	$x/\hat{r}$	$\hat{r}$	(rotating),cartesian
2	Spacecraft position	$y/\hat{r}$	$\hat{r}$	(rotating),cartesian
3	Spacecraft position	$z/\hat{r}$	$\hat{r}$	(rotating),cartesian
4	Spacecraft velocity	$v_x/\hat{v}$	$\hat{v}$	(rotating),cartesian
5	Spacecraft velocity	$v_y/\hat{v}$	$\hat{v}$	(rotating),cartesian
6	Spacecraft velocity	$v_z/\hat{v}$	$\hat{v}$	(rotating),cartesian
7	Control step size	$h/\hat{t}$	$\hat{t}$	n/a
8	Spacecraft azimuth angle	$\sin\varphi$	n/a	polar
9	Spacecraft azimuth angle	$\cos\varphi$	n/a	polar
10	Spacecraft elevation angle	$\sin\vartheta$	n/a	polar
11	Spacecraft elevation angle	$\cos\vartheta$	n/a	polar
12	Spacecraft azimuth rate	$\dot{\varphi}/\hat{t}$	$\hat{t}$	polar
13	Spacecraft elevation rate	$\dot{\vartheta}/\hat{t}$	$\hat{t}$	polar
14	Range	$\frac{\|\|\mathbf{r}_T-\mathbf{r}_{SC}\|\|}{\hat{r}}$	$\hat{r}$	n/a
15	Range rate	$\frac{(\mathbf{v}_T-\mathbf{v}_{SC})(\mathbf{r}_T-\mathbf{r}_{SC})}{\hat{v}\|\|\mathbf{r}_t-\mathbf{r}_{SC}\|\|}$	$\hat{v}$	n/a
16	Abs. target position	$x_T/\hat{r}$	$\hat{r}$	(rotating),cartesian
17	Abs. target position	$y_T/\hat{r}$	$\hat{r}$	(rotating),cartesian
18	Abs. target position	$z_T/\hat{r}$	$\hat{r}$	(rotating),cartesian
19	Abs. target velocity	$v_{x,T}/\hat{v}$	$\hat{v}$	(rotating),cartesian
20	Abs. target velocity	$v_{y,T}/\hat{v}$	$\hat{v}$	(rotating),cartesian
21	Abs. target velocity	$v_{z,T}/\hat{v}$	$\hat{v}$	(rotating),cartesian
22	Rel. target position	$(x_T-x)/\hat{r}$	$\hat{r}$	(rotating),cartesian
23	Rel. target position	$(y_T-y)/\hat{r}$	$\hat{r}$	(rotating),cartesian
24	Rel. target position	$(z_T-z)/\hat{r}$	$\hat{r}$	(rotating),cartesian
25	Rel. target velocity	$(v_{x,T}-v_x)/\hat{v}$	$\hat{v}$	(rotating),cartesian
26	Rel. target velocity	$(v_{y,T}-v_y)/\hat{v}$	$\hat{v}$	(rotating),cartesian
27	Rel. target velocity	$(v_{z,T}-v_z)/\hat{v}$	$\hat{v}$	(rotating),cartesian
28	Propellant mass	$m_p/m_{p,0}$	$m_{p,0}$	n/a

5

Low-Thrust Gravity Assist Trajectory Optimisation Implementation

This research was initiated with the idea of extending the current multi-phase framework of InTrance to accommodate Gravity Assists (GAs), thereby making it a truly versatile and smart global trajectory optimisation method. Both for high- and low-thrust applications, the domain of possible mission scenarios is greatly increased due to the enormous ΔV potential of gravity assist manoeuvres. InTrance can easily be extended to high-thrust, and with the inclusion of GA optimisation, trajectories to the outer planets and beyond can be optimised in an autonomous fashion. However, the main idea behind InTrance is the optimisation of low-thrust trajectories, which also stand to benefit from the inclusion of GAs; the almost instantaneous ΔV due to the slingshot manoeuvre around a celestial body can, for one, significantly decrease transfer times and/or propellant usage. Besides the GA accounting for part of the ΔV budget itself, due to the Oberth effect, thrusting at later stages will be more efficient.

The initial version of InTrance by Dachwald [17] was, to the best of the authors knowledge, the first method that could achieve autonomous trajectory design using Reinforcement Learning (RL), albeit solely for single-phase low-thrust heliocentric trajectories. The updated and revised version by Ohndorf [59] further increased the capabilities of the method, allowing the autonomous design of both helio- and planetocentric low-thrust trajectories, which could be comprised of multiple phases. It then became possible to, for instance, design intricate multiple asteroid rendezvous missions; optimising the trajectory leg from an Earth-bound orbit to the first asteroid, optimising all the transfer time, propellant usage, and dwell time at that asteroid, after which the S/C continued to the next asteroid, and so on. By adding the capability for GAs, InTrance is brought one step closer to becoming that fully autonomous method that can design optimal trajectories for *any* mission, as no mission specific assumptions or models are used. Instead, the problem is tackled from a RL perspective, which essentially derives its own underlying model.

This thesis has focussed on the development of such a versatile and autonomous method, that is, to build on the multi-phase framework of InTrance for the design of GA trajectories. A central question in the work was how GAs can optimally be implemented in InTrance, which is discussed in Section 5.1. The remainder of this chapter discusses the changes that have been made to InTrance to allow GA-trajectories to be optimised, starting with the GA-model in Section 5.3, followed by the final and initial conditions of phases ending at and starting from a GA-body in Section 5.4, after which the changes to the chromosome and fitness function are discussed in Section 5.5 and 5.6, respectively.

5.1. Single Phase vs. Multiphase Gravity Assist Strategy

Theoretically, a NC should be able to steer a S/C to perform a GA without externally being directed to do so, and autonomously find a sequence of gravity assists whenever favourable. However, from the work of Dachwald [17], Ohndorf [59] and Carnelli [14], GAs are hardly ever found with neurocontrol. This can be explained by the fact that a gravity assist solution is within a very small subset of the solution space, and even when an individual performs a GA, it is very unlikely it will be an optimal one and will therefore go extinct

in favour of more optimal non-GA performing individuals. Hence, a method has to be implemented which actively directs the search to perform GAs.

GAs can be implemented within InTrance in two fundamentally different ways. The first being to re-tackle the problem that Carnelli [13] faced, that is; can a single NC be constructed that autonomously finds optimal (possibly multi-) GA trajectories to reach some target? The major upside of this method is that physical correctness is assured at every step along the trajectory, and that no external tools are necessary to compute an optimal sequence of gravity assists. Thereby staying true to the beliefs behind InTrance; to be a global trajectory optimisation method that requires no expert knowledge or supervision. The second approach is to break-up the trajectory to some target that performs an intermediate GA in two phases; where a phase n is to end at a GA-body, and a new phase $n+1$ starts from that point onward. This method is conceptually much simpler to implement, but has some caveats. The major downside of such a method is that it would require the user to specify a sequence of GAs. Both strategies will be discussed in the subsequent sections; detailing how such a method could work, what problems could be encountered, and possible methods to alleviate those problems.

5.1.1. Single Neurocontroller

Theoretically, according to Kolmogorov's theory (Section 3.5.3), a suitably sized and trained single NC should be able to model any arbitrary underlying function, and hence should also be able to represent a steering strategy that directs a S/C along a trajectory while performing (multiple) optimal GAs.

Within the single NC approach, a GA would simply be located somewhere along the trajectory leg of a certain phase, see Figure 5.1. The NC will have to learn the benefits of performing a GA, and a proper fitness function should drive the optimisation to perform them. The major benefit of this strategy is its physical validity and independence of a user-supplied GA-sequence, staying true to the definition of a *smart* low-thrust trajectory optimisation method as given in Section 3.3. Whenever a GA is favourable, the NC should perform one, without the user indicating at which body or at what time to perform one.

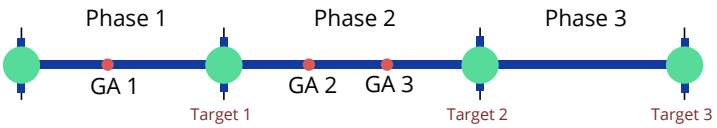


Figure 5.1: Single NC GA-approach within the multi-phase framework; a GA is some intermediate action within a phase.

The optimisation of GA trajectories with a single NC within Dachwald's version of InTrance has been tackled before by Carnelli [13]. The hypothesis was that since a single NC was found to perform solar photonic assists by Dachwald [17], GAs should also be possible. Unfortunately, Carnelli was unsuccessful in this approach and had to revert to a separate local optimisation scheme whenever crossing a SOI. Carnelli implemented a local optimisation method termed steepest ascent, utilising the gradient of the fitness function to find the optimum value. Whenever a trajectory found itself inside a SOI, the steepest ascent algorithm was employed to find the optimal injection point, which was then passed on as the new state to the NC. The downsides of this method are that it results in an instantaneous jump in the position of the S/C and that this method is only applicable to single GA and single-phase trajectories. Carnelli's work is no longer implemented in the current multi-phase version of InTrance.

The first problem that Carnelli encountered when using a single NC to optimise GAs was that only a very small percentage of simulated trajectories actually crossed a SOI, and therefore having a very limited learning environment. Carnelli tried to increase the number of SOI crossings by artificially inflating the size of the SOI, but to no avail. Carnelli furthermore found that GA performing individuals always performed a GA sub-optimal, that is, they did not achieve to penetrate the B-plane at the optimal aiming point distance. This resulted in premature death as their fitness was lower than non-GA performing individuals. In the hopes of alleviating this problem, Carnelli split the population into two sub-species; a group that performed one or more GAs, and a group that did not. This subdivision should give the GA performing individuals a chance to optimise their aiming point, before competing with non-GA individuals. Carnelli furthermore gave an additional boost in fitness to GA-performing individuals by multiplying the conventional fitness function J with a term dependent on how many GAs it performed. However, even with all of these alterations, InTrance

would eventually still converge to a non-GA performing solution or an infeasible one.

Carnelli lists another difficulty for the single NC strategy; the gravity field around a GA-body is dynamic and therefore information about a possible GA location cannot be passed to offspring when analysing different launch dates. One possible adaptation to alleviate this problem is to supply the NC with the relative states between the S/C and possible GA-bodies. However, since a GA sequence is not supplied, these states should be supplied relative to all possible GA-bodies. Supplying the NC with relative states for all possible bodies at each control step greatly increases the size of the ANN (n bodies $\times$ 6 coordinates) and therefore results in a much larger search space, which greatly hinders the optimisation of the internal parameters.

Although some more work with this approach could be done, including implementation of a more suitable fitness function, this approach is not further explored within this work. Even if the single NC GA approach could successfully be implemented to perform a GA, it is very unlikely to work for more intricate GA-sequences, such as an EVEV-GA[(1)]. After all, the chances of simulating such a trajectory are very slim, and even slimmer for generating four *optimal* GAs. Instead, a more promising strategy is to use a separate NC for each GA, which is discussed next.

5.1.2. Multiple Neurocontrollers

Within the multi-NC approach, a GA is a target in itself and marks the end of a phase, see Figure 5.2. This example shows the same sequence as in Figure 5.1, but is now a six phase trajectory instead of the three phase trajectory in the single NC GA approach. The major advantage of the multi-NC GA approach is that a NC can solely be trained to reach a GA-body and optimise that GA, with no risk of non-GA performing individuals outperforming GA-performing individuals. After all, not reaching a GA target is a constraint violation which is heavily penalised trough the fitness function as discussed in Section 4.3. The downside of this method is that a sequence of gravity assists has to be supplied, which is not directly in line with the definition of a *smart* low-thrust trajectory optimisation method. However, these sequences can be supplied by external heuristic sequencing algorithms, which could even be directly implemented within InTrance in the future.

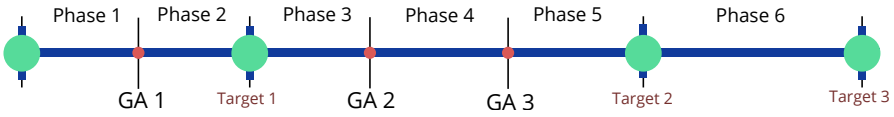


Figure 5.2: Multi-NC GA-approach within the multi-phase framework; a GA is a target in itself and ends a phase.

Since a GA is now a target in itself, the NC is supplied with the relative position of the S/C to the GA-body at each control step, as detailed in Section 4.5.1. Therefore, information about a possible GA location can be passed on to offspring without increasing the search space of the NC of the GA-phase, as would have been required in the single NC approach discussed above. By ending a phase right after a GA has been performed, the optimisation is steered to first optimise each phase in reaching its target. Only once each phase has reached its respective target, the optimisation shifts to decreasing the phase transition gaps and optimisation of the overall objective function. Therefore, a NC has ample opportunity to learn the benefits of a GA and how to properly perform one.

5.2. Gravity Assist Architecture

The above discussed multi-NC GA approach has been implemented within InTrance and the flow of this approach is shown in Figure 5.3 for an exemplary two phase mission with a single GA. The first phase starts at the launch body and ends with a GA when exiting the SOI of the GA-body. The second phase then starts on the rim of the SOI of the GA-body and takes the S/C towards its final target where it either performs a flyby or rendezvous, or is captured.

The evolutionary algorithm provides a candidate solution ξ_j, which is essentially the chromosome containing the internal parameters of both NCs and initial conditions of both phases. The internal NC parameters $\boldsymbol{\pi}_1$

[(1)] Earth-Venus-Earth-Venus-Gravity Assist; common in trajectories to Mercury.

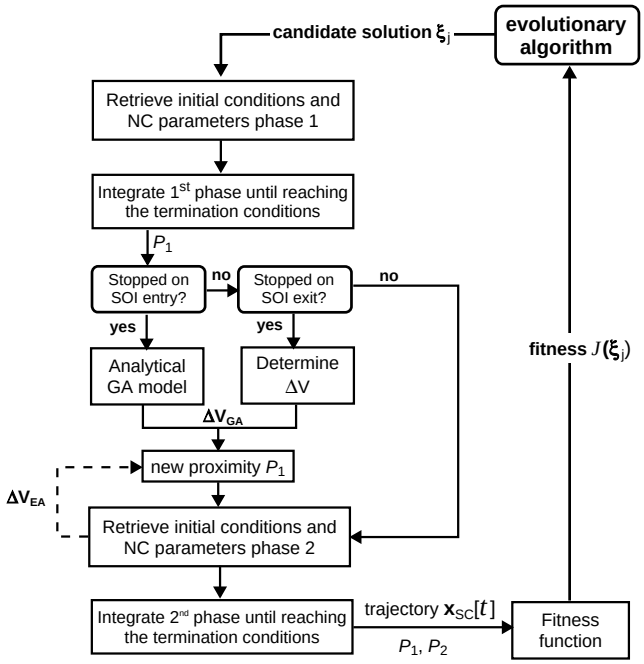


Figure 5.3: Multi-phase low-thrust gravity assist trajectory optimisation within InTrance. Figure shows the flow of an exemplary two phase mission; the first phase ending with a GA, the second starting from the GA-body towards its final target.

and initial conditions $\boldsymbol{x}_{SC,1}(t_0)$ of the first phase are then extracted from the chromosome from which the NC is initialised and integration commences from the initial state. The NC is sampled at each control time step to provide a control $\boldsymbol{u}(t_i)$ and integration continues until reaching a termination condition, exactly similar to the architecture as shown in Figure 4.1. Additional termination conditions have been added to support GAs, depending on whether the GA is modelled analytically or numerically, see Section 5.3. When using the analytical GA model, the integration is stopped when entering the SOI. These entry conditions are then supplied to the analytical GA model such that the exit state and resulting $\boldsymbol{\Delta V}$ can be determined. When using the numerical model, the trajectory is integrated until crossing the SOI for the second time, i.e. when exiting the SOI, after which the $\boldsymbol{\Delta V}$ is readily available. If the particular individual never crosses the first target's SOI, the integration is stopped from any of the other termination conditions (max MET, max integration steps, etc.).

The proximity is determined at each integration step, but contrary to any of the other trajectory types already implemented within InTrance, when one of the new termination conditions are triggered, a new proximity for the GA-phase is determined. This new proximity is a function of the actual achieved $\boldsymbol{\Delta V}$ due to the gravity assists, and the $\boldsymbol{\Delta V}_{\mathrm{EA}}$ which is an initial condition of the following phase and actively optimised by the EA. This $\boldsymbol{\Delta V}_{\mathrm{EA}}$ and the new proximity functions are described in more detail in Sections 5.4 and 5.6, respectively.

With the integration of the first phase terminated and the proximity determined, the second phase can be initialised by extracting the internal NC parameters $\boldsymbol{\pi}_2$ and initial conditions $\boldsymbol{x}_{SC,2}(t_0)$ of the second phase from the same individual ξ_j. These initial conditions of the second phase are different than those of other trajectory types, and are described in detail in Section 5.4. Integration then starts from the initial conditions and the NC of the second phase is again sampled at every control step. Integration continues until reaching one of the regular termination conditions relating to the flyby, rendezvous or capture problem. This results in a complete trajectory which is then evaluated through the fitness function as discussed in Section 4.3. The evolutionary algorithm then supplies a new individual and the process repeats until convergence.

5.3. Gravity Assist Model

Two gravity assist models have been implemented in InTrance. The first is an analytical method derived from Keplerian two-body dynamics as described in Section 2.2. This method has the advantage of not having to integrate while within the SOI, where the control and integration step sizes are the smallest due to the increased

dynamic gravitational environment. A possible downside of this method is that no thrust can be applied while inside the SOI, thereby not taking effect of the Oberth effect and having no possibility to alter the closest approach distance from the instance of entering the SOI onwards. As Section 2.4 showed, especially when using thrust to lower the closest approach, the effects can be large. However, a lower closest approach distance can also be achieved by having a different aiming point on the B-plane, which is fully determined by the entry conditions into the SOI. Furthermore, historical missions have not allowed thrusting while performing a GA due to its increase in mission risk.

Nevertheless, it is interesting to investigate the effect of thrust while performing a GA on the overall trajectory. InTrance autonomously determines when to engage the thruster, and it would be interesting to see if InTrance returns a more optimal solution when allowing thrust during the GA portion versus without. To this end, a numerical model for the GA portion has also been implemented, which simply records the entry state into the SOI and continues the integration until exiting the SOI. The $\boldsymbol{\Delta V}$ due to the gravity assist portion is then readily available by subtracting the entry velocity from the exit velocity in the heliocentric frame. The numerical model furthermore has the added benefit that it ensures physical correctness with a high accuracy (depending on the integration scheme) and can take disturbing bodies into account. However, it is computationally much heavier and increases the search space.

5.3.1. Analytical Gravity Assist Model

The analytical gravity assist model is shown in Algorithm 5.1 and is derived from the geometry and equations described in Section 2.4. The algorithm's input is made up out of the the heliocentric position and velocity vectors of both the S/C and GA-body at the entry date into the SOI ($\boldsymbol{R}_1$, $\boldsymbol{V}_1$, $\boldsymbol{R}_{b,1}$, $\boldsymbol{V}_{b,1}$), the entry date itself (T_{ent}), the standard gravitational parameter of the GA-body (μ_b), and a C++ body object (pointer) of the GA-body (*body*), which is necessary to retrieve certain parameters of that body. The algorithm will then determine the resulting exit state out of the SOI ($\boldsymbol{R}_2$, $\boldsymbol{V}_2$) and resulting GA parameters $\boldsymbol{\Delta V}$, the time spent within the SOI (T_{GA}), and the closest approach distance r_p.

The algorithm first determines the S/C state relative to the GA-body in lines 1-3, after which the angles γ, β and deflection angle δ (see Figure 2.5) can be determined from the eccentricity e, hyperbolic excess velocity v_∞ and b-point aiming distance b in lines 4-9. The closest approach distance r_p and time spent within the SOI T_{GA} are then determined from the hyperbolic anomaly H, the semi-major axis a and eccentricity e in lines 10-13. The exit state is determined from realising that the result of a GA is a rotation of the incoming velocity vector around the angular momentum vector in the GA-bodycentric frame. The position vector is rotated over an angle α and the velocity vector over an angle δ through rotation matrix

$$\mathrm{R}(\varphi, \boldsymbol{h}) = \begin{bmatrix} \cos\varphi & \frac{h_3}{h}\sin\varphi & -\frac{h_2}{h}\sin\varphi \\ -\frac{h_3}{h}\sin\varphi & \cos\varphi & \frac{h_1}{h}\sin\varphi \\ \frac{h_2}{h}\sin\varphi & -\frac{h_1}{h}\sin\varphi & \cos\varphi \end{bmatrix}, \tag{5.1}$$

in which $\boldsymbol{h} = [h_1,\ h_2,\ h_3]$ and $h = \|\boldsymbol{h}\|$. This is shown in lines 14-17. The heliocentric state of the GA-body at the instance of exiting the SOI is then retrieved in lines 18 and 19, after which the heliocentric exit state and resulting $\boldsymbol{\Delta V}$ is determined in lines 20-22. Line 23 determines the velocity difference between the exit state of the S/C and the velocity of the GA body at the instance the S/C exits the SOI. This $\boldsymbol{\Delta V}_b$ will be elaborated on in Section 5.4.2.

5.4. Initial and Final States of Gravity Assist Phases

A phase ending with a GA ends when the S/C exits the SOI. When using the analytical GA model, the integration is stopped when the S/C enters the SOI, after which the exit state is determined analytically and added to the state vector history. Hence, the trajectory within the SOI is not integrated and therefore results in a gap in the state history of that phase. When using the numerical model, the trajectory is integrated until exiting the SOI, so that the GA-phase again ends with a state positioned on the rim of the SOI. When the evaluated individual does not enter the SOI, the integration is stopped at one of the termination conditions (e.g. max. MET or max. arrival date) and whatever state the S/C has attained at that point is the final state of that phase.

The initial conditions of a phase succeeding a GA-phase are directly supplied by the Evolutionary Algorithm (EA). The initial position has to be on the rim of the SOI, as a GA-phase always ends on the rim of the SOI when a GA has been performed. To this end, two additional parameters have been encoded onto the chromosome;

Algorithm 5.1: $[\boldsymbol{R}_2,\ \boldsymbol{V}_2,\ \boldsymbol{\Delta V},\ \boldsymbol{\Delta V}_b,\ T_{GA},\ r_p]$ = AnalyticalGravAssModel($\boldsymbol{R}_1,\ \boldsymbol{V}_1,\ \boldsymbol{R}_{b,1},\ \boldsymbol{V}_{b,1},\ T_{\text{ent}},\ \mu_b$, body)

Input : Heliocentric position and velocity vectors of both the GA-body and SC at instance of entering the SOI $\boldsymbol{R}_1$, $\boldsymbol{V}_1$, $\boldsymbol{R}_{b,1}$, $\boldsymbol{V}_{b,1}$, time of entry into SOI T_{ent}, the standard gravitational parameter of the GA-body μ_b, and the body object *body*

Output: Heliocentric position and velocity vectors of the SC when exiting the SOI $\boldsymbol{R}_2$, $\boldsymbol{V}_2$, two types of $\boldsymbol{\Delta V}$ which are the result of the GA, the time spent within the SOI T_{GA}, and the closest approach distance r_p.

Result: Find the exit state and resulting ΔV due to a gravity assist.

// Determine GA-bodycentric state of SC at SOI entry

$\boldsymbol{r}_1 = \boldsymbol{R}_1 - \boldsymbol{R}_{b,1}$

$\boldsymbol{v}_1 = \boldsymbol{V}_1 - \boldsymbol{V}_{b,1}$

$r_1 = \|\boldsymbol{r}_1\|, \quad v_1 = \|\boldsymbol{v}_1\|$

// determine GA-geometry

$v_\infty = \sqrt{v_1^2 - \frac{2\mu}{r_1}}$ *// hyperbolic excess velocity*

$\beta = \frac{\boldsymbol{r}_1 \cdot \boldsymbol{v}_1}{r_1 v_1}$ *// angle between velocity direction and distance vector*

$\gamma = \pi - \beta$ *// b-distance angle*

$b = r \sin\gamma$ *// b-distance*

$e = \sqrt{1 + b^2 \frac{v_\infty^4}{\mu^2}}$ *// eccentricity*

$\delta = 2\text{asin}\left(\frac{1}{e}\right)$ *// deflection angle*

// determine time spent within SOI and closest approach distance

$a = \left(\frac{v_1^2}{\mu} - \frac{2}{r_1}\right)^{-1}$ *// semi-major axis*

$H = \text{acosh}\left[\left(1 + \frac{r}{a}\right)\sin\frac{\delta}{2}\right]$ *// hyperbolic anomaly*

$T_{\text{GA}} = 2\sqrt{\frac{a^3}{\mu}}\,(e\sinh H - H)$

$r_p = -\frac{\mu}{v_\infty^2}\,(1 - e)$

// determine GA-bodycentric SC SOI exit state

$\boldsymbol{h} = \boldsymbol{r}_1 \times \boldsymbol{v}_1$ *// angular momentum vector*

$\alpha = \pi - \delta + 2\gamma$ *// position rotation angle*

$\boldsymbol{r_2} \leftarrow \mathtt{R}(\alpha, \boldsymbol{h}) \cdot \boldsymbol{r_1}$

$\boldsymbol{v_2} \leftarrow \mathtt{R}(\pm\delta, \boldsymbol{h}) \cdot \boldsymbol{v_1}$

// determine heliocentric SC SOI exit state and ΔV both relative to entry velocity and body velocity

$\boldsymbol{R_{b,2}} \leftarrow \mathtt{getPosVec}\left(T_{\text{ent}} + T_{\text{GA}}, \text{body}\right)$ *// retrieve heliocentric position at SOI exit of GA-body*

$\boldsymbol{V_{b,2}} \leftarrow \mathtt{getVelVec}\left(T_{\text{ent}} + T_{\text{GA}}, \text{body}\right)$ *// retrieve heliocentric velocity at SOI exit of GA-body*

$\boldsymbol{R_2} = \boldsymbol{R}_{b,2} + \boldsymbol{r}_2$

$\boldsymbol{V_2} = \boldsymbol{V}_{b,2} + \boldsymbol{v}_2$

$\boldsymbol{\Delta V} = \boldsymbol{V_2} - \boldsymbol{V_1}$

$\boldsymbol{\Delta V}_b = \boldsymbol{V_2} - \boldsymbol{V_{b,2}}$

the initial launch position azimuth α_{GA} and the initial launch position elevation δ_{GA}. A bodycentric reference frame centred at the GA-body is then used to set the initial position at a distance R_{SOI} with an azimuth and elevation supplied by the EA, see Figure 5.4. The azimuth and elevation are allowed to be varied by the EA within user defined bounds, indicated in red in Figure 5.4 for the azimuth angle. This initial bodycentric position is then converted to Cartesian elements and added to the heliocentric position of the GA-body at the launch date, resulting in the heliocentric initial position of a phase which directly follows a GA-phase. The launch date of this phase is also a parameter on the chromosome and is optimised within user defines bounds.

The initial velocity of a phase following a GA-phase is either defined relative to the entry velocity into the SOI of the preceding phase or relative to the GA-body velocity at launch. Both methods will be tested and both

have their advantages and drawbacks, as will be detailed next.

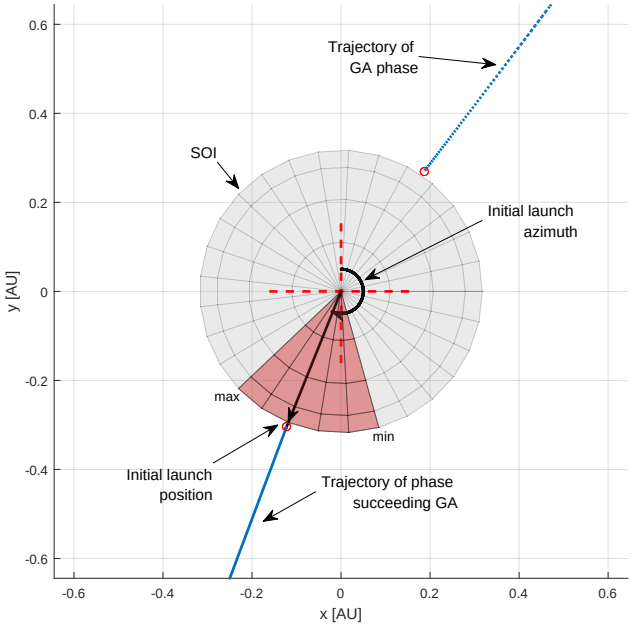


Figure 5.4: Definition of initial position of a phase directly following a GA in the XY-plane. Elevation angle completes the right-handed system.

5.4.1. Velocity Relative to SOI Entry Velocity in GA-Phase

For clarity, the phase ending with a GA will be refereed to as phase 1, the subsequent phase which starts from the rim of the SOI of the GA-body is referred to as phase 2. Within this approach, the initial velocity of phase 2 is given by the heliocentric entry velocity into the SOI as found in phase 1 plus a $\boldsymbol{\Delta V}_{\mathrm{EA}}$ generated by the evolutionary algorithm, see Figure 5.5. Hence, once the actually generated $\boldsymbol{\Delta V}$ due to the GA in phase 1 is equal to the required $\boldsymbol{\Delta V}_{\mathrm{EA}}$ needed to reach the next target in phase 2, and the final position of phase 1 is equal to the initial position of phase 2, a physically valid GA has been performed.

The initial velocity of the second phase is shown with a red vector in Figure 5.5, starting from the initial position of the second phase. The green vector is the entry velocity into the SOI as found in the first phase, and the purple vector shows the $\boldsymbol{\Delta V}_{\mathrm{EA}}$ as supplied by the EA. This $\boldsymbol{\Delta V}_{\mathrm{EA}}$ is optimised within user defined bounds by the EA, to which end three parameters are encoded onto the chromosome; the magnitude of the velocity increment ΔV_{EA}, the azimuth angle $\alpha_{\Delta V}$ and the elevation angle $\delta_{\Delta V}$. These angles are defined in a similar fashion as for the initial position and are shown in Figure 5.5; the only difference is that the reference frame is centred at the initial position of the second phase on the rim of the SOI.

The advantage of this method is the physical correctness and straightforward use of the $\boldsymbol{\Delta V}$, which is as its definition the difference in heliocentric velocity at exit and entry into the SOI. The downside, however, is that the two phases are inherently coupled. The second phase will only be able to reach its target if the entry velocity into the SOI of the first phase is sufficiently close to the actual optimal injection velocity. Whenever a vastly wrong GA, or no GA at all, is performed in the first phase; the second phase is also unable to reach its target. This goes against the overall idea of the multi-phase framework, in which each phase is first optimised separately and independent from other phases, and only after all phases reach their respective targets, the focus shifts to finding the overall optimal trajectory. Therefore, in an effort to partly decouple the phases, an alternative approach is detailed next.

5.4.2. Velocity Relative to Gravity Assist Body

Instead of defining the initial velocity of the second phase relative to the entry velocity into the SOI of the first phase, it is defined relative to the velocity of the GA-body at launch of the 2nd phase, see Figure 5.6. The initial velocity of the second phase is then given by the GA-body velocity at launch plus a $\boldsymbol{\Delta V}_{\mathrm{EA,b}}$ supplied

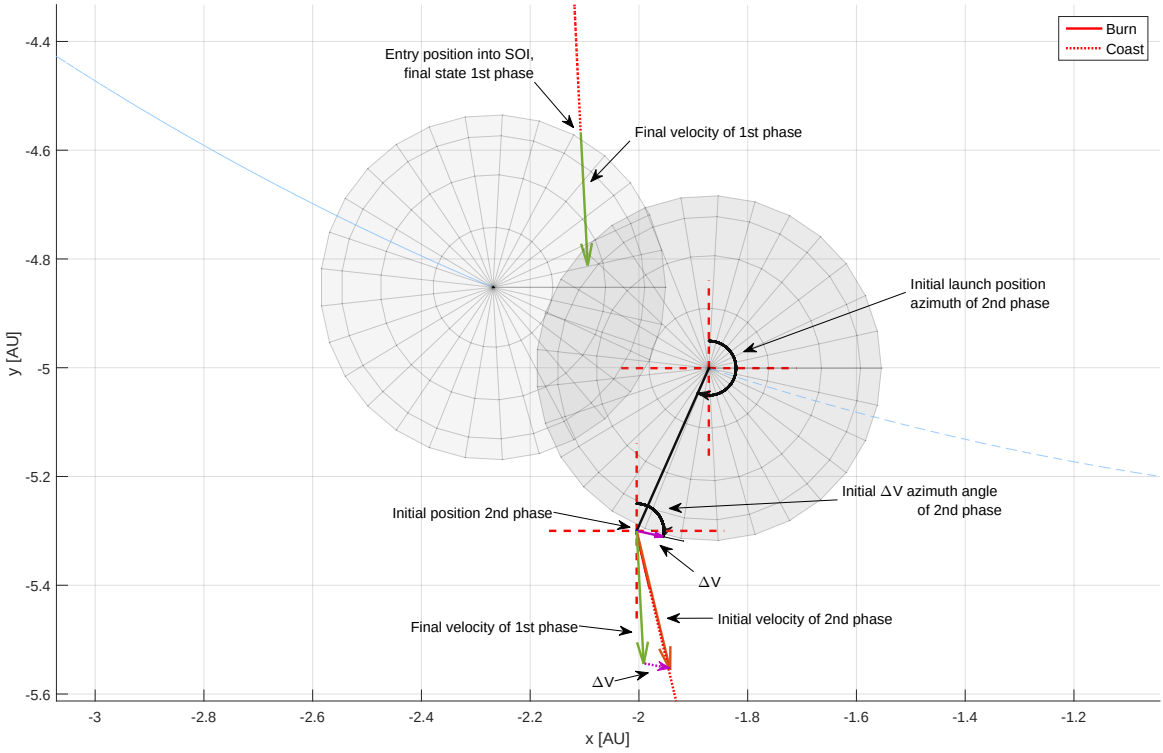


Figure 5.5: Definition of initial state in a phase directly following a GA using the entry velocity of the preceding phase.

by the EA. The $\boldsymbol{\Delta V}_{\mathrm{EA,b}}$ in this approach cannot be interpreted as a classical $\boldsymbol{\Delta V}$ due to a GA, but rather as a 'launch velocity' or $C3$ relative to the GA-body as if it were supplied by a launch vehicle. Nevertheless, if the $\boldsymbol{\Delta V}$ generated due to the GA is also defined relative to the GA-body at the instance the S/C exits the SOI, it can be compared to the initial launch $\boldsymbol{\Delta V}_{\mathrm{EA,b}}$ of the second phase. Once these values match, and the final position of phase 1 and initial position of phase 2 match, a physically valid GA has been performed.

