POLITECNICO DI TORINO

Master's Degree in Aerospace Engineering

Master's Degree Thesis

Multiple-burn trajectory optimization and guidance design for In-Orbit-Service missions

Supervisors

Prof.ssa Elisa CAPELLO

Dott. Mauro MANCINI

Ing. Vincenzo ROMANO

Ing. Christophe ROUX

Candidate

Marco CAVALLO

July 2025

Abstract

In-Orbit-Service (*IOS*) has moved from visionary proposal to operational necessities for keeping orbits safe, sustainable, and affordable. In fact repairing, refuelling, upgrading or de-orbiting aging satellites lowers replacement costs and reduces collision risk.

The lowering of the launch prices has sharpened commercial interest; indeed, market studies foresee a multi-billion-dollar commercial sector expanding at a high growth rate over the coming decade. For these reasons, this thesis work aims to plan and validate impulsive rendezvous manoeuvres through a modular MATLAB tool for IOS that combines an optimization process based on an *EA (Evolutionary Algorithm)* with a 3-DOF orbital simulator. The aim of the optimization process is to minimize the propellant consumption in terms of the ΔV provided by a classic *Lambert's problem*. For this purpose, three evolutionary algorithms with a dedicated *cost function* have been tested: *Particle Swarm Optimization (PSO)*, *Differential Evolution (DE)* and *Covariance Matrix Adaptation (CMA-ES)*. When compared to reference transfers optmizations from literature, the three algorithms implementations show negligible differences with those in literature, demonstrating their high reliability.

For multi-target optimization the algorithms were combined with a *Travelling Salesman Problem (TSP)* since the propellant consumption and the time of flight *(TOF)* depend strongly on the order in which targets satellites are visited. This *TSP-EA* formulation is particularly valuable for IOS and ADR (*Active Debris Removal*) missions, where a single chaser may need to service or de-orbit several spacecraft distributed across different orbits.

All the optimals parameters provided by the EA (i.e. the time of flight, the target satellite departure true anomaly and the two ΔV) are passed to the orbital simulator. This one propagates the spacecrafts states via a classic fourth-order *Runge–Kutta* (`RK4`) integrator in which at each integration sub-step the dynamic model computes the translational state derivatives of the spacecrafts and the `GNC` algorithm updates the throttle command and thrust direction of the chaser. The guidance logic is open-loop during the first burn, in which the thrust is aligned with the optimized Lambert's $\Delta V_{1_{target}}$ direction and lasts until this one is fully erogated by the propulsion system. After this first impulse the algorithm switches to other two closed-loop phases: the coasting and the second burn. The first one lasts until a predefined target *Argument of Latitude* (θ_{target}) is reached without any propulsion. The second one starts immidiately after the coasting phase and during that the algorithm continuosly adjusts thrust direction to match the target satellite velocity. This second burn lasts until the Lambert's $\Delta V_{2_{target}}$ is reached.

Representative campaigns covering scenarios such as SSO-SSO (*Sun-Synchronous*

Orbit) transfers, LEO (*Low Earth Orbit*) two-impulse paths and multi-target servicing missions show close agreement between the desired results and the simulation ones in terms of ΔV, time of flight, relative velocity, and orbital parameters.
The resulting framework delivers fast and reliable insight into propellant needs and guidance performance, helping mission designers to obtain quicker evaulation of In-Orbit-Servicing scenarios.

Agli eterni disorganizzati

Table of Contents

List of Tables IX

List of Figures XII

Acronyms XVIII

1 Introduction 1

1.1 IOS hystorical review . . . 3

1.1.1 AVIO's In-Orbit Services (IOS) Projects . . . 7

1.2 Classification . . . 8

1.3 Commercial interest . . . 10

1.3.1 Aging satellites impact . . . 11

1.3.2 End-User insights . . . 12

1.4 Rendezvous and proximity operations . . . 13

1.4.1 Definition . . . 13

1.4.2 RPO phases . . . 15

1.4.3 Trajectory and Rendezvous optimization in literature . . . 16

1.4.4 Guidance algorithms in literature . . . 18

2 Foundamentals of astrodynamics 24

2.1 Overview of the laws . . . 24

2.1.1 Universal Law of Gravitation . . . 24

2.1.2 The N-Body Problem . . . 25
2.1.3 The Two-Body Problem . . . 26
2.1.4 Constants of the motion . . . 27
2.1.5 Trajectory Equation . . . 28
2.1.6 Types of orbits . . . 30
2.1.7 Coordinate Systems . . . 33
2.1.8 Classical Orbital Parameters . . . 36
2.1.9 Orbital elements determination from **r** and **v** . . . 37
2.1.10 Determining **r** and **v** from orbital elements . . . 38
2.2 External disturbances . . . 39
2.2.1 Atmospheric Drag . . . 39
2.2.2 J_2 Effect . . . 39
2.2.3 Gravity Gradient . . . 40
2.2.4 Magnetic Torque . . . 41
2.2.5 Solar Radiation . . . 41
2.3 Orbital manoeuvres . . . 42
2.3.1 Impulsive manoeuvres . . . 43

3 Evolutionary Algorithms 52
3.1 Overview . . . 52
3.2 Particle Swarm Optimization . . . 54
3.3 Differential Evolution . . . 56
3.4 CMA-ES . . . 61
3.5 Algorithms performance evaluation . . . 65
3.5.1 Objective Function . . . 65
3.5.2 Case study 1 – Pontani 2D “case b” . . . 66
3.5.3 Case study 2 – Pontani 3D “case 3” . . . 76
3.5.4 Algorithm Selection . . . 85

4 Travelling Salesman Problem 86
4.1 TSP formulation 86
4.2 TSP logic implementation 87
4.2.1 **1st** Scenario - TSP 87
4.2.2 **2nd** Scenario - TSP 89

5 3DOF Orbital Simulator 91
5.1 Orbital Simulator overview 91
5.2 RK4 numerical integrator 93
5.3 Dynamic model 95
5.4 Guidance algorithm 96
5.4.1 Flowchart of the GNC algorithm 98
5.5 Auxiliary functions 99

6 Simulations results 100
6.1 Test case I 101
6.2 Test case II 105
6.3 Test case III 109
6.4 Test case IV 112
6.5 Test case V 117
6.6 Test case VI 121
6.7 Test case VII 124
6.8 Test case VIII 129

7 Conclusions and future improvements 134

A Objective function 138

B Travelling Salesman Problem 140

C GNC algorithm 143

C.1 **1st** Burn logic 143

C.2 Guidance code 144

Bibliography 148

List of Tables

1.1 AVIO's IOS platform main services 8
1.2 Types of in-orbit services [17] 10

2.1 Specific impulse of various propellant types [43] 43

3.1 Orbital elements of Pontani's 2D transfer 67
3.2 Reference solution for Pontani's 2D transfer 67
3.3 Parameters of the PSO run 68
3.4 Transfer obtained with PSO (Pontani 2D case) 69
3.5 PSO performance (2D case). 69
3.6 Parameters of the DE run. 70
3.7 Transfer obtained with DE/rand/1/bin (Pontani 2D case) 71
3.8 DE/rand/1/bin performance (2D case) 71
3.9 Transfer obtained with DE/best/1/bin (Pontani 2D case) 71
3.10 DE/best/1/bin performance (2D case) 72
3.11 Transfer obtained with DE/current-to-best/1/bin (Pontani 2D case) 72
3.12 DE/current-to-best/1/bin performance (2D case) 73
3.13 CMA-ES configuration (Hansen & Ostermeier) [55] 74
3.14 Transfer obtained with CMA-ES (Pontani 2D) 74
3.15 Performance summary 75
3.16 Orbital elements of Pontani's 3D transfer 77

3.17 Reference solution for Pontani's 3D transfer 77
3.18 Parameters of the PSO run (Pontani 3D case) 78
3.19 Transfer obtained with PSO (Pontani 3D case) 78
3.20 PSO performance (2D case). 78
3.21 Transfer obtained with DE/rand/1/bin (Pontani 3D case) 79
3.22 DE/rand/1/bin performance (3D case) 80
3.23 Transfer obtained with DE/best/1/bin (Pontani 3D case) 80
3.24 DE/best/1/bin performance (3D case) 81
3.25 Transfer obtained with DE/current-to-best/1/bin (Pontani 2D case) 81
3.26 DE/current-to-best/1/bin performance (3D case) 82
3.27 Transfer obtained with CMA-ES (Pontani 2D) 83
3.28 Performance summary 83

4.1 Orbital elements of the satellites - scen. 1 88
4.2 Summary of transfer legs - scen. 1 88
4.3 Orbital elements of the satellites - scen. 2 89
4.4 Summary of transfer legs - scen. 2 90

5.1 Auxiliary functions overview 99

6.1 Environmental constants used in the simulation 101
6.2 Approximation of AVUM vehicle (AVIO) 101
6.3 Starting orbital elements of Chaser and Target 101
6.4 Optimization results for the manoeuvre 102
6.5 Simulation performance results 102
6.6 Orbital parameters – desired vs achieved 103
6.7 Starting orbital elements of Chaser and Target 105
6.8 Optimization results for the manoeuvre 106
6.9 Simulation performance results 106

6.10 Orbital parameters – desired vs achieved . . . 107
6.11 Starting orbital elements of Chaser and Target . . . 109
6.12 Optimization results for the manoeuvre . . . 109
6.13 Simulation performance results . . . 110
6.14 Orbital parameters – desired vs achieved . . . 110
6.15 Starting orbital elements of Chaser and Target . . . 113
6.16 Optimization results for the manoeuvre . . . 114
6.17 Simulation performance results . . . 114
6.18 Orbital parameters – desired vs achieved . . . 115
6.19 Starting orbital elements of Chaser and Target . . . 117
6.20 Spacecraft data . . . 117
6.21 Optimization results for the manoeuvre . . . 118
6.22 Simulation performance results . . . 118
6.23 Orbital parameters – desired vs achieved . . . 119
6.24 Starting orbital elements of Chaser and Target . . . 121
6.25 Spacecraft data . . . 121
6.26 Optimization results for the manoeuvre . . . 122
6.27 Simulation performance results . . . 122
6.28 Orbital parameters – desired vs achieved . . . 123
6.29 Orbital elements for the chaser and three targets . . . 124
6.30 Optimal manoeuvre results for each leg . . . 125
6.31 Simulation results for each leg . . . 125
6.32 Orbital parameters for each leg – desired vs achieved . . . 127
6.33 Orbital elements for the chaser and three targets . . . 129
6.34 Optimal manoeuvre results for each leg . . . 129
6.35 Simulation results for each leg . . . 130
6.36 Orbital parameters for each leg – desired vs achieved . . . 132

List of Figures

1.1 Workflow of the thesis toolchain, from algorithm design to validation and future extensions 3

1.2 Final phases of a space walk from the HST second servicing mission, astronauts Mark Lee (right) and Steven Smith work on HST while perched on the Shuttle's remote manipulator arm [8] 4

1.3 Concept of on-orbit servicing by the MEV (the left one) [12] 5

1.4 RISE mission render (on the left) and official patch (on the right) [13] 5

1.5 Evolution timeline of In-Orbit Servicing missions from Skylab (1973) to RISE (planned 2028). 6

1.6 Vega In-orbit Service (VIS) [15] 7

1.7 ALEK-2 render model (on the left) and AVUM+ subsystems model (on the right) [15, 16] 8

1.8 In-Orbit Operations classification [17] 9

1.9 On-Orbit Satellite Servicing Market Size 2024–2034 [18] 11

1.10 On-Orbit Satellite Servicing Market by Orbit Type [18] 12

1.11 On-Orbit Satellite Servicing Market by End-User [18] 13

1.12 RPO formal definitions [19] 14

1.13 RDV mission profile [26] 16

1.14 Open-Loop guidance scheme [24] 19

1.15 Closed-loop ZEM/ZEV guidance 20

1.16 Closed-loop PN guidance 20

1.17 Closed-loop LQR guidance with online path generation 21
1.18 Closed-loop LQR guidance with offline path generation 22
1.19 Closed-loop APF guidance with control law 23

2.1 The N-Body problem [44] 25
2.2 Heliocentric-ecliptic coordinate system [45] 34
2.3 Geocentric-equatorial coordinate system [43] 35
2.4 Perifocal frame [43] 36
2.5 Classical orbital elements [43] 37
2.6 JB-2006 model [24] 39
2.7 Earth's oblateness [24] 40
2.8 Solar radiation [24] 42
2.9 Propellant mass fraction versus ΔV for typical specific impulses [43] 44
2.10 Hohmann transfer [43] 45
2.11 Bi-elliptic transfer from inner orbit 1 to outer orbit 4 [43] 45
2.12 Orbits for which the bi-elliptical transfer is either less efficient or more efficient than the Hohmann transfer [43] 46
2.13 Main orbit (0) and two phasing orbits, faster (1) and slower (2). T_0 is the period of the main orbit [43] 47
2.14 Two non-coplanar orbits about F (on the left). A view down the line of intersection of the two orbital planes (on the right) [43] . . . 49
2.15 Lambert's problem [43] 50
2.16 Lambert's problem solving logic 51