The initial velocity of this approach is shown with a red vector in Figure 5.6. The blue vectors indicate the velocity of the GA body at the launch date of the second phase, and the purple vectors the initial $\boldsymbol{\Delta V}_{\mathrm{EA,b}}$. The velocity increment $\boldsymbol{\Delta V}_{\mathrm{EA,b}}$ is again optimised by the EA within user defined bounds, defined in the same polar reference frame as described in Section 5.4.1.

The major upside of this definition of an initial launch $\boldsymbol{\Delta V}_{\mathrm{EA,b}}$ is that the second phase is decoupled from the first phase. The NC of the second phase is then free to optimise towards its first goal of reaching the next target, and only after all targets in all phases have been reached, the optimisation focusses on decreasing phase transition gaps and maximising the actual objective function. The downside of this method is that it is less intuitive and therefore more difficult to interpret.

5.5. Chromosome

The EA chromosome $\boldsymbol{\xi}_i$ contains both the internal parameter set ${}^{(j)}\boldsymbol{\pi}$ of the ANN of all NCs for each phase j, and the simulation parameters which affect either S/C design or mission design of all phases j. The two to be optimised S/C design parameters of each phase are the propellant mass $m_{p,0}$ and the characteristic power output of the electric propulsion system $P_{e,0}$. The propellant mass is optimised in such a way that enough propellant should be brought on-board for both the current phase plus enough reserve for all succeeding phases. The characteristic power output can be optimised if a mission requires so, or simply be set at a constant value if a specific propulsion system is chosen. The simulation parameters affecting the mission design are the launch/departure date t_0, the hyperbolic excess velocity $\boldsymbol{v}_\infty$ at t_0, and the initial state vector $\boldsymbol{x}_{SC}(t_0)$. Two additional simulation parameters have been added if the phase starts from a GA-body; the launch position azimuth α_{GA} and launch position elevation δ_{GA}, as described in Section 5.4. Table 5.1 shows the encoding of the above described simulation parameters.

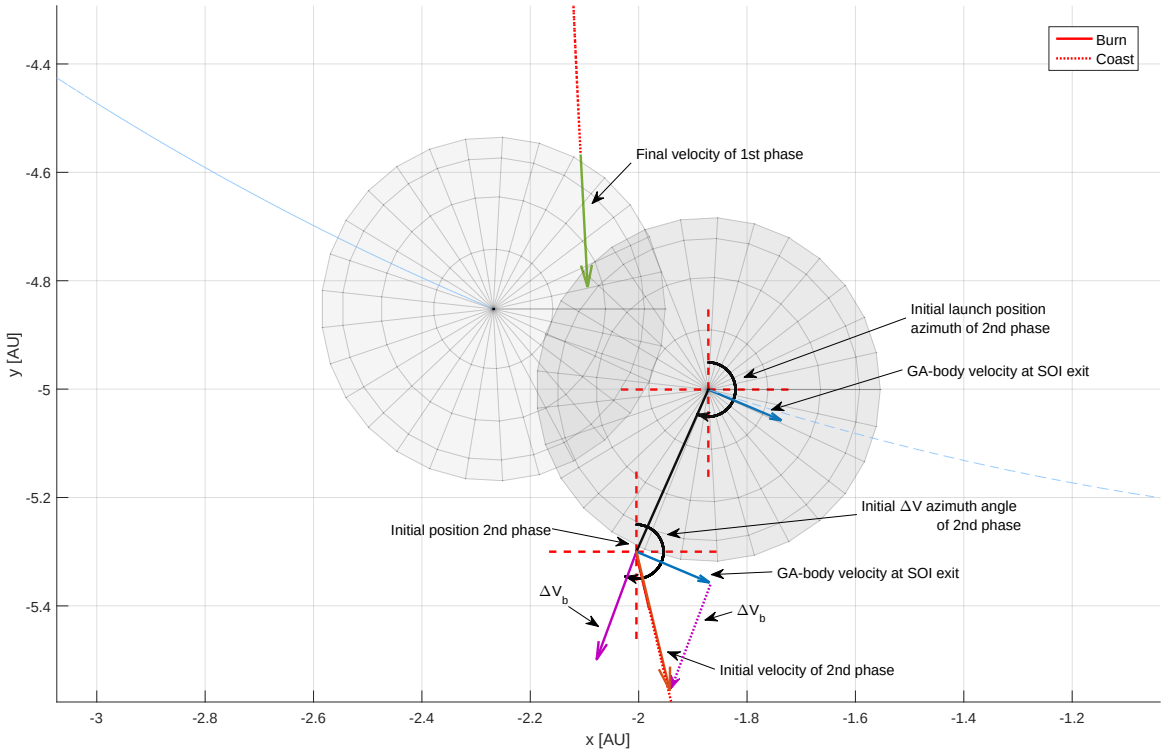


Figure 5.6: Definition of initial state in a phase directly following a GA using the body velocity at SOI exit.

Table 5.1: Simulation parameter encoding onto the chromosome.

Relative Locus	Allele	Associated Variable	Affected Initial Condition
0	r_0	t_0	t_0
1	r_1	v_∞	$\mathbf{v}_\infty$
2	r_2	α_∞	$\mathbf{v}_\infty$
3	r_3	δ_∞	$\mathbf{v}_\infty$
4	r_4	$m_{p,0}$	$\mathbf{x}_{SC}(t_0)$
5	r_5	$P_{e,0}$	$\mathbf{x}_{SC}(t_0)$
6	r_6	r, v	$\mathbf{x}_{SC}(t_0)$
7	r_7	r, v	$\mathbf{x}_{SC}(t_0)$
8	r_8	r, v	$\mathbf{x}_{SC}(t_0)$
9	r_9	φ, ϑ	$\mathbf{x}_{SC}(t_0)$
10	r_{10}	φ, ϑ	$\mathbf{x}_{SC}(t_0)$
11	r_{11}	φ, ϑ	$\mathbf{x}_{SC}(t_0)$
12	r_{12}	ζ, Φ	$\mathbf{x}_{SC}(t_0)$
13	r_{13}	ζ, Φ	$\mathbf{x}_{SC}(t_0)$
14	r_{14}	ζ, Φ	$\mathbf{x}_{SC}(t_0)$
15	r_{15}	α_{GA}	$\mathbf{x}_{SC}(t_0)$
16	r_{16}	δ_{GA}	$\mathbf{x}_{SC}(t_0)$

InTrance can optimise the initial launch $\mathbf{v}_\infty$, thereby simulating the launch velocity supplied by a launch vehicle. However, its parameters on the chromosome are only used in the very first phase, when the S/C is launched. No use is made of alleles r_1 to r_3 after the very first phase in the classical version of InTrance, which is why they are re-used for a phase following a GA and supply the ΔV_{EA}, the azimuth angle $\alpha_{\Delta V}$ and the elevation angle $\delta_{\Delta V}$, as described in Section 5.4.

5.6. Fitness

The fitness determination within InTrance in the multi-phase framework has been detailed in Section 4.3, and remains largely the same with the implementation of GAs. However, since a new target state –the gravity assist– has been added, new proximity functions and transition conditions have been defined.

In case of the gravity assist problem, the deviation δ is not used. Furthermore, there is no need to keep track of the best solution along the trajectory, as entering the SOI always stops the integration. It would be logical to define the gravity assist problem analogous to the rendezvous problem; with a δ_r which compares the current distance to the gravity assist body at each integration step to the maximum allowed distance (R_{SOI}), and a δ_v which compares the ΔV supplied by the gravity assists with the ΔV which is actually used by the second phase and optimised by the evolutionary algorithm. However, the required closest approach distance is unknown a priori, hence it is not possible to define a deviation δ_r. After all, the definition of δ_r drives the spacecraft as close as possible to the desired target distance. In principle, a velocity deviation δ_v could be defined as detailed in Section 4.3.1. However, it is often the case that the δ_r of the second phase reaches a very large negative number; orders of magnitude larger than that of a velocity deviation of the GA phase. Therefore, a better result in the second phase will have a much larger effect on the overall fitness than a better result in the first phase. This will result in the optimisation to focus more on the second phase and almost neglect the first phase, giving rise to physically invalid trajectories in which no GA is performed.

In order to overcome the above mentioned problems, use has been made of dedicated proximity functions for the gravity assist phase. The proximity function can be regarded as a sub-fitness function of the GA-phase, driving the NC to perform an optimal GA which supplies the $\Delta\mathbf{V}$ required by its succeeding phase to reach its target. Initially, the proximity function is defined in such a way that it drives the optimisation towards reaching the SOI of the gravity assist body. Once the SOI is entered, the integration is stopped and the gravity assist model takes over. The $\Delta\mathbf{V}_{\mathrm{GA}}$ determined by the GA model is then compared to the required $\Delta\mathbf{V}_{\mathrm{EA}}$ as determined by the evolutionary algorithm. The proximity function of the gravity assist problem is then defined as:

$$P_{\mathrm{GA}} = \begin{cases} -c_1 \left(r - R_{\mathrm{SOI}}\right)^2 - M_1 & \text{if } r > R_{\mathrm{SOI}} \\ \min\left[-c_2 \cdot \mathrm{abs}\left(\frac{\Delta\mathbf{V}_{GA} - \Delta\mathbf{V}_{EA}}{\Delta\mathbf{V}_{EA}}\right)\right] + c_2 \cdot 0.01p & \text{if } r \leq R_{\mathrm{SOI}} \end{cases}, \tag{5.2}$$

where the min() function here means the minimum proximity in one of the three Cartesian velocity directions x, y, z. Additionally, c_1 and c_2 are scaling factors, p the maximum allowed percent deviation between the two $\Delta\mathbf{V}$ vectors, $r = |\mathbf{r}|$, and M_1 a sufficiently large negative value such that performing a gravity assist is always favoured over not reaching the SOI, regardless of how large the deviation between the two $\Delta\mathbf{V}$ vectors is.

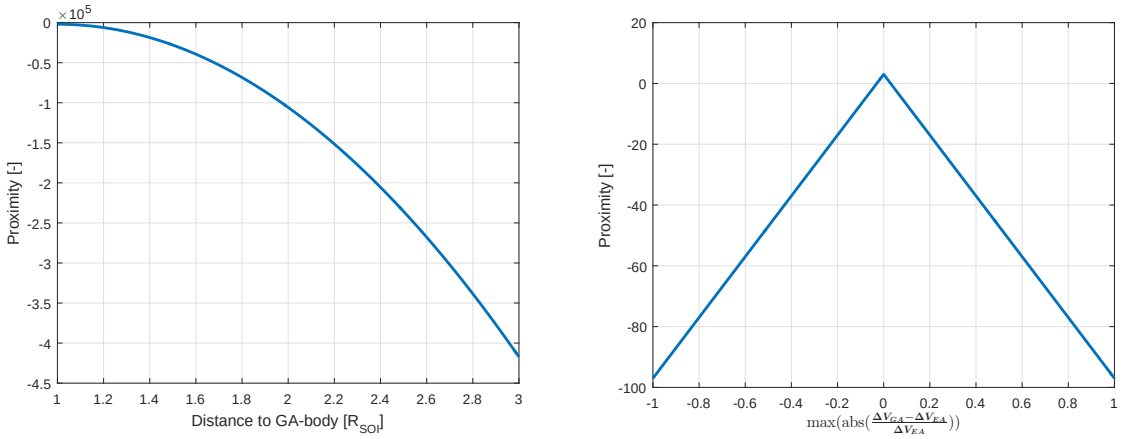
Figure 5.7 shows a plot of the proximity functions of the gravity assist problem. The figure on the left side shows the function when the spacecraft has not yet reached the SOI, with a scaling factor of $c_1 = 10^6$ and $M_1 = 2000$. The figure on the right shows the proximity function when the S/C performs a GA, with scaling factor $c_2 = 100$ and the maximum allowed deviation $p = 3\%$. These scaling factors have been used throughout this work for all simulated trajectories. It is clear that the sub-fitness of an individual performing a gravity assist is much higher than that of non-GA performing individuals, regardless of the deviation between the two velocity increment vectors.

Once the sub-fitness functions J_P (see Section 4.3.3) becomes positive, then all proximity constraints have been met and the gravity assist has provided the required $\boldsymbol{\Delta V}$ needed by its subsequent phase to reach its respective target. The overall fitness $J(\boldsymbol{\xi}_j)$ for individual $\boldsymbol{\xi}_j$ is then determined by the objective function $\mathcal{O}$ (such as transfer time minimisation) and the phase transition condition violations V_i as

$$J(\boldsymbol{\xi}) = \begin{cases} J_P & \text{if } J_P < 0 \\ \mathcal{O} \cdot \prod_{i \in (dw,m,s)} \frac{1}{1+V_i} & \text{if } J_P \leq 0 \end{cases}. \tag{5.3}$$

Hence, once the phase-specific constraints have been met, any fitness improvement results from an increased value of the respective objective function $\mathcal{O}$ or from a decrease of the phase transition condition violations V_i, or both.

The phase transition condition violations V_i provide a measure on how well two phases connect. Naturally, the position at the end of the first phase and the initial position of the second phase cannot be too far apart. The same is true for the final and initial velocity of the two phases, the propellant mass, and the arrival and departure dates. Since the proximity function is defined in such a way that the final velocity of the GA-phase



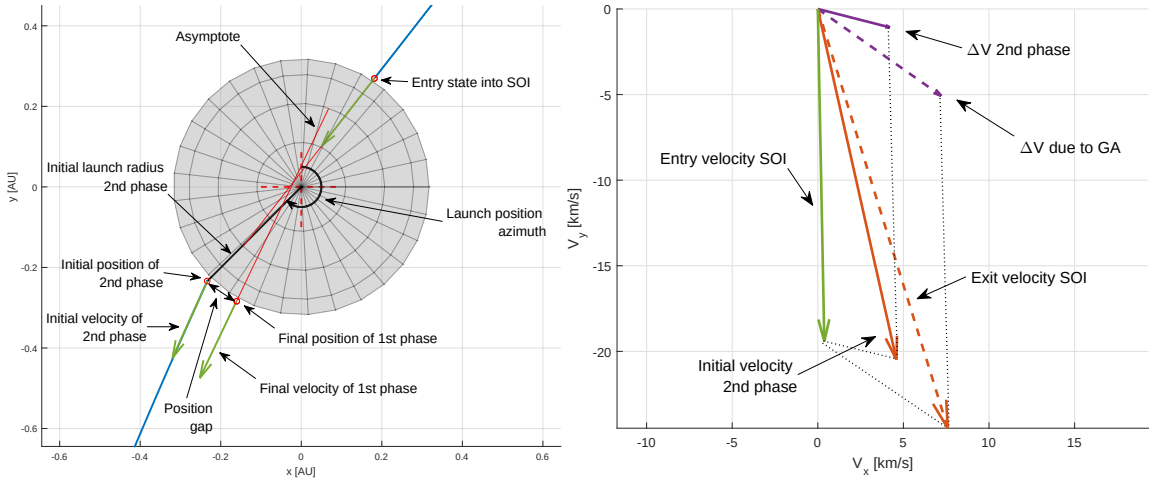
(a) if $r > R_{\text{SOI}}$ (b) if $r \leq R_{\text{SOI}}$

Figure 5.7: Proximity functions of the analytical gravity assist phase. $c_1 = 10^6$, $M_1 = 2000$, $c_2 = 100$, $p = 3$.

and initial velocity of its succeeding phase are approximately equal when optimised, there is no need for a phase transition condition on the velocity. However, since a p% deviation between the velocity is allowed in the proximity (see eq. 5.3), a transition conditions is applied to the velocity to further decrease the gap. The second phase's initial position can be anywhere within predefined bounds on the edge of the SOI, and hence also requires a phase transition condition. Example geometry of these position and velocity gaps are shown in Figure 5.8. The state phase transition condition between a GA-phase and its succeeding phase is formulated as:

$$V_{\text{pos}} = \begin{cases} \frac{\|\boldsymbol{r_2} - \boldsymbol{r_1}\|}{\|\mathbf{r}_2\|} & \text{if } V_{\text{pos}} > \theta \\ 0 & \text{if } V_{\text{pos}} \leq \theta \end{cases}, \quad V_{\text{vel}} = \begin{cases} \max\left(\frac{\text{abs}(\boldsymbol{v}_2 - \boldsymbol{v}_1)}{\boldsymbol{v_2}}\right) & \text{if } V_{\text{vel}} > \theta \\ 0 & \text{if } V_{\text{vel}} \leq \theta \end{cases}, \quad V_{\text{state}} = \max\left(V_{\text{pos}}, V_{\text{vel}}\right). \tag{5.4}$$

Hence, the position transition condition is the distance between the GA-bodycentric final position of the first phase ($\mathbf{r}_1$) and initial position of the second phase ($\mathbf{r}_2$), divided by the radius of the SOI of the gravity assist body $\|\mathbf{r}_2\| = R_{\text{SOI}}$. The velocity transition condition is the maximum difference in any of the three Cartesian directions divided by the initial velocity of the 2nd phase in that direction. When any of the transition condition violations are smaller than some user defined threshold θ, it is set to zero and indicates there is no violation. The state transition condition is then given by the maximum of the velocity and position transition conditions.



(a) Position gap (b) Velocity gap

Figure 5.8: Example geometry to showcase transition conditions between a GA-phase and its succeeding phase.

5.6.1. Algorithm

The determination of the proximity when using the analytical GA model is shown in Algorithm 5.2. The algorithm is called at each integration step and starts with retrieving the then current heliocentric state and converts them to the GA-bodycentric reference frame. The proximity is then set as defined in equation 5.2, with $r > R_{\text{SOI}}$, awarding an individual for getting closer to the SOI. If the S/C has not entered the SOI, the algorithm stops and integration continues. If the S/C has entered the SOI, then a termination condition flag is set which stops the integration loop (line 11). The analytical GA model (see Algorithm 5.1) is then called which returns the $\boldsymbol{\Delta V}$ due to the GA and SOI exit state. This exit state is then saved on line 13 such that it can be accessed by the second phase and used to determine the transition conditions later on. The $\boldsymbol{\Delta V}$ due to the GA is then compared to the $\boldsymbol{\Delta V}_{\text{EA}}$ used in the initial conditions of the 2nd phase, which first has to be converted to Cartesian elements (lines 14-15). The proximity when the S/C has performed a GA is then determined and set in lines 17 and 18, where line 16 determines which $\boldsymbol{\Delta V}$ definition to use (see Section 5.4). Lastly, if the S/C collides with the GA-body, the proximity is set to a large negative value.

If the S/C has entered the SOI or if any of the other termination conditions have been reached, the integration loop is stopped and the evaluation of the next phase commences. If no termination conditions are reached, the trajectory is propagated to the next state and the proximity is again determined.

Algorithm 5.2: Proximity(s, b, $\boldsymbol{\Delta V}_{\text{EA}}$)

Input : spacecraft object s, body object b, initial ΔV of next phase $\boldsymbol{\Delta V}_{\text{EA,pol}}$ in polar coordinates
Output: none
Result: Determine and set the proximity at each integration step.

// Retrieve current heliocentric state, date, μ_b, and R_{SOI} and convert state to GA-bodycentric
1 date = `getDate()`
2 $\boldsymbol{R} = s \rightarrow getPosVec()$, $\boldsymbol{V} = s \rightarrow getVelVec()$
3 $\boldsymbol{R}_b = b \rightarrow getPosVec()$, $\boldsymbol{V}_b = b \rightarrow getVelVec()$
4 $\boldsymbol{r} = \boldsymbol{R} - \boldsymbol{R}_b$, $\boldsymbol{v} = \boldsymbol{V} - \boldsymbol{V}_b$, $r = \|\boldsymbol{r}\|$, $v = \|\boldsymbol{v}\|$
5 $R_{\text{SOI}} = b \rightarrow$ `RSOI`(date)
6 $\mu_b = b \rightarrow$ `getGM`
7 `setStateSOIExit`($\boldsymbol{R}, \boldsymbol{V}$, date)

// determine and set deviation/proximity
8 prox = $-c_1 (r - R_{\text{SOI}})^2 - M_1$
9 `setProx`(0, prox)

// if SC is within the SOI, determine different deviation/proximity.
10 **if** $0.9 R_{SOI} < r \leq R_{SOI}$ **then**
11 `setIsSOIEntry`(true) *// termination condition -> stops the integration loop*
// determine and set heliocentric exit state of SC
12 $[\boldsymbol{R}_2,\ \boldsymbol{V}_2,\ \boldsymbol{\Delta V}_{\text{GA}},\ \boldsymbol{\Delta V}_{\text{GA,b}},\ T_{GA},\ r_p]$ = `AnalyticalGravAssModel`$\left(\boldsymbol{R}, \boldsymbol{V}, \boldsymbol{R_b}, \boldsymbol{V_b}, \text{date}, \mu_b, b\right)$
13 `setStateSOIExit`($\boldsymbol{R}_2,\ \boldsymbol{V}_2,$ date + T_{GA})
// convert initial $\boldsymbol{\Delta V}$ of subsequent phase from polar to Cartesian coordinates
14 $\Delta V_{\text{EA}} = \boldsymbol{\Delta V}_{\text{EA,pol}}(0)$, $\alpha = \boldsymbol{\Delta V}_{\text{EA,pol}}(1)$, $\delta = \boldsymbol{\Delta V}_{\text{EA,pol}}(2)$
15 $\boldsymbol{\Delta V}_{EA} = \Delta V_{\text{EA}} \cdot [\sin\alpha\cos\delta,\ \cos\alpha\cos\delta,\ \sin\delta]$
// determine and set proximity
16 **if** *relative to entry velocity* **then** $\boldsymbol{\Delta V} = \boldsymbol{\Delta V}_{\text{GA}}$ **else** $\boldsymbol{\Delta V} = \boldsymbol{\Delta V}_{\text{GA,b}}$
17 prox = $\min\left[-c_2 \cdot \text{abs}\left(\frac{\boldsymbol{\Delta V} - \boldsymbol{\Delta V}_{\text{EA}}}{\boldsymbol{\Delta V}_{\text{EA}}}\right)\right] + c_2 \cdot 0.01 p$
18 `setProx`(0, prox)
// penalise if impacting the surface
19 **if** $r_p < (b \rightarrow getMeanRadius())$ **then** `setProx`$(0, -1e7)$
end

6

Verification and Validation

InTrance has undergone significant validation efforts by both Ohndorf [59] and Dachwald [17] to validate the overall search behaviour and optimised solution plus all components such as the EA and numerical integrator. Therefore, the current validation effort focusses on the newly implemented components, and the analytical GA-model in particular. It is generally not possible to determine whether a solution to the low-thrust trajectory optimisation problem is the actual global optimum, therefore, in order to assess the convergence behaviour and validity of the obtained solution, it will be compared to reference solutions found in literature. To his end, a low-thrust New Horizons like trajectory is re-computed using the developed extended version of InTrance to assess the overall validity and optimality of the gravity assist implementation within InTrance.

6.1. Analytical Gravity Assist Model

An analytical gravity assist model has been incorporated within InTrance to reduce complexity and computation time. Since this is an analytical model, based on two-body dynamics (see Section 5.3), its validity in a more realistic environment has to be evaluated. To this end, several gravity assists have been simulated using an external RK4(5) integrator implemented in MATLAB, which can then be compared to the results from the analytical GA-model.

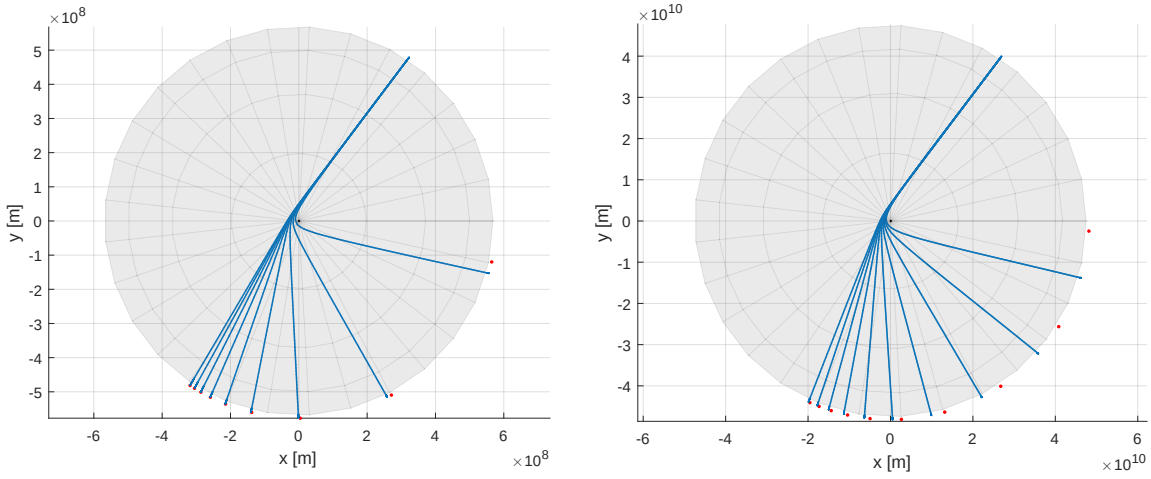
Figure 6.1 shows a selection of simulated gravity assists at both Jupiter and Mars in the planetocentric frame, all starting from the same initial position[(1)] on the rim of the SOI and having the same initial velocity direction, but a different entry velocity ($\approx v_\infty$) magnitude. The trajectories are integrated using a constant step-size of $h = 10$ s for the trajectories at Mars and $h = 50$ s for the trajectories at Jupiter. The trajectories are integrated from the initial conditions in the heliocentric frame under the main attracting gravitational force from the Sun, the gravitational attraction of the GA-body is then modelled as a disturbing potential. The heliocentric states of the GA-bodies have been retrieved through SPICE at an arbitrary start date of 27 sept 2007. The resulting SOI exit states have also been computed with the analytical GA-model given in Algorithm 5.1, and are indicated with red dots in Figure 6.1.

A lower entry velocity results in a lower closest approach distance, giving rise to a larger deflection angle δ, as described in Section 2.3. This effect can clearly be seen in the simulated trajectories, where the trajectories with the largest entry velocity ($v_{\infty,max} = 20$ km/s at Jupiter and 5 km/s at Mars) are only deflected over a few degrees, whereas the trajectories with the lowest entry velocities ($v_{\infty,min} = 5$ km/s at Jupiter and 1 km/s at Mars) are deflected over approximately 115°. The positional SOI exit deviation between the numerical and analytical method can clearly be seen to increase with increasing deflection angles. The deviation is especially apparent in the case of a Jupiter GA, in which the S/C spends more time inside the SOI then in the Mars case.

The deviations between the analytical and numerical gravity assists have been plotted in Figure 6.2, using the same initial conditions as in the above simulations. The deviations are shown in percentages, determined as

$$\Delta r = \frac{\|\boldsymbol{r}_{\text{ana}} - \boldsymbol{r}_{\text{num}}\|}{\|\boldsymbol{r}_{\text{num}}\|} \times 100, \quad \Delta v = \frac{\|\boldsymbol{v}_{\text{ana}} - \boldsymbol{v}_{\text{num}}\|}{\|\boldsymbol{v}_{\text{num}}\|} \times 100, \quad \Delta\delta = \frac{\delta_{\text{ana}} - \delta_{\text{num}}}{\delta_{\text{num}}} \times 100, \tag{6.1}$$

(1) Initial Cartesian position: $\mathbf{r} = R_{\text{SOI}} \cdot [\cos 56°, \sin 56°, 0]^{\text{T}}$; initial Cartesian velocity: $\mathbf{v} = v_\infty \cdot [-\cos 53°, -\sin 53°, 0]^{\text{T}}$

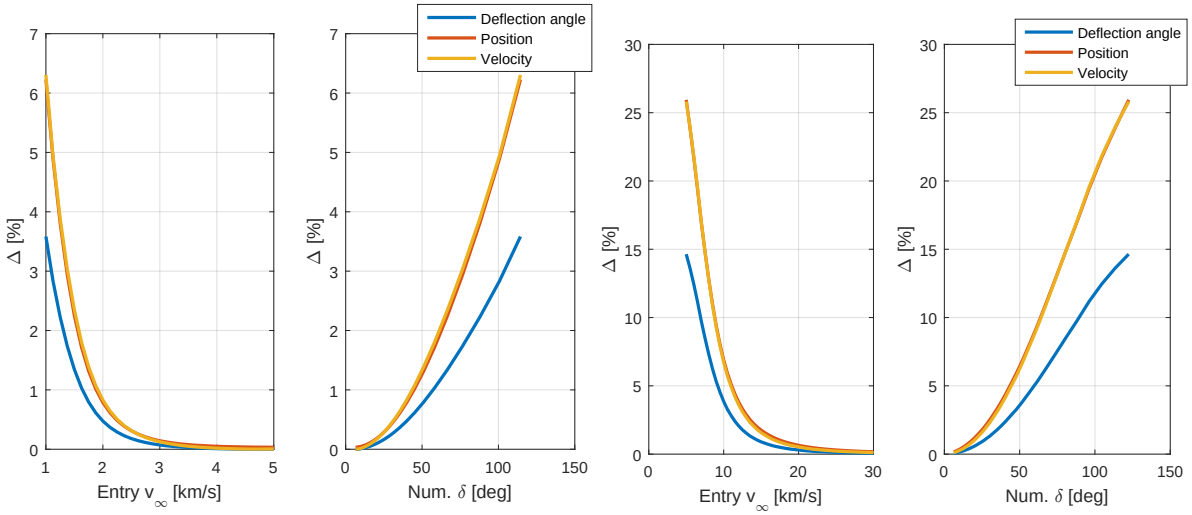


(a) Mars; entry $v_\infty = 1 : 0.5 : 5$ km/s, $h = 10$ s.

(b) Jupiter; entry $v_\infty = 5.5 : 1.5 : 20$ km/s, $h = 50$ s.

Figure 6.1: Trajectory while performing a GA at both Mars (left) and Jupiter (right) for varying entry velocities from RK4(5) integrator (blue lines) and corresponding analytical GA-model exit positions (red dots).

where r and v are, respectively, the position and velocity in the GA-bodycentric frame at SOI exit and δ the deflection angle. Indices *num* and *ana* indicate whether the parameters are determined from the numerical or analytical method, respectively. These figures again show that the deviation of these parameters between the two methods quickly increase with increasing deflection angle. The deviation between the positions and velocities are almost identical, going up to about 6.3% in the Mars case and about 26% in the Jupiter case. The deviation in deflection angle between the two methods is smaller, attaining a maximum of 3.5% in the Mars scenario and 15% in the Jupiter case. The deviation in the deflection angle is most telling, as the position and velocity error follow a practically linear relation with the SOI radius. The deviation is partly explained by the inclusion of a gravitational acceleration from the Sun in the numerical method, thereby stepping away from the simplifying two-body dynamics of the analytical model. The analytical method furthermore simplifies the problem by making certain assumptions, such as assuming the entry velocity into the SOI to be equal to the incoming hyperbolic excess velocity v_∞. It should also be noted that the numerical method does not necessarily give the correct result, as it is highly dependent on the integration step size h, for which no variable stepsize method has been implemented in the simple RK4(5) implementation used here. Especially for the extreme gravity assists, the step size is rather important due to the rapidly changing gravitational environment around the point of closest approach.



(a) Mars

(b) Jupiter

Figure 6.2: Deviation between analytical GA-model and numerically integrated trajectory in deflection angle δ, exit position r_2 and exit velocity v_1 versus the entry $v_\infty = v_1$ and numerically determined deflection angle. Entry position same as in Figure 6.1.

It is problem dependent whether these deviations can be deemed acceptable. Generally, deflection angles in missions are not as extreme as simulated above and a trajectory resulting from a preliminary design tool such as InTrance is usually run through a local optimiser to generate a higher fidelity solution, which are usually able to smooth out these gaps. Furthermore, as detailed in Section 5.6, InTrance allows a certain user defined state gap between phases. As long as the deviation from the analytical GA model is in the same order of magnitude as the allowed state transition gap, the analytical model can generally be deemed acceptable. If more extreme gravity assists are simulated, or when a higher fidelity solution is desired, use can be made of the implemented numerical method.

6.2. New Horizons Validation Case

In order to assess the overall validity of the GA implementation in InTrance, an optimised solution is compared to results found in literature. A New Horizons like trajectory will be optimised, utilising low-thrust NEP and a Jupiter GA to generate the required ΔV to perform a flyby of Pluto. The reference solutions are generated by Vasile et al. [75] using a direct collocation method based on a finite element transcription implemented in the software DITAN [8], and by Carnelli [13] using his implementation of gravity assists in the single-phase framework version of InTrance.

6.2.1. Validation Data

The software DITAN [8] has been used by Vasile et al. [75] to optimise trajectories to Pluto and beyond. They generated trajectories using SEP and aero gravity assists at Jupiter to reach Pluto, but relied on NEP and conventional GAs to generate more favourable mission scenarios. With NEP, they calculated several trajectories to Pluto which performed GAs at several different bodies along the way. One of their simplest and fastest solutions was termed 'fast transit', a low-thrust version of the New Horizons mission, which performs a GA at Jupiter and a fast flyby (v_∞ = 15.337 km/s) of Pluto. For increased scientific return, they also computed trajectories which perform slow flybys (v_∞ = 50 m/s) of Pluto, trajectories which are captured by Pluto, trajectories which perform a Petit Tour (flybys/GA at the inner planets) prior to performing a flyby of Pluto, and lastly trajectories which perform a Grand Tour (flybys/GA at the outer planets) prior to reaching Pluto.

Carnelli [13] worked on implementing gravity assists back in 2005 in the original single-phase version of InTrance developed by Dachwald [17]. As described in Sections 4.1 and 5.1, Carnelli relied on a local optimisation scheme to determine the optimal injection point into the B-plane, after having been unsuccessful in using a NC to directly optimise gravity assists. Carnelli validated his implementation with the New Horizons like solution found by Vasile et al., described above.

The results by Vasile et al. [75] and Carnelli [13] are summarised in Table 6.1 and show excellent agreement between them. Both reference trajectories have been optimised to have the lowest overall mission time, with the solution from DITAN resulting in a MET of 3181 days and Carnelli's implementation resulting in 3185 days. Vasile et al. constrained their optimisation to reach a specific v_∞ of 15.4 km/s at Pluto for which it used a total of 34.5 kg of propellant. Contrary, Carnelli constrained his solution by allowing a maximum propellant usage of 34.5 kg and does not report on the resulting v_∞ at Pluto.

Table 6.1: Validation data for a low-thrust New Horizons like trajectory.

	Vasile et al. [75]	Carnelli [13]
Launch V_∞	12 km/s	12 km/s
Launch date	Jan. 19, 2006	Jan. 4, 2006
Jupiter encounter	Feb. 23, 2007	Feb 16, 2007
Pluto arrival	Oct. 4, 2014	Sep. 23, 2014
Mission elapsed time	3181 days	3185 days
Total thrust time	6600 hours	Unknown
Pluto v_∞	15.337 km/s	Unknown
Mass at departure	600 kg	600 kg
Mass at arrival	565.5 kg	565.3 kg
Thrust	34 – 40 mN	40 mN
Specific impulse	1700 – 4000 s	3000 s

6.2.2. InTrance Input Parameters

An optimisation run of InTrance is performed with the intention of generating a trajectory that is broadly similar to the above discussed validation data. To this end, only the steering strategy is optimised globally, constraining the maximum allowed propellant mass, the launch, and arrival dates to be in the vicinity of the reference trajectory.

The trajectory is split in two phases, with the first starting at launch from Earth and ending with entry into the SOI of Jupiter. The second phase starts on the rim of the SOI after having performed a GA, as described in Section 5.4, and ends with a flyby of Pluto. The S/C drymass is set at 565 kg and both phases are propelled by a single NEP thruster with bang-bang control, a maximum thrust of 40 mN and specific impulse of 3000 s.

The set of mission defining input parameters used in this optimisation run of InTrance is completed by:

Table 6.2: Mission defining parameters used to generate an optimised New Horizons like low-thrust trajectory similar to the validation date. Values in square brackets indicate ranges.

	Phase 1 – Earth to Jupiter	**Phase 2 – Jupiter to Pluto**
Launch date	[Jan. 18, 2006 – Jan. 20, 2006]	[Jan. 20, 2007 – Feb. 09, 2007]
Arrival date	[Jan. 20, 2007 – Feb. 09, 2007]	[Sep. 30, 2014 – Oct. 30, 2014]
Launch V_∞ or ΔV_{GA}	[5.00 – 12] km/s	[4 – 6.5] km/s
Target	Jupiter GA	Pluto flyby
Maximum target distance	N/A	10^7 km
Dry mass	565 kg [75]	565 kg
Propellant consumption	[10 – 22] kg	[5 – 13] kg

6.2.3. Optimisation Run

The solution from InTrance is plotted in Figure 6.3, with the left portion showing the complete heliocentric trajectory and the right showing a close-up to properly display the thrust arcs and direction (green arrows). The S/C is launched from Earth on Jan. 19, 2006 with a launch V_∞ of 12 km/s. A continuous thrust of 40 mN is supplied from launch for 184 days, about 50% of the flight time of the first phase. The S/C enters the SOI of Jupiter on Jan. 26, 2007 and performs its closest approach 30 days later, on Feb. 24, 2007. The S/C leaves the SOI another 31 days thereafter, resulting in a $\Delta V = \left\| \boldsymbol{V}_{2,i} - \boldsymbol{V}_{1,f} \right\|$ of 5.29 km/s, which commences the second phase on Mar. 26, 2007. Thrust is applied from departure for 54 days; about 2% of the flight time of the second phase. The S/C finally performs a flyby of Pluto on Oct. 3, 2014 with a relative v_∞ to Pluto of 15.371 km/s. Note that the trajectory while the S/C is within the SOI is computed with an external RK4(5) integrator and is not an output of InTrance, and is solely added for presentation purposes.

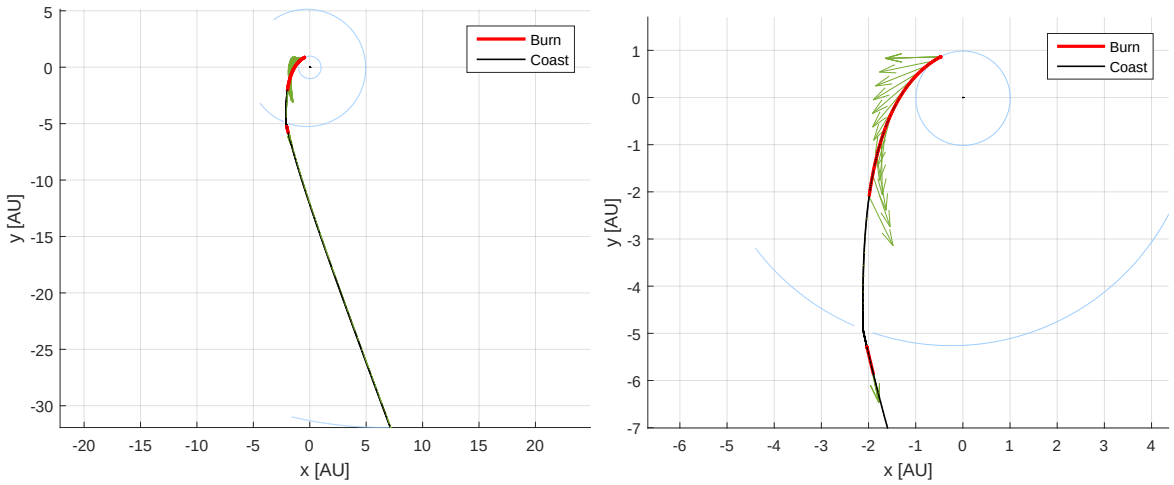


Figure 6.3: Heliocentric trajectory from an InTrance run of the validation case.

The results of the simulation are summarised in Table 6.3, together with the results from Vasile et al. [75]

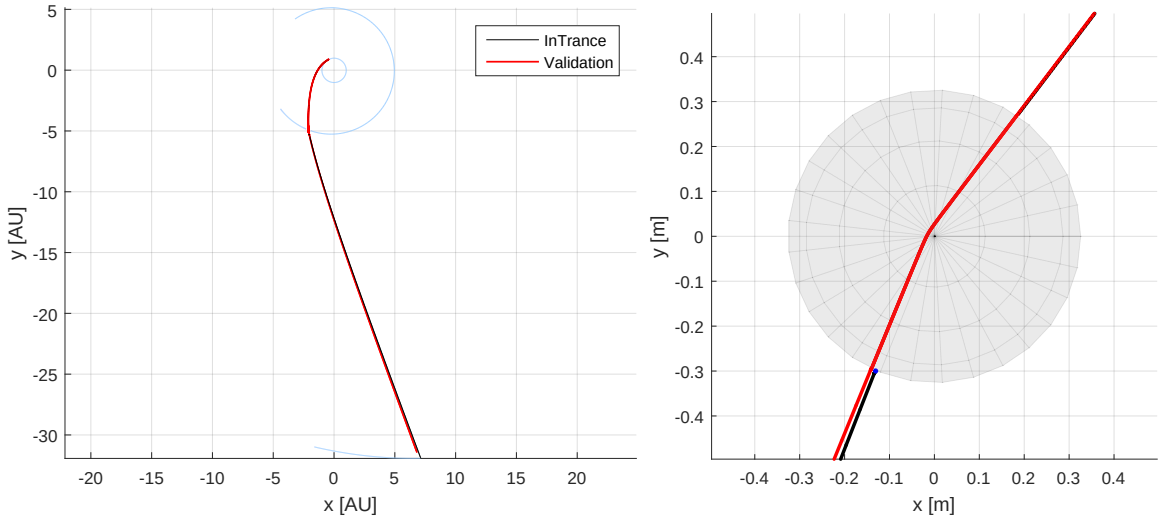
and Carnelli [13], and shows very good overall agreement. In terms of flight time, the generated trajectory by InTrance is one day faster than the solution computed with DITAN and five days faster than the result from Carnelli. All three results use about the same amount of propellant, to within 0.4 kg, and the three v_∞ at Pluto are almost identical. The solution generated by InTrance uses about the same amount of propellant as the one generated with DITAN, but its total thrust time is 13% shorter. This must be caused by a lower thrust force or lower I_{sp} used within DITAN, versus the fixed 40 mN used in InTrance. Vasile et al. do not explicitly state the thrust force for this specific trajectory, but state that they used a thrust force ranging between 34 and 40 mN for all generated trajectories.

Table 6.3: Results and reference data for the New Horizons like validation simulation.

	Vasile et al. [75]	InTrance	Carnelli [13]
Launch V_∞	12 km/s	12 km/s	12 km/s
Launch date	Jan. 19, 2006	Jan. 19, 2006	Jan. 4, 2006
Jupiter encounter	Feb. 23, 2007	Feb. 24, 2007	Feb 16, 2007
Pluto arrival	Oct. 4, 2014	Oct. 3, 2014	Sep. 23, 2014
Mission elapsed time	3181 days	3180 days	3185 days
Total thrust time	6600 hours	5739 hours	Unknown
Pluto v_∞	15.337 km/s	15.371 km/s	Unknown
Mass at departure	600 kg	600 kg	600 kg
Mass at arrival	565.5 kg	565.1	565.3 kg
Thrust	34 – 40 mN	40 mN	40 mN
Specific impulse	1700 – 4000 s	3000 s	3000 s

6.2.4. External Integration

In order to further asses the validity of the result of InTrance, the trajectory is integrated from the initial conditions at launch from Earth with an external RK4(5) integrator implemented in MATLAB. At each integration step, the control as supplied by InTrance is applied. The states of the attracting bodies are retrieved from InTrance and the integration is performed in the heliocentric reference frame, with each phases' initial and target bodies as the only disturbing bodies. The result of this external integration is shown in Figure 6.4 in red, together with the result of InTrance in black, in both the heliocentric frame and a close-up of the GA portion in the GA-bodycentric frame.



(a) Heliocentric

(b) GA-bodycentric

Figure 6.4: External RK4(5) integration of InTrance's result of the validation case.

It can be seen, at least on this scale, that the externally integrated trajectory during the first phase is broadly equal to the solution from InTrance. A clear deviation can be seen in the planetocentric frame in the initial

conditions of the second phase as supplied by InTrance, which is mostly caused by the inaccuracy in the analytical GA model. The exit position out of the SOI is shown with a blue dot, which is very close to the initial position of the second phase in InTrance. The deviation between the analytically determined exit position and the numerical integration is about $0.017R_{SOI,♃} \approx 11R_{♃}$. The two trajectories diverge further over time, which is attributed to the relatively large step size (up to 10 days) used in the external integration. The external integration uses the control step size returned in the output files of InTrance as integration step size to accurately supply the thrust, however, the internal integration step size of InTrance usually differs from the control step size. InTrance therefore generates a higher fidelity solution than can be generated externally afterwards.

The heliocentric distance and velocity of the S/C are plotted in Figure 6.5, which further show the deviation between the solution generated by InTrance and the externally re-integrated solution. The deviation is small for the first phase, but jumps at the start of the second phase due to the inaccuracy of the analytical GA-model and state gaps between the two phases. The deviation grows over time due to the difference in integration step size, which is especially apparent in the second phase.

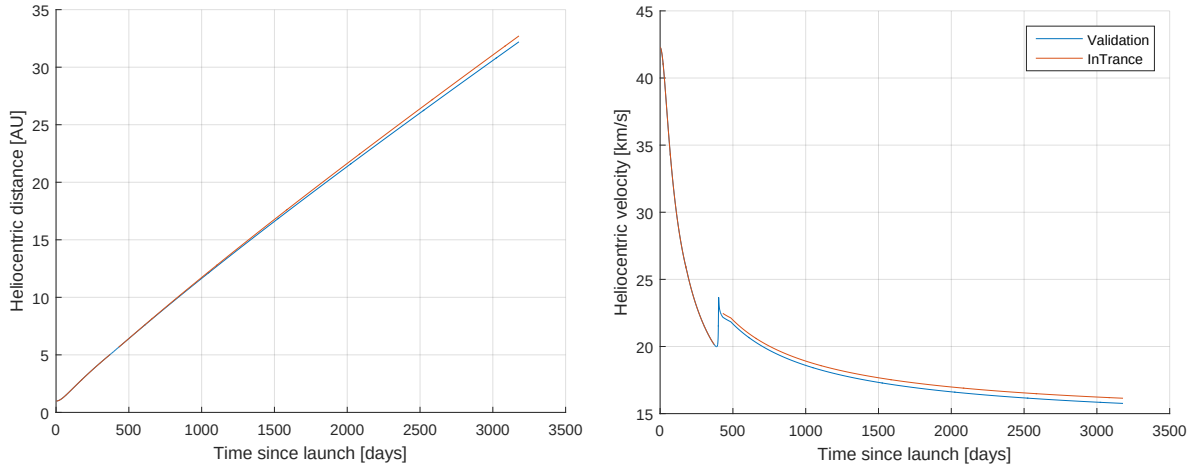


Figure 6.5: Heliocentric distance and velocity of both InTrance and the externally re-integrated solution.

6.2.5. Initial and Final Conditions at Gravity Assist

Figure 6.6 shows a close-up of the GA portion of the trajectory in the heliocentric frame. Both the output of InTrance and the externally integrated GA portion are plotted, including the corresponding initial velocity of the second phase. The right side of this figure shows a close up of the initial state of the second phase (indicated by ②), and includes the exit states as computed by the analytical GA model used in the InTrance optimisation (indicated by ①) and by an external integration from the entry conditions onward (indicated by ③).

The resulting trajectory from InTrance has a state phase transition gap of $\epsilon_T = 16.7$ s, $\epsilon_r = 4.86 \cdot 10^5$ km= $0.0098\mathrm{R_{SOI}}$ and $\epsilon_v = 290$ m/s between its analytically determined SOI exit position (①) and initial condition of the 2$^{\mathrm{nd}}$ phase (②). These values give phase transition violations (see Section 5.6) of $V_{\mathrm{pos}} = 0.00983$ and $V_{\mathrm{vel}} = 0.00979$, which are less than the user defined allowed threshold of 0.01. The gap can be decreased further by decreasing the threshold, but is deemed satisfactory for the current application.

The complete set of both heliocentric and GA-bodycentric coordinates of the three states shown in Figure 6.6 are tabulated in Table 6.4. Note that only the initial distance of the 2$^{\mathrm{nd}}$ phase is exactly equal to the SOI radius of Jupiter (as computed by InTrance), as per its definition. Ideally, the exit positions of both the analytical GA model and numerical integration should also end exactly on the rim of the SOI. However, the analytical method rotates the very first state that is within the SOI, which due to a finite step size, is not exactly on the rim of the SOI. The external numerical re-integration does not end exactly on the rim of the SOI again due to its finite stepsize, ending the integration with the first state that is larger than the SOI radius.

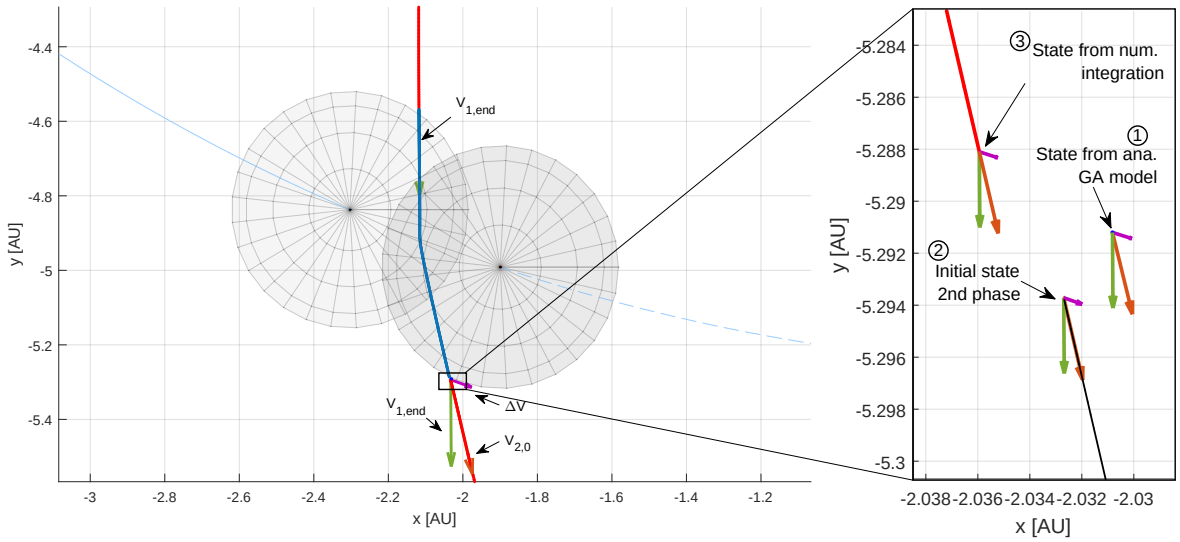


Figure 6.6: Close-up of the gravity assist portion of the validation case in the heliocentric frame.

Table 6.4: The SOI exit states computed using the analytical GA model and an external numerical integration plus the actual initial state of the second phase in InTrance.

		Analytical GA Model ①	Numerical Integration ③	Initial State 2nd Phase ②
Heliocentric Position $\boldsymbol{R}$ [AU]	x	-2.0308	-2.0359	-2.0327
	y	-5.2912	-5.2881	-5.2937
	z	0.0824	0.0809	0.0816
	$\|\boldsymbol{R}\|$	5.6681	5.6671	5.6711
Planetocentric Position $\boldsymbol{r}$ [AU]	x	-0.1309	-0.1354	-0.1328
	y	-0.2999	-0.297	-0.3024
	z	0.0192	0.0177	0.0183
	$\|\boldsymbol{r}\|$	0.3278	0.3269	0.3308
Heliocentric Velocity $\boldsymbol{V}$ [km/s]	$\dot{x}$	5.3216	5.0500	5.0399
	$\dot{y}$	-21.7725	-21.7292	-21.8708
	$\dot{z}$	1.0229	0.9439	1.0354
	$\|\boldsymbol{V}\|$	22.4367	22.3283	22.4679
Planetocentric Velocity $\boldsymbol{v}$ [km/s]	$\dot{x}$	-6.7321	-7.0031	-7.0317
	$\dot{y}$	-17.7315	-17.6867	-17.8299
	$\dot{z}$	1.276	1.197	1.2885
	$\|\boldsymbol{v}\|$	19.0093	19.0603	19.203
$\Delta\boldsymbol{V} = \boldsymbol{V}_{\text{ext}} - \boldsymbol{V}_{\text{ent}}$ [km/s]	$\dot{x}$	2.1209	2.1179	2.1207
	$\dot{y}$	4.5545	4.5671	4.5544
	$\dot{z}$	-0.0631	-0.0636	-0.0631
	$\|\Delta\boldsymbol{V}\|$	5.0245	5.0346	5.0244

6.3. Applicability of the Implementation

The implementation of gravity assists within InTrance is deemed verified and validated. The generated results for a New Horizons like reference case show excellent agreement with literature; validating the overall search behaviour and optimality. The external re-integration from the initial conditions at Earth indicate the correct implementation of phase transitions, implementation of initial conditions for each phase and implementation of additional chromosome parameters. It is clear there is a deviation between a numerical re-integration and the analytical GA model implemented within InTrance, however, its accuracy for non-extreme cases is deemed sufficient. Furthermore, if higher fidelity solutions are required, use can be made of the numerical model.

7

Mission Analysis

Trajectory analysis and design is a crucial part of the feasibility analysis of any space mission. Trajectory design is a complicated multi-asset task, with connections into an coming from many other design groups. Trajectory design is, for instance, constrained by the available launcher, but also sets constraints on the maximum achievable payload mass for certain launch velocities, transit times and launch masses. The trajectory sets bounds on the achievable orbits at the target body, but also receives requirements on those orbits determined from the science objectives and available instruments. Preliminary trajectory analysis plays a vital part in analysing the feasibility of a mission, for which an easy-to-use and robust method should be available. It will be demonstrated that InTrance can be used as a preliminary gravity assist low-thrust trajectory design tool by analysing two main missions: a low-thrust adaptation of New Horizons and Dawn.

7.1. Low-Thrust New Horizons

The low-thrust adaptation of New Horizons has been introduced in Chapter 6 and formed part of the validation effort. However, the generated validation trajectory was optimised in such a way that it mimicked the results found in literature. It was found that a gain could be achieved by further optimising the initial conditions, for which the results will be discussed here. First, the mission defining and simulation parameters will be detailed in Section 7.1.1, after which the results are presented in Section 7.1.2. These results are then compared to the actual high-thrust New Horizons mission and to similar low-thrust missions from literature, in Sections 7.1.3 and 7.1.4, respectively. The improvement due to the inclusion of a GA is investigated in Section 7.1.5 by comparing the resulting trajectory with a direct transfer from Earth to Pluto, which is also computed with InTrance. Lastly, a third trajectory is generated and presented in Section 7.1.6 in which thrusting is allowed while performing the gravity assists, such that the effects of a powered gravity assist on the overall trajectory design can be analysed.

7.1.1. Simulation & Mission Defining Input Parameters

The trajectory is split in two phases, with the first starting at launch from Earth and ending with entry into the SOI of Jupiter. The second phase starts from the rim of the SOI of Jupiter after having performed a GA, and ends with a flyby of Pluto. The analytical gravity assist model as described in Section 5.3.1 has been used to determine the $\boldsymbol{\Delta V}$ due to the gravity assist and the resulting exit position out of the SOI. The second phase is initialised relative to the entry velocity, as explained in Section 5.4.1. The drymass of the S/C has been set, similarly to the validation case, at 565 kg and both phases are propelled with a single NEP thruster with a specific impulse of 3000 s. Bang-bang control is applied, meaning the thruster is either fully engaged at 40 mN or completely turned off. The only disturbing potentials taken into account are the gravitational attraction of initial and final target bodies of a respective phase.

Both NCs are represented by a 3 layer ANN with 35 neurons in the hidden layer in which the sigmoid is used as activation function. The population is comprised of 50 individuals during the SSS (30 epochs) and thereafter reduced to 30 individuals. The hypercube is initialised at 1.0 during the SSS and 0.2 thereafter, and is allowed to shrink with a factor of 0.09 after a successful epoch or after a predefined number of bad epochs. The

mutation probability of the chromosome is set at 0.9, and at 0.05 for a genome. The trajectory of both phases is integrated in the heliocentric frame under the influence of a gravitational attraction from the Sun with an RK4(5) integrator using dynamic step size control.

The goal of the optimisation is to minimise the transfer time to Pluto while making use of a GA at Jupiter. The S/C has to perform a flyby of Pluto within 10^7 km and has to arrive before Oct. 30, 2014. The allowed transition threshold between phases has been set at 0.05. InTrance has to optimise both the initial conditions of each phase plus all internal parameters of both NCs. The initial condition windows are quite small for some parameters, which are a result of prior InTrance runs with larger windows and lower accuracy. The complete set of mission defining parameters is given by:

Table 7.1: Mission defining parameters used to generate an optimised New Horizons like low-thrust trajectory. Values in square brackets indicate ranges.

	Phase 1 – Earth to Jupiter	**Phase 2 – Jupiter to Pluto**
Launch date	[Jan. 05, 2006 – Jan. 25, 2006]	[Mar. 19, 2007 – Apr. 08, 2007]
Arrival date	[Jan. 20, 2007 – Feb. 09, 2007]	[Feb. 02, 2014 – Oct. 30, 2014]
Launch V_∞ or ΔV_{GA}	[5.00 – 12] km/s	[4 – 6.5] km/s
Launch V_∞ or ΔV_{GA} azimuth	[-10 – 10] deg	[100 – 120] deg
Launch V_∞ or ΔV_{GA} elevation	[0 – 10] deg	[0 – 15] deg
Departure azimuth GA	N/A	[180 – 220] deg
Departure elevation GA	N/A	[-8 – 8] deg
Target	Jupiter GA	Pluto flyby
Maximum target distance	N/A	10^7 km
Dry mass	565 kg [75]	565 kg
Propellant consumption	[10 – 22] kg	[5 – 13] kg
Specific Impulse	3000 s [75]	3000 s
Maximum thrust	40 mN [75]	40 mN

7.1.2. Results

The resulting trajectory from the above described InTrance run is shown in Figure 7.1, with the left portion showing the complete heliocentric trajectory and the right showing a close-up to better display the thrust arcs and corresponding direction of the thrust vectors (green arrows). The S/C is launched from Earth on Jan. 17, 2006 with a launch V_∞ of 12 km/s. A continuous thrust arc follows with a total duration of 185 days, which is about 50% of the total flight time of the first phase. The S/C enters the SOI of Jupiter on Jan. 25, 2007 to perform its closest approach of Jupiter on Feb. 23, 2007 at a distance of 27.5 $R_♃$. The second phase commences from the rim of the SOI after having performed a GA which generated a ΔV of 6.01 km/s on Mar. 25, 2007. A thrust arc follows from departure for a total of 77 days, or 3% of the total flight time in the second phase. The S/C performs its flyby of Pluto on Jul. 05, 2014 at a distance of $3.1 R_{SOI,♇}$ with a relative v_∞ to Pluto of 15.9 km/s.

The resulting trajectory and its characteristics are summarised in Table 7.2. The total MET is 3,091 days, of which the S/C spends 6,299 hours thrusting and 59.0 days inside the SOI of Jupiter. A total of 30.8 kg propellant is consumed, of which 21.7 kg in the first phase and 9.1 kg during the second phase.

Table 7.2: Results of the InTrance optimisation run of the New Horizons like low-thrust gravity assist trajectory.

Launch V_∞	12.02 km/s	ΔV due to GA	6.01 km/s
Launch date	Jan. 17, 2006	Closest approach Pluto	$3.1 R_{SOI,♇}$ km
Jupiter encounter	Feb. 23, 2007	Pluto v_∞	15.9 km/s
Pluto arrival	Jul. 05, 2014	Mass at departure	595.8 kg
Mission elapsed time	3,091 days	Mass at arrival	565.2 kg
Total thrust time	6,299 hours	Thrust	40 mN
Closest approach Jupiter	27.5 $R_♃$	Specific impulse	3000 s

A close-up of the gravity assist portion of the trajectory is shown in Figure 7.2, both in the planetocentric (left) and heliocentric (right) reference frames. The trajectory inside the SOI is integrated externally from InTrance

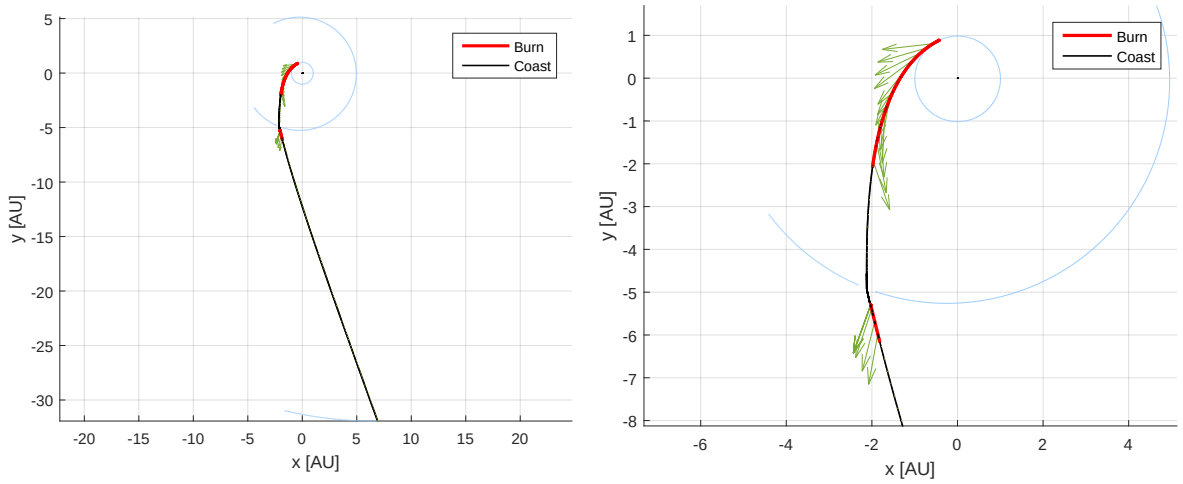


Figure 7.1: Heliocentric InTrance optimised New Horizons like low-thrust trajectory. Close-up of thrust direction on the right side.

and solely added for presentation purposes. The incoming $\mathbf{v}_\infty$ is deflected over an angle of 18.14° in the planetocentric frame, and the S/C approaches Jupiter closest at a distance of 27.5 $R_\mathrm{♃}$. The resulting $\mathbf{\Delta V}$ due to the gravity assist of 6.01 km/s is indicated with a purple arrow in the heliocentric plot, relative to the SOI entry velocity.

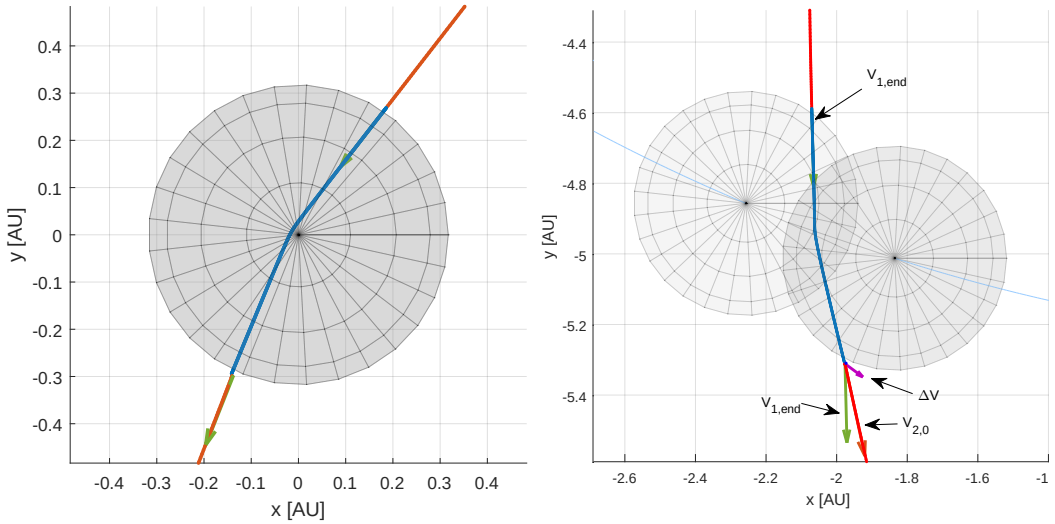


Figure 7.2: Close-up of the gravity assist portion of the resulting trajectory from InTrance for the New Horizons like low-thrust mission in both the planetocentric (left) and heliocentric (right) reference frames.

The initial conditions of the second phase are defined relative to the arrival conditions of the first phase, as detailed in Section 5.4.1. The optimised initial state of the second phase is shown in Figure 7.3, in which the optimised initial $\mathbf{\Delta V}$ vector is indicated in purple. InTrance found an optimal initial $\mathbf{\Delta V}_\mathrm{EA}$ with a magnitude of 5.60 km/s, an azimuth angle of 111.2° and an elevation angle of 9.6°. The initial launch position is located on the rim of the SOI with an azimuth angle of 201.9° and elevation of 3.1°. The initial velocity $\boldsymbol{V}_{2,0}$ is then determined as the velocity at SOI entry $\boldsymbol{V}_{1,\mathrm{end}}$ plus $\mathbf{\Delta V}_\mathrm{EA}$.

7.1.3. Compared to High-Thrust New Horizons Mission

The resulting InTrance trajectory, together with the actual New Horizons high-thrust trajectory, is plotted in Figure 7.4 in the heliocentric frame on the left hand side. The heliocentric velocity and distance of both trajectories are plotted, respectively, in the middle and right-hand side. The trajectory of the high-thrust New Horizons missions has already been described in Section 2.1, and will not be further elaborated upon here.

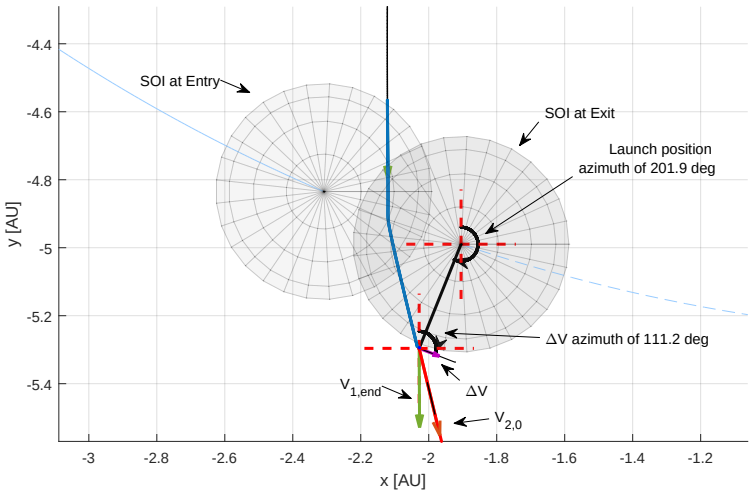


Figure 7.3: Initial state of the phase departing from Jupiter with velocities optimised relative to the entry velocity.

The high-thrust New Horizons trajectory has been retrieved through SPICE[1], which does not completely reflect the actual flown mission, but rather one of the solutions found during its baseline design. The low-thrust solution found by InTrance cannot directly be compared to New Horizon's trajectory, as InTrance does not take any thrust restriction times into account. However, a qualitative analysis is possible.

It can be seen that the low-thrust version computed by InTrance reaches Pluto faster and therefore performs a flyby with a higher relative velocity to Pluto. The higher velocity is disadvantageous as it decreases the scientific return, however, could easily be decreased by applying a braking manoeuvre. After all, the low-thrust S/C uses less propellant than New Horizons, and can therefore take additional propellant on-board. Overall, the two trajectories are quite similar upto the gravity assist, where the low-thrust S/C exits with a higher velocity, resulting in the divergence of the two trajectories.