3.1 PSO strategy [51] 56
3.2 Schematic workflow of the Differential Evolution algorithm 57
3.3 Simple DE mutation scheme in 2-D parametric space [53] 58
3.4 Different possible trial vectors formed due to binomial crossover between the target and the mutant vectors in 2-D search space [53] 59
3.5 Schematic workflow of the CMA–ES algorithm 62

3.6 Evolution of the CMA-ES search cloud over six generations on a 2D convex objective. Each panel shows background contours of the cost function, sampled offspring (black dots), and the standard deviation contour of the Gaussian distribution (orange dashed). As generations advance, the cloud deforms and shrinks, homing in on the optimum [56] 64

3.7 Pontani 2D case b reference trajectory [47] 68

3.8 PSO cost evolution 69

3.9 Transfer trajectory obtained with PSO (Pontani 2D case) 70

3.10 DE/rand/1/bin cost evolution 71

3.11 DE/best/1/bin cost evolution 72

3.12 DE/current-to-best/1/bin cost evolution 73

3.13 Transfer trajectory obtained with DE (Pontani 2D case) 73

3.14 CMA-ES cost convergence 75

3.15 Transfer trajectory obtained with CMA-ES (Pontani 2D) 75

3.16 Algorithms comparison (Pontani 2D) 76

3.17 Pontani 2D case b reference trajectory [47] 77

3.18 PSO cost evolution 78

3.19 Transfer trajectory obtained with PSO (Pontani 3D) 79

3.20 DE/rand/1/bin cost evolution 80

3.21 DE/best/1/bin cost evolution 81

3.22 DE/current-to-best/1/bin cost evolution 82

3.23 Transfer trajectory obtained with DE (Pontani 3D case) 82

3.24 CMA-ES cost convergence 83

3.25 Transfer trajectory obtained with CMA-ES (Pontani 3D) 84

3.26 Algorithms comparison (Pontani 3D) 84

4.1 Best tours found for the three debris clouds in the classic TSP variant for the fragmentation caused by Iridium 33-Cosmos 2251 collision and the 2007 Fengyun-1C event caused by a Chinese anti-satellite missile test [58] 86

4.2 TSP trajectory sequence - scen. 1 89
4.3 TSP trajectory sequence - scen. 2 90

5.1 Flow chart of the orbital simulator process 92
5.2 The fourth-order Runge–Kutta method evaluates the derivative four times once at the initial point, twice at two intermediate trial points, and once at the final trial point and then combines these four estimates to compute the next value of the unknown y [59] . . 94
5.3 Guidance flowchart 98

6.1 Propagation velocity evolution 103
6.2 Propagation velocity chasing 103
6.3 Rendezvous trajectory 104
6.4 Orbital elements evolution 104
6.5 Propagation velocity evolution 107
6.6 Propagation velocity chasing 107
6.7 Rendezvous trajectory 108
6.8 Orbital elements evolution 108
6.9 Propagation velocity evolution 110
6.10 Propagation velocity chasing 110
6.11 Rendezvous trajectory 111
6.12 Orbital elements evolution 112
6.13 Propagation velocity evolution 115
6.14 Propagation velocity chasing 115
6.15 Rendezvous trajectory 116
6.16 Orbital elements evolution 116
6.17 Propagation velocity evolution 119
6.18 Propagation velocity chasing 119
6.19 Optimal trajectory 120

6.20 Simulated trajectory 120
6.21 Orbital elements evolution 120
6.22 Propagation velocity evolution 122
6.23 Propagation velocity chasing 122
6.24 Optimal trajectory 123
6.25 Simulated trajectory 123
6.26 Orbital elements evolution 124
6.27 Propagation velocity evolution – leg 1 126
6.28 Propagation velocity chasing – leg 1 126
6.29 Propagation velocity evolution – leg 2 126
6.30 Propagation velocity chasing – leg 2 126
6.31 Propagation velocity evolution – leg 3 126
6.32 Propagation velocity chasing – leg 3 126
6.33 Mass evolution for each leg 127
6.34 Combined rendezvous trajectories 128
6.35 Orbital elements evolution 128
6.36 Mass evolution for each leg 130
6.37 Propagation velocity evolution – leg 1 131
6.38 Propagation velocity chasing – leg 1 131
6.39 Propagation velocity evolution – leg 2 131
6.40 Propagation velocity chasing – leg 2 131
6.41 Propagation velocity evolution – leg 3 131
6.42 Propagation velocity chasing – leg 3 131
6.43 Combined rendezvous trajectories 132
6.44 Orbital elements evolution 133

C.1 $|V_{1_{des}} - V_C|$ undesired peek during a simple plane change maneouver of 1° 143

Acronyms

AOM

AVUM Orbital Module

APF

Artificial Potential Field

ALEK

AVUM Life Extension Kit

ADR

Active Debris Removal

AVUM

Attitude and Vernier Upper Module

CAGR

Compound Annual Growth Rate

CMA-ES

Covariance Matrix Adaptation - Evolution Strategy

CPU

Central Processing Unit

COTS

Commercial Off The Shelf

DE

Differential Evolution

DOF

Degree Of Freedom

EA

Evolutionary Algorithm

GEO

Geostationary Earth Orbit

GTO

Geostationary Transfer Orbit

GNC

Guidance and Navigation Control

IOS

In-Orbit Servicing

LEO

Low Earth Orbit

LQR

Linear Quadratic Regulatore

MEO

Medium Earth Orbit

MPC

Model Predictive Control

OOS

On-Orbit Servicing

PD

Proportional Derivative

PN

Proportional Navigation

PSO

Particle Swarm Optimization

RACS

Redundant Attitude Control System

RK4

Runge Kutta 4th order

RPO

Rendezvous and Proximity Operations

SCP

Sequential Convex Programming

SSO

Sun Synchronous Orbit

TOF

Time Of Flight

TSP

Travelling Salesman Problem

USD

United States Dollar

ZEM

Zero Effort Mission

ZEV

Zero Effort Velocity

Chapter 1

Introduction

In the last decade, in-orbit servicing (IOS) and active debris removal (ADR) have become vital for keeping low-Earth orbit safe and cost-effective. Fixing, refuelling, upgrading, or safely removing old satellites cuts replacement costs and reduces collision risk. These tasks need new methods that can make these systems faster and more reliable. The aim of this thesis work is to develop, implement and validate a MATLAB computational tool capable of designing optimal impulsive rendezvous maneuvers that minimize the propellant consumption (ΔV), while also determining the most efficient sequence for visiting multiple targets. To achieve this goal, the work is structured around the following detailed objectives:

1. **Evolutionary Algorithm design and tuning**
    - Implementation of the evolutionary algorithm (Particle Swarm Optimization, Differential Evolution, CMA-ES) in MATLAB.
    - Construction of an *Objective Function* based on the Lambert equation which will be called by the EA for the optimization process.
    - Systematically select and tune EA parameters such as population size, selection criteria and stopping conditions through benchmark tests on canonical rendezvous scenarios (e.g. Pontani optimal rendezvous [1]).
2. **Travelling Salesman Problem for optimal multi-target sequencing**
    - Extend the optimizer to solve a Travelling Salesman Problem (TSP) variant: given three distinct rendezvous targets, determine the visitation sequence that yields the lowest cumulative ΔV.

- Model inter-target transfers via successive solutions of the Lambert problem, and insert the resulting ΔV costs into a combinatorial search framework.

3. **Guidance Algorithm Design**
   - Design of a three-phase impulsive guidance law (based on the optimization results of the EA) composed of:
     (a) *Burn 1*: initial impulse computed to inject the chaser onto the transfer arc.
     (b) *Coasting*: passive propagation along the transfer arc, during which the state is monitored.
     (c) *Burn 2*: terminal impulse to correct residual errors and complete the manouver.
4. **Integration with a 3DOF numerical propagator**
   - Couple the EA optimizer (including the TSP module) with a three-degree-of-freedom (3DOF) propagator based on a *Runge-Kutta4* numerical integrator to simulate the real execution of impulsive sequences in three-dimensional space.
   - Integrate the guidance algorithm as a function to be called in the RK4 numerical inetgrator.
5. **Performance evaluation**
   - Apply the developed code to realistic missions scenarios documenting the results (e.g. LEO to LEO or SSO manouver).
6. **Tool modularity and future extensions**
   - Design the MATLAB code architecture with modular components optimizer: Lambert solver, TSP sequencer, GNC function, propagator, etc. to facilitate future extensions.
   - Provide comprehensive documentation and example workflows that allow other users and mission designers to adapt the tool to different mission profiles and operational scenarios.

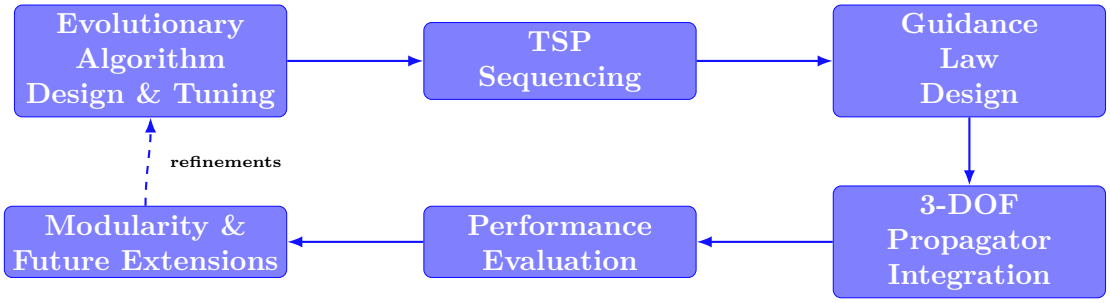


Figure 1.1: Workflow of the thesis toolchain, from algorithm design to validation and future extensions

By achieving these goals, this thesis delivers a flexible, easy-to-use tool for planning optimal impulsive rendezvous and multi-target missions, with potential useful applications in satellite in-orbit servicing.

1.1 IOS hystorical review

Since the 1960s, engineers have searched for extending the life of satellites and building larger systems by working directly in orbit. The first real test of these ideas came with the Skylab mission in 1973, when astronauts repaired solar panels, fixed antennas, and replaced critical components proving that human and robotic operations in microgravity were possible [2].

The Space Shuttle program of the 1980s then accelerated on-orbit servicing. Early Shuttle flights worked on the Solar Maximum Mission satellite, repairing its attitude control system and electronics, and similar operations were later carried out on Palapa B2 and Westar 6 [3]. Between 1993 and 2009, five dedicated Shuttle missions serviced the Hubble Space Telescope (HST), correcting its flawed mirror, swapping out old instruments, and adding new technology. These successes demonstrated how planning satellites for easy servicing can save time and money [4].

In the 1990s, Japan's ETS-VII mission became the first fully robotic servicer. ETS-VII tested autonomous rendezvous and docking, robotic arms, refuelling, and part replacement without astronauts on board [5]. Almost simultaneously, the International Space Station (ISS) began to take shape: from 1998 to 2011, modules and trusses were launched and assembled piece by piece, creating the largest human-made structure in orbit [6].

A major robotic milestone arrived with Orbital Express in 2007. This mission flew two free-flying satellites ASTRO (the servicer) and NEXTSat (the client) which performed automatic rendezvous, capture, refuelling, and exchange of Orbital

Replacement Units (ORUs), all without crewed spacecraft [7].

Figure 1.2: Final phases of a space walk from the HST second servicing mission, astronauts Mark Lee (right) and Steven Smith work on HST while perched on the Shuttle's remote manipulator arm [8]

From 2010 onward, several new programs have pushed IOS further: the first fully commercial life-extension missions in geostationary orbit began with Northrop Grumman's Mission Extension Vehicle family. MEV-1, launched in 2019, successfully docked with Intelsat 901 in February 2020 and provided both east–west and north–south station-keeping for over a year, proving that a dedicated servicer can extend a satellite's operational life without human intervention [9]. Its sister ship, MEV-2, followed in 2020 to service Intelsat 10-02 in early 2021 and remains on standby to repeat the process with other clients [9].

Meanwhile, Astroscale's ELSA-d mission tackled the complementary challenge of active debris removal in Low Earth Orbit. Launched in 2021, ELSA-d used a two-satellite chase-and-capture architecture to test magnetic and mechanical docking mechanisms, demonstrating precise proximity operations that will underpin both debris mitigation and future life-extension services [10].

Building on these successes, defense and space agencies have launched new development programs aimed at more sophisticated inspection, maintenance, and refueling capabilities. DARPA's Robotic Servicing of Geosynchronous Satellites (RSGS) will fly a servicer equipped with dual robotic arms and advanced vision systems to autonomously berth with commercial GEO satellites for on-orbit repair and refueling [11].

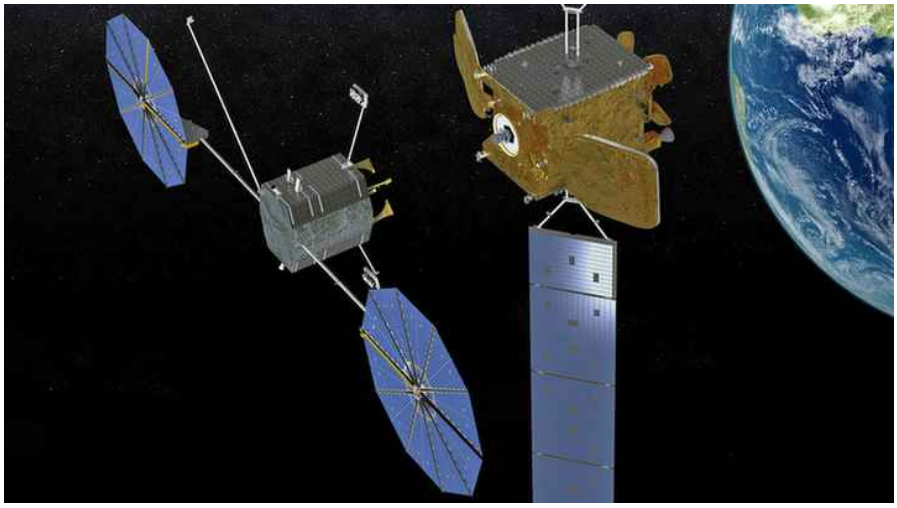


Figure 1.3: Concept of on-orbit servicing by the MEV (the left one) [12]

At the same time, ESA together with commercial partner D-Orbit has approved the Responsive In-Orbit Servicing Experiment (RISE), scheduled for launch in 2028. RISE will perform rendezvous, docking, and fuel transfer in GEO, further expanding the toolbox of services available to satellite operators and paving the way for a fully serviced, sustainable orbit [13].

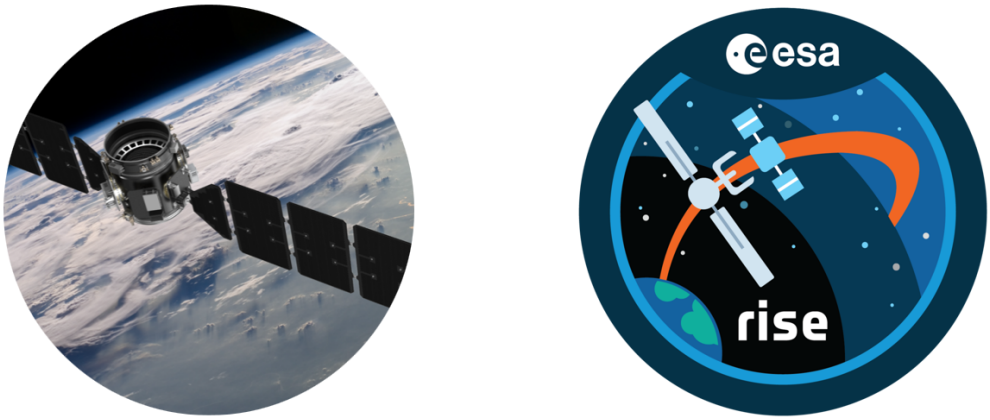


Figure 1.4: RISE mission render (on the left) and official patch (on the right) [13]

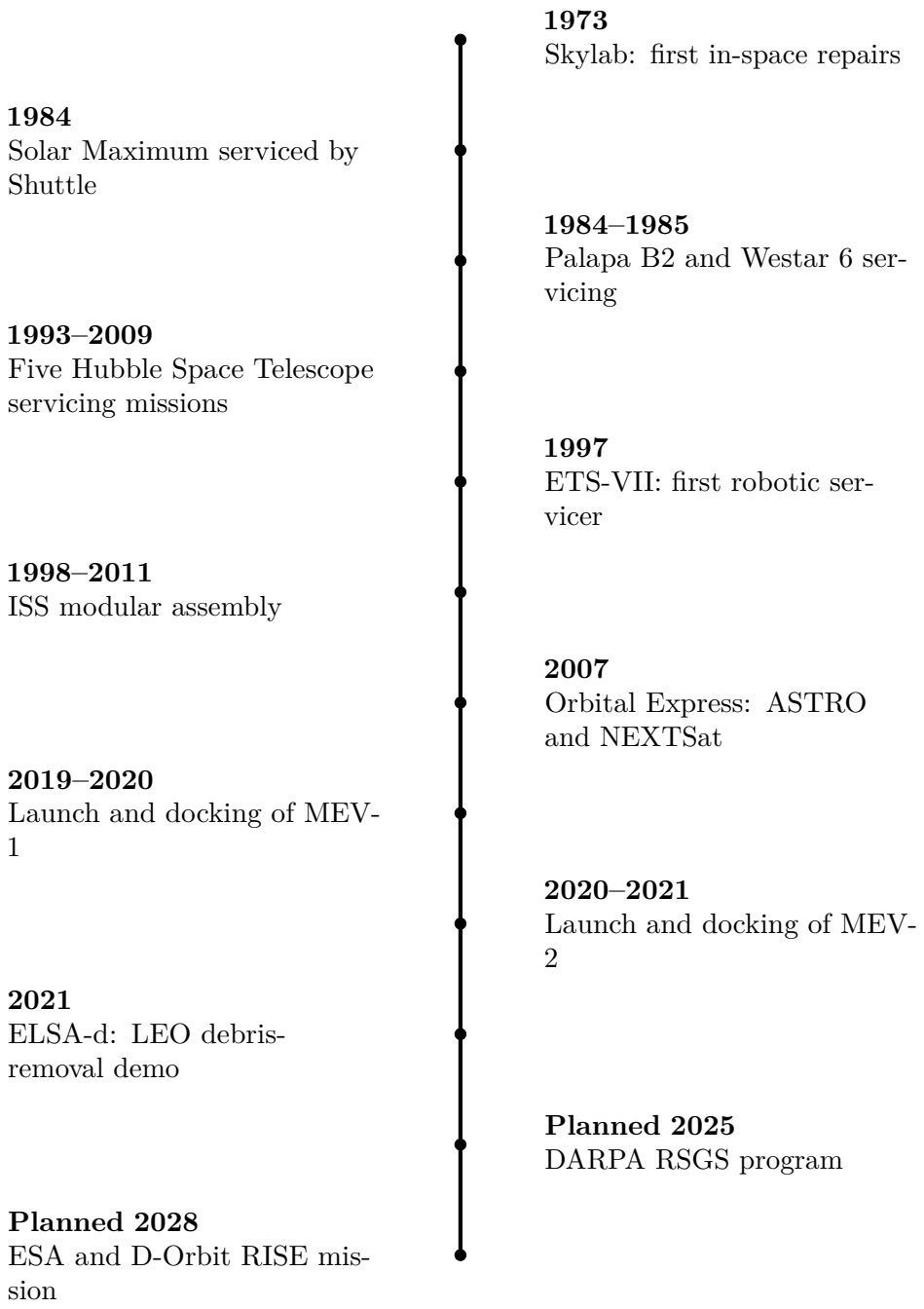


Figure 1.5: Evolution timeline of In-Orbit Servicing missions from Skylab (1973) to RISE (planned 2028).

1.1.1 AVIO's In-Orbit Services (IOS) Projects

AVIO is developing a comprehensive *In-Orbit Services* (IOS) capability branded as Vega In-Orbit Services (VIS) fully integrated with the Vega C launch system. VIS is conceived as a modular, end-to-end solution that leverages the Vega C payload fairing to both deploy primary customer satellites and then undertake secondary "last-mile" operations, including rendezvous, capture, satellite transportation, refuelling, debris removal, and controlled de-orbiting [14]. The IOS vision emphasizes sustainability and commercial viability, offering a broad spectrum of core services (rendezvous, debris removal, satellite transport) and a roadmap toward future capabilities such as in-orbit assembly, life-extension, and on-orbit refurbishment [14].

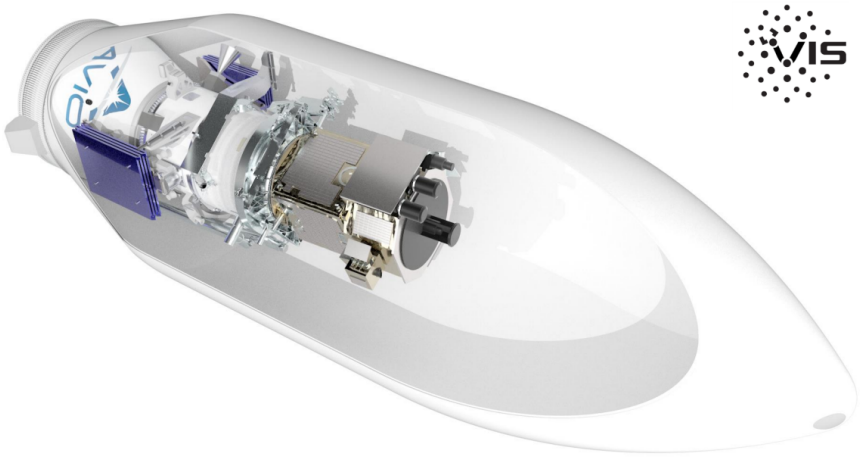


Figure 1.6: Vega In-orbit Service (VIS) [15]

A key element of AVIO's IOS architecture is the *AVUM Orbital Module* (AOM), which transforms Vega C's fourth stage (AVUM+) into a fully functional service module capable of extended LEO operations. After launch and orbital insertion, AVUM+ hands over control to the *AVUM Life Extension Kit* (ALEK), activating additional avionics, power conditioning, and thermal control systems that sustain the module for up to two months on orbit [15, 16]. The AOM provides "smart" propulsion (via the Vega's bi-propellant main engine and RACS), power (through deployable solar arrays delivering up to 3.6 kW, with 2 kW reserved for payloads), attitude control (reaction wheels, magnetic torquers, and star trackers), and a robust On-Board Data Handling system, all packaged within a reinforced PLA-1194 adapter to interface seamlessly with both the launcher and serviced payloads [15, 16].

The table below shows the main services that AVIO's IOS platform can perform

through VIS missions

Service	Description
Rendezvous & Proximity Operations	Autonomous guidance, navigation, and control to approach and capture target objects
Satellite Transport & Deployment	Delivery of secondary payloads to precise orbits following primary mission release
In-Orbit Refuelling & Life Extension	Docking and propellant transfer to extend operational lifetimes
Debris Removal & Safe Disposal	Capture of obsolete satellites or debris followed by controlled re-entry or storage
De-Orbit Maneuvers	Precision boosts to ensure safe atmospheric re-entry, minimizing collision risks

Table 1.1: AVIO's IOS platform main services

This modular, COTS-driven approach maximizing reuse of Vega C heritage hardware while integrating novel avionics positions AVIO's IOS as a versatile solution for the growing demand in sustainable, commercial orbital logistics and servicing [14, 16].