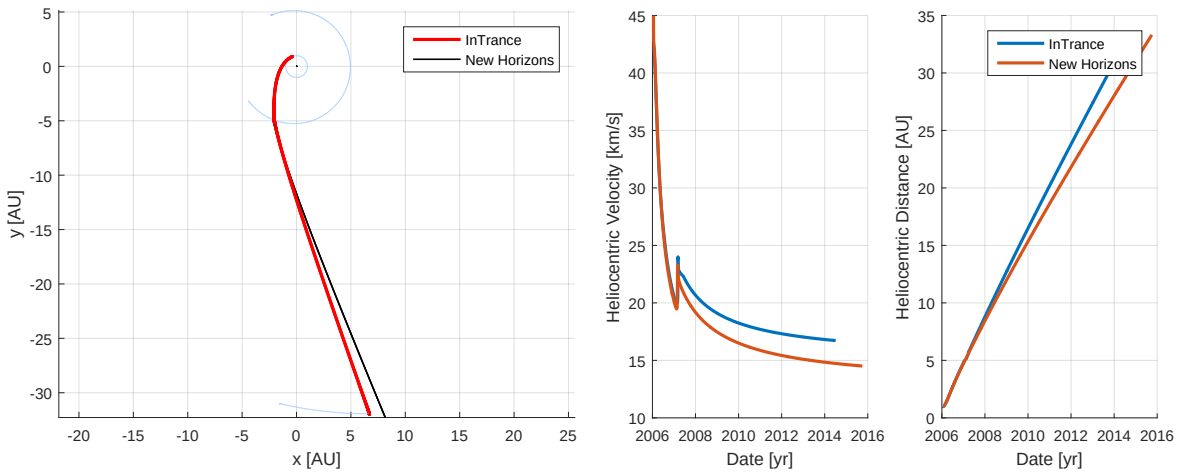


Figure 7.4: Heliocentric trajectory, velocity and distance of the actual New Horizons missions and a low-thrust alternative computed with InTrance.

Some mission defining parameters of both the high-thrust New Horizons trajectory and the low-thrust GA solution found by InTrance are tabulated in Table 7.3. The low-thrust version is launched 2 days prior to the New Horizon case with a slightly lower V_∞ due to its higher mass. Both trajectories perform a gravity assist at around the same time, but the InTrance solution has a lower closest approach (27.5 vs. 32.2R_φ), which results in a higher generated ΔV. The low-thrust version generated by InTrance is therefore faster during the second phase, resulting in a MET of 3,091 days versus 3,103 days for New Horizons. InTrance's goal was to perform a

[1] SPICE New Horizons kernel available at: `https://naif.jpl.nasa.gov/naif/data_archived.html`

flyby with a minimum distance to Pluto equal to about 3.5 times the SOI, which explains the large difference between the two closest approach distances. InTrance is capable of lowering this distance, if desired, but the current result suffices within the intended use as a preliminary design.

Table 7.3: Comparison of low-thrust NEP trajectories to Pluto generated by InTrance and the actual high-thrust New Horizons trajectory. New Horizons data from SPICE files unless stated otherwise.

	InTrance	New Horizons	InTrance - NH
Launch V_∞	12.02 km/s	12.56 km/s	-0.54 km/s
Launch date	Jan. 17, 2006	Jan. 19, 2006	-2 days
Jupiter encounter	Feb. 23, 2007	Feb. 28, 2007	-5 days
Pluto arrival	Jul. 05, 2014	Jul. 19, 2014	-14 days
Mission elapsed time	3,091 days	3,103 days	-12 days
Total thrust time	6,299 hours	N/A	N/A
Closest approach Jupiter	27.5 $R_♃$	32.24 $R_♃$	-4.74 $R_♃$
ΔV due to GA	6.01 km/s	5.14 km/s	+0.87 km/s
Closest approach Pluto	3.1$R_{\mathrm{SOI},♇}$	64,088 km	+3.1$R_{\mathrm{SOI},♇}$
Pluto v_∞	15.9 km/s	13.8 km/s	+2.1 km/s
Mass at departure	595.8 kg	478 kg [36]	+117.8 kg
Mass at arrival	565.0 kg	426.4 kg[1] [36]	+138.6
Thrust	40 mN	4 × 4.4 N [29]	-17.56 N

[1] Including a 25.4 kg propellant reserve.

This comparison has shown that low-thrust missions which make use of intermediate GAs can be excellent alternatives to conventional high-thrust missions, even those to the outer planets. InTrance has found a result which is more time-optimal than the actual trajectory using less propellant, even though the S/C has more than 100 kg additional launch mass. The S/C drymass used in the InTrance run is derived from similar optimisations in literature [75], however, could easily be set to equal the actual New Horizons mass. The low-thrust alternative would then result in a smaller launch mass, leaving additional room for payload or to further decrease the flight time by taking along more propellant.

7.1.4. Compared to Literature

InTrance has optimised the conditions within larger windows than those set during the validation case, therefore, an even faster solution is found than those found by Vasile et al. and Carnelli as discussed in Section 6.2. The most prominent results from the InTrance validation run and those generated by Vasile et al. [75] and Carnelli [13] are repeated in Table 7.4, together with the current optimised solution from InTrance. InTrance has found more favourable initial conditions which result in a decrease of about 3% in flight time and a decrease of about 11% in required propellant mass. It should be noted that the closest approach distance at Pluto is not reported by Vasile et al. and Carnelli; if their solutions performed a closer flyby of Pluto then the MET would naturally be higher than the one found with InTrance.

Table 7.4: InTrance's optimised trajectory to Pluto with a GA at Jupiter compared to the validation literature.

	Vasile et al.	Validation	Carnelli	InTrance
Launch V_∞	12 km/s	12 km/s	12 km/s	12 km/s
Launch date	Jan. 19, 2006	Jan. 19, 2006	Jan. 4, 2006	Jan. 17, 2006
Jupiter encounter	Feb. 23, 2007	Feb. 24, 2007	Feb 16, 2007	Feb. 23, 2007
Pluto arrival	Oct. 4, 2014	Oct. 3, 2014	Sep. 23, 2014	Jul. 05, 2014
Mission elapsed time	3,181 days	3,180 days	3,185 days	3,091 days
Total thrust time	6,600 hours	5,739 hours	Unknown	6,299 hours
Pluto v_∞	15.337 km/s	15.371 km/s	Unknown	15.9 km/s
Mass at departure	600 kg	600 kg	600 kg	595.8 kg
Mass at arrival	565.5 kg	565.1	565.3 kg	565.0 kg
Thrust	34 – 40 mN	40 mN	40 mN	40 mN
Specific impulse	1700 – 4000 s	3000 s	3000 s	3000 s

7.1.5. Improvement due to Gravity Assist

The improvement due to the inclusion of a GA over a direct transfer is investigated by also optimising a direct Earth-Pluto transfer with InTrance. Jupiter is excluded from the list of disturbing bodies in the direct transfer case to make sure InTrance does not implicitly perform a GA. This is equivalent to launching at a date where Earth and Pluto would have the same relative geometry, but where Jupiter is not in the vicinity. During an early run with Jupiter included as a disturbing body, InTrance did find a GA at Jupiter, albeit a far from optimal one. The direct transfer is optimised to minimise the propellant usage and constrained to still have a reasonably close MET to the solution with a GA.

The mission defining input parameters are shown in Table 7.5. The launch and arrival date windows are the same as used in the trajectory with a Jupiter GA as given in Section 7.1.1. Due to the higher mass, the maximum launch V_∞ is slightly lower (11.65 vs 12.04 km/s), determined from setting the maximum launcher C3 capacity at 7200 km^2/s^4 with an exponent of 0.01725 [75]. The S/C can use anywhere between 50 and 500 kg of fuel to reach Pluto, its goal is then to minimise this number while abiding to the other constraints.

Table 7.5: Mission defining parameters used to generate a direct Earth-Pluto transfer. Values in square brackets indicate ranges.

	Phase 1 – Earth to Pluto
Launch date	[Jan. 05, 2006 – Jan. 25, 2006]
Arrival date	[May. 23, 2014 – Oct. 30, 2014]
Launch V_∞	[5.00 – 11.653] km/s
Launch V_∞ azimuth	[-10 – 10] deg
Launch V_∞ elevation	[0 – 10] deg
Target	Pluto flyby
Maximum target distance	10^7 km
Dry mass	565 kg
Propellant consumption	[50 – 500] kg
Specific Impulse	3000 s
Maximum thrust	40 mN

The optimised direct transfer resulting from a run of InTrance is shown in Figure 7.5, together with the GA performing solution discussed above. The left-hand figure shows both heliocentric trajectories and their corresponding thrust arcs, the middle the heliocentric velocity over time and the right-hand plot the heliocentric distance over time. A major noticeable difference between the trajectories is the long continuous thrust arc of the direct transfer; engaging the thrusters for 1083 days, compared to a total of 262 days in the GA trajectory. This long continuous thrust arc provides almost the same ΔV as the GA, with the difference in velocity between the two trajectories converging to about 1 km/s after the thruster is disengaged.

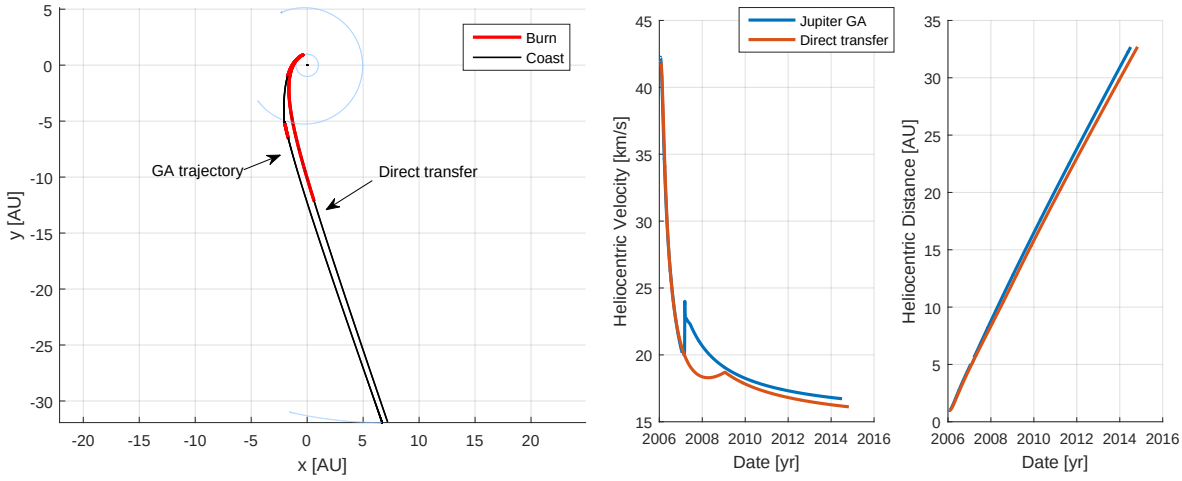


Figure 7.5: Heliocentric trajectory, velocity and distance of two low-thrust NEP trajectories to Pluto generated with InTrance, one direct transfer and one including a Jupiter GA.

The two trajectories are characterised by the parameters shown in Table 7.6. Due to the optimisation goal of propellant consumption minimisation, the direct transfer utilises the entire allowed flight time; from the lower constraint of the launch date to the upper constraint of the arrival date. This results in a MET of 3,200 days, an increase of 3.5% from the 3,091 days with a Jupiter GA. In order to reach Pluto within this time, the thruster burns a total of 126.8 kg; an almost 312% increase from the 30.8 kg in the GA case.

Table 7.6: Comparison of low-thrust NEP trajectories to Pluto generated by InTrance, both as a direct transfer and with a GA at Jupiter.

	Jupiter GA	Direct transfer	Jupiter GA - Direct
Launch V_∞	12 km/s	11.6 km/s	-0.4 km/s
Launch date	Jan. 17, 2006	Jan. 25, 2006	-8 days
Jupiter encounter	Feb. 23, 2007	N/A	N/A
Pluto arrival	Jul. 05, 2014	Oct. 30, 2014	-117 days
Mission elapsed time	3,091 days	3,200 days	-109 days
Total thrust time	262.5 days	1,092 days	-829.5 days
Closest approach Jupiter	27.5 $R_{♃}$	N/A	N/A
ΔV due to GA	6.01 km/s	N/A	N/A
Closest approach Pluto	$3.1 R_{\mathrm{SOI},♇}$	$0.69 R_{\mathrm{SOI},♇}$	N/A
Pluto v_∞	15.9 km/s	15.5 km/s	+0.39 km/s
Mass at departure	595.8 kg	691.8	-96 kg
Mass at arrival	565.0 kg	565.0	-
Thrust	40 mN	40 mN	-
Specific impulse	3000 s	3000 s	-

This demonstration clearly shows the benefits of adding GAs to low-thrust trajectories; with the inclusion of a GA the transfer time is reduced by almost 3.5% and the required propellant is reduced by over 75%. Although demonstrating the benefit of a GA in low-thrust trajectories in this case, the inclusion of GAs does not always result in more optimal trajectories. However, even if a less than optimal trajectory is found with the inclusion of a GA, it might still be interesting to perform one for increased scientific return.

7.1.6. Influence of Thrust During Gravity Assist

The above simulations all made use of the analytical GA-model, which prevents thrusting while inside the SOI. It was shown in Section 2.4 that low-thrust can have a large influence on a gravity assist, especially when applied for a long duration to decrease the closest approach distance. To investigate whether a powered GA is beneficial for the low-thrust New Horizons case, another run of InTrance is performed in which the analytical GA-model is omitted. In this way, InTrance will simply continue integrating until having exited the SOI, which marks the end of the first phase. The mission defining parameters are the same as those given in Section 7.1.1, with the exception of shifting the arrival date at and departure date from Jupiter by +60 days, which is the expected duration of the time inside the SOI. The numerical GA has only been implemented for a body-relative initial state of the second phase (see Section 5.4.2); the windows for the $\Delta\mathbf{V}_{\mathrm{GA}}$ are therefore different from those used before, which were defined relative to the entry state of the SOI. The windows for the initial conditions of the second phase relative to the body-velocity are now given by:

Table 7.7: Different mission defining parameters used in the numerical GA computation. Values in square brackets indicate ranges.

	Phase 2 – Jupiter to Pluto
Departure ΔV_{GA}	[18 – 21] km/s
Departure ΔV_{GA} azimuth	[180 – 220] deg
Departure ΔV_{GA} elevation	[0 – 15] deg

The resulting trajectory in both the helio- and planetocentric frame is shown in Figure 7.6, together with the trajectory computed with the analytical GA-model discussed before. InTrance has not found a solution which performed a powered GA to be more favourable than the optimal solution which performs an unpowered GA. The resulting trajectory, especially for the first phase, is very similar to the result obtained before with the analytical GA. With the analytical model, the GA generated a ΔV of 6.01 km/s, which is 100 m/s higher than the 5.91 km/s found in the numerically computed trajectory. The closest approach in both solutions is

located at a distance of 27.5 $R_{\oplus}$, and the deflection angle in the numerical computation is 18.24° versus 18.15° in the analytically computed trajectory. However, InTrance was unable to close the time gap between the two phases, rendering the second phase to commence 5 days prior to actually having exited the SOI. Therefore, a more thorough comparison of the two trajectories is meaningless. However, the numerically computed solution was generated to investigate whether the low-thrust New Horizons mission would benefit from a powered GA, which has not been found the case.

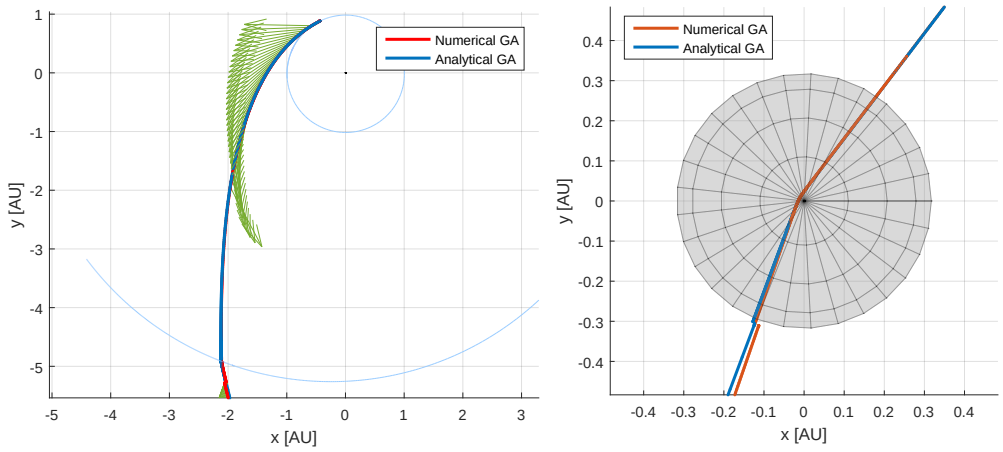


Figure 7.6: Helio- (left) and planetocentric (right) portions of the New Horizons trajectories computed with the analytical GA-model (blue) and numerical GA-model (red). Green arrows denote the thrust vectors of the trajectory computed with the numerical GA-model.

The optimisation with a numerically computed GA is significantly more computationally expensive. The total optimisation run duration was 33h:26m, whereas the previously discussed run with the analytical computation took 7h:17m. A run which also defined the initial conditions of the second phase relative to the GA-body and used the analytical GA-model took 8h:17m. All these results have been computed using four cores on an Intel® Core™ i7-4700HQ. Recomputing the same problems will result in different runtimes, as the chromosome is initialised at random, but a clear deviation can be seen in the numerical versus analytical computations. The decrease in efficiency is caused by the small required (dynamically computed) step-size while inside the SOI; testament by the number of output points: 269 for the first phase with the analytical model; and 725 in the numerical computation, with 459 points inside the SOI.

7.2. Dawn

Although the previous version of InTrance by Ohndorf [59] did not have the capability to independently optimise trajectories such as the low-thrust New Horizons adaptation of the previous section, it was able to perform a gravity assist a Jupiter, simply due to its massive gravitational potential. Although the resulting GA is far from optimal, it does show that Jupiter is a relatively easy GA target. Nevertheless, optimising such a trajectory with neurocontrol is no trivial task, demonstrated by the fact that previous attempts to optimise gravity assists with NCs, such as the one by Carnelli [14], failed. Having shown very good results for the optimisation of a GA at Jupiter, it is interesting to investigate whether InTrance is capable of finding optimal GA trajectories at smaller bodies. To this end, a challenging case is found in the re-optimisation of the Dawn mission.

Dawn was the first mission to orbit a main belt asteroid and the first to orbit two extraterrestrial bodies with a single craft, an achievement which would not have been possible with conventional high-thrust chemical propulsion. Dawn was launched in September 2007 on its way to Mars, where it performed a gravity assist to reach its first primary target Vesta. After having orbited Vesta for some 1.5 years, Dawn departed on its way to Ceres. For more details on the Dawn mission, the reader is referred to Section 2.1.2.

The simulation and mission defining input parameters for the nominal simulation are described in Section 7.2.1, after which the resulting trajectory is discussed in Section 7.2.2. The resulting trajectory is then first compared to the Dawn's actual trajectory in Section 7.2.3, followed by a comparison to a non-GA performing trajectory in Section 7.2.4. The influence of a powered gravity assist is investigated in Section 7.2.5, and the full three-phase single optimisation is lastly tackled in Section 7.2.6.

7.2.1. Simulation & Mission Defining Input Parameters

Due to the challenging nature of the Dawn trajectory, the optimisation effort here is initially focussed on the trajectory from Earth to Vesta, hence excluding the transfer to Ceres. Due to a stay time at Vesta (+250 days), the phase from Vesta to Ceres can be optimised independently. A propellent reserve for this transfer is taken into account in the optimisation of the transfer from Earth to Vesta. The first optimisation is split in two phases; the first from Earth to Mars where a GA is performed, and the second from Mars to ending with a rendezvous with Vesta. The second, separate, optimisation is a single phase mission, starting from Vesta and ending with a rendezvous of Ceres. A single optimisation run of the complete three phase trajectory is discussed in Section 7.2.6.

Initial State after a Gravity Assist

Empirical evidence from early optimisation runs of InTrance on the Dawn trajectory showed that the optimisation of the second phase (Mars to Vesta) could not reach Vesta within the predefined bounds. This was caused by the fact that the initial velocity of the first phase was defined as the final velocity of the first phase, plus a $\boldsymbol{\Delta V}$ optimised by the EA. Due to the small size of Mars' SOI, it takes some time before InTrance finds individuals which cross into the SOI. Hence, until that happens, the second phase is initialised on the rim of the SOI, but with a completely trivial velocity, i.e. the velocity which the S/C attained at one of the stopping conditions outside of entering the SOI. Once the S/C does finally perform a GA, the NC of the second phase has been trained with completely different initial velocities. Therefore, it performed worse in reaching Vesta after a GA was performed, rendering the GA individuals to go extinct or favouring reaching Vesta over performing a physically valid GA.

The first possible solution lay in increasing the importance of the proximity of the GA phase through a larger scaling factor, see equation 5.2. Although improving the search behaviour, InTrance was still unable to generate a physically valid trajectory. The focus then shifted to decoupling the second phase from the first phase, such that the second phase's training can focus on reaching Vesta, for which the first phase then has to supply the required $\boldsymbol{\Delta} V$. This led to the implementation of a GA-body relative initial state for a phase starting directly after a GA, as discussed in Section 5.4.2. This new state definition has been used in all Dawn scenarios for which results are discussed in this work.

Modification of Initial Conditions

Correct formulation of the azimuth and elevation angles of both the 'launch' position and the $\boldsymbol{\Delta V}_{\mathrm{EA}}$ of a phase starting after a GA (see Section 5.4) is generally no trivial task as it depends both on the geometry of the planets at the departure date plus on the resulting GA of the previous phase. Windows for these parameters cannot be chosen too strict, as the NC needs some initial freedom to find physically valid trajectories, only after which the state gaps are reduced. However, they also cannot be too large, as the value found from the chromosome is used in a linear mapping between the minimum and maximum allowed value for each parameter. Hence, when the windows are large, it is difficult for the EA to supply the accuracy needed for small changes in the initial parameters.

In order to reduce the dependency on the correct formulation of the initial conditions after a GA (position on SOI and direction of ΔV), InTrance is allowed to vary the windows of these parameters in the Dawn simulations. The evolutionary algorithm hones in on promising regions by extending and contracting the allowed window until the minimum and maximum values are within 5 degrees of each other. If the window is larger than 5 degrees after a successful epoch, it is contracted by $(h_{\max} - h_{\min})/2$, centred around the midpoint.

Input Parameters Optimisation 1 – Earth to Vesta

The analytical gravity assist model as described in Section 5.3.1 has been used to determine the $\boldsymbol{\Delta V}$ due to the gravity assist and the resulting exit position out of the SOI. The drymass is taken to be the same as the Dawn S/C, namely 747 kg [73], plus an additional 70kg which serves as propellant reserve for the subsequent optimisation for the phase from Vesta to Ceres. The thrust is supplied by a NSTAR engine, modelled after the one used in the actual Dawn S/C [73], and is allowed to be varied in both direction and magnitude by InTrance.

Both NCs are represented by a 3-layer ANN with 40 neurons in the hidden layer for phase 1 and 35 for phase 2, and the sigmoid function is used as activation function. The population is comprised of 50 individuals

during both the SSS (30 epochs) and main optimisation. The hypercube is initialised at 1.0 during the SSS and at 0.2 in the following optimisation run. The mutation probability of the chromosome is set at 0.9, and at 0.05 for a genome. The trajectory of both phases are integrated in the heliocentric frame under the influence of a gravitational attraction from the Sun and initial and final bodies of a phase, with an RK4(5) integrator using dynamic step size control.

The goal of the optimisation is to minimise the transfer time to Vesta while making use of a GA at Mars. The S/C has to rendezvous with Vesta before Feb. 12, 2019 at a maximum final relative distance of $1.5 \cdot 10^6$ km and maximum relative velocity of 1 km/s. The arrival and departure dates are chosen to be in relative proximity to Dawns actual trajectory, as generated with SPICE. The allowed transition threshold between phases has been set at 0.1. InTrance has to optimise both the initial conditions of each phase plus all internal parameters of both NCs. The complete set of mission defining parameters is given by:

Table 7.8: Mission defining parameters used to re-compute the Dawn trajectory from Earth to Vesta. Values in square brackets indicate ranges.

	Phase 1 – Earth to Mars	**Phase 2 – Mars to Vesta**
Launch date	[Sep. 22, 2007 – Oct. 02, 2007]	[Jan. 28, 2009 – Feb. 17, 2009]
Arrival date	[Jan. 23, 2009 – Feb. 12, 2009]	[May. 19, 2011 – Oct. 16, 2011]
Launch V_∞ or $\Delta V_{GA,b}$	3.362 km/s [62]	[2 – 3] km/s
Launch V_∞ or $\Delta V_{GA,b}$ azimuth	[-20 – 20] deg	[-15 – -7.5] deg
Launch V_∞ or $\Delta V_{GA,b}$ elevation	[-20 – 40] deg	[-60 – -40] deg
Departure azimuth GA	N/A	[-20 – 0] deg
Departure elevation GA	N/A	[-70 – -40] deg
Target	Mars GA	Vesta rendezvous
Maximum target distance	N/A	$1.5 \cdot 10^6$ km
Maximum relative target velocity	N/A	1 km/s
Dry mass	747 kg + 70 kg reserve propellant	747 kg + 70 kg reserve propellant
Propellant consumption	[100 – 160] kg	[100 – 160] kg
Thruster	NSTAR	NSTAR

Input Parameters Optimisation 2 – Vesta to Ceres

A single phase rendezvous is easily tackled by InTrance; physically valid solutions are usually generated within minutes, and the entire optimisation takes about half an hour. Contrary, the inclusion of a GA vastly complicates the problem and InTrance runs usually take about 10 hours depending on the desired accuracy and chosen convergence limits (using four cores on an Intel® Core™ i7-4700HQ).

Due to the relative simplicity of the problem, the maximum relative position and velocity to Ceres at the rendezvous are set lower than was previously the case. There are furthermore only 20 epochs in the SSS and the hypercube is initialised at a larger size of 1.0, other ANN and EA parameters are equal to those of the two-phase Earth-Vesta transfer discussed above. The dwell time at Vesta was set to be a minimum of 250 days. The mission defining parameters are then given by;

Table 7.9: Mission defining parameters used to re-compute the Dawn trajectory from Vesta to Ceres. Values in square brackets indicate ranges.

	Phase 1 – Vesta to Ceres
Launch date	[Jan. 24, 2012 – Mar. 14, 2012]
Arrival date	[Aug. 31, 2014 – Nov. 19, 2014]
Target	Ceres rendezvous
Maximum target distance	$1.0 \cdot 10^6$ km
Maximum relative target velocity	500 m/s
Dry mass	747 kg
Propellant consumption	[50 – 100] kg
Thruster	NSTAR

7.2.2. Results

The resulting heliocentric trajectory from the combined results of the two above described optimisation runs is shown in Figure 7.7, in which the green arrows denote thrust vectors. The S/C is launched from Earth with a launch V_∞ of 3.362 km/s on Sep. 23, 2007. After about 16 months, the S/C performs a Mars GA with a closest approach of 2.39$R_♂$ on Jan. 27, 2009. The GA results in an inclination change of 3.3° and a ΔV of 2.4 km/s. Vesta is finally reached after another 2 years and 4 months, on May. 19, 2011. The second optimisation finds a departure from Vesta on Mar. 03, 2012, resulting in a Vesta dwell time of 293 days in which the science operations are to be performed. The flight from Vesta to Ceres takes about 2 years and 7 months, resulting in a rendezvous with Ceres on Oct. 29, 2012. An enlarged three-dimensional view of the heliocentric trajectory is shown in Figure B.1 in Appendix B.

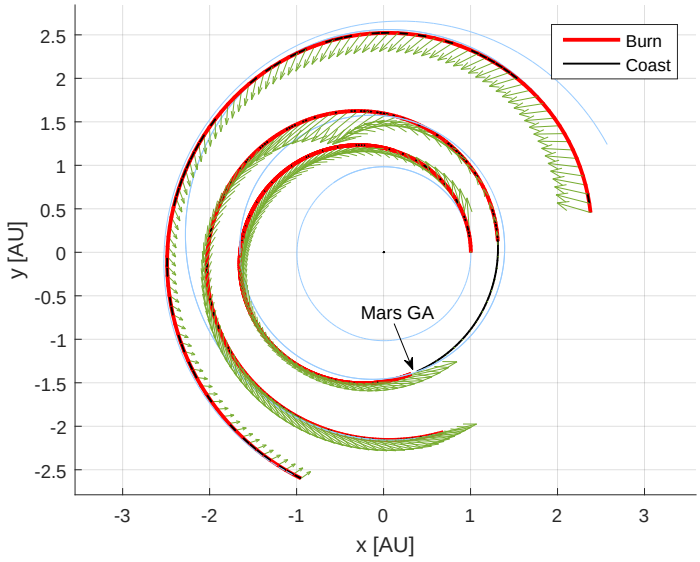


Figure 7.7: Dawn trajectory as optimised by InTrance in the heliocentric frame. Green arrows denote thrust vectors.

The characteristics of the resulting trajectory are summarised in Table 7.10. The total mission elapsed time is 2,594 days, including the Vesta dwell time of 293 days. Of the total flight duration, hence excluding the dwell time at Vesta, about 86% is spend thrusting. The launch mass is found as 1,056.1 kg, of which 308 kg (29%) is propellant. A rendezvous was deemed successful if the relative distance and velocity to the targets were less than $1.5 \cdot 10^6$ km and 1 km/s, respectively. The final position and velocity relative to Vesta at the end of the second phase are $1.07 \cdot 10^6$ km and 716 m/s. The final position and velocity relative to Ceres at the end of the final phase are $0.96 \cdot 10^6$ km and 499 m/s. Although these values are still relatively large, as a rendezvous should decrease the relative difference to zero, it is deemed sufficient for the current application. InTrance is a global optimisation tool, not capable of supplying the accuracy to reduce the relative states to zero. With InTrance intended as a preliminary design phase trajectory optimisation tool, the found results suffice and can generally be improved by using a local optimisation scheme.

Table 7.10: Results of the InTrance optimisation run of the Dawn trajectory including a Mars GA.

Launch V_∞	3.362 km/s	Closest approach Mars	2.39 $R_♂$
Launch date	Sep. 23, 2007	ΔV due to GA	2.41 km/s
Mars encounter	Jan. 27, 2009	Rel. distance Vesta	$1.07 \cdot 10^6$ km
Vesta arrival	May. 19, 2011	Rel. velocity Vesta	716 m/s
Vesta dwell time	293 days	Rel. distance Ceres	$0.96 \cdot 10^6$ km
Vesta departure	Mar. 03, 2012	Rel. velocity Ceres	499 m/s
Ceres arrival	Oct. 29, 2014	Mass at departure	1,056.1 kg
Mission elapsed time	2,594 days	Mass at arrival	747.8 kg
Total thrust time	1,973 days	Thruster	NSTAR

The planetocentric GA portion of the trajectory is plotted in Figure 7.8 in both the XY- and YZ planes. Red

lines indicate the InTrance trajectory, the blue line is an external numerical integration solely added for presentation purposes, and the green arrow denotes the exit state as computed by the analytical GA-model. The exit velocity out of the SOI of the first phase and the initial velocity of the second phase are completely matched, but a position gap of $5.0 \cdot 10^4$ km = 14.7 $R_{♂}$ remains. These values are within the allowed state transition threshold of 0.1, and cannot be reduced much further. InTrance has trouble delivering the required accuracy to reduce the positional gap as it depends on the values of two alleles on the chromosome, see Section 5.5. For a slight modification of the launch position, the entire steering strategy has to be updated to accommodate these new initial conditions. The chromosome in this optimisation has a total length of 1718 alleles, meaning that new optimal values for all other 1716 alleles have to be found, which is not likely to occur in a few evaluations. Therefore, these individuals perform worse and go extinct, leaving the best found solution with a small state gap.

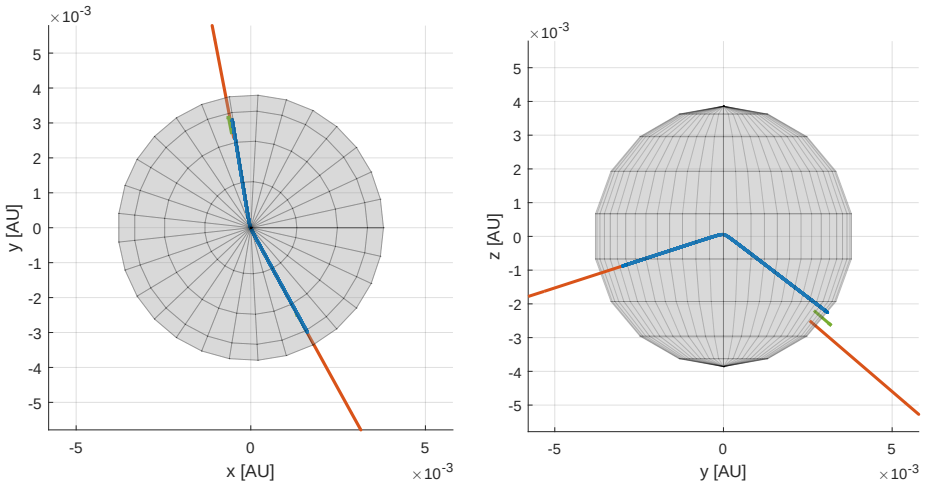


Figure 7.8: Close-up of the Mars GA portion in the planetocentric frame. Red lines indicate the trajectory as computed by InTrance, the blue line indicates the externally integrated trajectory while inside the SOI and the green arrow indicates the exit state as computed by the analytical GA-model.

The initial conditions of the second phase are defined relative to GA-body, as detailed in Section 5.4.2. The optimised initial position of the second phase was already shown in the planetocentric frame in Figure 7.8, the initial state in the heliocentric frame is shown in Figure B.2 in Appendix B. InTrance found an optimal initial $\Delta V_{\mathrm{EA,b}}$ of 2.39 km/s relative to the velocity of Mars at departure, with an azimuth of -10.86° and elevation of -40.17°. The initial position is located on the rim of the SOI with an azimuth of -10.76° and elevation of -43.87°.

7.2.3. Compared to Dawn's Actual Trajectory

The resulting trajectory generated with InTrance, together with Dawn's actual mission trajectory, is plotted in Figure 7.9 in the heliocentric frame on the left hand side. The heliocentric velocity and distance of both trajectories are plotted, respectively, in the middle and right-hand side. The trajectory of the Dawn missions has already been described in Section 2.1, and will not be further elaborated upon here. The actual Dawn trajectory has been retrieved through SPICE[(2)], which does not completely reflect the actual flown mission, but rather one of the solutions found during its baseline design. The two trajectories cannot be compared directly as no thrust restriction windows are taken into account in the InTrance simulation. However, a qualitative analysis is possible.