Figure 1.7: ALEK-2 render model (on the left) and AVUM+ subsystems model (on the right) [15, 16]

1.2 Classification

Although no official or universally agreed definition currently exists and terms like in-space services or on-orbit services are often used interchangeably the European Space Policy Institute report [17] identifies three main categories of operations:

- In-Orbit Servicing which refer to the provision of support services by a spacecraft (servicer) to another space object (serviced) while in orbit;
- In-Orbit Manufacturing which are defined as the use of innovative techniques, such as space resources or 3D printers, to build items and components directly in outer space;
- In-Orbit Assembly which is characterized as the assembly or combination of modular platforms to form a new object as well as the integration of upgrade payloads in orbit

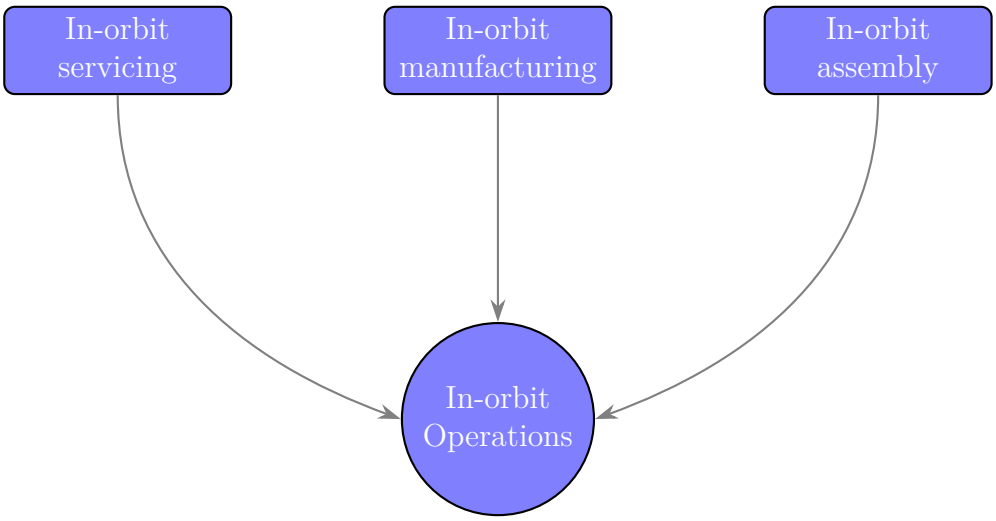


Figure 1.8: In-Orbit Operations classification [17]

The collection of several definitions of In-Orbit Operations enables to identify several categories of orbital services such as, but not limited to, maintenance, tugging and inspection [17]:

Category	Activity	Brief Description
Maintenance	Repair	Replace or fix components to extend operational life.
	Reconfiguration	Modify payloads/modules to change mission profile.
	Refuelling	Transfer propellant or fluids to sustain operations.
	Recharging	Restore power via docking or power-beaming.
	Upgrade	Add or swap components to enhance capabilities.
Tugging & Towing	Station-keeping	Dock to maintain orbit or attitude.
	Orbit correction	Adjust orbital trajectory.
	Relocation	Move spacecraft to a new orbital slot.
	De-orbiting	Transfer to graveyard orbit or enable controlled re-entry.
	Recycling	Recover materials from spent stages for reuse.
Inspection	In-orbit inspection	Assess satellite health and detect anomalies.

Table 1.2: Types of in-orbit services [17]

1.3 Commercial interest

As reported in the Precedence Research report on the size of on-orbit satellite servicing market [18], the surge in commercial interest in in-orbit servicing has been fueled by advancements in technology and a decline in satellite launch costs. As low Earth orbit (LEO) satellites become more prevalent, the feasibility of servicing these satellites has increased. Notably, sectors such as telecommunications have expressed strong interest in life-extension services. Operators of satellites in geostationary orbit (GEO) are particularly keen to ensure their satellites remain functional, as these represent critical assets for communication services.

The global on-orbit satellite servicing market size accounted for USD 4.22 billion in 2024 and is expected to exceed around USD 11.56 billion by 2034, growing at a CAGR of 10.60% from 2025 to 2034. The rising demand to maintain and repair satellites in space, which is essential to extending the life of satellites and

improving their performance, contributes to the growth of the on-orbit satellite servicing market.

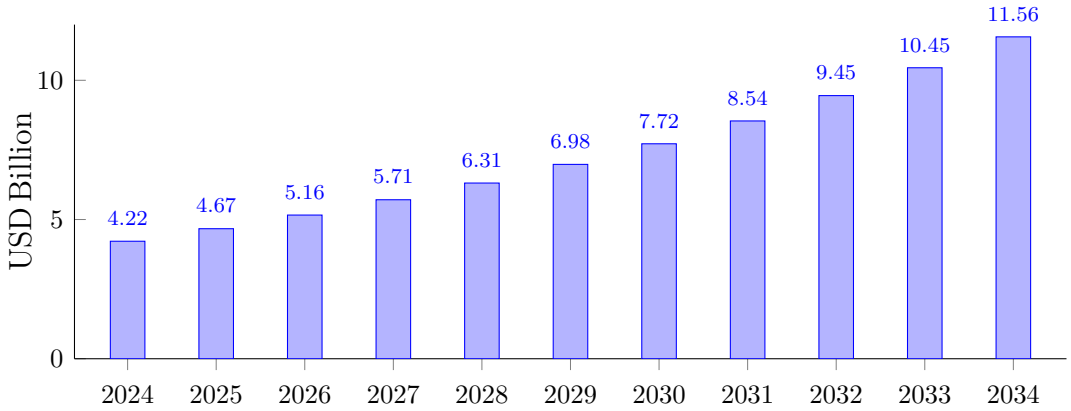


Figure 1.9: On-Orbit Satellite Servicing Market Size 2024–2034 [18]

1.3.1 Aging satellites impact

The aging satellite population is one of the key drivers of the on-orbit satellite servicing market. For instance, in September 2022, SpaceX launched 3,399 Starlink satellites into orbit, including prototypes and spacecraft. Numerous satellite enterprises are experiencing growth in the U.S., driven by the increasing need for communication and space-based internet services. Several satellites in orbit near the end of their operational lifespans often require maintenance or replacement to sustain functionality.

Typically, satellites are designed to operate for a specific duration, beyond which they may encounter technical issues or diminished performance. The provision of on-orbit satellite servicing also serves to tackle technical challenges that emerge throughout a satellite's active tenure. Exposure to rigorous space conditions can lead to failures or declines in performance, and on-orbit servicing offers a means to detect and rectify these issues. Consequently, this aids in maintaining the operational integrity and dependability of satellite-dependent services like communication, navigation, and remote sensing.

With an increasing number of satellites nearing the conclusion of their operational life, there is an anticipated surge in demand for on-orbit satellite services, which is poised to stimulate innovation and investment within this sector.

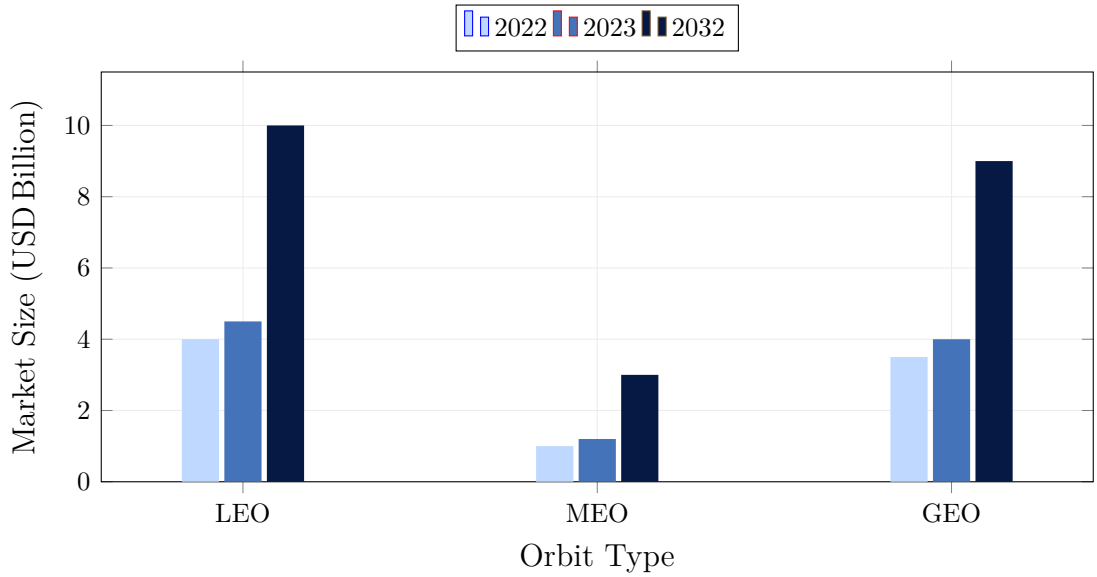


Figure 1.10: On-Orbit Satellite Servicing Market by Orbit Type [18]

On the basis of orbit type, the market is divided into LEO (low earth orbit), MEO (medium earth orbit), and GEO (geostationary earth orbit). The provision of on-orbit satellite servicing also serves to tackle technical challenges that emerge throughout a satellite's active occupation.

1.3.2 End-User insights

The commercial segment dominated the on-orbit satellite servicing market in 2024. The dominance of this segment is credited to the presence of on-orbit satellite servicing companies, which are integrated with advanced technology such as robotic arms, sensors, and software to command, interact, and manipulate the satellite. Additionally, commercial companies use automation, robotics, and navigation technology to maneuver satellites to position them for servicing.

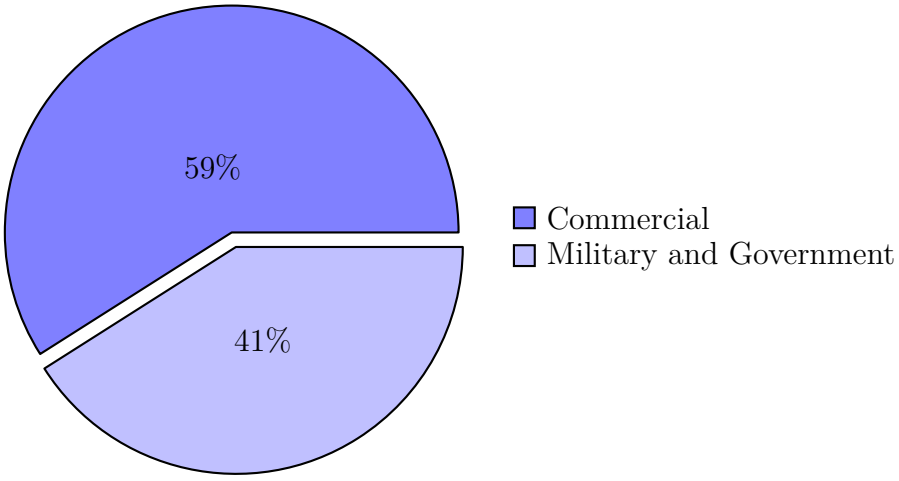


Figure 1.11: On-Orbit Satellite Servicing Market by End-User [18]

The military and government segment is projected to expand rapidly in the on-orbit satellite servicing market over the coming years. The growth of this segment is owed to the utilization of OOS for satellite intelligence gathering, navigation, and communication by the military and government. Along with that, they are also being used for disaster management, agriculture, and forestry.

1.4 Rendezvous and proximity operations

1.4.1 Definition

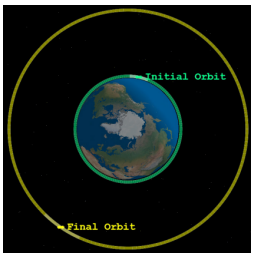
The concept of rendezvous and proximity operations (RPO) has existed for decades, but it has recently received more attention from the industry due to increased interest in activities like in-orbit servicing. Generally, RPO involves orbital maneuvers in which two spacecraft achieve the same orbit and approach each other at close distances. Such rendezvous procedures might subsequently lead to docking, although docking is not necessarily required. A formal definition of RPO is presented in Fig. 1.12 below [19].

As said before, activities such as in-orbit servicing, formation flying, and active debris removal have made RPO more relevant than ever. In-orbit servicing requires two satellites to match orbits, stay in close proximity, and often dock or grapple for instance, to transfer fuel [11, 20]. Formation flying describes multiple satellites keeping fixed relative positions to collect large numbers of Earth images[21] or boost system resilience [22]. In fact, formation flying is just a special case of RPO, since it defines the relative motion framework. Active debris removal seeks to

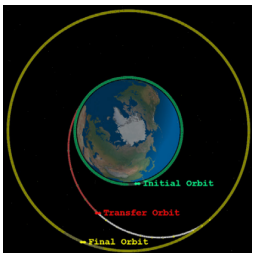
intercept and secure defunct satellites or stray fragments before their orbits decay naturally (especially in GEO or crowded LEO altitudes) [23].

A key distinction is between cooperative and non-cooperative RPO. In cooperative RPO, the chaser and target exchange status, position, attitude, and other data sometimes routed through the ground. Docking with the ISS is a classic example, with two-way communications throughout the approach. Non-cooperative RPO, by contrast, involves only one-way data gathering by the chaser; the target doesn't actively share its state [24]. Servicing inactive satellites or removing debris typically falls under this category.

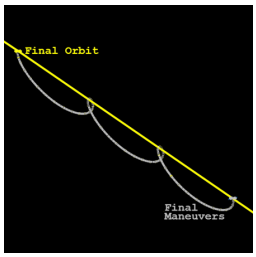
Rendezvous and Proximity Operations



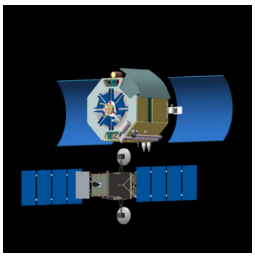
Initial configuration.



The transfer orbit, which effects rendezvous.



Example of a maneuver for proximity operations.



Final configuration just before docking.

Definitions:

Rendezvous matching the plane, altitude, and phasing of two (or more) satellites.

Proximity operations two (or more) satellites in roughly the same orbit intentionally perform maneuvers to affect their relative states.

Docking subset of proximity operations, where one satellite intentionally performs maneuvers to physically contact another satellite.

Cooperative RPO information (position, velocity, health/status, etc.) transfer is two-way via crosslinks, ground contact, etc. Example: docking with the ISS.

Non-cooperative RPO information transfer between vehicles is one-way only. Example: active debris removal.

Figure 1.12: RPO formal definitions [19]

1.4.2 RPO phases

As previously mentioned, a rendezvous is comprised of a list of orbital maneuvers in which two vehicles enter the same orbit and converge until very close, and then stop. Usually, the *Chaser* takes an active trajectory that brings it towards an inactive *Target*, the latter merely maintains its position. The Target may be cooperative (e.g., an active spacecraft prepared to support) or non-cooperative (e.g., an old satellite, bit of debris). Following rendezvous, the two spacecraft may establish the rigid union. If both are free flyers and both approach and dock actively, the procedure is termed *docking*, preferred for crewed missions since it can be performed rapidly in case of emergencies (e.g., for the evacuation of the ISS). If instead, through the use of a robot arm, the passive module is brought into fixed interface, the procedure is referred to as *berthing*. Berthing is usually best tailored for in-orbit servicing situations, particularly if the Target is non-cooperative in that instance it is better in this case to consider the procedure *capture* instead of berthing. Following that, once the Chaser and Target approach is very near, the operations must comply with stringent rules on the manner in which the two approach one another, sense each other, and interact. Constraints are put on the relative pose and velocity, on the sensor and algorithm accuracy, and even on the very path approach trajectory. In order to handle the complexity, the whole process is divided into different stages each with its own design methods, coordinate frames, and requirements for performance [24, 25]:

- **Launch and Orbit Injection**: Put the Chaser into the same lower orbit as the Target, having the same inclinations and RAAN. The selection of launch location and window is dictated by this requirement.
- **Phasing**: Compensate for any RAAN or inclination misalignments, then trim the phase angle between Chaser and Target. Bring the Chaser near in orbital position for the start of the relative navigation while still under ground-station control.
- **Far-Range Rendezvous ("Homing")**: Slow the Chaser relative velocity down to align it in the same orbit as the Target. Guidance from then on is relative and may be automated.
- **Close-Range Rendezvous**:
  - *Closing*: Achieve the accuracies in relative position, velocity, and attitude necessary for the final approach.
  - *Final Approach*: Execute along a predetermined path towards the Target in order to achieve the state necessary for mating.
- **Mating**: Mate the Chaser and Target with stringent tolerancing on residual

angular/linear velocity and pointing. If berthing, this will also control the motion of the robotic arm in capture.

Although numerous strategies may be conceived and implemented, an illustrative mission profile is presented in Fig. 1.13

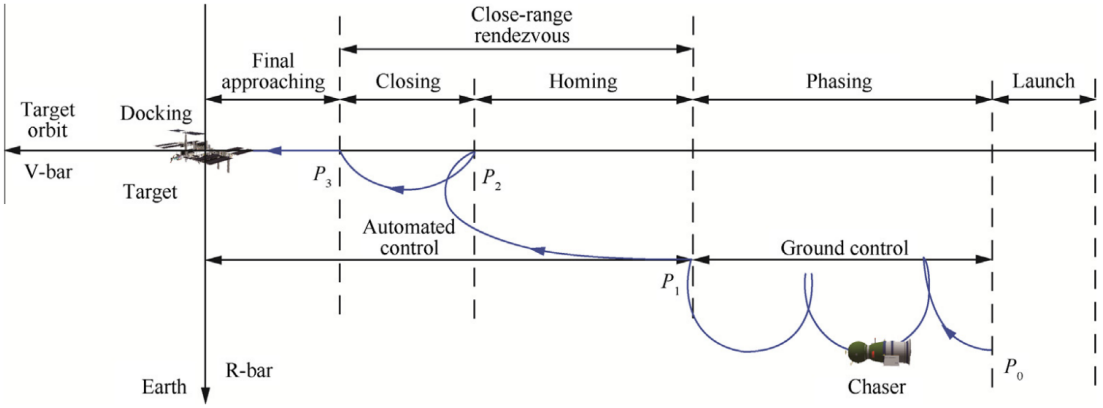


Figure 1.13: RDV mission profile [26]

1.4.3 Trajectory and Rendezvous optimization in literature

A wide variety of numerical and analytical techniques have been proposed for optimal spacecraft trajectory design. Beyond the *Lambert–EA* approach adopted in this work, four major families dominate the literature. For each family, the main principles are outlined together with key strengths, weaknesses, and detailed applications specifically related to Rendezvous and Proximity Operations (RPO).

Indirect Methods

Indirect or *optimum-control* methods apply Pontryagin's Maximum Principle to derive first-order necessary conditions of optimality. This yields a two-point boundary-value problem (TPBVP) in the state $\mathbf{x}$ and costate $\boldsymbol{\lambda}$ variables,

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \qquad \dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \mathbf{x}}, \qquad \left.\frac{\partial H}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}^\star} = 0, \tag{1.1}$$

where $H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = \boldsymbol{\lambda}^\mathsf{T} f(\mathbf{x}, \mathbf{u}) + L(\mathbf{x}, \mathbf{u})$ is the Hamiltonian.

- *Strengths* — High accuracy and formal guarantees of optimality; computationally efficient once a convergent initial guess is supplied.
- *Weaknesses* — Extreme sensitivity to the initial costate guess; implementation effort grows quickly with problem dimensionality, phasing, and complex path constraints [27, 28].

- *Applications in RPO* — Indirect methods have been widely employed for cooperative rendezvous missions where fine accuracy is required, such as spacecraft docking missions to space stations and multiple spacecraft cooperative maneuvers. These methods enable accurate final-approach trajectories and have been effectively applied to missions requiring strict trajectory constraints like station-keeping and refueling missions [29].

Primer-Vector Theory

Primer-vector theory offers an analytical framework for impulsive, fuel-optimal transfers. The *primer vector* $\mathbf{p}(t)$, equal to the costate of the velocity, must satisfy

$$\mathbf{p}(t_i) = \mathbf{0}, \qquad \|\mathbf{p}(t)\| \leq 1 \ \ \forall t, \tag{1.2}$$

with impulses applied precisely at the switching times t_i where $\|\mathbf{p}\|$ touches unity.

- *Strengths* — Closed-form insight into optimal ΔV split and timing; low computational cost once analytical solutions exist.
- *Weaknesses* — Limited to impulsive maneuvers and to problems that admit primer analytics (typically low-eccentricity or planar cases).
- *Applications in RPO* — Primer-vector theory is particularly suited for impulsive rendezvous maneuvers in nearly circular orbits, such as servicing satellites in geostationary orbit or performing inspection and maintenance missions in low Earth orbit. Its analytical simplicity allows for real-time trajectory adjustments during close-proximity operations, crucial for rapid-response scenarios [30].

Direct Transcription and Collocation

Direct methods discretize state and control histories on a grid, transforming the continuous optimal-control problem into a nonlinear program (NLP). Either finite-difference collocation or spectral pseudospectral schemes enforce the dynamics and continuity constraints.

- *Strengths* — Robust to the initial guess; can handle multi-phase structures, interior-point constraints, and low-thrust dynamics in a unified way.
- *Weaknesses* — Produces large, sparse NLPs that demand efficient sparse solvers and careful mesh refinement .
- *Applications in RPO* — Direct transcription and collocation techniques are extensively applied to trajectory optimization for spacecraft proximity rendezvous, particularly beneficial for missions involving intricate constraints

like collision avoidance, debris removal operations, and multi-phase transfers. The techniques efficiently manage dynamic challenges involved in spacecraft rendezvous operations to produce accurate trajectory planning for complex missions. [31, 32].

Sequential Convex Programming (Successive Convexification)

Sequential convex programming (SCP) linearizes non-linear dynamics and non-convex constraints around the current iterate $(\mathbf{x}_k, \mathbf{u}_k)$, yielding a convex sub-problem typically a quadratic or second-order cone program solved iteratively.

1. Initialize with a dynamically feasible trajectory.
2. Linearize dynamics and constraints about $(\mathbf{x}_k, \mathbf{u}_k)$.
3. Solve the convexified problem to obtain $(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})$.
4. Repeat until convergence.

- *Strengths* — Rapid convergence in practice; accommodates stringent state/-control limits by construction.
- *Weaknesses* — Convergence is local; poor linearization trust regions can lead to sub-optimal solutions or divergence [33].
- *Applications in RPO* — SCP is widely applied to time-critical rendezvous missions, like autonomous spacecraft docking, close-proximity operations under strict safety conditions, and coordinated missions by several vehicles. SCP's capacity to converge quickly makes it a good option for real-world applications for automated rendezvous and docking operations, such as ISS servicing missions and debris mitigation initiatives. [34].

Overall, indirect and primer methods excel when analytical insight and maximum optimality are predominant, whereas direct transcription and SCP provide robustness and constraint handling for modern, complex, multi-phase RPO missions at the cost of larger numerical optimization problems.

1.4.4 Guidance algorithms in literature

As previously said, in this thesis work we combine an optimization algorithm together with a guidance logic based on an optimization of the Lambert's problem. More generally choosing the right navigation and guidance algorithms for the chaser spacecraft's GNC system is critical to ensuring successful rendezvous and proximity operations and in this section are provided some major examples from literature. There are two main GNC logic: open-loop algorithm and closed-loop algorithm.

Open-Loop guidance

We pre-compute an optimal steering schedule on the ground using a detailed vehicle model and upload it to the spacecraft before launch. Once in flight, the on-board system looks up the appropriate commands from a table and applies them automatically based on the vehicle's elapsed time or current position [24].

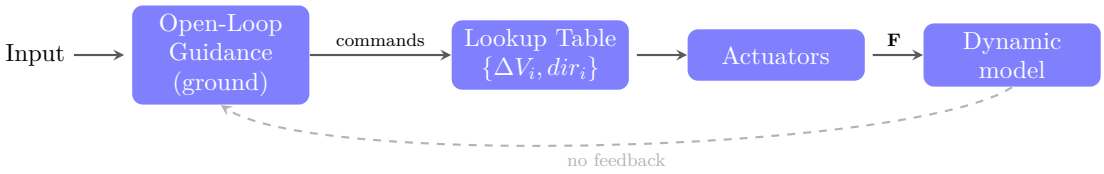


Figure 1.14: Open-Loop guidance scheme [24]

The advantages and limitations of this kind of algorithm are briefly described. Firstly the spacecraft only needs to read commands from a table so no complex software is required onboard and there is an overall reduction in onboard computation which saves power in flight. On the other hand if something goes off course (for example thruster performance or unexpected forces) the system cannot adapt. In addition to this, to fix any drift or misalignment the spacecraft, additional maneuvers are often needed, adding operational work. In this thesis we will see a ground correction of the key factor due to the disturbances that the Lambert problem doesn't take into account [35, 36].

Closed-Loop guidance

Closed-loop guidance continuously updates steering commands based on real-time state estimates and desired trajectories, providing robustness against disturbances and model uncertainties. Major classes are:

- **Acceleration-based algorithms**
  - *Zero-Effort-Mission / Zero-Effort-Velocity (ZEM-ZEV)*
    The ZEM term is the distance between the Chaser and the Target considering that the Target is moving along a known predefined path while the ZEV term can be defined as the end-of-mission velocity offset with no acceleration applied [24].

$$\text{ZEM} = \mathbf{r}_f - t_{go}\,\mathbf{v} + \int_t^{t_f} \frac{(t_f - \tau)\,\mathbf{g}(\tau)}{t_{go}}\,d\tau, \tag{1.3}$$

$$\text{ZEV} = \mathbf{v}_f - \left[\mathbf{v} + \int_t^{t_f} \mathbf{g}(\tau)\,d\tau\right], \tag{1.4}$$

    where $t_{go} = t_f - t$ represents tha time-to-go, i.e. the mission flight time necessary to achieve the Target and it's defined by the user. The

commanded acceleration is

$$\mathbf{a} = \frac{6}{t_{go}^2}\,\mathrm{ZEM} \;-\; \frac{2}{t_{go}}\,\mathrm{ZEV}. \tag{1.5}$$

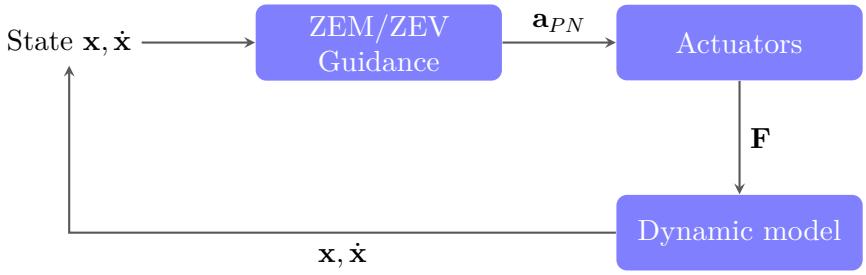


Figure 1.15: Closed-loop ZEM/ZEV guidance

It is usually combined with a Sliding Mode Controller and its usage could be a good option for Mars Landing [37].