Overall, the two trajectories are broadly similar in shape when viewed in the heliocentric frame. InTrance finds a trajectory with a slightly lower pericenter and higher apocentre after the Mars GA. From a three-dimensional view (see Figure B.3) it furthermore becomes clear that the actual trajectory is deflected slightly more with regards to the inclination angle (3.5 vs. 3.3°). From the heliocentric velocity plot it can be seen that the result from InTrance has a higher velocity some time after the Mars GA, which is not a result of a higher ΔV due to the GA but due to the earlier application of thrust. The velocity when approaching Vesta is

[(2)] SPICE New Horizons kernel available at: `https://naif.jpl.nasa.gov/naif/data_archived.html`

quite different, as the S/C in Dawn's actual trajectory has to slow down to a relative velocity of about 0.5 m/s, whereas InTrance found a solution with a relative velocity of 716 m/s. The velocity of the phase from Vesta to Ceres is quite different, which is explained by their different departure dates from Vesta.

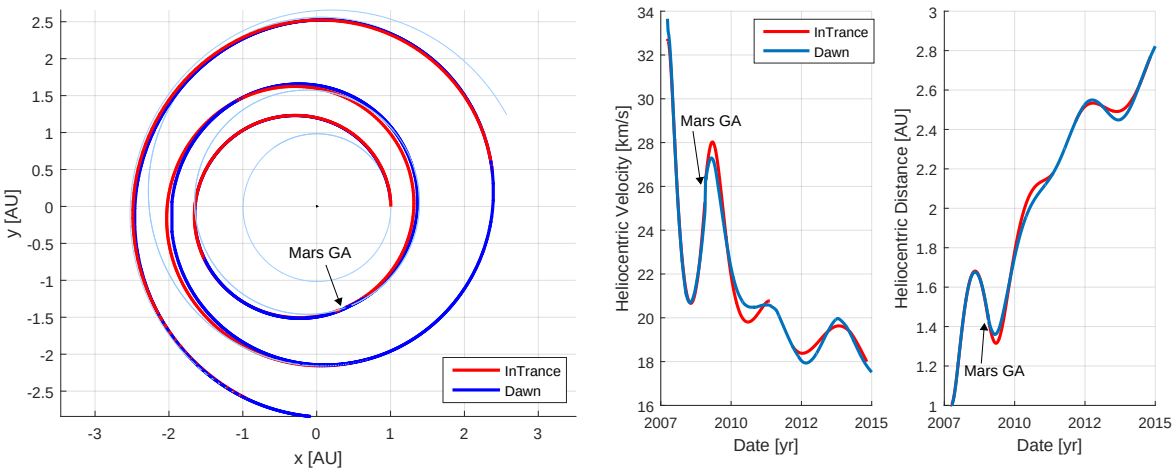


Figure 7.9: Heliocentric trajectory, velocity and distance of the actual Dawn missions and the resulting trajectory from InTrance.

Some mission defining parameters of both the actual Dawn trajectory and the solution found by InTrance are tabulated in Table 7.11. The solution from InTrance is launched 4 days prior to the actual Dawn case and performs its closest approach of Mars 8 days prior. The GA in InTrance supplies a ΔV of 2.41 km/s and an inclination change of 3.3°, whereas the GA in SPICE supplies a ΔV of 2.9 km/s and an inclination change of 3.5°. The solution from InTrance arrives at Vesta on May. 19, 2011, 84 days prior to Dawn's actual arrival. This large deviation is in part caused by the different final relative states to Vesta; InTrance found a relative final position and velocity of $1.07 \cdot 10^6$ km and 716 m/s, whereas the trajectory from SPICE rendezvouses at 6,700 km with a relative velocity of 0.5 m/s. Another source for the discrepancy could be the thrust restriction windows, which are included in the solution from SPICE but not in InTrance. Both trajectories have a broadly similar stay time at Vesta; 293 days from InTrance and 288 days from SPICE. Due to the earlier arrival of the InTrance solution, it departs Vesta 84 days prior to the SPICE solution, however, both have the same arrival date at Ceres. It is unknown how much propellant is used in each phase in the SPICE files, but the overall propellant carried on-board is 184 kg more than required in the solution from InTrance. Therefore, it could be the case that more propellant is consumed in the transfer from Vesta to Ceres, justifying a lower transfer time. Ceres and Vesta furthermore have different orbital periods, rendering a more favourable configuration in the trajectory from SPICE. However, the overall MET of both trajectories are within four days of each other; 2,594 days for the solution from InTrance and 2,590 days for the trajectory from SPICE.

7.2.4. Improvement due to Gravity Assist

The Mars GA in Dawn's trajectory was not strictly necessary, but added to increase technical margins [63], hence it is interesting to investigate the benefits of the added GA. The improvement due to the inclusion of a GA over a direct transfer is investigated by also optimising a double-RV mission to Ceres and Vesta excluding a GA. The input parameters for this simulation are broadly similar to those of the Ceres and Vesta phases in the above discussed simulations, and hence will not be repeated here. The main difference is that the direct transfer is allowed to launch from Earth earlier as its transfer to Vesta will take longer by excluding the GA. The optimisation goal is again to minimise the transfer time.

The optimised direct transfer trajectory resulting from InTrance is shown in Figure 7.10, together with the GA performing solution discussed above. The left-hand figure shows both heliocentric trajectories, the middle the heliocentric velocity over time and the right-hand plot the heliocentric distance over time. The direct transfer can be seen to be launched from Earth earlier and arriving at Ceres later. Since the S/C in the direct transfer reaches Vesta later, it also departs later, however, the direct transfer reaches Ceres earlier due to a more favourable relative geometry of Vesta and Ceres. An enlarged three-dimensional view of the two heliocentric trajectories is available in Appendix B.

Table 7.11: Comparison of InTrance results and actual Dawn mission. Dawn data from SPICE kernels unless stated otherwise.

	InTrance	Dawn	InTrance - Dawn
Launch V_∞	3.362 km/s	3.362 km/s [62]	-
Launch date	Sep. 23, 2007	Sep. 27, 2007	-4 days
Mars encounter	Jan. 27, 2009	Feb. 04, 2009	-8 days
Vesta arrival	May. 19, 2011	Jul. 21, 2011[1]	-63 days
Vesta dwell time	293 days	288 days	+5 days
Vesta departure	Mar. 03, 2012	May. 26, 2012	-84 days[2]
Ceres arrival	Oct. 29, 2014	Oct. 29, 2014[3]	-
Mission elapsed time	2,594 days	2,590 days	+4 days
Total thrust time	1,973 days	2,300 days[4] [63]	327 days
Closest approach Mars	2.39 $R_♂$	1.15 $R_♂$	+1.24 $R_♂$
ΔV due to GA	2.4 km/s	2.9 km/s	-0.5 km/s
Rel. distance Vesta	$1.07 \cdot 10^6$ km	~ 6,700 km	$+1.06 \cdot 10^6$ km
Rel. velocity Vesta	716 m/s	0.5 m/s	+715 m/s
Rel. distance Ceres	$0.96 \cdot 10^6$ km	~ 3,500 km[5]	$+0.95 \cdot 10^6$ km
Rel. velocity Ceres	499 m/s	222 m/s[5]	+277 m/s
Mass at departure	1,056.1 kg	1,240 kg[4] [63]	-183.9 kg
Mass at arrival	747.8 kg	747 kg[6]	+0.8 kg
Thruster	NSTAR	NSTAR	-

[1] Arrival date defined as the date when the S/C has a relative distance to Vesta of $1.07 \cdot 10^6$ km. The actual date when Dawn enters into orbit is around Aug. 11, 2011.
[2] Although having a broadly similar stay-time, the actual Dawn mission requires another 74 days to further decrease the relative velocity and distance to Vesta as indicated in the above footnote. InTrance currently cannot supply this accuracy and hence is neglected.
[3] Arrival date defined as the date when the S/C has a relative distance to Ceres of $0.96 \cdot 10^6$ km. The actual date when Dawn enters into orbit is around Jan. 31, 2015.
[4] This number includes the additional fuel/thrust required to further decrease the rendezvous conditions plus the propellant/thrust required to enter and exit an orbit around both Vesta and Ceres, which is not included in the InTrance optimisation run. The actual Dawn S/C carries an additional 45.5 kg of hydrazine for the reaction control subsystem [73], which is not taken into account in the InTrance simulations.
[5] Final relative state provided in the SPICE kernels.
[6] The dry mass of Dawn is 747 kg, however, there is undoubtedly a propellant reserve left which is not specified in available literature.

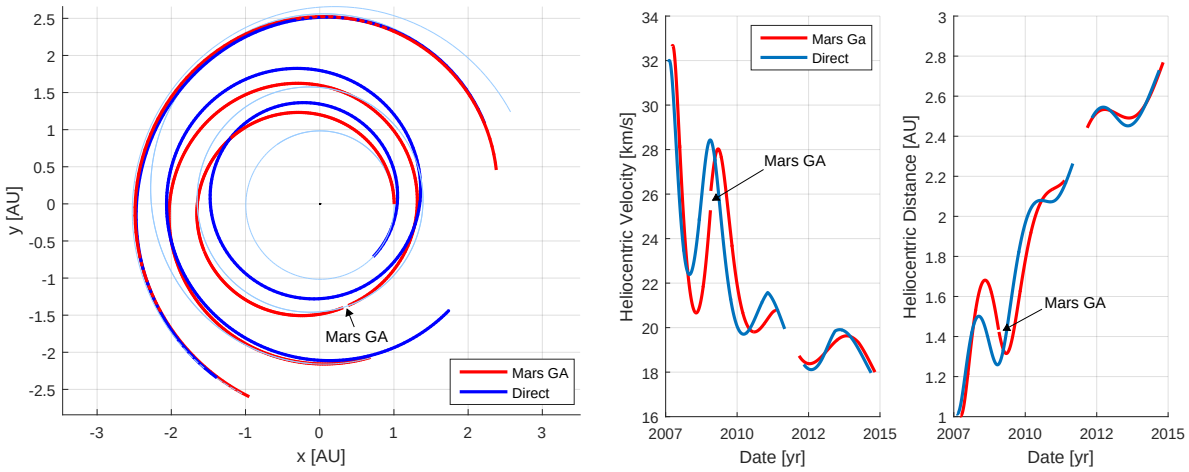


Figure 7.10: Heliocentric trajectory, velocity and distance of the Dawn trajectory computed with InTrance both with and without a Mars GA.

The two trajectories are characterised by the parameters shown in Table 7.12. The solution with the GA launches 46 days later than the direct transfer case, and arrives at Vesta 105 days prior to the direct transfer; the inclusion of a GA hence results in a much shorter transfer time to Vesta. The stay time at Vesta in the direct transfer case is optimised to 245 days, versus 293 days in the Mars GA case. Although the S/C in the direct transfer case departs Ceres 61 days later, it arrives at Ceres 49 days earlier, again caused by favourable geome-

try between Vesta and Ceres. Both scenario's then have a similar MET, 2,594 days including the Mars GA and 2,598 in the direct transfer, and use about the same amount of fuel (308.3 vs 300.7 kg).

Table 7.12: Comparison of InTrance results for the Dawn mission, both with and without a Mars GA.

	Mars GA	Direct transfer	Mars GA - Direct
Launch V_∞	3.362 km/s	3.362 km/s [62]	-
Launch date	Sep. 23, 2007	Aug. 08, 2007	+46 days
Vesta arrival	May. 19, 2011	Sep. 01, 2011	-105 days
Vesta dwell time	293 days	245 days	+48 days
Vesta departure	Mar. 03, 2012	May. 03, 2012	-61 days
Ceres arrival	Oct. 29, 2014	Sep. 10, 2014	+49 days
Mission elapsed time	2,594 days	2,589 days	+5 days
Total thrust time	1,973 days	1,668 days	+305 days
Rel. distance Vesta	$1.07 \cdot 10^6$ km	$0.84 \cdot 10^6$ km	$+0.23 \cdot 10^6$ km
Rel. velocity Vesta	716 m/s	418 m/s	+298 m/s
Rel. distance Ceres	$0.96 \cdot 10^6$ km	$0.99 \cdot 10^6$ km	$+0.03 \cdot 10^6$ km
Rel. velocity Ceres	499 m/s	498 m/s	+1 m/s
Mass at departure	1,056.1 kg	1,047.8 kg	+8.3 kg
Mass at arrival	747.8 kg	747.1 kg	+0.7
Thruster	NSTAR	NSTAR	-

It can indeed be seen that both scenarios can be flown with broadly similar total MET and propellant usage, and that a Mars GA is only included for increased technical margins. Due to the inclusion of a Mars GA, the S/C arrives at Vesta 105 days earlier, rendering a longer possible dwell time (+20%) for scientific operations. There is furthermore more flexibility in the mission, as staying even longer at Vesta renders a more favourable geometry such that Ceres can still be reached at around the same date.

7.2.5. Influence of Thrust During Gravity Assist

To investigate whether a powered Mars gravity assist is beneficial for the Dawn mission, another run of InTrance is performed in which the analytical GA-model is omitted. This way, InTrance integrates the trajectory until exiting the SOI, which marks the end of the first phase. The mission defining parameters are exactly equal to those presented in Table 7.8, with the exception that the arrival at and departure date from Mars are 5 days later to accommodate the time spent within the SOI. The resulting trajectory, combined with the prior obtained results with the analytical GA-model, is shown in Figure 7.11 in the heliocentric frame (left) and as a close-up of the GA portion in the planetocentric frame (right).

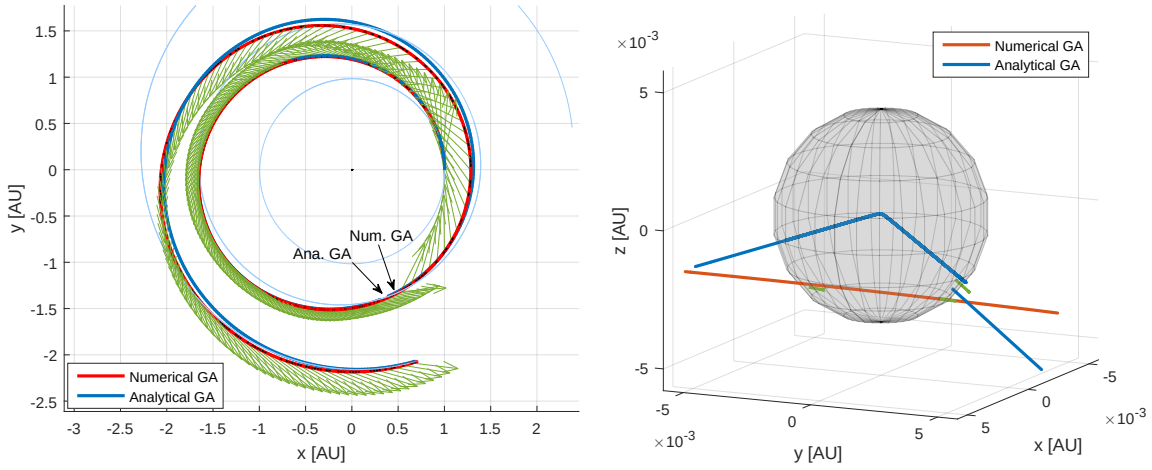


Figure 7.11: Helio- (left) and planetocentric (right) portions of both Dawn trajectories computed with the analytical GA-model (blue) and numerical GA-model (red). Green arrows denote the thrust vectors of the trajectory computed with the numerical GA-model.

From the trajectory in the heliocentric frame, especially for the first phase (Earth - Mars), both trajectories are almost identical on this scale. However, from the close-up on the GA-portion in the planetocentric frame, it becomes clear that no classical GA has been performed. Although the S/C travels through the SOI, it does so at a distance from Mars which is too far to have much of an effect on the trajectory. Instead, the inclination change and ΔV needed to arrive at Vesta are supplied by the thrusters, which are engaged at full throttle during the entire flight time of 1329 days. While inside the SOI, the thrusters supply a ΔV of 1.3 km/s. The flight time is 4.3 days shorter than was the case for the result computed with the analytical GA-model, and hence achieved the goal of minimising the transfer time, although requiring 45 kg (+21%) of additional fuel. Multiple runs, even when further constricting the propellant mass to be in line with the results from the analytical model (104 kg for the first phase), did not result in performing the expected GA. Instead, GAs very similar in shape to the result in Figure 7.11 were found, although with a longer flight time and larger gaps between phases.

7.2.6. Optimisation of the Three-Phase Scenario – Earth to Ceres

The previously discussed trajectories are the result of concatenating the trajectories from two separate runs of InTrance, the first optimising the trajectory from Earth to Vesta with a Mars GA, and the second optimising the trajectory from Vesta to Ceres. However, InTrance can optimise more than two phases simultaneously, therefore, the problem is re-tackled as a single optimisation for the complete three phase mission. The input parameters are exactly the same as those given in Section 7.2.1, except that the the input parameters from Table 7.9 now govern the third phase and no additional 70 kg propellant is taken as reserve in the transfer from Earth to Vesta.

The heliocentric trajectories in the XY-plane, velocity and distance of both the three-phase single optimisation and the previously discussed concatenated runs are shown in Figure 7.12. An enlarged three-dimensional view of the heliocentric trajectory is available in Appendix B. Visually, both optimisations result in trajectories with a similar shape, but the trajectory from the single optimisation run arrives at and departs from Vesta earlier, and arrives at Ceres later.

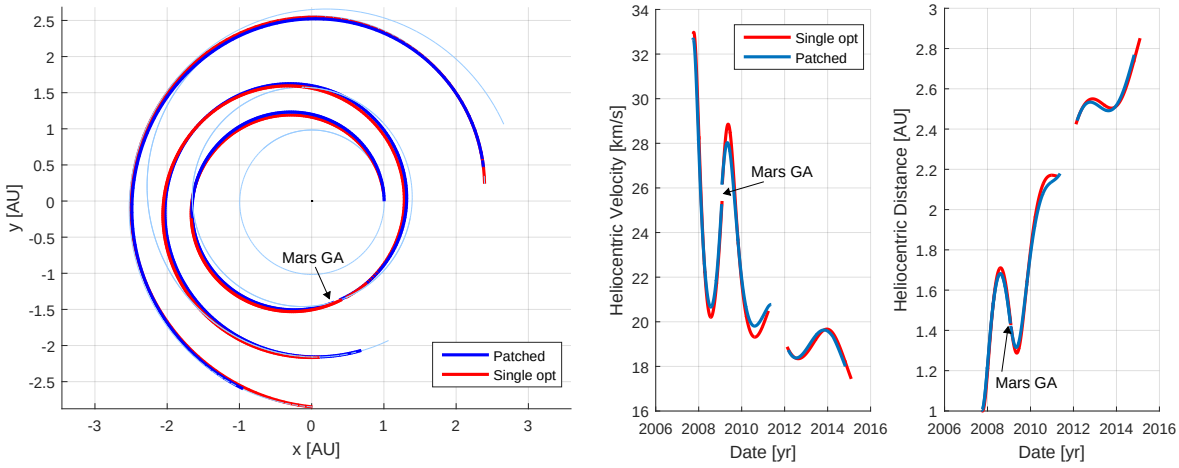


Figure 7.12: Heliocentric trajectory, velocity and distance of the Dawn trajectory computed with InTrance both through a single optimisation of the complete mission (red) and the previously described concatenated simulations (blue).

A close-up of the gravity assist portion in the planetocentric frame is shown in Figure 7.13, which shows a positional gap between the phases which InTrance is unable to bridge to within the specified transition condition threshold of 0.1. Although the velocity gap is almost fully bridged, with a difference of only 76 m/s, the position gap is too large at 39.2 $R_{♂}$, resulting in a state phase transition condition violation (equation 5.4) of 2.48. This state gap was mostly persistent over multiple runs of the same problem, but whenever the gap was closed, the constraints on the final relative states to Vesta and Ceres were not matched.

The main trajectory defining parameters are compared to those of the two separate patched optimisations in Table 7.13. Although the trajectory of the patched optimisation runs starts 7 days prior and performs its closest approach of Mars 7 days prior to the single optimisation run, it arrives at Vesta 49 days later. Although

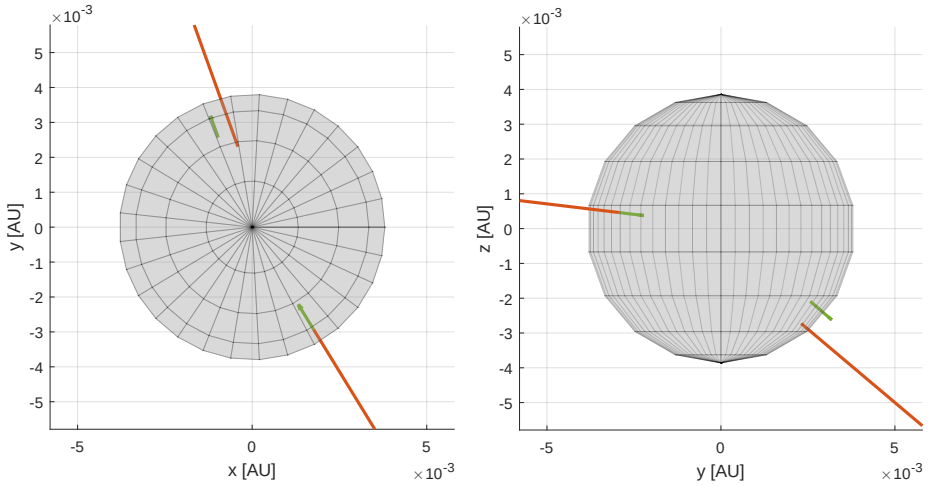


Figure 7.13: Close-up of the Mars GA portion in the planetocentric frame for the three-phase optimisation run. Red lines indicate the trajectory as computed by InTrance, the blue line indicates the externally integrated trajectory while inside the SOI and the green arrow indicates the exit state as computed by the analytical GA-model.

this gives rise to the notion that the transfer from Mars to Vesta is not optimal in the two separate optimisation runs, the result is misleading. First off, the initial position on the rim of the SOI of Mars is more favourable in rendezvousing with Vesta in the three-phase optimisation, but in reality cannot be reached, indicated by the existence of the large positional gap between the phases. Next, the S/C consumes 20 kg of additional fuel in the three-phase optimisation run in the phase from Mars to Vesta, therefore reaching a higher velocity and hence shorter transfer time. Lastly, the relative velocity to Vesta is 264 m/s higher than in the patched trajectory, which would take additional time to decrease. The transfer from Vesta to Ceres takes 120 additional days in the three-phase optimisation, further giving rise to the notion that this trajectory is not the global optimal solution, even when neglecting the positional phase gap.

Table 7.13: Comparison of Dawn trajectories computed with InTrance, both for a single optimisation of the complete mission and the patched results of the previously discussed two separate optimisation runs.

	Patched	Single opt.	Patched - Single opt.
Launch V_∞	3.362 km/s	3.362 km/s	-
Launch date	Sep. 23, 2007	Sep. 30, 2007	-7 days
Mars encounter	Jan. 27, 2009	Feb. 03, 2009	-7 days
Vesta arrival	May. 19, 2011	Mar. 31, 2011	+49 days
Vesta dwell time	293 days	318 days	-25 days
Vesta departure	Mar. 03, 2012	Feb. 12, 2012	+20 days days
Ceres arrival	Oct. 29, 2014	Feb. 06, 2015	-100 days
Mission elapsed time	2,594 days	2,686 days	-92 days days
Total thrust time	1,973 days	2,363 days	-390 days
Closest approach Mars	2.39 $R_♂$	3.54 $R_♂$	-1.15 $R_♂$
ΔV due to GA	2.4 km/s	1.69 km/s	+0.71 km/s
Rel. distance Vesta	$1.07 \cdot 10^6$ km	$0.93 \cdot 10^6$ km	$+0.14 \cdot 10^6$ km
Rel. velocity Vesta	716 m/s	980 m/s	-264 m/s
Rel. distance Ceres	$0.96 \cdot 10^6$ km	$1.08 \cdot 10^6$ km	$-0.12 \cdot 10^6$ km
Rel. velocity Ceres	499 m/s	724 m/s	+225 m/s
Mass at departure	1,056.1 kg	1,101.3 kg	-45.2 kg
Mass at arrival	747.8 kg	756.8 kg	+9 kg
Thruster	NSTAR	NSTAR	-

InTrance is unable to optimise this three-phase trajectory directly to within the predefined accuracy requirements. The dimensions of this problem are much larger than those of the two-phase simulation discussed before; testament by the fact that the chromosome in the three-phase scenario has a length 2632 alleles, com-

pared to 1718 in the two-phase run. The search space is furthermore much larger due to the inclusion of the additional phase, which can clearly be seen in the optimisation runtime; almost 36 hours for the single three-phase mission and 8.75+0.75 hours for the two separate simulations. InTrance is unable to find favourable initial conditions together with a proper steering strategy, and eventually hits the maximum value of the ΔV_b window of 3.0 km/a at the start of the 2$^{\text{nd}}$ phase and optimises a steering strategy which brings the S/C to the respective targets. Although an almost physically valid trajectory results, neglecting the positional phase gap at the GA, this is not the global optimal solution, neither in time nor in propellant usage. The EA becomes stuck in a local optimum; whenever more favourable initial conditions are found, the steering strategy is trained with different conditions, rendering a trajectory which no longer reaches the respective targets, and therefore individuals which go extinct. Although only slightly changing the initial conditions after the GA at Mars, which are describes by a few alleles, all other 2600+ alleles have to change to values which are not necessarily close to their previous values. After all, a small change in initial conditions, under the same steering strategy, has tremendous effects on the trajectory over a long time frame.

8

Conclusion and Recommendations

This chapter concludes this thesis by first providing a summary of this work, which simultaneously highlights the answer to the research questions as specified in Section 1.2. A critical analysis on the obtained results and implementation follows in Section 8.2, from which recommendations for future research are given in Section 8.3. The recommendations are split in *fundamental research* and *application oriented* recommendations; the former focussing on the fundamental components of InTrance, and the latter on areas in which the developed method can be used and improved upon from an astrodynamics standpoint.

8.1. Summary

It was quickly realised that the outer planets of the Solar System were out of reach with classical chemical propulsion as sole means of supplying the ΔV budget. Gravity assists were some of the first innovations which made it possible to explore targets beyond Mars' orbit, utilising a momentum exchange between an intermediate flyby body and the spacecraft, resulting in an increase of the spacecraft's velocity relative to a third body such as the Sun. The very first missions to the outer planets, such as the Pioneers and Voyagers, all made use of gravity assists to increase their velocity relative to the Sun, and most mission still do to this day.

With all major bodies within the Solar System explored by at least a single fly-by, modern-day missions are becoming increasingly more demanding, up to a point where classical chemical propulsion combined with gravity assists can no longer supply the required ΔV. More and more is relied upon low-thrust propulsion, characterised by its (very) low thrust force; long continuous thrust arcs, often lasting months at a time; and high specific impulse. Low-thrust can be supplied by methods from two different categories; electric propulsion and internal-reaction mass free propulsion. The former makes use of power to electrically accelerate and expel mass, although with a much smaller mass flow rate than classical chemical propulsion. The latter does not rely on any propellant, but rather makes use of the (electro-)magnetic environment with methods such as solar sails, electrodynamic tethers and laser propulsion.

To even further increase the possible ΔV budget, thereby allowing more intricate missions, more payload, or a lower transfer time; use is made of low-thrust propulsion combined with gravity assists. A prominent example is the Dawn mission which orbited both Vesta and Ceres and made use of a Mars GA, a mission which would not have been possible with classical chemical propulsion. Dawn made use of a Mars GA to increase technical margins, but it would have been possible to perform the mission without one. On the contrary, a mission which heavily relies on both low-thrust and gravity assists is BepiColombo, performing a total of six gravity assists to eventually be captured by Mercury, with very low propellant usage.

8.1.1. Research Motivation and Framework

Traditional high-thrust trajectory optimisation is relatively straightforward compared to its low-thrust counterpart; thrust is only applied at a few instances and each operation is modelled as an instantaneous burn. These relatively simple trajectories were traditionally designed by a team of astrodynamics experts by hand, followed by simple numerical integration. Contrary, low-thrust can be applied for months at a time, where

at each time step the direction and magnitude can be varied, resulting in a much larger search space. With the inclusion of gravity assists, the optimisation of the trajectory becomes even more difficult. Traditional trajectory optimisation tools heavily rely on astrodynamics expertise to generate an initial guess to a local optimisation scheme. Global optimisation tools exist, but usually require heavy modification for each new mission scenario, are not capable of optimising *any* trajectory, and still require experts in the fields of astrodynamics, optimal-control theory and optimisation. There is a clear need for a *smart* global low-thrust trajectory optimisation tool, capable of optimising low-thrust trajectories from only a broad description of the mission.

One such *smart* global low-thrust trajectory optimisation tool, termed InTrance, was developed by Dachwald [17] and later extended by Ohndorf [59]. InTrance tackled the problem from the novel perspective of artificial intelligence and machine learning, using a method termed evolutionary neurocontrol to design multi-phase low-thrust trajectories. Neuroevolution, or evolutionary neurocontrol when applied to control problems, combines biologically inspired Artificial Neural Networks (ANNs) with Evolutionary Algorithms (EAs). The internal parameters of the ANN and initial conditions of each phase are optimised by the EA, which then serves as an agent; supplying the S/C with a steering strategy at each integration step. Although InTrance can optimise a wide variety of trajectories from only a basic description of the mission, it lacks the generic support for gravity assists.

With missions becoming more intricate, nowadays making use of both low-thrust and gravity assists, the above described need is extended to the need for a *smart* low-thrust *gravity assist* optimisation tool. It was the aim of this thesis to supply in the need for such a tool, and the solution was found in extending the above described InTrance to allow for the optimisation of low-thrust gravity assist trajectories. The objective was formally defined as:

Research Objective: *To develop and use a smart low-thrust trajectory optimisation tool using evolutionary neurocontrol by extending InTrance, capable of robustly optimising preliminary low-thrust gravity assist trajectories in both transfer time and propellant usage which should only be dependent on basic mission defining inputs.*

This developed *smart* low-thrust gravity assist trajectory optimisation tool was then used to optimise gravity assists at bodies with both small and large spheres of influence, to optimise trajectories which make use of gravity assists at bodies with both small and large spheres of influences, and to provide insight into the effects and efficiency increase due to the inclusion of both powered and unpowered gravity assists.

8.1.2. Approach and Results

InTrance splits the trajectory into multiple phases, each with their own respective targets and governed by a distinct ANN, also termed Neurocontroller (NC). The internal parameters of all NCs plus all initial conditions of each phase are encoded on a single chromosome, which is optimised in its entirety by an EA. Gravity Assists (GAs) have been implemented as a target in itself, marking the end of a phase when crossing into the Sphere of Influence (SOI) of the GA-body. As an example, a flyby mission of Pluto which performs a GA at Jupiter would be characterised by two phases; the first taking the S/C from Earth up to entering the SOI of Jupiter; the second phase then commences on the rim of the SOI after the GA and takes the S/C to Pluto where it performs a flyby.

Sub-fitness functions first drive each phase to reach their respective targets, and only after all targets of all phases have been reached, the focus is shifted to the overall optimisation objective such as minimising the transfer time or propellant usage of the entire trajectory, and to minimise gaps in phase transitions. The phase starting after a GA has an initial position on the rim of the SOI of the GA-body, optimised by the EA. The initial velocity of this phase is either the entry velocity into the SOI of the previous phase plus a $\Delta\mathbf{V}$ optimised by the EA, or the velocity of the GA body at the departure date plus a $\Delta\mathbf{V}$ optimised by the EA. The sub-fitness function of the phase ending with a gravity assist first drives the S/C to enter the SOI of the GA-body. Once the SOI is entered, the integration is stopped and an analytical GA-model is employed to calculate the SOI exit state and resulting $\Delta\mathbf{V}$ vector. The sub-fitness of this phase is then determined from the difference between the actual generated $\Delta\mathbf{V}$ due to the GA and the initial $\Delta\mathbf{V}$ as determined by the EA of its subsequent phase. The phase ending with the GA is therefore inherently coupled to its subsequent phase; initial conditions at the departure body plus a steering strategy have to be optimised which render the S/C to perform a GA which supplies the $\Delta\mathbf{V}$ needed by its subsequent phase to reach its respective target.

The implementation has been verified and validation by external re-integration and by comparison of InTrance optimised trajectories with literature. The analytical GA-model has firstly been verified by simulating the gravity assist portion from a fixed initial position on the rim of the SOI for both Mars and Jupiter for varying entry velocities, and comparing the integrated exit states with those computed with the analytical model. The simulated trajectories had deflection angles ranging from a few degrees to >120 degrees. The deviation between the numerically integrated simulations and analytical model grows with increasing deflection angle, or equivalently, with decreasing entry velocity. The simulations at Mars show a deviation in both SOI exit velocity and position of about 6% for the most extreme deflection angles, whereas due to the larger SOI the deviations reach values of up to 26% at Jupiter. It is problem dependent whether these deviations can be deemed acceptable. Generally, deflection angles will not reach values as high as 120 degrees, and some deviation is permissible given InTrance is designed for preliminary phase-A mission design. When higher accuracy is required, or when an extreme gravity assist is expected, the analytical GA can be omitted which will result in InTrance integrating the trajectory until exiting the SOI. However, due to the vast variation in the gravitational environment, especially around the point of closest approach, a small integration step-size has to be applied which does hinder the performance.

A low-thrust version of the New Horizons mission to Pluto with a Jupiter GA was optimised by InTrance and compared to literature. The windows for the initial and arrival conditions at each phase were set to resemble the validation trajectories as closely as possible, thereby focusing on the optimisation of the steering strategy and gravity assist. The resulting trajectory was compared to two different trajectories from literature [13, 75] and all showed very good agreement. All three solutions had a total flight time within 5 days (0.15%) of each other, use about the same amount of propellant to within 0.4 kg (1.1%) and achieved a flyby velocity relative to Pluto to within 34 m/s (0.22%) deviation. The resulting InTrance trajectory was furthermore integrated with an RK4(5) integrator in MATLAB from the initial conditions at Earth onward while applying the thrust history supplied from InTrance. The externally re-computed trajectory does not completely agree with the solution from InTrance, which is mostly caused due to a state gap after the GA and a different integration step-size between the two computations. However, the deviation was small enough to further validate the implementation of the GA within InTrance.