– *Proportional Navigation (PN)*
Proportional Navigation is a classic guidance law, first developed for tactical missiles and later adapted for spacecraft rendezvous. It drives the line-of-sight (LOS) rate to zero by commanding a lateral acceleration proportional to the LOS angular rate. Defining

$$\lambda = \arctan\!\left(\tfrac{z}{x}\right), \qquad \dot{\lambda} = \frac{x\,\dot{z} \;-\; z\,\dot{x}}{r^2}, \qquad V_c = -\frac{x\;v_x + z\;v_z}{r}$$

the acceleration normal to the LOS is

$$\mathbf{a}_{PN} = N\,V_c\,\dot{\lambda}\,\mathbf{n}_{LOS},$$

where N (the navigation constant) is typically chosen between 3 and 5 to balance responsiveness and smoothness.

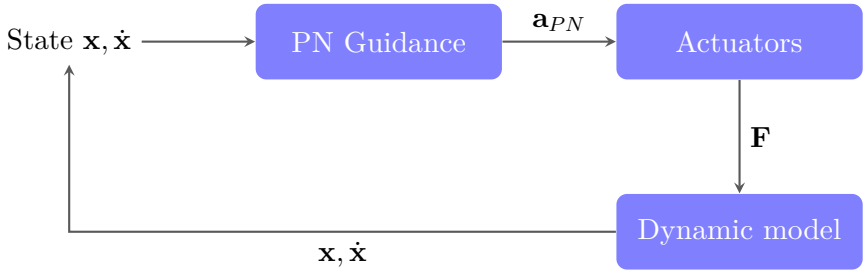


Figure 1.16: Closed-loop PN guidance

Nowadays augmented versions of the PN are used in CubeSats equipped with only optical sensor to achieve autonomous and robust proximity with uncooperative satellites [38].

- **Control-based algorithms**
  - *LQR-based guidance*
    Linear-Quadratic Regulator (LQR) controller combines a reference trajectory provided by guidance with a state-feedback controller to minimize a quadratic cost on tracking error and control effort. Starting from the linearized dynamics

$$\dot{\mathbf{x}} = A\,\mathbf{x} + B\,\mathbf{u},$$

    we define the state-error $\mathbf{x}_e = \mathbf{x} - \mathbf{x}_{\text{ref}}$ and seek the control law

$$\mathbf{u} = K_{\text{LQR}}\,\mathbf{x}_e,$$

    where K_{LQR} is chosen to minimize the infinite-horizon cost

$$J = \tfrac{1}{2}\int_0^\infty \left(\mathbf{x}_e^\top Q\,\mathbf{x}_e \;+\; \mathbf{u}^\top R\,\mathbf{u} \;+\; 2\,\mathbf{u}^\top N\,\mathbf{x}_e\right) dt.$$

    Here Q, R, and N are designer-specified weighting matrices that balance state-tracking accuracy against control effort. The optimal gain is computed by solving the associated Algebraic Riccati Equation ([39]) to obtain

$$K_{\text{LQR}} = -\,R^{-1}\,(B^\top P + N^\top),$$

    where P is the positive-definite solution of

$$A^\top P + P\,A - (P\,B + N)\,R^{-1}(B^\top P + N^\top) + Q = 0.$$

    The matrices Q and R can be retuned onboard as the rendezvous geometry evolves to maintain robustness and performance.

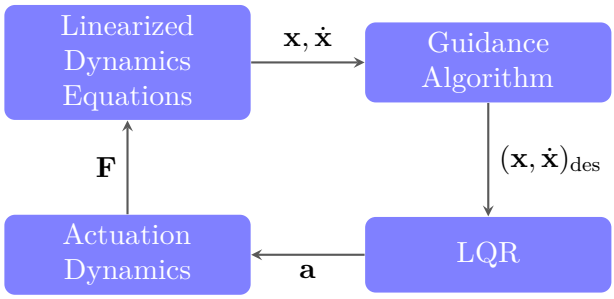


Figure 1.17: Closed-loop LQR guidance with online path generation

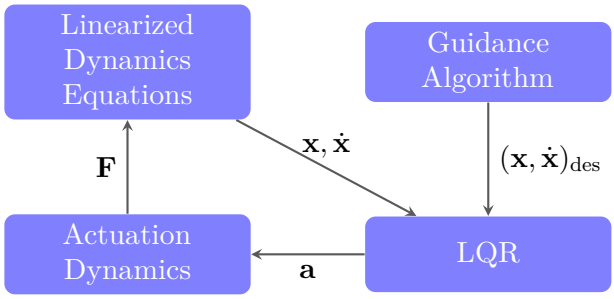


Figure 1.18: Closed-loop LQR guidance with offline path generation

This algorithm has been widely used in proximity operation scenarios such as distance control maneouvres in satellites constellations [40]

– *MPC-based guidance*
Model Predictive Control (MPC) formulates, at each sampling instant k, a finite-horizon optimal control problem using the linearized dynamics

$$\dot{\mathbf{x}} = A\,\mathbf{x} + B\,\mathbf{u},$$

and minimizes the quadratic cost

$$J_k = \tfrac{1}{2}\sum_{i=0}^{N-1}\Big[(\mathbf{x}_{k+i|k} - \mathbf{x}_{\text{ref}})^\top Q\,(\mathbf{x}_{k+i|k} - \mathbf{x}_{\text{ref}}) \;+\; \Delta\mathbf{u}_{k+i}^\top R\,\Delta\mathbf{u}_{k+i}\Big],$$

where $\Delta\mathbf{u}_{k+i} = \mathbf{u}_{k+i} - \mathbf{u}_{k+i-1}$, subject to actuator limits and LOS-cone constraints. The resulting Quadratic Program is solved online, the first control increment $\Delta\mathbf{u}_k$ is applied, then the horizon "recedes" and the procedure repeats with updated state measurements. This approach is widely used for autonomous rendezvous in elliptical orbits and for robust proximity operations under uncertainty such as rendezvous with a spinning orbital object [41].

- **Collision-Avoidance algorithms**

  – *Artificial Potential Field (APF)*
  Artificial Potential Field guidance treats the chaser and obstacles as charged particles moving under electrostatic-like forces. Since its introduction in the 1990s for mobile robots, APF has been adapted for spacecraft proximity operations due to its conceptual simplicity and capability for real-time path planning [24]. An attractive potential

$$U_a(\mathbf{x}) = \tfrac{1}{2}\,K_a\,\|\mathbf{x} - \mathbf{x}_{\text{goal}}\|^2$$

pulls the vehicle toward the target, while a repulsive potential

$$U_r(\mathbf{x}) = \begin{cases} \frac{1}{2}\,K_r\left(\frac{1}{\eta(\mathbf{x})} - \frac{1}{\eta_0}\right)^2, & \eta(\mathbf{x}) < \eta_0, \\ 0, & \eta(\mathbf{x}) \geq \eta_0, \end{cases} \qquad \eta(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_{\text{obs}}\|,$$

pushes it away from any convex obstacle within a safety radius η_0. The gradient of the total potential

$$\mathbf{F} = -\nabla\big(U_a + U_r\big)$$

yields a guidance vector which, when normalized and scaled by a designer-chosen maximum speed $\dot{x}_{\max}$, defines the commanded velocity direction and magnitude.

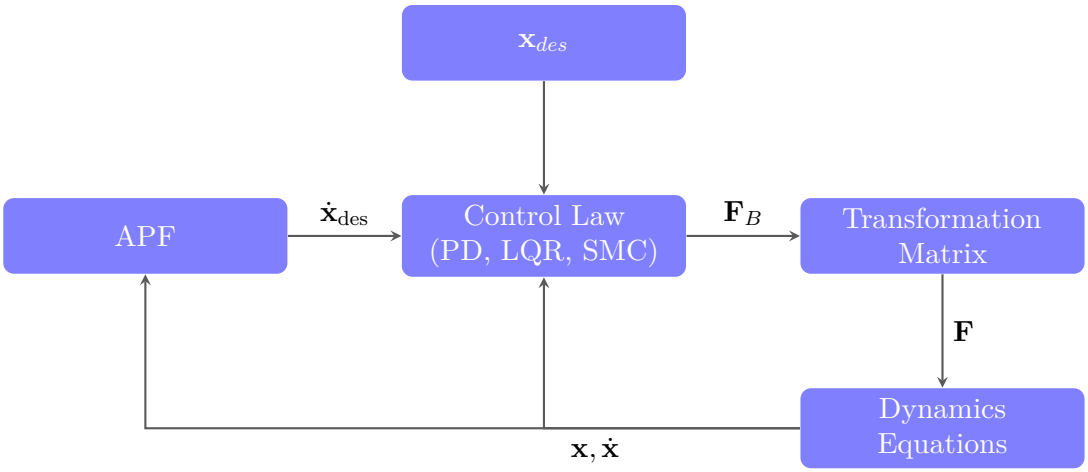


Figure 1.19: Closed-loop APF guidance with control law

APF offers online reactivity to moving obstacles and for this reason is widely used in LEO formation flying for collision avoidance [42].

Chapter 2

Foundamentals of astrodynamics

2.1 Overview of the laws

This chapter provides a concise overview of the essential theoretical concepts on which the physics of the problem analyzed in this thesis is based. The fundamental laws governing the motion of spacecraft and planets, as well as the primary elements used to define orbital parameters, will be described. Finally, concepts related to propulsion will be introduced in order to present the key characteristics of orbital manoeuvres. References [24], [25], and [43] were used to draft this chapter.

2.1.1 Universal Law of Gravitation

The foundation of celestial mechanics was laid by Isaac Newton, who formulated the law of universal gravitation. In its classical form, the law states that two point masses M and m, separated by a distance r, attract each other with a force whose magnitude is

$$F \;=\; G\,\frac{M\,m}{r^2} \tag{2.1}$$

where $G = 6.673 \times 10^{-11}\,\mathrm{m^3\,kg^{-1}\,s^{-2}}$ is the universal gravitational constant.

If we introduce the vector $\mathbf{r}$ to denote the displacement from mass M to mass m, then the gravitational force exerted on m by M can be written in vector form as

$$\mathbf{F} \;=\; -\,G\,\frac{M\,m}{|\mathbf{r}|^3}\,\mathbf{r}. \tag{2.2}$$

The negative sign indicates that the force is attractive, pulling m directly toward M (i.e., in the opposite direction of $\mathbf{r}$).

Although real celestial bodies are not perfect point masses, Newton's law remains valid for extended objects provided they are spherically symmetric or their separation is much larger than their own dimensions. In such cases, one can treat each body as if its entire mass were concentrated at its center when calculating the gravitational interaction.

2.1.2 The N-Body Problem

Given a spacecraft traveling in space, it is subjected at each instant to N different gravitational forces from N different bodies with distinct gravitational masses (see Figure 2.1). Other forces may act on the spacecraft—such as thrust, aerodynamic drag, or solar radiation pressure—but they are neglected in the following discussion.

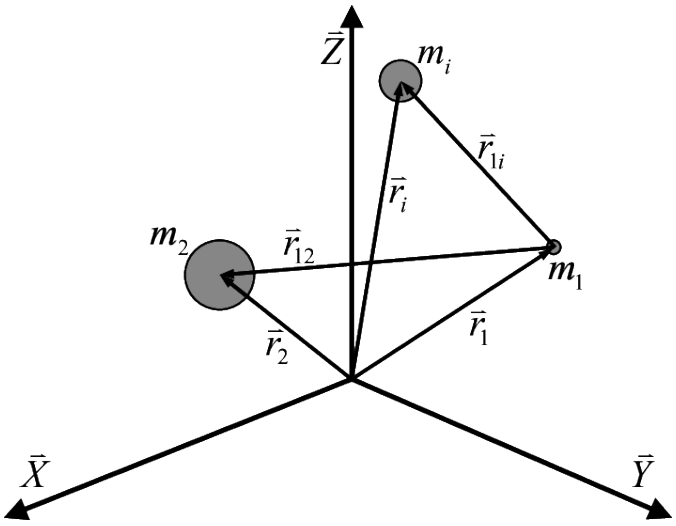


Figure 2.1: The N-Body problem [44]

Considering a system in Euclidean space of N bodies, $\{m_1, m_2, m_3, \ldots, m_N\}$, one of which is the body whose motion we wish to study (e.g., the spacecraft), the following assumptions are made:

- The bodies are spherically symmetric. This allows us to treat each body as though its entire mass were concentrated at its geometric center.
- The masses of all bodies are constant over time.
- There are no external or internal forces acting on the system other than the mutual gravitational forces, which act along the line joining the centers of the bodies.

Assume an inertial reference frame centered at an arbitrary point O. The gravitational force exerted on body i by body n can be written as

$$\mathbf{F}_{in} \;=\; -\,G\,\frac{m_n\,m_i}{r_{ni}^3}\,\mathbf{r}_{ni}, \quad n \neq i, \quad \mathbf{r}_{ni} \;=\; \mathbf{r}_i - \mathbf{r}_n. \tag{2.3}$$

Here, G is the universal gravitational constant, m_i and m_n are the masses of bodies i and n, respectively, and $\mathbf{r}_i$, $\mathbf{r}_n$ are their position vectors relative to O. By summing over all $n \neq i$, the total gravitational force on body i becomes

$$\mathbf{F}_i \;=\; \sum_{n=1,\,n\neq i}^{N} \mathbf{F}_{in} \;=\; -\sum_{\substack{n=1\\ n\neq i}}^{N} G\,\frac{m_n\,m_i}{r_{ni}^3}\,\mathbf{r}_{ni}. \tag{2.4}$$

Combining (2.4) with Newton's second law, $\mathbf{F}_i = m_i\,\ddot{\mathbf{r}}_i$, and introducing relative position vectors between bodies i and j, $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$, one obtains the following system of second-order differential equations:

$$\ddot{\mathbf{r}}_i \;=\; -\sum_{\substack{n=1\\ n\neq i}}^{N} G\,\frac{m_n}{r_{ni}^3}\,\mathbf{r}_{ni} \;=\; -\,G\,\frac{m_i + m_j}{r_{ij}^3}\,\mathbf{r}_{ij} \;-\; \sum_{\substack{n=1\\ n\neq i,\,n\neq j}}^{N} G\,m_n\!\left(\frac{\mathbf{r}_{nj}}{r_{nj}^3} \;-\; \frac{\mathbf{r}_{ni}}{r_{ni}^3}\right), \tag{2.5}$$

for each $i = 1,2,\ldots,N$. In (2.5), the first term represents the direct gravitational interaction between bodies i and j; the summation that follows accounts for perturbations due to all remaining bodies in the system. Because there are N bodies, one obtains N coupled second-order differential equations (or equivalently $2N$ first-order equations). In general, this system cannot be solved analytically and must be integrated numerically.

2.1.3 The Two-Body Problem

When the full N-body system is reduced to only two bodies say masses m_i and m_j all indices $n \neq i$, $n \neq j$ vanish, so that the perturbation sum in (2.5) disappears. The equation of motion for the relative vector $\mathbf{r}_{ij}$ then simplifies to

$$\ddot{\mathbf{r}}_{ij} \;=\; -\,G\,\frac{m_i + m_j}{r_{ij}^3}\,\mathbf{r}_{ij}. \tag{2.6}$$

If one body (say m_j) is much more massive than the other ($m_j \gg m_i$), one obtains the *restricted two-body problem*. In that case, the smaller mass m_i exerts a negligible force on m_j, and the origin of coordinates may be taken at the center of the larger mass. Defining $M = m_j$ and $m = m_i$, the relative distance $\mathbf{r} = \mathbf{r}_i - \mathbf{r}_j$ becomes

equivalent to the absolute position of the small body relative to the large one. Equation (2.6) reduces to

$$\ddot{\mathbf{r}} \;=\; -\,G\,\frac{(M+m)}{r^3}\,\mathbf{r}. \tag{2.7}$$

Finally, because $M \gg m$, one may neglect m in the numerator to arrive at the familiar form

$$\ddot{\mathbf{r}} \;=\; -\,G\,\frac{M}{r^3}\,\mathbf{r}. \tag{2.8}$$

It is customary to define the standard gravitational parameter μ of the larger body as

$$\mu \;=\; G\,M,$$

so that (2.8) can also be written as

$$\ddot{\mathbf{r}} \;=\; -\,\mu\,\frac{\mathbf{r}}{r^3}.$$

These equations form the basis of classical celestial mechanics for a spacecraft (or planet) moving under the gravitational attraction of a single dominant body.

2.1.4 Constants of the motion

The gravitational field is conservative and radial, so an object moving under the influence of gravity alone does not lose or gain mechanical energy but only exchanges one form of energy (kinetic) for another form called potential energy. Since the gravitational field is radial, the angular momentum of the satellite about the center of the reference frame (the large mass) remains constant.

Conservation of Mechanical Energy

The energy constant of motion can be derived by taking the scalar product between the equation of motion and the velocity $\dot{\mathbf{r}}$:

$$\dot{\mathbf{r}} \cdot \ddot{\mathbf{r}} \;+\; \frac{\mu}{r^3}\,\mathbf{r} \cdot \dot{\mathbf{r}} \;=\; 0 \tag{2.9}$$

From (2.9), one obtains

$$\frac{d}{dt}\left(\frac{V^2}{2} \;-\; \frac{\mu}{r}\right) \;=\; 0, \tag{2.10}$$

where $V = \|\dot{\mathbf{r}}\|$ is the speed and $\mu = G\,M$ is the standard gravitational parameter. Equation (2.10) implies that

$$\mathcal{E} \;=\; \frac{V^2}{2} \;-\; \frac{\mu}{r}$$

is constant in time. By choosing the integration constant of the potential energy to be zero at $r \to \infty$, the total specific mechanical energy $\mathcal{E}$ is defined as above.

Conservation of Angular Momentum

The angular-momentum constant of motion results from taking the cross product of the equation of motion with $\mathbf{r}$:

$$\mathbf{r} \times \ddot{\mathbf{r}} \; + \; \frac{\mu}{r^3}\, \mathbf{r} \times \mathbf{r} \; = \; 0. \tag{2.11}$$

Since $\mathbf{r} \times \mathbf{r} = \mathbf{0}$, the second term in (2.11) vanishes. Applying the product-rule form of differentiation to the first term yields

$$\frac{d}{dt}\Big(\mathbf{r} \times \dot{\mathbf{r}}\Big) \; = \; 0. \tag{2.12}$$

Define the specific angular momentum vector as

$$\mathbf{h} \; = \; \mathbf{r} \times \dot{\mathbf{r}}.$$

Because $\mathbf{h}$ is constant in time, $\mathbf{r}$ and $\dot{\mathbf{r}}$ must always lie in the same plane perpendicular to $\mathbf{h}$. Consequently, the motion of the satellite remains confined to a fixed plane in space, commonly called the *orbital plane*.

2.1.5 Trajectory Equation

It is possible to obtain a partial solution that determines the size and shape of the orbit. A fully explicit, time-dependent solution requires a double integration and is therefore more difficult to derive. By taking the cross product of the equation of motion

$$\ddot{\mathbf{r}} \; = \; -\mu\, \frac{\mathbf{r}}{r^3}$$

with the specific angular-momentum vector $\mathbf{h} = \mathbf{r} \times \dot{\mathbf{r}}$, one arrives at

$$\ddot{\mathbf{r}} \times \mathbf{h} \; = \; \frac{\mu}{r^3}\,(\mathbf{r} \times \mathbf{h}) \; = \; \frac{\mu}{r^3}\,\Big(\mathbf{r} \times (\mathbf{r} \times \dot{\mathbf{r}})\Big). \tag{2.13}$$

Since $\mathbf{r} \times (\mathbf{r} \times \dot{\mathbf{r}}) = (\mathbf{r} \cdot \dot{\mathbf{r}})\, \mathbf{r} - r^2\, \dot{\mathbf{r}}$, equation (2.13) can be shown after a few algebraic manipulations to become

$$\frac{d}{dt}\Big(\dot{\mathbf{r}} \times \mathbf{h}\Big) \; = \; \frac{d}{dt}\Big(\mu\, \frac{\mathbf{r}}{r}\Big). \tag{2.14}$$

Integrating both sides of (2.14) with respect to time introduces a constant (vector) of integration $\mathbf{B}$. Hence,

$$\dot{\mathbf{r}} \times \mathbf{h} \; = \; \mu\, \frac{\mathbf{r}}{r} \; + \; \mathbf{B}. \tag{2.15}$$

Taking the scalar (dot) product of $\mathbf{r}$ with both sides of (2.15) gives

$$\mathbf{r} \cdot \Big(\dot{\mathbf{r}} \times \mathbf{h}\Big) \; = \; \mu\, \frac{\mathbf{r} \cdot \mathbf{r}}{r} \; + \; \mathbf{r} \cdot \mathbf{B}.$$

But $\mathbf{r} \cdot (\dot{\mathbf{r}} \times \mathbf{h}) = 0$ since $\mathbf{r}$, $\dot{\mathbf{r}}$, and $\mathbf{h} = \mathbf{r} \times \dot{\mathbf{r}}$ are mutually perpendicular. Therefore,

$$0 \;=\; \mu\, r \;+\; \mathbf{r} \cdot \mathbf{B} \;\;\Longrightarrow\;\; \mathbf{r} \cdot \mathbf{B} \;=\; -\,\mu\, r.$$

Define the angle ν between the vectors $\mathbf{B}$ and $\mathbf{r}$. Then

$$\mathbf{r} \cdot \mathbf{B} \;=\; r\, B\, \cos\nu \;=\; -\,\mu\, r \quad\Longrightarrow\quad B\, \cos\nu \;=\; -\,\mu.$$

Since $B = \|\mathbf{B}\|$, we may write

$$\frac{B}{\mu}\, \cos\nu \;=\; -\,1. \tag{2.16}$$

In practice, one chooses the direction of $\mathbf{B}$ so that ν is measured from pericenter (where r is minimum), yielding

$$r \;=\; \frac{h^2/\mu}{1 + (B/\mu)\, \cos\nu}.$$

Equation (2.1.5) is the trajectory equation expressed in polar coordinates, where

$$r \;=\; \frac{\dfrac{h^2}{\mu}}{1 + \frac{B}{\mu}\, \cos\nu}.$$

Here, ν is the true anomaly (the angle between $\mathbf{B}$ and $\mathbf{r}$), and B/μ is identified as the eccentricity e. When $\nu = 0$, r attains its minimum value; when $\nu = \pi$, r is maximum. To determine the type of conic section represented by (2.1.5), compare it with the general polar-coordinate equation of a conic:

$$r \;=\; \frac{p}{1 + e\, \cos\nu}, \tag{2.17}$$

where p is the semilatus rectum and e is the eccentricity. By identifying

$$p \;=\; \frac{h^2}{\mu}, \qquad e \;=\; \frac{B}{\mu},$$

one sees that the orbit is elliptical if $0 \le e < 1$, parabolic if $e = 1$, and hyperbolic if $e > 1$. Equation (2.17) therefore confirms that the trajectory of a two-body system under a $1/r^2$ force law is a conic section.

Building on the trajectory equation derived above, we note that

$$r \;=\; \frac{p}{1 + e\cos\nu}, \qquad p \;=\; \frac{h^2}{\mu}, \quad e = \frac{B}{\mu},$$

where p is the semilatus rectum and e is the eccentricity. From this form and the preceding analysis, one can draw the following key conclusions:

1. The locus of possible trajectories under a central $1/r^2$ force is confined to the family of conic sections (circle, ellipse, parabola, hyperbola), as determined by the value of e.
2. The focus of each conic section corresponds to the center of attraction (i.e., the central mass), confirming that all two-body orbits share the same focal geometry.
3. The specific mechanical energy

$$\mathcal{E} = \frac{V^2}{2} - \frac{\mu}{r}$$

   remains constant along the orbit, implying that a satellite does not gain or lose total energy as it moves on its conic trajectory.
4. The motion of the satellite is confined to a fixed plane in inertial space (the orbital plane), since the specific angular momentum vector $\mathbf{h} = \mathbf{r} \times \dot{\mathbf{r}}$ is constant in both magnitude and direction.
5. The magnitude of the specific angular momentum

$$h = \|\mathbf{r} \times \dot{\mathbf{r}}\|$$

   is invariant, reinforcing that the areal velocity is constant (Kepler's second law).

2.1.6 Types of orbits

The term *conic* originates from the fact that each of these orbital paths can be defined as the intersection of a plane and a right circular cone. Every conic orbit possesses two foci, which determine its geometric properties:

- *Circle:* both foci coincide at the center of the circle.
- *Ellipse:* the two foci lie on the major axis, separated by twice the focal distance.
- *Parabola:* one focus lies at a finite distance, while the other effectively resides at infinity; it represents the boundary between bound (elliptical) and unbound (hyperbolic) trajectories.
- *Hyperbola:* the two foci lie on the transverse axis, beyond the vertices, indicating an unbound trajectory.

A parabolic orbit corresponds to exactly the minimum specific energy required for a spacecraft to escape the central body's gravitational influence.

For any conic section, the following geometric parameters are defined:

- *Semilatus rectum* p: the distance from the focus to the curve measured along a line perpendicular to the major axis (i.e., half the width of the conic at the focus).
- *Semi-major axis* a: half the length of the chord passing through both foci.
- *Focal distance* c: half the distance between the two foci.
- *Semi-minor axis* b: half the length of the chord passing through the center of the conic and perpendicular to the major axis.