The InTrance optimised low-thrust version of New Horizons used in the validation case aimed to mimic the trajectories from literature by settings very strict windows on the input dates and propellant mass. However, it was found that a gain could be achieved by optimising within a larger search space. Therefore, InTrance has been used to re-optimise the low-thrust New Horizons trajectory with larger windows for the initial and arrival conditions, resulting in a flight time reduction of about 90 days (-3%) and a propellant mass reduction of 4.1 kg (-11%). Compared to New Horizons' actual high-thrust trajectory, the InTrance result arrived at Pluto 12 days earlier, although at a further distance and larger relative velocity. InTrance found a GA with a lower closest approach to Jupiter, resulting in a larger generated ΔV (+17%). The S/C's drymass was 565.0 kg in the InTrance simulations, similar to the validation data used before, but the actual high-thrust New Horizon's drymass was 426.4[(1)] kg. InTrance furthermore used a lower launch C_3 of 144 km^2s^{-2}, versus the 157 km^2s^{-2} in the high-thrust mission. Hence, the low-thrust trajectory from InTrance reached Pluto faster, with 138.6 kg of additional drymass (=payload) and used 20.8 kg (-41%) less propellant. Although the two trajectories cannot be compared directly, as InTrance uses a simplified environment (no SRP, not all disturbing bodies, etc.) and does not take operational constraints into account, InTrance has shown to be able to find low-thrust gravity assist trajectories which are most likely better than its high-thrust counterpart in terms of flight time and propellant usage. The effect of the gravity assist on the overall trajectory was assessed through comparison with a direct Earth-Pluto transfer, also optimised with InTrance. This direct transfer was optimised with respect to propellant usage and constrained to still have a reasonably close MET to the solution with a Jupiter GA. Jupiter has been excluded from the list of disturbing bodies to prevent an implicit GA, which would be equivalent to launching at a later date when Earth and Pluto have the same relative geometry, but where Jupiter is not in the vicinity. The direct-transfer utilises the entire allowed flight time of 3,200 days, which is 109 days more when compared to the solution with a Jupiter GA. It then requires a total of 126.8 kg of propellant to achieve this flight time, which is 96 kg more than for the Jupiter GA trajectory. Lastly, it was investigated whether allowing thrusting while inside the SOI of Jupiter would be beneficial. By omitting the analytical GA-model and have InTrance integrate the trajectory until exiting the SOI, thrust can be applied while performing the GA. However, InTrance did not find a favourable powered GA solution, and returned a trajectory very similar to the one generated using the analytical GA-model.

[(1)] Including a propellant reserve of 25.4 kg.

The second mission which was optimised with InTrance is the Dawn mission; a double asteroid rendezvous mission with a Mars GA. The resulting trajectory is a concatenation of two separate optimisation runs of InTrance, the first from Earth to Vesta with the intermediate Mars GA and the second from Vesta to Ceres. The resulting trajectory is very similar to Dawn's actual trajectory in terms of MET, being four days longer but also having a five days longer stay time at Vesta. InTrance finds a solution which has a faster transfer to Vesta (-84 days), and therefore departs earlier from Vesta, but arrives at Ceres on the same day as the actual trajectory. The longer flight time in the phase from Vesta to Ceres is accredited to more favourable geometry between the two bodies at a later departure date from Vesta and a different thrust strategy. The trajectories are difficult to compare in much further detail, as the actual Dawn S/C brought along more propellant (+184 kg, +17%) to, amongst others, account for the orbiting and de-orbiting of both Vesta and Ceres. Furthermore, InTrance is not capable of supplying the accuracy to rendezvous with both bodies with the same relative distance and velocity as in the actual Dawn trajectory, which would require additional propellant. Comparing the trajectory to a trajectory which does not perform a GA shows the GA in this mission was mostly included to increase technical margins, as both missions can be flow with a broadly similar MET (5 days difference, 0.2%) and a broadly similar propellant usage (7.6 kg difference, 2.5%). However, the direct transfer trajectory has to launch from Earth sooner (-46 days) and arrives at Vesta later (+105 days). Due to a later departure from Vesta with more favourable geometry with respect to Ceres, and a shorter dwell time at Vesta (245 vs. 293 days), the overall MET is broadly similar. The effect of a powered GA is difficult to assess as InTrance found a solution which does not perform a classical GA. Instead, the required change in both inclination and velocity is supplied by the thrusters, which are engaged longer and burn an additional 45 kg of propellant. The complete three-phase mission has also been tackled by InTrance, but was unable to escape a local optimum and unable to close the gap between phases to within the allowed threshold.

8.2. Analysis of Results and Performance

The extended version of InTrance with GA optimisation capabilities has shown to produce broadly similar, or sometimes even better, trajectories compared to the actual flown missions and literature. The trajectories of the actual missions required a team of experts in the fields of astrodynamics, optimal control theory and optimisation, and many different software tools. On the contrary, InTrance was initialised with only a broad description of the missions such as departure and arrival bodies, desired launch and arrival windows and allowed propellent mass. The novel application of evolutionary neurocontrol to the low-thrust gravity assist optimisation problem has shown to be promising, which is quite remarkable considering a neurocontroller has no a-priori knowledge of what a gravity assist is, and has to learn how to properly perform a GA through evolution.

8.2.1. Analysis of Performance, Implementation and Robustness

The optimisation is very sensitive to the initial conditions of a phase starting after a GA. The New Horizons trajectory, for instance, did not converge to a valid solution when provided with relatively large windows (+100°) for the initial direction of the launch position on the SOI and ΔV due to the gravity assist. The first issue is that InTrance retrieves a single allele value which is used in a linear mapping from the minimum to maximum window parameter, hence rendering that when the window is large, a slight change in allele value has a large effect on the parameter. Next, when only slightly changing the initial launch position angles, represented by two alleles, all other 2000+ internal ANN parameters also have to be re-optimised. After all, a slight change in initial conditions will result in providing the NC with different states for which it is not trained, giving a completely different steering strategy. As it is unlikely all 2000+ parameters are re-optimised within a single epoch to accommodate these new initial conditions, therefore these individuals go extinct in favour of individuals which result in a state gap but do fulfil all other constraints. Multiple runs of the problem where therefore required to first provide insight into promising windows of the initial conditions, which eventually resulted in windows with a size varying between 10 and 40 degrees. The same problem was encountered in the Dawn simulations, but was then tackled by allowing InTrance to alter the windows of the initial conditions, thereby both shifting the midpoint and expanding and contracting the size of the windows. This resulted in a simpler initialisation of the problem, where InTrance found initial conditions outside of the initially specified window for the direction of the ΔV of the GA.

The Dawn simulations had to be run multiple times with different initial (random) chromosome parameters before a valid trajectory which fulfilled all constraints was found. This problem did not occur in the optimi-

sation of the New Horizons trajectory, which was quite robust and returned trajectories with a similar flight time (the optimisation goal) to within 2 days for each run once proper windows for the initial angles after a GA were found. InTrance has difficulty in decreasing the gap between the SOI exit position and initial position of a phase starting from a GA-body with a small SOI, which was also seen in the single optimisation of the three-phase Dawn mission. This is again attributed to the EA, which was not designed with the required accuracy for a GA in mind. After all, InTrance was designed as a phase-A mission analysis tool, not having to provide the accuracy which one would achieve with a local optimisation scheme. This gap can either be decreased by changing the entry position into the SOI, resulting in a different exit position, or to alter the initial position of a phase starting from the GA-body. The difficulty in altering the launch position of the phase starting after the GA was already described above. The second option, decreasing the positional gap from optimising to a different entry state into the SOI, is no trivial task for InTrance either. As was shown with the validation of the analytical GA-model, an increase of 500 m/s in entry velocity at Mars could result in the deflection angle increasing from 70° to 120°, rendering a much larger state gap. To put this into perspective, InTrance was developed to be capable of rendezvousing with bodies at a relative velocity of 500 m/s. Hence, the EA cannot always supply the accuracy to change the entry state by a few tens of m/s required to reach an exit position equal to the initial position of the second phase at small SOIs. Multiple runs of InTrance usually alleviate the problem, at least for the two-phase Dawn scenario, due to different (random) initialisation of allele values.

It was found that the analytical GA-model performed both more robust and more efficiently than numerically integrating the trajectory inside the SOI. Empirical evidence of monitoring the optimisation showed that the NC had more difficulty in learning the benefits of a GA when numerically integrating the trajectory inside the SOI. Many individuals performed a GA which did not result in much of a change in velocity nor direction, as was seen in the final Dawn result. Contrary, the analytical model showed much quicker convergence to GA performing individuals and the resulting trajectories were much more robust. The optimisations using the analytical model was furthermore more than four times as efficient with respect to computation time for the New Horizons case. This is caused by the small required (dynamically computed) step-size while inside the SOI, which makes up 60% of the total steps in the first phase, but only 14% of the total flight time. Due to Mars' small SOI size, there was no clear difference in runtime in the Dawn simulations.

GAs are implemented as targets in itself, meaning they have to be provided to InTrance as a target of a respective phase with a full set of input parameters such as propellant usage and arrival date. This decision was taken as previous work in optimising a GA with neurocontrol without specifically specifying the target was unsuccessful [14]. Usually, a GA sequence is unknown a priori and a maximum propellant mass is defined for the complete mission, rather than for each separate phase. The current implementation somewhat goes against the idea of InTrance being a smart global low-thrust trajectory optimisation tool, as defining a GA sequence and corresponding input parameters still requires expertise in astrodynamics. However, this work has shown evolutionary neurocontrol can be used to optimise low-thrust gravity assist trajectories, which can be build upon by including a heuristic method which solves the combinatorial GA-sequence problem beforehand.

The proximity functions for the phase ending with a gravity assist make use of scaling factors, as discussed in Section 5.6. These scaling factors are used to place a larger weight on performing a GA than on reaching the target in the subsequent phases, which was seen to help the initial convergence to GA performing individuals. These scaling factors were set somewhat arbitrarily, but showed good results for all generated trajectories. However, it might be possible that a different value for these factors could result in faster convergence, which has not been analysed.

8.2.2. Analysis of Results

It has been demonstrated that InTrance can be used for preliminary design of low-thrust gravity assist trajectories, performing different types of GAs at bodies with both small and large SOIs. The gravity assist in the Dawn trajectory is distinctively different from the one in the New Horizons case. New Horizons mostly made use of its Jupiter GA to increase its in-plane velocity, whereas Dawn makes use of the GA mostly as an out-of-plane manoeuvre. The SOI's of both planets are furthermore very different, with the SOI of Jupiter being 83.5 times larger in radius than the one of Mars; testament by the fact that the flight time within the SOI in the New Horizons case was about 60 days, whereas Dawn spends only about 5 days inside the SOI of Mars. Optimising a GA at Mars is therefore much more difficult, as the number of generated trajectories which actually enter the SOI is initially much smaller. A slightly different entry state furthermore has a much

larger effect than would be the case at Jupiter, as can be seen in the validation of the analytical GA-model; at Mars a 1 km/s relative entry velocity has a deflection angle of about 1°, and an entry velocity of 5 km/s gives a deflection angle of about 120°; to achieve the same deflection angles at Jupiter one would require an entry velocity of 5.5 km/s and 20 km/s, respectively.

The efficiency increase due to the inclusion of a GA has been investigated for both the low-thrust New Horizons mission and the Dawn mission. The generated trajectories were compared to InTrance optimised direct-transfer trajectories, and in both cases the inclusion of the gravity assist was favourable. InTrance optimised the propellant mass for the New Horizons direct-transfer case and was constrained to having a broadly similar flight time as the trajectory with a GA. Multi-objective optimisation with respect to both time and propellant usage is currently not available in InTrance. The inclusion of a Jupiter GA in the New Horizons case resulted in a reduction of 109 days (-3.4%) in flight time and a propellant saving of 96 kg (-75.7%). The increase in efficiency is partly caused by the S/C being launched with a higher C_3 (+3.4%), which is possible due to its lower initial wet mass, but mostly caused by the velocity increase of 6.01 km/s due to the GA. The effect of the Mars GA in the Dawn scenario is less apparent, as it was included to increase technical margins, and the baseline mission could be flown with one [63]. It was found that the MET of the mission with a Mars GA was 5 days longer (+0.2%) and required 7.6 kg (+2.5%) of additional propellant compared to the direct-transfer trajectory. However, due to the inclusion of the GA the S/C can be launched 46 days later, arrives at Vesta 105 days earlier, spends 48 additional days at Vesta (+19.6%), and arrives at Ceres on the same date. Hence, with slightly more propellant usage, there is both an increase in launch margins plus increased scientific return, both from the Mars flyby and from the longer dwell time at Vesta.

The effect of a powered GA is difficult to assess for the New Horizons and Dawn missions. InTrance did not find a New Horizons solution which performed a powered GA to be more optimal than one which performed an unpowered GA. This effect was expected, as thrust is supplied in the first half of the first phase which directs the S/C to the optimal b-plane target point. If a closer flyby of Jupiter would be beneficial, it could fully be reached by altering the b-plane penetration point prior to entering the SOI. Employing the numerical GA-model for the Dawn mission resulted in a trajectory which performed a GA which did not alter the flight direction. Instead, the required changes in inclination and velocity to reach Vesta were supplied by the thrusters, thereby burning 21% more fuel than in the trajectory computed with the analytical GA-model. A multi-objective optimisation with respect to both time and propellant usage is needed to properly assess the effects of the powered GA on the Dawn trajectory, which is currently not implemented. However, it is not that expected a powered GA would be much more beneficial than an unpowered one, as again the required closest approach can be reached by altering the b-plane target point prior to entering the SOI. The result computed with the analytical model furthermore does not apply thrust for the first 118 days, indicating that the unpowered GA is able to provide the required velocity and direction change without thrusting. Lastly, due to the relatively low mass of Mars, the flighttime inside the SOI is less than 5 days, in which low-thrust cannot make much of a difference. An increase in efficiency for powered GAs is expected to be seen in SEP missions to the outer planets, in which the maximum thrust decreases with increasing distance from the Sun. For instance, when performing a GA at Jupiter to the outer planets; thrusting for the entire flighttime within the SOI of +60 days can make a large difference as corrections later on are difficult [75].

It was one of the goals to optimise trajectories which perform multiple GAs, however, this goal was not investigated due to time constraints. Optimisation of trajectories with multiple consecutive GAs could be difficult with the current implementation, as each GA-phase is optimised to provide the ΔV required by its subsequent phase. Hence, when GA-phases follow each other, all these phases are inherently linked to each other, whereas InTrance traditionally optimises each phase separately in reaching its target prior to optimising the overall objective and state gaps between phases. Although principally not a problem as all phases are co-evolved, it does significantly increase the difficulty of the problem as now all initial and arrival conditions have to be optimised at the same time. Furthermore, it was found that the three-phase Dawn mission was unable to find an optimal trajectory, whereas the two-phase mission excluding the transfer from Vesta to Ceres could, attributed to the larger problem dimension. When optimising a trajectory with two GAs, the minimum number of required phases is three, which already significantly increases the problem dimensions due to the internal parameters of the additional NC. As an example, considering a three-layered ANN with 35 nodes in its hidden layer results in +1100 alleles per additional phase. It is expected that a trajectory with two consecutive GAs at bodies with large SOIs, such as a Grand Tour with GAs at Jupiter and Neptune [75], can be optimised with the current optimisation when the windows of initial conditions following a GA are sufficiently small. Intricate trajectories, such as low-thrust gravity assist trajectories to Europa, which can utilise

as many as 16 GAs [74], are not expected to be within the capability of the current framework.

8.3. Recommendations for Further Work

This work has resulted in the development of a *smart* low-thrust gravity assist optimisation tool and demonstrated both the effects of gravity assists on low-thrust trajectories, and has shown to be a viable alternative for the design of such trajectories for preliminary phase-A design. However, the method can still be further improved upon, thereby further increasing InTrance's capability to become a truly *smart* and versatile low-thrust trajectory optimisation method, capable of optimising *any* low-thrust mission. The recommendations for further work are split in *fundamental research* and *application oriented* recommendations, the former with a focus on InTrance's components such as the ANN and EA, and the latter with a focus on further improving InTrance's GA and mission analysis capabilities.

8.3.1. Fundamental Research

An interesting research topic is the quest for the optimal NC input set for the low-thrust trajectory optimisation problem. The currently implemented input set of InTrance for each NC is comprised of; the S/C state in three different representations, two of which are relative to the major body, and one relative to the target body; the state of the target body in Cartesian coordinates; the control step size h; the range; the range rate; and the available propellant mass m_p. The state relative to the major body is both supplied to the NC in Cartesian coordinates and polar coordinates, whereas the relative state to the target is only supplied in Cartesian coordinates. It is clear that half of these inputs are heavily correlated and redundant, which can hinder the learning process [43]. For instance, the ANN should be capable of learning how to determine the relative state of the S/C with respect to the target body, thereby eliminating the need for 6 out of the 28 input nodes. Furthermore, as the S/C state in polar coordinates is a simple transformation from the Cartesian state; this is another redundant input set. The dimension of the ANN, assuming the complete input set of 28 nodes, a single hidden layer of 30 nodes, and an output layer of 6 nodes, is equal to 1148 parameters (connection weights, thresholds and sigmoid temperature parameter) that are subject to optimisation. When eliminating the correlated input nodes (14 nodes), the ANN size is reduced by 39% to 700 optimisation parameters. Reducing the input set will mean that the network has to discover such dependencies itself, which could be costly, but the optimisation procedure of the internal parameters itself would be more efficient. It would be interesting to apply Input Variable Selection (IVS) methods [49] to first find how correlated the different input parameters are, and subsequently determine which parameters the ANN relies on most. If it is found that the ANN is not efficiently able to learn these dependencies without providing them as inputs, an alternative solution could be found in using indirect encodings, such as employed within HyperNEAT [70], which are specifically designed to recognise and utilise such dependencies [43]. The large problem dimensions can already be seen to hinder the optimisation, testament by the fact that the complete three-phase Dawn mission could not be optimised to within the allowed thresholds. Removing redundant input parameters significantly reduces the problem dimensions, and could be a possible solution to allow direct optimisation of a three-phase mission.

The quest for the optimal ANN topology would be another interesting research topic in reducing the problem dimensions and increasing efficiency. The topology should be large enough to represent the intricate dynamics, but small enough to allow efficient computation. Such optimal topologies can be generated with Topology- and Weight Evolving Artificial Neural Networks (TWEANNs), of which NeuroEvolution of Augmenting Topologies (NEAT) [71] is a particularly popular method. The reader is referred to the Literature Review [43] performed prior to this work for a discussion on TWEANNs. Gomez et al. [33] found that methods such as NEAT can significantly outperform conventional neuroevolution on both Markov and non-Markov control problems. NEAT was able to robustly and consistently optimise the non-Markov double-pole balancing problem, whereas conventional neuroevolution failed in nearly 40% of their tests. TWEANNs can furthermore find recurrent connections, which are required to represent non-Markov control problems, and could benefit the optimisation of low-thrust trajectories. If such a method would decrease the problem dimensions or perform the optimisation more efficient, InTrance's convergence limits can be decreased which might result in the needed accuracy increase with respect to phase transition conditions.

Diploid Genetic Algorithms (DGAs) provide another possible alternative to the currently implemented EA. DGAs make use of the biological diploidy and dominance mechanisms. In the diploid structure, two homologue chromosomes are twisted together into a duplex structure. In the reproduction cycle, meiosis splits these

two chromosomes into four haploid chromosomes, which are recombined with other haploid chromosomes to form a new diploid chromosome. Diploidy organisms furthermore require two genes for each biological trait, but only the dominant trait is expressed. The recessive gene then helps to keep genetic diversity which allows organism to evolve more efficiently to changes in the environment. Haploid genetic algorithms, as the one implemented in InTrance, have a goal of converging the entire population to an identical chromosome. Therefore, once convergence is achieved, haploid genetic algorithms are unable to adapt to changes in the environment. On the contrary, due to the recessive genes of a diploid chromosome in DGAs, genetic diversity is maintained which helps in adapting to a new environment [83]. Since InTrance first focusses on reaching each respective phases' target, state gaps between phases inevitably occur. Once each phases' target is reached, InTrance shifts its attention to decreasing these state gaps between phase transitions and optimising the overall objective such as transfer time minimisation. In order to decrease the state gaps, the initial condition are usually varied, which has shown to be no trivial task on large problems such as the three-phase Dawn scenario. DGAs might be more suitable for this type of problem, as they are better suited to handle these changes in initial conditions. DGAs are furthermore more robust than conventional haploid genetic algorithms [10], which could help to overcome having to run InTrance multiple times before finding a valid solution for some problems.

8.3.2. Application Oriented

Gravity assists have been implemented as specific targets in their own respective phase, thereby requiring a complete set of input parameters including arrival and departure dates. Ideally, a gravity assist sequence should not have to be provided, in-line with the original goal of InTrance to be a truly smart low-thrust trajectory optimisation method which does not rely on the astrodynamics expertise of the user. More general, this also applies to targets in the multi-phase framework, such as multi-asteroid rendezvous missions as Dawn. It would be ideal to only provide a list of targets, from which InTrance can find its own sequence with respective parameters for each phase. It is therefore recommended to combine InTrance with a heuristic method to solve the combinatorial problem beforehand.

InTrance is a global optimisation tool, and hence provides solutions to the low-thrust trajectory optimisation problem with limited accuracy. Although the accuracy is sufficient for InTrance's original intended use as a phase-A mission analysis tool, the correct optimisation of GAs requires a better accuracy than a flyby or rendezvous mission, even in the preliminary design phase. A gravity assist is usually applied to increase the heliocentric velocity by a few km/s, however, if there is a velocity gap of 500 m/s between the phases arriving at and departing from the GA-body, this has a large impact on the phase starting from the GA-body, which can give a distorted view of realistic flight times and required propellant mass. Besides the recommendations of the previous section, the combination of InTrance with a local optimisation scheme could supply the needed accuracy. The base version of InTrance would then first be used to find (near-)global optimal trajectories, after which its solution can be used as an initial guess to a local optimiser.

Multi-objective optimisation is not supported in InTrance, which leads to having to specify the maximum flight times and maximum propellant usage for each phase. When the goal is to minimise transfer time, InTrance will often use the entire allowed propellant mass, without giving any indication to how much longer the flight time would be with less propellant. Contrary, the reverse is true in minimising the propellant usage. Multi-objective optimisation for both flight time and propellant usage would be a desirable addition from a usability standpoint. One method to implement multi-objective optimisation might be to evolve two different populations simultaneously, one which minimises transfer time and the other which minimises propellant usage. A multi-tournament reproduction between the two populations, with a relative weighting to each criteria set by the user, would provide solutions which are both time and propellant optimal.

From a low-thrust gravity assist optimisation standpoint, it would be interesting to determine InTrance's capability in optimising multi-GA trajectories. As was previously discussed, it is expected that the current implementation can optimise two consecutive GAs at bodies with large SOIs if the initial conditions are specified sufficiently close to the actual optimum. It is however unlikely that InTrance can optimise trajectories in which many consecutive GAs at bodies with small SOIs are performed due to the vast increase in problem dimensions with each additional phase, and limited accuracy of the EA. However, if the solutions of the previous section are implemented and result in a sharp decrease of the problem dimensions and increase in accuracy, it might eventually be within the capabilities of InTrance.

A

Reference Frames

The two main reference frames used within this work are the *inertial Cartesian reference frame* and the *inertial polar ecliptic reference frame*. Both frames are inertial reference frames, in which bodies, whose net force acting upon them is zero, are not accelerated. Hence, they are either at rest or move with constant velocity in a straight line. An inertial reference frame describes time and space homogeneously, isotropically and in a time-independent manner [44].

The inertial Cartesian reference frame has extensively been used throughout this work to showcase trajectories, either in the heliocentric frame (Ecliptic J2000) or in the bodycentric frame (=Ecliptic J2000 heliocentric coordinates of S/C - Ecliptic J2000 heliocentric coordinates of GA-body). The inertial polar ecliptic reference frame is used within InTrance to supply the ANN with a double exhaustive set of S/C coordinates.

The inertial Cartesian and inertial polar ecliptic reference frames are described in Section A.1 and A.2, respectively. Orbital elements are used in the derivation of the analytical GA-model and other GA related parameters, and are lastly described in Section A.3. The reference frames used in the defining of initial conditions after a GA have already been described in Section 5.4 and hence will not be repeated here.

A.1. Inertial Cartesian Reference Frame

The inertial Cartesian reference frame $\mathcal{J}$: $(\mathbf{e_x}, \mathbf{e_y}, \mathbf{e_z})$, shown in Figures A.1 and A.2, is a right-handed frame defined through its orthogonal unit vectors $\mathbf{e_x}$, $\mathbf{e_y}$ and $\mathbf{e_z}$. A vector $\mathbf{r}$ can then be defined by its projections along these vectors. The inertial heliocentric frame (also termed the Ecliptic J2000 frame) is an inertial Cartesian reference frame, centered at the Sun, in which the XY-plane is defined by the Earth's mean orbital plane, and the principal axis by the vernal equinox direction of epoch J2000 [56].

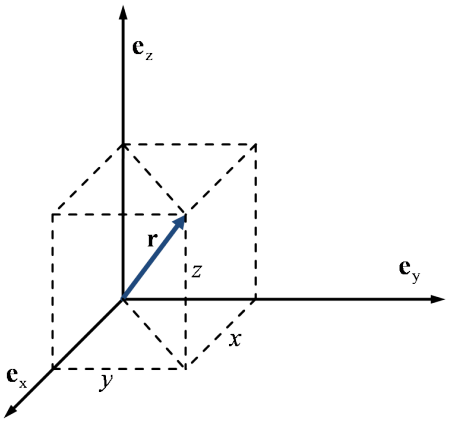


Figure A.1: The Inertial Cartesian reference frame $\mathcal{J}$. Courtesy of [59].

The spacecraft's position $\mathbf{r}$, velocity $\dot{\mathbf{r}} = \mathbf{V}$ and acceleration $\ddot{\mathbf{r}} = \dot{\mathbf{V}} = \mathbf{a}$ in the $\mathcal{J}$ frame are then described by:

$$\mathbf{r} = x\mathbf{e_x} + y\mathbf{e_y} + z\mathbf{e_z} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{A.1}$$

$$\dot{\mathbf{r}} = \dot{x}\mathbf{e_x} + \dot{y}\mathbf{e_y} + \dot{z}\mathbf{e_z} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \tag{A.2}$$

$$\ddot{\mathbf{r}} = \ddot{x}\mathbf{e_x} + \ddot{y}\mathbf{e_y} + \ddot{z}\mathbf{e_z} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix}. \tag{A.3}$$

A.2. Inertial Polar Ecliptic Reference Frame

Translational motion of a S/C is better described in a polar reference frame due to the nature of the problem. The frame is again centered at the Sun, and its principal axis points towards the vernal equinox at epoch J2000. The frame is shown in Figure A.2.

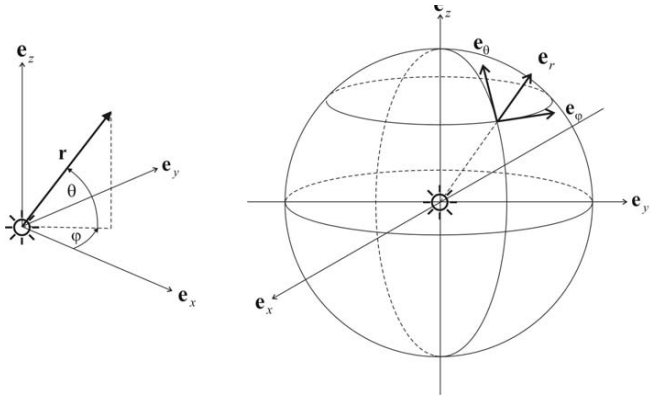


Figure A.2: The Inertial Polar Ecliptic Reference Frame $\mathcal{P}$. Courtesy of [17].

The polar reference frame $\mathcal{P} : \left(\mathbf{e_r}, \mathbf{e}_\varphi, \mathbf{e}_\vartheta\right)$ is an orthogonal right-handed polar frame, where $\mathbf{e_r}$ points along the sun-spacecraft line, where the vector $\mathbf{e_r}$ points towards the S/C, $\mathbf{e}_\vartheta$ lies in the $\mathbf{e_r}-\mathbf{e_z}$-plane and points along the direction of increasing ϑ, and $\mathbf{e}_\varphi$ completes the right-handed coordinate system. The azimuth angle (φ) is the angle between $\mathbf{e_x}$ and the projection of $\mathbf{e_r}$ onto the ecliptic. The elevation angle ϑ is the angle between the ecliptic plane and $\mathbf{r}$. The unit vectors, expressed in $\mathcal{J}$ components, are described by

$$\mathbf{e_r} = \begin{bmatrix} \cos\varphi\cos\vartheta \\ \sin\varphi\cos\vartheta \\ \sin\vartheta \end{bmatrix}, \tag{A.4}$$

$$\mathbf{e}_\varphi = \begin{bmatrix} -\sin\varphi \\ \cos\varphi \\ 0 \end{bmatrix}, \tag{A.5}$$

$$\mathbf{e}_\vartheta = \begin{bmatrix} -\cos\varphi\sin\vartheta \\ -\sin\varphi\sin\vartheta \\ \cos\vartheta \end{bmatrix}. \tag{A.6}$$

The position vector $\mathbf{r}$ is defined as $\mathbf{r} = r\mathbf{e_r}$, where r is the shortest distance from the origin to that point. The

derivatives of the base vectors are given as [59]

$$\dot{\mathbf{e}}_{\mathbf{r}} = \frac{d\mathbf{e}_{\mathbf{r}}}{dt} = \dot{\varphi}\cos\vartheta\mathbf{e}_{\varphi} + \dot{\vartheta}\mathbf{e}_{\vartheta}, \tag{A.7}$$

$$\dot{\mathbf{e}}_{\varphi} = \frac{d\mathbf{e}_{\varphi}}{dt} = -\dot{\varphi}\left(\cos\vartheta\mathbf{e}_{\mathbf{r}} - \sin\vartheta\mathbf{e}_{\vartheta}\right), \tag{A.8}$$

$$\dot{\mathbf{e}}_{\vartheta} = \frac{d\mathbf{e}_{\vartheta}}{dt} = -\dot{\varphi}\sin\vartheta\mathbf{e}_{\varphi} - \dot{\vartheta}\mathbf{e}_{\mathbf{r}}, \tag{A.9}$$

from which the velocity is found as

$$\mathbf{V} = \dot{\mathbf{r}} = \frac{d\left(r\mathbf{e}_{\mathbf{r}}\right)}{dt} = \dot{r}\mathbf{e}_{\mathbf{r}} + r\dot{\mathbf{e}}_{\mathbf{r}} \tag{A.10}$$

$$= \dot{r}\mathbf{e}_{\mathbf{r}} + r\dot{\varphi}\cos\vartheta\mathbf{e}_{\vartheta} + r\dot{\vartheta}\mathbf{e}_{\vartheta}, \tag{A.11}$$

and the acceleration as

$$\mathbf{a} = \dot{\mathbf{V}} = \ddot{\mathbf{r}} = \frac{d}{dt}\left(\dot{r}\mathbf{e}_{\mathbf{r}} + r\dot{\varphi}\cos\vartheta\mathbf{e}_{\vartheta} + r\dot{\vartheta}\mathbf{e}_{\vartheta}\right) \tag{A.12}$$

$$= \left(\ddot{r} - r\dot{\vartheta}^2 - r\dot{\varphi}^2\cos^2\vartheta\right)\mathbf{e}_{\mathbf{r}} \tag{A.13}$$

$$+ \left(2\dot{r}\dot{\varphi}\cos\vartheta + r\ddot{\varphi}\cos\vartheta - 2r\dot{\varphi}\dot{\vartheta}\sin\theta\right)\mathbf{e}_{\varphi} \tag{A.14}$$

$$+ \left(2\dot{r}\dot{\vartheta} + r\ddot{\vartheta} + r\dot{\varphi}^2\cos\vartheta\sin\vartheta\right)\mathbf{e}_{\varphi}. \tag{A.15}$$

A.3. Orbital Elements

The six classical orbital elements are the semi-major axis a, the eccentricity e, the inclination i, the longitude of ascending node or right ascension of ascending node (dependent on whether defined w.r.t. the ecliptic or equator) Ω, the argument of periapsis ω and the time of (last) pericenter passage τ. The elements a, e, and ω are integration constants originating from integrating the differential equations for the motion of a body (point mass) in the orbital plane, with respect to a non-rotating reference frame that is fixed to the center of the other (attracting) body. The time of (last) pericenter passage τ is another integration constant, used to link time and position in the orbit. The above four mentioned elements completely describe the orbit within the orbital plane, the last two elements (Ω and i) describe the orientation of the orbital plane relative to the reference frame. The equations to describe the conic sections originate from Wakker [76], unless stated otherwise.

The geometry of the problem is shown in Figures A.3 to A.5. The first two figures shown the in-plane geometry for, respectively, an elliptic and hyperbolic orbit. Figure A.5 shows the 3D geometry, relating the orientation of the orbital plane to the reference frame, in which the ecliptic is usually taken as the principal plane.

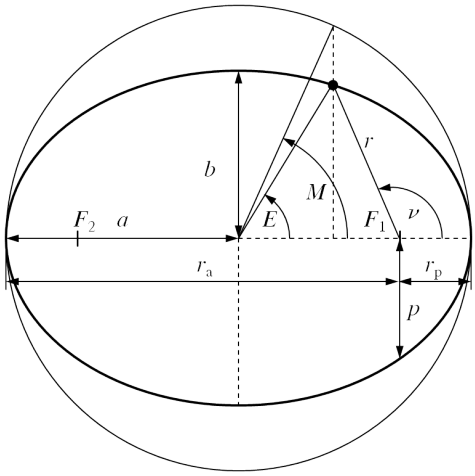


Figure A.3: Geometry of an elliptical orbit.

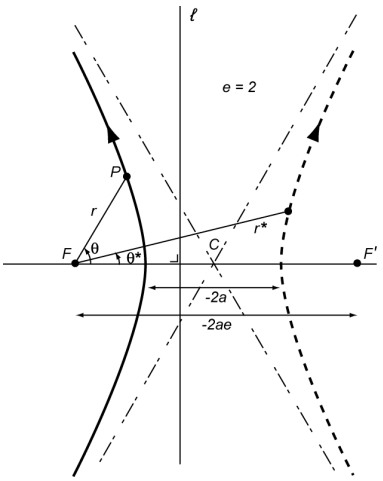


Figure A.4: Geometry of a hyperbolic orbit.

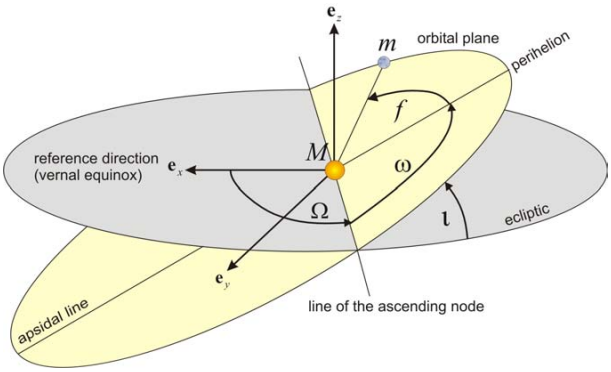


Figure A.5: Geometry of the orbital elements w.r.t. the ecliptic.

Semi-major axis *a*

The semi-major axis, together with the eccentricity, determines the size and shape of the conic section. The distance between the two extreme points of an ellipse is termed the major axis. Half of its value, hence from the center of the ellipse to one extreme, is called the semi-major axis, as depicted in Figure A.3. Analogous for hyperbolic orbits, the semi-major axis is the distance between the extreme point of the hyperbola and the extreme point of a mirrored hyperbola over the intersection of its asymptotes, see Figure A.4. The semi-major axis of a parabola is undefined. In the case of a closed orbit, the semi-major axis can be determined from the periapsis and apoapsis radii r_p and r_a as:

$$a = \frac{r_a + r_p}{2} = \left(\frac{2}{r} - \frac{V^2}{\mu}\right)^{-1}. \tag{A.16}$$

The semi-major axis, for both hyperbolic and elliptical orbits, can also be determined from the total energy per unit mass ($\mathscr{E}$) and gravitational parameter (μ) of the main body as:

$$a = -\frac{\mu}{2\mathscr{E}}, \tag{A.17}$$

in which $\mathscr{E}$ is computed as:

$$\mathscr{E} = \frac{V^2}{2} - \frac{\mu}{r}. \tag{A.18}$$

Note that the semi-major axis is negative by definition in the case of a hyperbolic section.

Eccentricity *e*

The eccentricity is the shape parameter; it is zero for circular orbits, between 0 and 1 for elliptical orbits, equal to 1 for parabolic orbits, and larger than 1 for hyperbolic orbits. For closed orbits, the eccentricity can be determined from the periapsis and apoapsis as

$$e = \frac{r_a + r_p}{r_a - r_p}. \tag{A.19}$$

Alternatively, for all types of conic sections, the eccentricity can be determined from the total energy per unit mass and the specific angular momentum as:

$$e = \sqrt{1 + \frac{2\mathscr{E}h^2}{\mu^2}}. \tag{A.20}$$

Lastly, the eccentricity can also be determined from the magnitude of the eccentricity vector ($e = |\mathbf{e}|$), which points to the orbit's pericenter, and is defined as [59]

$$\mathbf{e} = \frac{\left(V^2 - \frac{\mu}{r}\right)\mathbf{r} - (\mathbf{r}\cdot\mathbf{V})\,\mathbf{V}}{\mu}. \tag{A.21}$$

Argument of Periapsis ω

The argument of periapsis denotes the angle of a line passing through the periapsis, measured from the ascending node, and is shown in Figure A.5. The argument of periapsis can be computed from a vector **n** pointing towards the ascending node and the eccentricity vector as:

$$\tilde{\omega} = \arccos\left(\frac{\mathbf{n}\cdot\mathbf{e}}{|\mathbf{n}|\,|\mathbf{e}|}\right). \tag{A.22}$$

The domain of the argument of periapsis is $(0,2\pi)$, hence, the following relations should be taken into account:

$$\omega = \begin{cases} \tilde{\omega} & (\mathbf{e}\cdot\mathbf{e_z} \geq 0) \\ 2\pi - \tilde{\omega} & (\mathbf{e}\cdot\mathbf{e_z} < 0) \end{cases}. \tag{A.23}$$

Time of (last) Pericenter Passage τ

The time of last pericenter passage is used to link time and position, and can be computed –for elliptical orbits– as:

$$t-\tau = \sqrt{\frac{a^3}{\mu}}\left[2\arctan\left(\sqrt{\frac{1-e}{1+e}}\tan\frac{\theta}{2}\right) - e\sqrt{1-e^2}\,\frac{\sin\theta}{1+e\cos\theta}\right], \tag{A.24}$$

where θ, also referred to as f, is the true anomaly; the angle between the ascending node vector **n** and current position vector r measured over the orbital plane (see Figure A.5), and is determined as:

$$\theta = \arccos\left(\frac{\mathbf{n}\cdot\mathbf{r}}{|\mathbf{n}|\,|\mathbf{r}|}\right). \tag{A.25}$$

Due to the complicated form of equation A.24, the time of last pericenter passage if often substituted by the mean anomaly (M). The mean anomaly is the angle between the fictitious orbit point at which an object would be if it moved with the mean motion

$$n = \frac{2\pi}{T} = \sqrt{\frac{\mu}{a^3}} \quad \text{(Elliptical orbits)}, \tag{A.26}$$

$$\bar{n} = \sqrt{\frac{\mu}{-a^3}} \quad \text{(Hyperbolic orbits)}, \tag{A.27}$$

where T is the orbital period; the time needed to complete a full revolution. The mean anomaly (see Figure A.3) is then defined as:

$$M = n\,(t-\tau) \quad \text{(Elliptical orbits)}, \tag{A.28}$$

$$\bar{M} = \bar{n}\,(t-\tau) \quad \text{(Hyperbolic orbits)}, \tag{A.29}$$

where t is the current time and τ the time of (last) pericenter passage.

From the mean anomaly, the true anomaly can be determined. Although the true anomaly does not link time and position, it does fully determine the location of the S/C in its orbit, and hence is often used when time is not of the essence. In order to determine the true anomaly, the eccentric anomaly (E) or the hyperbolic anomaly (F) –depending on the type of orbit– are required, see Figure A.3. The eccentric and hyperbolic anomalies can be computed iteratively from M as:

$$M = E - e\sin E \quad \text{(Elliptical orbits)}, \tag{A.30}$$

$$\bar{M} = \sinh F - F \quad \text{(Hyperbolic orbits)}. \tag{A.31}$$

The true anomaly is then given by:

$$\theta = \begin{cases} \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2} & (e<1) \\ \sqrt{\frac{e+1}{e-1}}\tanh\frac{F}{2} & (e>1) \end{cases}. \tag{A.32}$$

Inclination *i*

The inclination (see Figure A.5) is the angle between the specific angular momentum vector $\mathbf{h} = \mathbf{r} \times \mathbf{V}$ and the reference frame's third vector $\mathbf{e_z}$, such that

$$i = \arccos\left(\frac{\mathbf{h} \cdot \mathbf{e_z}}{h}\right). \tag{A.33}$$

Hence, the inclination defines the angle between the orbital plane and reference frame, rotated over the x-axis. The inclination is constrained between 0 and π. Orbits with an inclination smaller than $\pi/2$ are called prograde, orbits with $i > \pi/2$ are called retrogade orbits.

Right Ascension of the Ascending Node Ω

The right ascension, or longitude, of the ascending node is the angle between the unit vector $\mathbf{e_z}$ and the location where the orbit crosses the principal frame from southward direction. Due to its definition, Ω is only defined for inclined orbits. The right ascension of the ascending node can be determined using the angular momentum vector and node vector:

$$\tilde{\Omega} = \arccos\left(\frac{\mathbf{e_x} \cdot \mathbf{n}}{|\mathbf{n}|}\right), \tag{A.34}$$

then,

$$\Omega = \begin{cases} \tilde{\Omega} & \left(\mathbf{n} \cdot \mathbf{e_y} \geq 0\right) \\ 2\pi - \tilde{\Omega} & \left(\mathbf{n} \cdot \mathbf{e_y} < 0\right) \end{cases}. \tag{A.35}$$

B

Results Dawn Simulations

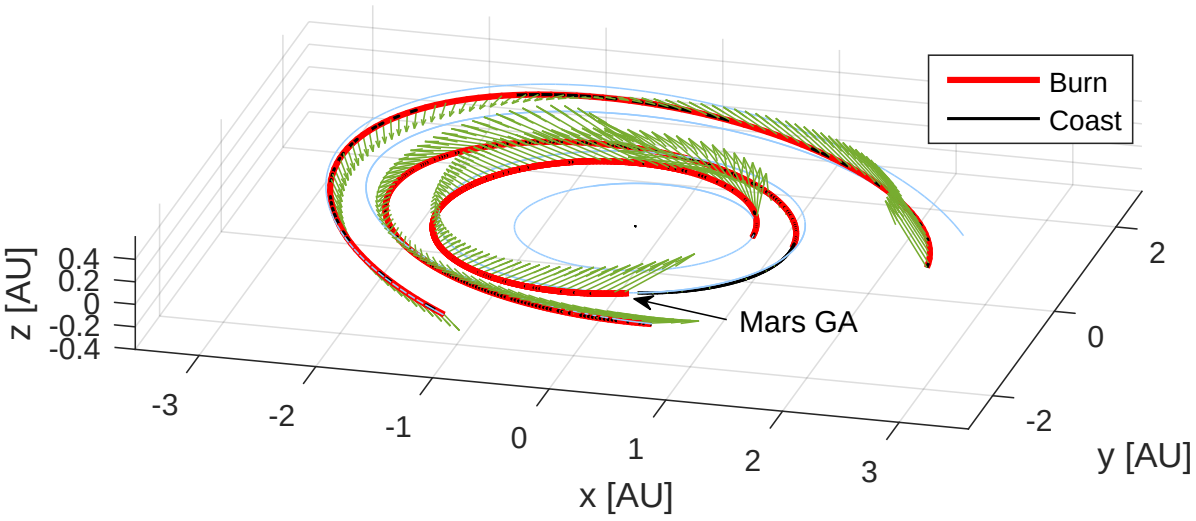


Figure B.1: Three-dimensional view of the Dawn trajectory as computed by InTrance in the heliocentric frame. Green arrows indicate thrust vectors.

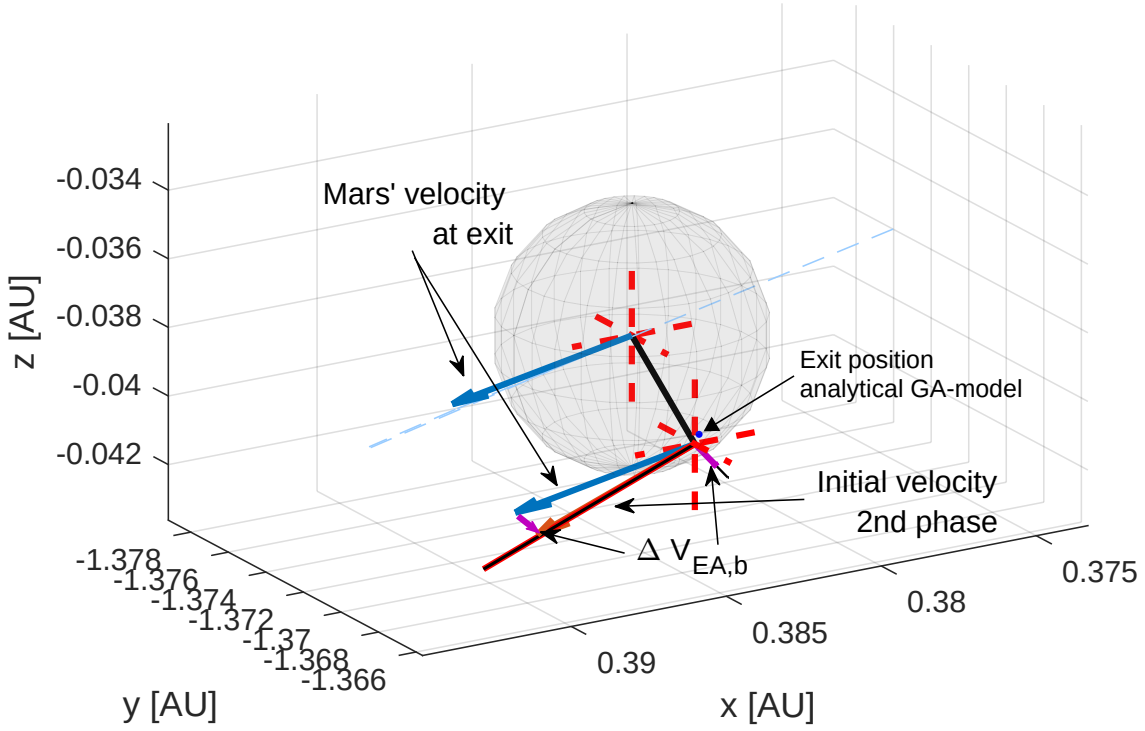


Figure B.2: Three-dimensional view of the initial state of the second phase in the heliocentric frame of the Dawn trajectory computed with InTrance.

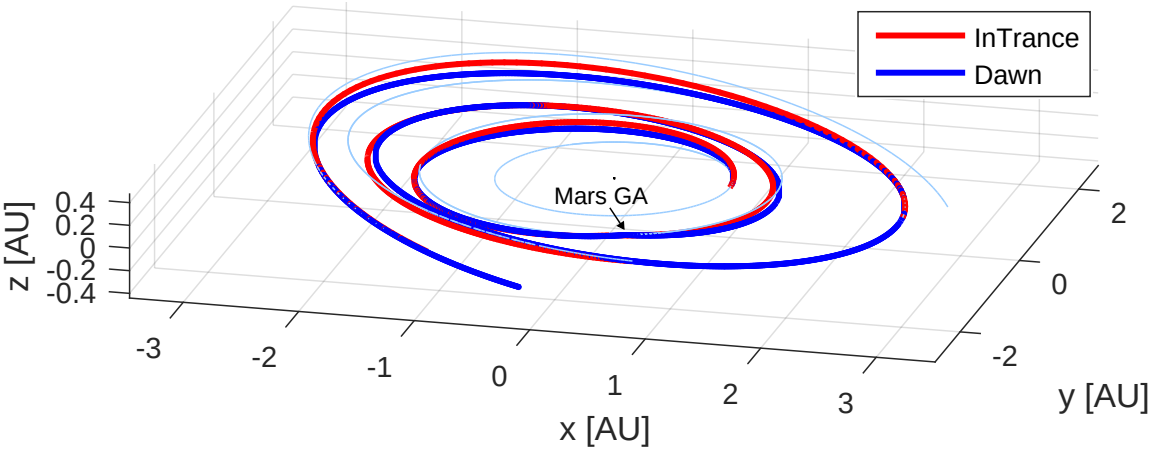


Figure B.3: Three-dimensional view of the heliocentric Dawn trajectory as computed by InTrance (red) compared to Dawn's actual trajectory generated through SPICE (blue).

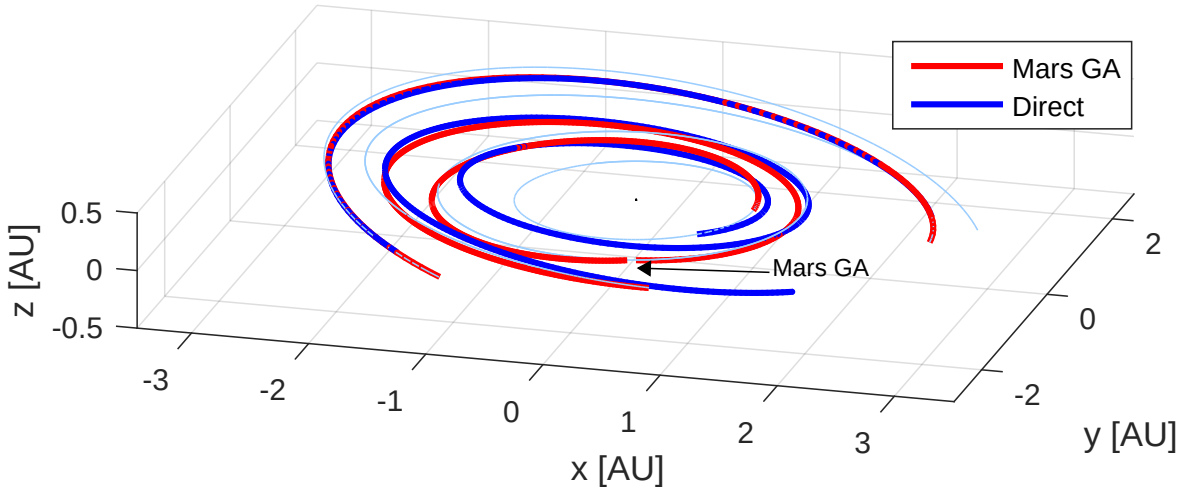


Figure B.4: Three-dimensional view of the heliocentric Dawn trajectory as computed by InTrance with a Mars GA (red) and without a GA (blue).

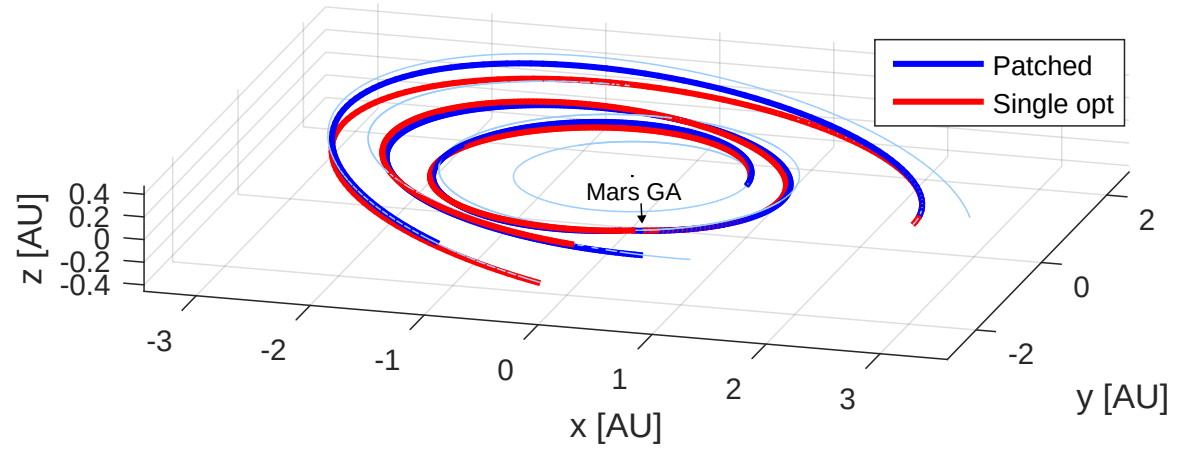


Figure B.5: Three-dimensional view of the heliocentric Dawn trajectory as computed by InTrance, both the concatenated results of the two separate simulations (blue) and the single optimisation of the complete missions (red).

C

InTrance Input Files

This appendix contains the InTrance input files used in generating the main results of Chapter 7. The input files for the low-thrust New Horizons adaptation with a Jupiter GA are given in Section C.1, followed by the input files for the Dawn re-calculation with a Mars GA for the optimisation from Earth to Vesta in Section C.2, and for the optimisation from Vesta to Ceres in Section C.3. All simulations are started by opening a command window in the directory where the executable of InTrance is located and then entering the command `intrance.exe NAME.inp`, where `NAME` should be substituted by the filename of the .inp file.

Each simulation is defined by a set of three input files per phase plus two overarching ones. The phase specific input files are; (1) the NC configuration files, the (2) spacecraft configuration file, and (3) the simulation configuration file. The first is used to define the size and activation function of the ANN, the second defines the throttle and S/C characteristics, and the third configuration file describes the mission defining simulation parameters such as the launch/arrival dates and target body and type, but also the integrator settings and internal representation of the EoM of the respective phase. The overarching input files contain parameters which govern all phases and are (1) the configuration parameters input file and (2) the EA parameters configuration file. The former details whether a simulation is a cold- or warmstart (see Section 4.4.1) and the directory of the other input files. The latter describes the EA parameters such as the population and hypercube size during both the SSS and the following optimisation run, the convergence criteria and mutation rate.

C.1. Low-Thrust New Horizons with Jupiter Gravity Assist

C.1.1. General Input Files

Configuration Parameters – coldstart.inp

```
COMMAND                         = optimize
COLDSTART                       = yes
NO_OF_EVAL_OBJECTS              = 4
SIM_PARAM_FILE_1                = p1.sim
SIM_PARAM_FILE_2                = p2.sim
EA_PARAM_FILE                   = coldstart.eap
SIM_DATA_FILE                   = coldstart.csv
TRAJ_DATA_FILE                  = coldstart.dat
GESOP_FILE                      = coldstart.gesop.txt
VRML_FILE                       = coldstart.wrl
CTRL_FILE                       = coldstart.ctr
BEST_CHROM_FILE                 = coldstart.eac
REPORT_FILE                     = coldstart.rep
```

Evolutionary Algorithm Parameters – coldstart.eap

```
SEARCH_SPACE_HYPERCUBE_SIZE     = 1.0
HYPERCUBE_START_SIZE            = 2.0E-1
HYPERCUBE_SHRINKING_FACTOR      = 90E-2
POPULATION_SIZE                 = 30
POPULATION_SIZE_SSS             = 50
SEARCH_SCAN_EPOCHS              = 30
FITNESS_FUNCTION_TYPE           = J_AND
CHROMOSOME_MUTATION_PROBABILITY = 0.9
GENOM_MUTATION_PROBABILITY      = 0.05
HYPERCUBE_UPPER_LIMIT           = 1.0E-4
IL_POP_CONV_FBC_MET             = 1.0E-6
IL_POP_CONV_FBC_NOT_MET         = 1.0E-5
IL_EA_CONV_FBC_MET              = 1.0E-5
IL_EA_CONV_FBC_NOT_MET          = 1.0E-5
```

C.1.2. Phase 1 Input Files

Neurocontroller Parameters – p1.ncp

```
NC_OUTPUT                       = direct
TRANSFER_FUNCTION               = sigmoid
HIDDEN_LAYERS                   = 1
NEURONS_IN_HIDDEN_LAYER1        = 35
```

Spacecraft Parameters – p1.scp

```
SC_TYPE                         = NEP
SC_NAME                         = "New Horizons"
PAYLOAD_MASS                    = 565kg
MIN_PROP_MASS                   = 10kg
MAX_PROP_MASS                   = 22kg
THROTTLE_TYPE           = bang-bang
MAX_THRUST              = 40E-3
SPECIFIC_IMPULSE        = 3000s
NEP_DECAY_CONSTANT      = -2.196450873E-5
```

Simulation Parameters – p1.sim

```
INTEGRATION_INTERVAL            = 500day
FLIGHT_TIME_MIN                 = 250day
INTEGRATION_STEPS               = 1500
MIN_OUTPUT_POINTS               = 100
DYN_INTEGRATION_INTERVAL        = yes
MODIFY_INIT_PARAMETERS          = yes
MODIFY_LAUNCH_DATE              = no
MODIFY_INIT_PROP_MASS           = no
MODIFY_INIT_VINF                = yes
USE_DSSC                        = yes
DISTURBING_BODIES               = disturbance.sim
DSSC_STEP_ANGLE_CONTROL         = yes
DSSC_MAX_STEP_ANGLE             = 1.0DEG
DSSC_MAX_STEP_SIZE              = 10DAY
DSSC_MIN_STEP_SIZE              = 0.5min
SIM_START_TIME_MIN              = 53740
SIM_START_TIME_MAX              = 53760
ARRIVAL_DATE_MIN                = 54120
ARRIVAL_DATE_MAX                = 54140
INITIAL_STATE                   = body
INITIAL_BODY_NAME               = EARTH
INITIAL_CENTRAL_BODY            = sun
INITIAL_VINF_MIN                = 5km/s
INITIAL_VINF_AZIMUTH_MIN        = -10deg
INITIAL_VINF_AZIMUTH_MAX        = 10deg
INITIAL_VINF_ELEV_MIN           = 0deg
INITIAL_VINF_ELEV_MAX           = 10deg
LAUNCHER_MAX_C3_CAPACITY        = 7200
LAUNCHER_C3_EXPONENT            = 0.01725
ANALYTIC_GRAVITY_ASSIST         = true
TARGET_BODY_NAME                = Jupiter
MIN_SOLAR_DISTANCE              = 0.2AU
INTEGRATOR                      = RK54F
MAX_RELATIVE_ERROR              = 1.0E-6
MAX_ABSOLUTE_ERROR              = 1.0E-6
USE_ITGR_STOPPER                = no
SC_CONF                         = p1.scp
NAV_TYPE                        = ANN
NAV_ANN_CONF                    = p1.ncp
TARGET_STATE                    = renewed gravity assist
OPTIMIZATION_GOAL               = minimum transfer time
STEERING_DYN_UNIT_CALC          = yes
STEERING_USE_RANGE              = yes
STEERING_USE_RANGE_RATE         = yes
STEERING_USE_ACC_THRUST_MAX     = yes
STEERING_USE_ACC_THRUST_MAX_DRY = yes
STEERING_USE_STEP_SIZE          = yes
STEERING_USE_TIME_UNTIL_PERI_SC = yes
STEERING_USE_TIME_UNTIL_PERI_TGT = yes
STEERING_USE_ABS_CART_POS_X     = yes
STEERING_USE_ABS_CART_POS_Y     = yes
STEERING_USE_ABS_CART_POS_Z     = yes
STEERING_USE_ABS_CART_VEL_X     = yes
STEERING_USE_ABS_CART_VEL_Y     = yes
STEERING_USE_ABS_CART_VEL_Z     = yes
STEERING_USE_ABS_POLAR_POS_R    = yes
STEERING_USE_ABS_POLAR_POS_AZI  = yes
STEERING_USE_ABS_POLAR_POS_ELE  = yes
STEERING_USE_ABS_POLAR_VEL_R    = yes
STEERING_USE_ABS_POLAR_VEL_AZI  = yes
STEERING_USE_ABS_POLAR_VEL_ELE  = yes
STEERING_USE_TGT_CART_POS_X     = no
STEERING_USE_TGT_CART_POS_Y     = no
STEERING_USE_TGT_CART_POS_Z     = no
STEERING_USE_TGT_CART_VEL_X     = no
STEERING_USE_TGT_CART_VEL_Y     = no
STEERING_USE_TGT_CART_VEL_Z     = no
STEERING_USE_TGT_CART_POS_REL_X = no
STEERING_USE_TGT_CART_POS_REL_Y = no
STEERING_USE_TGT_CART_POS_REL_Z = no
STEERING_USE_TGT_CART_VEL_REL_X = no
STEERING_USE_TGT_CART_VEL_REL_Y = no
STEERING_USE_TGT_CART_VEL_REL_Z = no
```

C.1.3. Phase 2 Input Files

Neurocontroller Parameters – p2.ncp

```
NC_OUTPUT                       = direct
TRANSFER_FUNCTION               = sigmoid
HIDDEN_LAYERS                   = 1
NEURONS_IN_HIDDEN_LAYER1        = 35
```

Spacecraft Parameters – p2.scp

```
SC_TYPE                         = NEP
SC_NAME                         = "New Horizons"
PAYLOAD_MASS                    = 565kg
MIN_PROP_MASS                   = 5kg
MAX_PROP_MASS                   = 13kg
THROTTLE_TYPE           = bang-bang
MAX_THRUST              = 40E-3
SPECIFIC_IMPULSE        = 3000s
NEP_DECAY_CONSTANT      = -2.196450873E-5
```

Simulation Parameters – p2.sim

```
INDEPENDENT_FLIGHT_PHASE         = no
TRANSITION_THRESHOLD_MASS_PROP   = 0.1
SIM_START_TIME_MIN               = 54178
SIM_START_TIME_MAX               = 54198
MET_MAX                          = 10JYR
INTEGRATION_INTERVAL             = 3000day
FLIGHT_TIME_MIN                  = 2000
INTEGRATION_STEPS                = 1000
MIN_OUTPUT_POINTS                = 200
DYN_INTEGRATION_INTERVAL         = yes
MODIFY_INIT_PARAMETERS           = no
MODIFY_LAUNCH_DATE               = no
MODIFY_INIT_PROP_MASS            = no
DISTURBING_BODIES                = disturbance.sim
USE_DSSC                         = yes
INITIAL_VINF_MIN                 = 4
INITIAL_VINF_MAX                 = 6.5
INITIAL_VINF_AZIMUTH_MIN         = 100deg
INITIAL_VINF_AZIMUTH_MAX         = 120deg
INITIAL_VINF_ELEV_MIN            = 0deg
INITIAL_VINF_ELEV_MAX            = 15.0deg
INITIAL_POS_AZIMUTH_MIN_GA       = 180deg
INITIAL_POS_AZIMUTH_MAX_GA       = 220deg
INITIAL_POS_ELEV_MIN_GA          = -8deg
INITIAL_POS_ELEV_MAX_GA          = 8.0deg
MODIFY_INIT_LAUNCH_POS_GA        = no
TRANSITION_THRESHOLD_STATE       = 0.01
DSSC_STEP_DISTANCE_CONTROL       = YES
DSSC_MAX_STEP_DISTANCE           = 0.1AU
DSSC_STEP_ANGLE_CONTROL          = YES
DSSC_MAX_STEP_ANGLE              = 1.0deg
DSSC_MAX_STEP_SIZE               = 10DAY
DSSC_MIN_STEP_SIZE               = 10min
INITIAL_STATE                    = body
INITIAL_BODY_NAME                = Jupiter
ARRIVAL_DATE_MIN                 = 56800
ARRIVAL_DATE_MAX                 = 56960
TARGET_STATE                     = body flyby
TARGET_BODY_NAME                 = Pluto
TGT_PROX_STATE_THRESHOLD         = 0.1
TGT_PROX_STATE_THRESHOLD_FINAL   = 0.05
TARGET_DIST_MAX_FINAL            = 1E7km
TARGET_DIST_MAX_INIT             = 1E7km
TARGET_DIST_MAX_SHRINK           = 0.8
TARGET_DIST_MAX_DECREASE         = 1.0E5km
TARGET_DIST_MAX_REDUCTION_USE_MAX = no
OPTIMIZATION_GOAL                = minimum transfer time
ACCURACY_FITNESS_FRACTION        = 0.01
MIN_SOLAR_DISTANCE               = 0.2AU
INTEGRATOR                       = RK54F
MAX_RELATIVE_ERROR               = 1.0E-6
MAX_ABSOLUTE_ERROR               = 1.0E-6
USE_ITGR_STOPPER                 = yes
SC_CONF                          = p2.scp
NAV_TYPE                         = ANN
NAV_ANN_CONF                     = p2.ncp
STEERING_DYN_UNIT_CALC           = yes
STEERING_USE_RANGE               = false
STEERING_USE_RANGE_RATE          = false
STEERING_USE_ACC_THRUST_MAX      = yes
STEERING_USE_ACC_THRUST_MAX_DRY  = yes
STEERING_USE_STEP_SIZE           = yes
STEERING_USE_TIME_UNTIL_PERI_SC  = yes
STEERING_USE_TIME_UNTIL_PERI_TGT = no
STEERING_USE_ABS_CART_POS_X      = yes
STEERING_USE_ABS_CART_POS_Y      = yes
STEERING_USE_ABS_CART_POS_Z      = yes
STEERING_USE_ABS_CART_VEL_X      = yes
STEERING_USE_ABS_CART_VEL_Y      = yes
STEERING_USE_ABS_CART_VEL_Z      = yes
STEERING_USE_ABS_POLAR_POS_R     = yes
STEERING_USE_ABS_POLAR_POS_AZI   = yes
STEERING_USE_ABS_POLAR_POS_ELE   = yes
STEERING_USE_ABS_POLAR_VEL_R     = yes
STEERING_USE_ABS_POLAR_VEL_AZI   = yes
STEERING_USE_ABS_POLAR_VEL_ELE   = yes
```

C.2. Dawn with Mars Gravity Assist — Earth to Vesta

C.2.1. General Input Files

Configuration Parameters – coldstart.inp

```
COMMAND                          = optimize
COLDSTART                        = yes
NO_OF_EVAL_OBJECTS               = 4
SIM_PARAM_FILE_1                 = Mars.sim
SIM_PARAM_FILE_2                 = Vesta.sim
EA_PARAM_FILE                    = coldstart.eap
SIM_DATA_FILE                    = coldstart.csv
TRAJ_DATA_FILE                   = coldstart.dat
GESOP_FILE                       = coldstart.gesop.txt
VRML_FILE                        = coldstart.wrl
CTRL_FILE                        = coldstart.ctr
BEST_CHROM_FILE                  = coldstart.eac
REPORT_FILE                      = coldstart.rep
ASTEROID_DATA_FILE1              = c:\ELEMENTS.NUMBR
```

Evolutionary Algorithm Parameters – coldstart.eap

```
SEARCH_SPACE_HYPERCUBE_SIZE      = 1.0
HYPERCUBE_START_SIZE             = 2.0E-1
HYPERCUBE_SHRINKING_FACTOR       = 90E-2
POPULATION_SIZE                  = 50
POPULATION_SIZE_SSS              = 50
SEARCH_SCAN_EPOCHS               = 30
FITNESS_FUNCTION_TYPE            = J_AND
CHROMOSOME_MUTATION_PROBABILITY  = 0.9
GENOM_MUTATION_PROBABILITY       = 0.05
HYPERCUBE_UPPER_LIMIT            = 1.0E-4
IL_POP_CONV_FBC_MET              = 1.0E-6
IL_POP_CONV_FBC_NOT_MET          = 1.0E-5
IL_EA_CONV_FBC_MET               = 1.0E-5
IL_EA_CONV_FBC_NOT_MET           = 1.0E-5
```

C.2.2. Phase 1 Input Files

Neurocontroller Parameters – Mars.ncp

```
NC_OUTPUT                       = direct
TRANSFER_FUNCTION               = sigmoid
HIDDEN_LAYERS                   = 1
NEURONS_IN_HIDDEN_LAYER1        = 40
```

Spacecraft Parameters – Mars.scp

```
SC_TYPE                         = NSTAR
SC_NAME                         = "Dawn"
PAYLOAD_MASS                    = 817kg
MIN_PROP_MASS                   = 100kg
MAX_PROP_MASS                   = 160kg
SOLAR_ARRAY_CHAR_POWER          = 9.8kW
POWER_VARIATION_EXPONENT        = 1.7
N_THRUSTERS                     = 1
THROTTLE_TYPE                   = variable
```

Simulation Parameters – Mars.sim

```
INTEGRATION_INTERVAL            = 800day
FLIGHT_TIME_MIN                 = 400day
INTEGRATION_STEPS               = 400
MIN_OUTPUT_POINTS               = 50
DYN_INTEGRATION_INTERVAL        = yes
MODIFY_INIT_PARAMETERS          = yes
MODIFY_LAUNCH_DATE              = no
MODIFY_INIT_PROP_MASS           = no
MODIFY_INIT_VINF                = yes
USE_DSSC                        = yes
DISTURBING_BODIES               = disturbance.sim
DSSC_STEP_ANGLE_CONTROL         = yes
DSSC_MAX_STEP_ANGLE             = 6.0DEG
DSSC_MAX_STEP_SIZE              = 60DAY
DSSC_MIN_STEP_SIZE              = 1day
DSSC_STEP_DISTANCE_CONTROL      = NO
DSSC_MAX_STEP_DISTANCE          = 0.50AU
DSSC_APPROACH_CONTROL           = yes
DSSC_APPROACH_STEP_SIZE_FACTOR  = 0.0
SIM_START_TIME_MIN              = 54365
SIM_START_TIME_MAX              = 54375
ARRIVAL_DATE_MIN                = 54854
ARRIVAL_DATE_MAX                = 54874
INITIAL_STATE                   = body
INITIAL_BODY_NAME               = EARTH
INITIAL_VINF_MIN                = 3.362km/s
INITIAL_VINF_MAX                = 3.362km/s
INITIAL_VINF_AZIMUTH_MIN        = -20deg
INITIAL_VINF_AZIMUTH_MAX        = 20deg
INITIAL_VINF_ELEV_MIN           = -206deg
INITIAL_VINF_ELEV_MAX           = 40deg
ANALYTIC_GRAVITY_ASSIST         = true
TARGET_BODY_NAME                = Mars
MIN_SOLAR_DISTANCE              = 0.2AU
INTEGRATOR                      = RK54F
MAX_RELATIVE_ERROR              = 1.0E-6
MAX_ABSOLUTE_ERROR              = 1.0E-6
USE_ITGR_STOPPER                = yes
SC_CONF                         = Mars.scp
NAV_TYPE                        = ANN
NAV_ANN_CONF                    = Mars.ncp
TARGET_STATE                    = renewed gravity assist
OPTIMIZATION_GOAL               = minimum transfer time
STEERING_DYN_UNIT_CALC          = yes
STEERING_USE_RANGE              = yes
STEERING_USE_RANGE_RATE         = yes
STEERING_USE_ACC_THRUST_MAX     = no
STEERING_USE_ACC_THRUST_MAX_DRY = no
STEERING_USE_STEP_SIZE          = no
STEERING_USE_TIME_UNTIL_PERI_SC = no
STEERING_USE_TIME_UNTIL_PERI_TGT = no
STEERING_USE_ABS_CART_POS_X     = yes
STEERING_USE_ABS_CART_POS_Y     = yes
STEERING_USE_ABS_CART_POS_Z     = yes
STEERING_USE_ABS_CART_VEL_X     = yes
STEERING_USE_ABS_CART_VEL_Y     = yes
STEERING_USE_ABS_CART_VEL_Z     = yes
STEERING_USE_ABS_POLAR_POS_R    = no
STEERING_USE_ABS_POLAR_POS_AZI  = no
STEERING_USE_ABS_POLAR_POS_ELE  = no
STEERING_USE_ABS_POLAR_VEL_R    = no
STEERING_USE_ABS_POLAR_VEL_AZI  = no
STEERING_USE_ABS_POLAR_VEL_ELE  = no
STEERING_USE_TGT_CART_POS_X     = no
STEERING_USE_TGT_CART_POS_Y     = no
STEERING_USE_TGT_CART_POS_Z     = no
STEERING_USE_TGT_CART_VEL_X     = no
STEERING_USE_TGT_CART_VEL_Y     = no
STEERING_USE_TGT_CART_VEL_Z     = no
STEERING_USE_TGT_CART_POS_REL_X = yes
STEERING_USE_TGT_CART_POS_REL_Y = yes
STEERING_USE_TGT_CART_POS_REL_Z = yes
STEERING_USE_TGT_CART_VEL_REL_X = yes
STEERING_USE_TGT_CART_VEL_REL_Y = yes
STEERING_USE_TGT_CART_VEL_REL_Z = yes
```

C.2.3. Phase 2 Input Files

Neurocontroller Parameters – Vesta.ncp

```
NC_OUTPUT                       = direct
TRANSFER_FUNCTION               = sigmoid
HIDDEN_LAYERS                   = 1
NEURONS_IN_HIDDEN_LAYER1        = 35
```

Spacecraft Parameters – Vesta.scp

```
SC_TYPE                         = NSTAR
SC_NAME                         = "Dawn"
PAYLOAD_MASS                    = 817kg
MIN_PROP_MASS                   = 100kg
MAX_PROP_MASS                   = 160kg
SOLAR_ARRAY_CHAR_POWER          = 9.8kW
POWER_VARIATION_EXPONENT        = 1.7
N_THRUSTERS                     = 1
THROTTLE_TYPE                   = variable
```

Simulation Parameters – Vesta.sim

```
INDEPENDENT_FLIGHT_PHASE          = no
TRANSITION_THRESHOLD_MASS_PROP    = 0.1
INTEGRATION_INTERVAL              = 1100day
FLIGHT_TIME_MIN                   = 600day
MET_MAX                           = 4.5JYR
INTEGRATION_STEPS                 = 400
MIN_OUTPUT_POINTS                 = 50
DYN_INTEGRATION_INTERVAL          = yes
MODIFY_INIT_PARAMETERS            = yes
MODIFY_LAUNCH_DATE                = no
MODIFY_INIT_PROP_MASS             = no
MODIFY_INIT_VINF                  = yes
INITIAL_VINF_MIN                  = 2
INITIAL_VINF_MAX                  = 3
INITIAL_VINF_AZIMUTH_MIN          = -15deg
INITIAL_VINF_AZIMUTH_MAX          = -7.5deg
INITIAL_VINF_ELEV_MIN             = -60deg
INITIAL_VINF_ELEV_MAX             = -40deg
INITIAL_POS_AZIMUTH_MIN_GA        = -20deg
INITIAL_POS_AZIMUTH_MAX_GA        = 0deg
INITIAL_POS_ELEV_MIN_GA           = -70deg
INITIAL_POS_ELEV_MAX_GA           = -40deg
MODIFY_INIT_LAUNCH_POS_GA         = yes
TRANSITION_THRESHOLD_STATE        = 0.1
USE_DSSC                          = no
DSSC_STEP_ANGLE_CONTROL           = yes
DSSC_MAX_STEP_ANGLE               = 6DEG
DSSC_MAX_STEP_SIZE                = 60DAY
DSSC_MIN_STEP_SIZE                = 1DAY
DSSC_STEP_DISTANCE_CONTROL        = NO
DSSC_MAX_STEP_DISTANCE            = 0.50AU
DSSC_APPROACH_CONTROL             = yes
DSSC_APPROACH_STEP_SIZE_FACTOR    = 0.0
SIM_START_TIME_MIN                = 54859
SIM_START_TIME_MAX                = 54879
ARRIVAL_DATE_MIN                  = 55700
ARRIVAL_DATE_MAX                  = 55850
INITIAL_STATE                     = body
INITIAL_BODY_NAME                 = MARS
TARGET_STATE                      = body rendezvous
TARGET_BODY_NAME                  = Vesta
TARGET_DIST_MAX_FINAL             = 1.5E6km
TARGET_DIST_MAX_INIT              = 1.5E6km
TARGET_DIST_MAX_SHRINK            = 0.95
TARGET_DIST_MAX_DECREASE          = 1.0E4km
TARGET_DIST_MAX_REDUCTION_USE_MAX = no
TARGET_RELVEL_MAX_FINAL           = 1000m/s
TARGET_RELVEL_MAX_INIT            = 1000m/s
TARGET_RELVEL_MAX_SHRINK          = 0.95
TARGET_RELVEL_MAX_DECREASE        = 1m/s
TARGET_RELVEL_MAX_REDUCTION_USE_MAX = no
OPTIMIZATION_GOAL                 = minimum transfer time
ACCURACY_FITNESS_FRACTION         = 0.5
MIN_SOLAR_DISTANCE                = 0.2AU
INTEGRATOR                        = RK54F
MAX_RELATIVE_ERROR                = 1.0E-8
MAX_ABSOLUTE_ERROR                = 1.0E-8
USE_ITGR_STOPPER                  = yes
SC_CONF                           = Vesta.scp
NAV_TYPE                          = ANN
NAV_ANN_CONF                      = Vesta.ncp
STEERING_DYN_UNIT_CALC            = yes
STEERING_USE_RANGE                = yes
STEERING_USE_RANGE_RATE           = yes
STEERING_USE_ACC_THRUST_MAX       = no
STEERING_USE_ACC_THRUST_MAX_DRY   = no
STEERING_USE_STEP_SIZE            = no
STEERING_USE_TIME_UNTIL_PERI_SC   = no
STEERING_USE_TIME_UNTIL_PERI_TGT  = no
STEERING_USE_ABS_CART_POS_X       = yes
STEERING_USE_ABS_CART_POS_Y       = yes
STEERING_USE_ABS_CART_POS_Z       = yes
STEERING_USE_ABS_CART_VEL_X       = yes
STEERING_USE_ABS_CART_VEL_Y       = yes
STEERING_USE_ABS_CART_VEL_Z       = yes
STEERING_USE_ABS_POLAR_POS_R      = no
STEERING_USE_ABS_POLAR_POS_AZI    = no
STEERING_USE_ABS_POLAR_POS_ELE    = no
STEERING_USE_ABS_POLAR_VEL_R      = no
STEERING_USE_ABS_POLAR_VEL_AZI    = no
STEERING_USE_ABS_POLAR_VEL_ELE    = no
STEERING_USE_TGT_CART_POS_X       = no
STEERING_USE_TGT_CART_POS_Y       = no
STEERING_USE_TGT_CART_POS_Z       = no
STEERING_USE_TGT_CART_VEL_X       = no
STEERING_USE_TGT_CART_VEL_Y       = no
STEERING_USE_TGT_CART_VEL_Z       = no
STEERING_USE_TGT_CART_POS_REL_X   = yes
STEERING_USE_TGT_CART_POS_REL_Y   = yes
STEERING_USE_TGT_CART_POS_REL_Z   = yes
STEERING_USE_TGT_CART_VEL_REL_X   = yes
STEERING_USE_TGT_CART_VEL_REL_Y   = yes
STEERING_USE_TGT_CART_VEL_REL_Z   = yes
```

C.3. Dawn with Mars Gravity Assist — Vesta to Ceres

C.3.1. General Input Files

Configuration Parameters – coldstart.inp

```
COMMAND                           = optimize
COLDSTART                         = yes
NO_OF_EVAL_OBJECTS                = 4
SIM_PARAM_FILE_1                  = Ceres.sim
EA_PARAM_FILE                     = coldstart.eap
SIM_DATA_FILE                     = coldstart.csv
TRAJ_DATA_FILE                    = coldstart.dat
GESOP_FILE                        = coldstart.gesop.txt
VRML_FILE                         = coldstart.wrl
CTRL_FILE                         = coldstart.ctr
BEST_CHROM_FILE                   = coldstart.eac
REPORT_FILE                       = coldstart.rep
```

Evolutionary Algorithm Parameters – coldstart.eap

```
SEARCH_SPACE_HYPERCUBE_SIZE       = 1.0
HYPERCUBE_START_SIZE              = 1.0
HYPERCUBE_SHRINKING_FACTOR        = 90E-2
POPULATION_SIZE                   = 50
POPULATION_SIZE_SSS               = 50
SEARCH_SCAN_EPOCHS                = 20
FITNESS_FUNCTION_TYPE             = J_AND
CHROMOSOME_MUTATION_PROBABILITY   = 0.9
GENOM_MUTATION_PROBABILITY        = 0.05
HYPERCUBE_UPPER_LIMIT             = 1.0E-4
IL_POP_CONV_FBC_MET               = 1.0E-6
IL_POP_CONV_FBC_NOT_MET           = 1.0E-5
IL_EA_CONV_FBC_MET                = 1.0E-5
IL_EA_CONV_FBC_NOT_MET            = 1.0E-5
```

C.3.2. Phase 1 Input Files

Neurocontroller Parameters – Ceres.ncp

```
NC_OUTPUT                       = direct
TRANSFER_FUNCTION               = sigmoid
HIDDEN_LAYERS                   = 1
NEURONS_IN_HIDDEN_LAYER1        = 35
```

Spacecraft Parameters – Ceres.scp

```
SC_TYPE                         = NSTAR
SC_NAME                         = "Dawn"
PAYLOAD_MASS                    = 747kg
MIN_PROP_MASS                   = 50kg
MAX_PROP_MASS                   = 100kg
SOLAR_ARRAY_CHAR_POWER          = 9.8kW
POWER_VARIATION_EXPONENT        = 1.7
N_THRUSTERS                     = 1
THROTTLE_TYPE                   = variable
```

Simulation Parameters – Ceres.sim

```
INDEPENDENT_FLIGHT_PHASE        = yes
TRANSITION_THRESHOLD_MASS_PROP  = 0.1
SIM_START_TIME_MIN              = 55950
SIM_START_TIME_MAX              = 56000
MET_MAX                         = 3.5JYR
INTEGRATION_INTERVAL            = 1100day
FLIGHT_TIME_MIN                 = 800
INTEGRATION_STEPS               = 110
MIN_OUTPUT_POINTS               = 1
DYN_INTEGRATION_INTERVAL        = yes
MODIFY_INIT_PARAMETERS          = yes
MODIFY_LAUNCH_DATE              = no
MODIFY_INIT_PROP_MASS           = no
USE_DSSC                        = no
DSSC_STEP_ANGLE_CONTROL         = YES
DSSC_MAX_STEP_ANGLE             = 5DEG
DSSC_MAX_STEP_SIZE              = 10DAY
DSSC_MIN_STEP_SIZE              = 1DAY
DSSC_STEP_DISTANCE_CONTROL      = NO
DSSC_MAX_STEP_DISTANCE          = 0.50AU
DSSC_APPROACH_CONTROL           = yes
DSSC_APPROACH_STEP_SIZE_FACTOR  = 0.0
DWELL_TIME_MIN                  = 100day
DWELL_TIME_MAX                  = 1.0JYR
INITIAL_STATE                   = body
INITIAL_BODY_NAME               = Vesta
ARRIVAL_DATE_MIN                = 56900
ARRIVAL_DATE_MAX                = 56980
TARGET_STATE                    = body rendezvous
TARGET_BODY_NAME                = Ceres
TGT_PROX_STATE_THRESHOLD        = 0.005
TGT_PROX_STATE_THRESHOLD_FINAL  = 0.001
TARGET_DIST_MAX_FINAL           = 1.0E6km
TARGET_DIST_MAX_INIT            = 1.0E6km
TARGET_DIST_MAX_SHRINK          = 0.8
TARGET_DIST_MAX_DECREASE        = 1.0E5km
TARGET_DIST_MAX_REDUCTION_USE_MAX = no
TARGET_RELVEL_MAX_FINAL         = 500m/s
TARGET_RELVEL_MAX_INIT          = 500m/s
TARGET_RELVEL_MAX_SHRINK        = 0.9
TARGET_RELVEL_MAX_DECREASE      = 1m/s
TARGET_RELVEL_MAX_REDUCTION_USE_MAX = no
ACCURACY_FITNESS_FRACTION       = 0.01
MIN_SOLAR_DISTANCE              = 0.2AU
INTEGRATOR                      = RK54F
MAX_RELATIVE_ERROR              = 1.0E-6
MAX_ABSOLUTE_ERROR              = 1.0E-6
USE_ITGR_STOPPER                = yes
SC_CONF                         = Ceres.scp
NAV_TYPE                        = ANN
NAV_ANN_CONF                    = Ceres.ncp
OPTIMIZATION_GOAL               = maximum dwell time
STEERING_DYN_UNIT_CALC          = yes
STEERING_USE_RANGE              = yes
STEERING_USE_RANGE_RATE         = yes
STEERING_USE_ACC_THRUST_MAX     = no
STEERING_USE_ACC_THRUST_MAX_DRY = no
STEERING_USE_STEP_SIZE          = no
STEERING_USE_TIME_UNTIL_PERI_SC = no
STEERING_USE_TIME_UNTIL_PERI_TGT = no
STEERING_USE_ABS_CART_POS_X     = yes
STEERING_USE_ABS_CART_POS_Y     = yes
STEERING_USE_ABS_CART_POS_Z     = yes
STEERING_USE_ABS_CART_VEL_X     = yes
STEERING_USE_ABS_CART_VEL_Y     = yes
STEERING_USE_ABS_CART_VEL_Z     = yes
STEERING_USE_ABS_POLAR_POS_R    = no
STEERING_USE_ABS_POLAR_POS_AZI  = no
STEERING_USE_ABS_POLAR_POS_ELE  = no
STEERING_USE_ABS_POLAR_VEL_R    = no
STEERING_USE_ABS_POLAR_VEL_AZI  = no
STEERING_USE_ABS_POLAR_VEL_ELE  = no
STEERING_USE_TGT_CART_POS_X     = no
STEERING_USE_TGT_CART_POS_Y     = no
STEERING_USE_TGT_CART_POS_Z     = no
STEERING_USE_TGT_CART_VEL_X     = no
STEERING_USE_TGT_CART_VEL_Y     = no
STEERING_USE_TGT_CART_VEL_Z     = no
STEERING_USE_TGT_CART_POS_REL_X = yes
STEERING_USE_TGT_CART_POS_REL_Y = yes
STEERING_USE_TGT_CART_POS_REL_Z = yes
STEERING_USE_TGT_CART_VEL_REL_X = yes
STEERING_USE_TGT_CART_VEL_REL_Y = yes
STEERING_USE_TGT_CART_VEL_REL_Z = yes
```

Bibliography

[1] C. Acton, N. Bachman, J. Diaz Del Rio, B. Semenov, E. Wright, and Y. Yamamoto. SPICE: A Means for Determining Observation Geometry. In *European Planetary Science Congress*, volume 6, EPSC–DPS2011–32, 2011.

[2] K. Alemany and R.D Braun. Survey of global optimization methods for low-thrust, multiple asteroid tour missions. In *AAS/AIAA 17th Space Flight Mechanics Meetings Jan. 28 – Feb. 1, 2007, Sezonda AZ*. AAS/AIAA, 2007.

[3] T. Bäck. Evolutionary algorithms. *SIGBIO Newsl.*, 12(2):26–31, June 1992. doi: 10.1145/130686.130691.

[4] T. Bäck. Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, Orlando, Florida, USA.*, ICEC-94, 1994. ISBN 0-7803-1899-4. doi: 10.1109/ICEC.1994.350042.

[5] E. Barrabés, G. Gómez, and J. Rodríguez-Canabal. *Lecture notes: Advanced Topics in Astrodynamics, Gravitational Assisted Trajectories*. Institut d'Estudis Espacials de Catalunya, Summer 2004. URL `http://www.ieec.cat/hosted/web-astro04/index.html`.

[6] V. Becerra. Solving complex optimal control problems at no cost with PSOPT. In *2010 IEEE International Symposium on Computer-Aided Control System Design*, pages 1391–1396. IEEE, 2010.

[7] J. Benkhoff, J. van Casteren, H. Hayakawa, H. Laakso, M. Novara, P. Ferri, H. R. Middleton, and R. Ziethe. BepiColombo—Comprehensive exploration of Mercury: Mission overview and science goals. *Planetary and Space Science*, 58(1-2):2–20, jan 2010. doi: 10.1016/J.PSS.2009.09.020.

[8] F. Bernelli-Zazzera, M. Vasile, N. Fornasari, and P. Masarati. Design of Interplanetary and Lunar Missions Combining Low Thrust and Gravity Assists. Technical report, ESA/ESOC Study Contract No. 14126/00/D/CS, 2002.

[9] J.T Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance control and dynamics*, 21(2):193–207, 1998.

[10] H. Bhasin and S. Mehta. On the applicability of diploid genetic algorithms. *AI & SOCIETY*, 31(2):265–274, May 2016. ISSN 1435-5655. doi: 10.1007/s00146-015-0591-x.

[11] J. Brophy, M.D. Rayman, and B. Pavri. Dawn: An ion-propelled journey to the beginning of the solar system. pages 1 – 10, 04 2008. ISBN 978-1-4244-1487-1.

[12] R.A Broucke and A. Prado. Jupiter Swingy-By Trajectories Passing Near the Earth, AAS 93-177. In *AAS-/AIAA Spaceflight Mechanics Meeting, Session XI: Orbit Transfers*, 1993.

[13] I. Carnelli. Optimization of Interplanetary Trajectories combining Low-Thrust and Gravity Assists with Evolutionary Neurocontrol. Master's thesis, Politechnico di Milano, 2005.

[14] I. Carnelli, B. Dachwald, and M. Vasile. Evolutionary Neurocontrol: A Novel Method for Low-Thrust Gravity-Assist Trajectory Optimization. *Journal of guidance, control, and dynamics*, 32(2), 2009.

[15] K. Chen, B. Aldrin, D. Landau, J. Longuski, and T. McConaghy. Powered earth-mars cycler with three-synodic-period repeat time. *Journal of Spacecraft and Rockets*, 42:921–927, 09 2005. doi: 10.2514/1.11610.

[16] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989. ISSN 1435-568X. doi: 10.1007/BF02551274.

[17] B. Dachwald. *Low-Thrust Trajectory Optimization and Interplanetary Mission Analysis Using Evolutionary Neurocontrol*. PhD thesis, Universität der Bundeswehr München, Faculty of Aerospace Engineering, Instute for Aerospace Engineering, 2004.

[18] B. Dachwald. Global optimization of low-thrust space missions using evolutionary neurocontrol. In *International workshop on global optimization*, pages 85–90, 2005.

[19] B. Dachwald. Optimal solar-sail trajectories for missions to the outer solar system. *Journal of Guidance, Control, and Dynamics*, 28(6):1187–1193, 2005.

[20] B. Dachwald. Optimization of very-low-thrust trajectories using evolutionary neurocontrol. *Acta Astronautica*, 57(2):175–185, 2005.

[21] B. Dachwald and L. Tsinas. A combined neural and genetic learning algorithm. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, Orlando, Florida, USA.*, ICEC-94, 1994. ISBN 0-7803-1899-4. doi: 10.1109/ICEC.1994.349968.

[22] C. Darwin. *On the origin of species*. New York: D. Appleton and Co., 1871. http://www.biodiversitylibrary.org/bibliography/28875.

[23] K. De Jong. Learning with genetic algorithms: An overview. *Machine Learning*, 3(2):121–138, 1988. ISSN 1573-0565. doi: 10.1007/BF00113894.

[24] K.A. De Jong. *Evolutionary Computation: A Unified Approach*. A Bradford book. 2006. ISBN 9780262041942.

[25] T. J. Debban, T. T. Mcconaghy, and J. M Longuski. Design and optimization of low-thrust gravity-assist trajectories to selected planets. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, California, 2002. AIAA 2002-4729.

[26] R. L. Dowling, W. J. Kosmann, M. A. Minovitch, and R. W. Ridenoure. The origin of gravity-propelled interplanetary space travel. *Proceedings of the 41st International Astronautical Federation*, 1990.

[27] L. J. Eshelman, R. A. Caruana, and J. D. Schaffer. Biases in the crossover landscape. In *Proceedings of the Third International Conference on Genetic Algorithms. George Mason University, San Francisco, CA, USA.*, pages 10–19. Morgan Kaufmann Publishers Inc., 1989. ISBN 1-55860-006-3.

[28] L. J. Fogel. *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1999. ISBN 0-471-33250-X.

[29] G. H. Fountain, D. Y. Kusnierkiewicz, C. B. Hersman, T. S. Herder, T. B. Coughlin, W. C. Gibson, D. A. Clancy, C. C. DeBoy, T. A. Hill, J. D. Kinnison, D. S. Mehoke, G. K. Ottman, G. D. Rogers, S. A. Stern, J. M. Stratton, S. R. Vernon, and S. P. Williams. The new horizons spacecraft. *Space Science Reviews*, 140(1): 23–47, Oct 2008. ISSN 1572-9672. doi: 10.1007/s11214-008-9374-8.

[30] M.S. Gashler and S.C. Ashmore. *Training Deep Fourier Neural Networks to Fit Time-Series Data*, pages 48–55. Springer International Publishing, Cham, 2014. ISBN 978-3-319-09330-7. doi: 10.1007/978-3-319-09330-7_7.

[31] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. volume 15 of *JMLR Proceedings*, pages 315–323. JMLR.org, 2011.

[32] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989. ISBN 0201157675.

[33] F. Gomez, J. Schmidhuber, and R. Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, 9(May):937–965, 2008.

[34] A. Gosavi. *Lecture notes: Neural networks and reinforcement learning. Department of Engineering Management and Systems Engineering*. Missouri University of Science and Tecnology. URL `http://web.mst.edu/~gosavia/neural_networks_RL.pdf`.

[35] A. Gosavi. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. Kluwer Academic Publishers, Norwell, MA, USA, 2003. ISBN 1402074549.

[36] Y. Guo and R. W. Farquhar. New Horizons Mission Design. *Space Science Reviews*, 140:49–74, 2008. doi: 10.1007/s11214-007-9242-y.

[37] B. Harvey and O. Zakutnyaya. *Russian Space Probes: Scientific Discoveries and Future Missions*. Springer Praxis Books. Praxis, 2011. ISBN 9781441981509.

[38] J.H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. A Bradford book. M.I.T.P., 1992. ISBN 9780262581110.

[39] D. Izzo. PyGMO and PyKEP: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization). 01 2012.

[40] D. Karaboga. *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer London, 2012. ISBN 9781447107217.

[41] A. Karpathy. *Lecture notes: Neural Networks Part 1: Setting up the Architecture , CS321n: Convolutional Neural Networks for Visual Recognition*. Stanford University, Winter 2017. URL `http://cs231n.github.io/`.

[42] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. A Bradford book. Bradford, 1992. ISBN 9780262111706.

[43] T.A.H. Kranen. Literature study on neuroevolution, interplanetary trajectory optimization and gravity assists. Technical report, Delft University of Technology; Faculty of Astrodynamics and Space Missions, 2017.

[44] L.D. Landau and E.M. Lifshitz. *Mechanics*. Elsevier Science, 1st edition, 1982. ISBN 9780080503479.

[45] Y.A. LeCun, L. Bottou, .B. Orr, and K. Müller. *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-642-35289-8. doi: 10.1007/978-3-642-35289-8_3.

[46] J.J. Lissauer and I. de Pater. *Fundamental Planetary Science: Physics, Chemistry and Habitability*. Cambridge University Press, 2013. ISBN 9780521853309.

[47] R.G. Madonna. *Orbital Mechanics*. Florida: Krieger Publishing Company, 1997.

[48] K.E. Mathias and L.D. Whitley. Initial performance comparisons for the delta coding algorithm. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence. Orlando, Florida, USA.*, 1994.

[49] R. May, G. Dandy, and H. Maier. chapter Review of Input Variable Selection Methods for Artificial Neural Networks. InTech, 2011. ISBN 9789533072432.

[50] T. T. Mcconaghy. GALLOP Version 4.5 User's Guide. Technical report, 2005.

[51] T. T. McConaghy, T. J. Debban, A. E. Petropoulos, and J. M. Longuski. Design and optimization of low-thrust trajectories with gravity assists. *Journal of Spacecraft and Rockets*, 40(3):380–387, 2003. doi: 10.2514/2.3973.

[52] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. Springer-Verlag, London, UK, UK, 1996. ISBN 3-540-60676-9.

[53] M.A. Minovitch. A Method for Determining Inteplanetary Free-Fall Reconnaissance Trajectories. Technical report, JPL, TM 312-130, 1961.

[54] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998. ISBN 0262631857.

[55] D.J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'89, pages 762–767, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[56] C.D. Murray and S.F. Dermott. *Solar System Dynamics*. Cambridge University Press, 1999. ISBN 9780521575973.

[57] NASA/JHUAPL/SwRI. Pluto dazzles in false color. `https://solarsystem.nasa.gov/resources/699/pluto-dazzles-in-false-color/?category=planets/dwarf-planets_pluto`. Accessed: 12/05/2019.

[58] H. Oberth. *Ways to Spaceflight*. Number 622 in NASA technical translation. National Aeronautics and Space Administration, 1972.

[59] A. Ohndorf. *Multiphase Low-Thrust Trajectory Optimization Using Evolutionary Neurocontrol*. PhD thesis, Delft University of Technology, 2016.

[60] A. Ohndorf, B. Dachwald, and B. Gill. Optimization of low-thrust earth-moon transfers using evolutionary neurocontrol. *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, pages 358–364, 2009. doi: 10.1109/CEC.2009.4982969.

[61] A.E. Petropoulos, J.M. Longuski, and N.X. Vinh. Shape-based analytic representations of low-thrust trajectories for gravity-assist applications. *AAS/AAIA*, 103:563–581, 01 2000.

[62] M. D. Rayman and K. C. Patel. The Dawn project's transition to mission operations: On its way to rendezvous with (4) vesta and (1) ceres. *Acta Astronautica*, 66(1):230 – 238, 2010. ISSN 0094-5765.

[63] M. D. Rayman, T. C. Fraschetti, C. A. Raymond, and C. T. Russell. Dawn: A mission in development for exploration of main belt asteroids Vesta and Ceres. *Acta Astronautica*, 58:605–616, 2006. doi: 10.1016/j.actaastro.2006.01.014.

[64] R. Rojas. *Neural Networks: A Systematic Introduction*. Springer Berlin Heidelberg, 1996. ISBN 9783540605058.

[65] C. T. Russell, F. Capaccioni, A. Coradini, M. C. De Sanctis, W. C. Feldman, R. Jaumann, H. U. Keller, T. B. Mccord, L. A. Mcfadden, S. Mottola, C. M. Pieters, T. H. Prettyman, C. A. Raymond, M. V. Sykes, D. E. Smith, M. T. Zuber, F. Capaccioni, M. C. De Sanctis, R. Jaumann, S. Mottola, H. U. Keller, and T. B. Mccord. Dawn Mission to Vesta and Ceres. *Earth Moon Planet*, 101:65–91, 2007. doi: 10.1007/s11038-007-9151-9.

[66] David S. Random walks: Training very deep nonlinear feed-forward networks with smart initialization. *CoRR*, abs/1412.6558, 2014.

[67] S.C. Shapiro. *Encyclopedia of Artificial Intelligence*. John Wiley & Sons, Inc. New York, 2 edition, 1992. ISBN 0471503053.

[68] J. Sims and S. N. Flanagan. Preliminary design of low-thrust interplanetary missions. 103, 01 2000.

[69] D. Sprecher. On the structure of continuous functions of several variables. *Transactions of the American Mathematical Society*, 115:340–355, 1964. doi: 10.1090/S0002-9947-1965-0210852-X.

[70] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009. doi: 10.1162/artl.2009.15.2.15202.

[71] K.O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10(2):99–127, June 2002. ISSN 1063-6560. doi: 10.1162/106365602320169811.

[72] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec 1997. ISSN 1573-2916. doi: 10.1023/A:1008202821328.

[73] V. C. Thomas, J. M. Makowski, G. M. Brown, J. F. McCarthy, D. Bruno, J. C. Cardoso, W. M. Chiville, T. F. Meyer, K. E. Nelson, B E. Pavri, D. A. Termohlen, M. D. Violet, and J. B. Williams. *The Dawn Spacecraft*, pages 175–249. Springer New York, New York, NY, 2012. ISBN 978-1-4614-4903-4. doi: 10.1007/978-1-4614-4903-4_10.

[74] M. Vasile and S. Campagnola. Design of low-thrust gravity assist trajectories to Europa. *Journal of the British Interplanetary Society*, 62(1):15–31, 1 2009. ISSN 0007-084X.

[75] M. Vasile, R. Biesbroek, L. Summerer, A. Galvez, and G. Kminek. Options for a Mission to Pluto and Beyond. In *13th AAS/AIAA Space Flight Mechanics Meeting*, Ponce, Puerto Rico, 2003.

[76] K.F. Wakker. *Fundamentals of Astrodynamics*. TU Delft Library, 2015. ISBN 9789461864192.

[77] J.R. Wertz. *Mission Geometry: Orbit and Constellation Design and Management*. Space technology library. Microcosm Press, 2009. ISBN 9781881883074.

[78] G. Whiffen. Mystic: Implementation of the static dynamic optimal control algorithm for high-fidelity, low-thrust trajectory design. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, page 6741, 2006.

[79] D. Whitley, K. Mathias, and P. Fitzhorn. Delta coding: An iterative search strategy for genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 77–84. Morgan Kaufmann, 1991.

[80] M. Wiering and M. van Otterlo. *Reinforcement learning : state-of-the-art*. Springer-Verlag Berling Heidelberg, Berlin; New York, 2012. ISBN 978-3-642-27644-6. doi: 10.1007/978-3-642-27645-3.

[81] O. Winter, E. Macau, H. Campos Velho, and V. Carruba. ASTER: a Brazilian mission to an asteroid. In *Asteroids, Comets, Meteors*, 2012.

[82] D. Włodzisław and K. Jerzy. Optimization and global minimization methods suitable for neural networks. *Neural Computing Surveys 2*, 1998.

[83] S. Yang. On the design of diploid genetic algorithms for problem optimization in dynamic environments. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1362–1369, July 2006. doi: 10.1109/CEC.2006.1688467.

[84] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[85] L. A. Young, S. A. Stern, H. A. Weaver, F. Bagenal, R. P. Binzel, B. Buratti, A. F. Cheng, D. Cruikshank, G. R. Gladstone, W. M. Grundy, D. P. Hinson, M. Horanyi, D. E. Jennings, I. R. Linscott, D. J. McComas, W. B. McKinnon, R. McNutt, J. M. Moore, S. Murchie, C. B. Olkin, C. C. Porco, H. Reitsema, D. C. Reuter, J. R. Spencer, D. C. Slater, D. Strobel, M. E. Summers, and G. L. Tyler. New Horizons: Anticipated Scientific Investigations at the Pluto System. *Space Science Reviews*, 140:93–127, October 2008. doi: 10.1007/s11214-008-9462-9.