These definitions provide the basis for distinguishing between different orbital shapes:

$$e \;=\; \begin{cases} 0, & \text{circle,} \\ 0 < e < 1, & \text{ellipse,} \\ e = 1, & \text{parabola,} \\ e > 1, & \text{hyperbola,} \end{cases}$$

where $e = \dfrac{c}{a}$ is the eccentricity.

Building on the geometric definitions of a conic orbit, one can derive additional relations among the orbital elements. Recall that the eccentricity is defined by

$$e \;=\; \frac{c}{a} \;=\; \sqrt{1 \;+\; \frac{2\,\mathcal{E}\,h^2}{\mu^2}}\,,$$

where $\mathcal{E}$ is the specific mechanical energy, h is the specific angular momentum, and μ is the gravitational parameter of the central body. The semilatus rectum p is related to a and e by

$$p \;=\; a\left(1 - e^2\right).$$

From these definitions, the periapsis distance $r_{\min}$ and apoapsis distance $r_{\max}$ can be written as

$$r_{\min} \;=\; \frac{p}{1+e} \;=\; a\,(1-e), \qquad r_{\max} \;=\; \frac{p}{1-e} \;=\; a\,(1+e).$$

Elliptical Orbit An ellipse ($0 \leq e < 1$) is a closed orbit with constant period T. For any point on an ellipse, the sum of distances to the two foci is constant:

$$r + r' \;=\; 2a,$$

where r and r' are the instantaneous distances from each focus. Denoting r_a and r_p as the apocenter and pericenter distances, respectively, one finds

$$r_a - r_p \;=\; 2c, \qquad e \;=\; \frac{2c}{2a} \;=\; \frac{r_a - r_p}{r_a + r_p}.$$

The orbital period of an ellipse follows Kepler's third law:

$$T \;=\; 2\pi \sqrt{\frac{a^3}{\mu}}\,.$$

Circular Orbit A circle is the special case $e = 0$, for which both foci coincide at the center. Hence, r is constant and equal to the semi–major axis a. The circular speed V_c at radius r_c satisfies

$$V_c \;=\; \sqrt{\frac{\mu}{r_c}}\,,$$

and the period reduces to

$$T_c \;=\; 2\pi \sqrt{\frac{r_c^3}{\mu}}\,.$$

Parabolic Orbit A parabola has $e = 1$ and represents the exact boundary between bound and unbound motion. In this case, one focus is effectively at infinity, and the semilatus rectum satisfies

$$p \;=\; 2\,r_p,$$

where r_p is the periapsis distance. The specific mechanical energy for a parabolic trajectory is zero, and the periapsis speed (escape velocity at r_p) is

$$V_p \;=\; \sqrt{\frac{2\,\mu}{r_p}}\,.$$

Since a parabolic trajectory is an open path, it does not have a well-defined orbital period.

Hyperbolic Orbit For $e > 1$, the trajectory is a hyperbola, also open, with periapsis distance

$$r_p \;=\; \frac{p}{1+e},$$

and the excess velocity at infinity V_∞ satisfies

$$\mathcal{E} \;=\; \frac{V_\infty^2}{2} \;>\; 0.$$

The hyperbolic excess speed is related to a by

$$a \;=\; -\,\frac{\mu}{2\,\mathcal{E}} \quad \text{(note that } a < 0\text{)},$$

and the asymptotic turning angle θ_∞ (measured from periapsis) obeys

$$\cos\theta_\infty \;=\; -\frac{1}{e}\,.$$

This concludes the summary of conic orbits and their key geometric and dynamical parameters.

2.1.7 Coordinate Systems

The first requirement for describing an orbit is to choose a suitable inertial reference frame. In the case of orbits around the Sun (e.g., planets, asteroids, comets, and deep-space probes), the heliocentric-ecliptic coordinate system is convenient. For satellites of the Earth, one typically uses the geocentric-equatorial system. To define any reference frame, we need to specify:

- The origin of the reference frame;
- The orientation of the fundamental plane (the X–Y plane);
- The principal direction (i.e., the direction of the X axis);
- The direction of the Z axis, which is perpendicular to the fundamental plane.

The Y axis is always chosen so that (X, Y, Z) form a right-handed set of coordinate axes.

The Heliocentric-Ecliptic Coordinate System

In the heliocentric-ecliptic system, the origin is at the center of the Sun. The fundamental plane coincides with the ecliptic, which is the plane of the Earth's revolution around the Sun. The line of intersection between the ecliptic plane and the Earth's equatorial plane defines the direction of the vernal equinox, commonly

denoted by $\hat{\boldsymbol{\gamma}}$. On the first day of spring (the March equinox), the line joining the center of the Earth and the center of the Sun points along the positive X–axis (i.e., toward $\hat{\boldsymbol{\gamma}}$).

Because the Earth's rotation axis precesses by roughly 50 arcseconds per year, the heliocentric-ecliptic frame is not strictly inertial over very long timescales. However, for typical mission durations of two to three years, this precession has a negligible effect, so the heliocentric-ecliptic frame can be treated as effectively inertial.

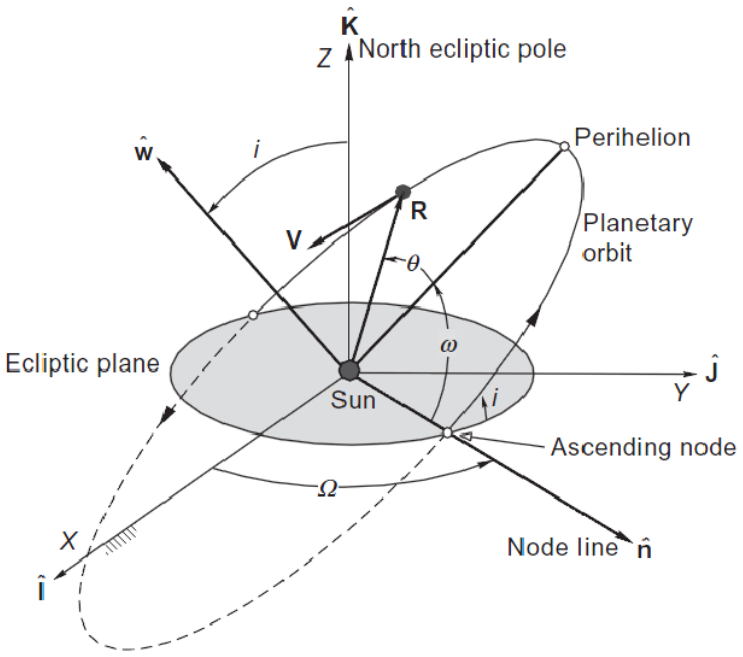


Figure 2.2: Heliocentric-ecliptic coordinate system [45]

The Geocentric-Equatorial Coordinate System

The geocentric-equatorial system has its origin at the center of the Earth. The fundamental plane is the Earth's equatorial plane, and the positive X–axis points toward the vernal equinox direction. The Z–axis points toward the north celestial pole, while the Y–axis completes a right-handed set of axes, lying in the equatorial plane 90° east of the X–axis.

Because this frame is fixed relative to the stars, it rotates with the Earth. In other words, as the Earth spins, the geocentric-equatorial coordinate frame also rotates in inertial space. Consequently, positions of satellites expressed in this frame must account for Earth's rotation when converting to inertial or Earth-fixed longitude and latitude.

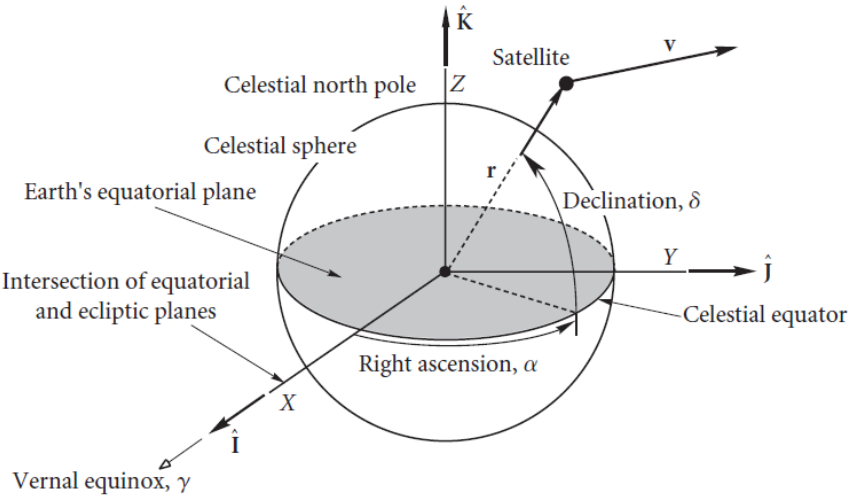


Figure 2.3: Geocentric-equatorial coordinate system [43]

The Perifocal Coordinate System

The perifocal frame is particularly useful for describing the instantaneous motion of a satellite in its orbital plane. Its origin is located at the focus of the Keplerian orbit (i.e., the center of attraction, such as the Earth's center for an Earth-orbiting satellite). The fundamental (reference) plane of the perifocal frame is the satellite's orbital plane.

- The p–axis ($\hat{\mathbf{p}}$) points from the focus toward the periapsis (closest approach) of the orbit.
- The q–axis ($\hat{\mathbf{q}}$) lies in the orbital plane, perpendicular to $\hat{\mathbf{p}}$, in the direction of motion at periapsis (i.e., 90° ahead of the periapsis direction).
- The w–axis ($\hat{\mathbf{w}}$) is perpendicular to the orbital plane and completes the right-handed triad ($\hat{\mathbf{p}}$, $\hat{\mathbf{q}}$, $\hat{\mathbf{w}}$). In fact, $\hat{\mathbf{w}} = \hat{\mathbf{p}} \times \hat{\mathbf{q}}$.

In this frame, the satellite's position vector $\mathbf{r}$ lies in the p–q plane and can be written as

$$\mathbf{r} = r\,\cos\nu\,\hat{\mathbf{p}} \;+\; r\,\sin\nu\,\hat{\mathbf{q}},$$

where r is the orbital radius and ν is the true anomaly measured from periapsis. The velocity vector $\dot{\mathbf{r}}$ also lies in the orbital plane and may be expressed in the p–q basis as

$$\dot{\mathbf{r}} = \dot{r}\,\hat{\mathbf{p}} \;+\; r\,\dot{\nu}\,\hat{\mathbf{q}},$$

with $\dot{r} = \frac{\mu}{h}\,e\,\sin\nu$ and $r\,\dot{\nu} = \frac{h}{r}$, where h is the specific angular momentum, μ is the gravitational parameter, and e is the eccentricity.

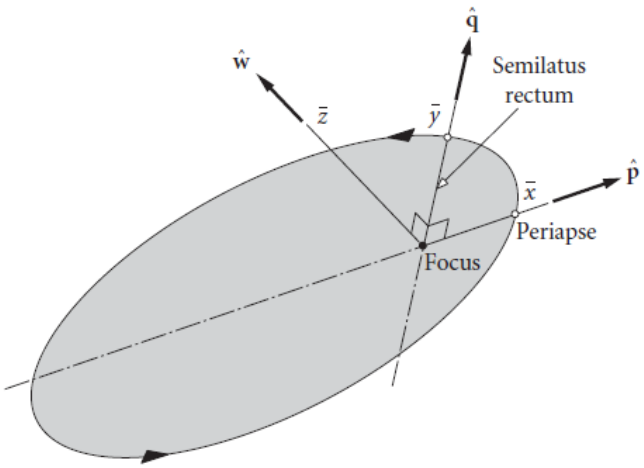


Figure 2.4: Perifocal frame [43]

2.1.8 Classical Orbital Parameters

In astrodynamics, a set of five independent quantities called *orbital parameters* is sufficient to specify the size, shape, and orientation of a Keplerian orbit. To indicate the location of a satellite along its orbit at a particular epoch, a sixth parameter is required. The classical set of six orbital elements (see Figure 2.5) is defined as follows:

- *Semi-major axis, a*: a constant that determines the overall size of the conic section.
- *Eccentricity, e*: a constant that defines the shape of the conic (circle, ellipse, parabola, or hyperbola).
- *Inclination, i*: the angle between the orbital plane's angular momentum vector $\mathbf{h}$ and the reference Z-axis.
- *Longitude of the ascending node, Ω*: the angle measured in the reference plane (from the reference direction $\hat{\mathbf{I}}$) to the point where the satellite crosses that plane moving northward (the ascending node).
- *Argument of periapsis, ω*: the angle in the orbital plane between the ascending node and the orbit's periapsis, measured in the direction of motion.
- *True anomaly, $\nu(t)$*: the angle between the orbital radius vector $\mathbf{r}(t)$ and the periapsis direction, measured in the direction of motion.

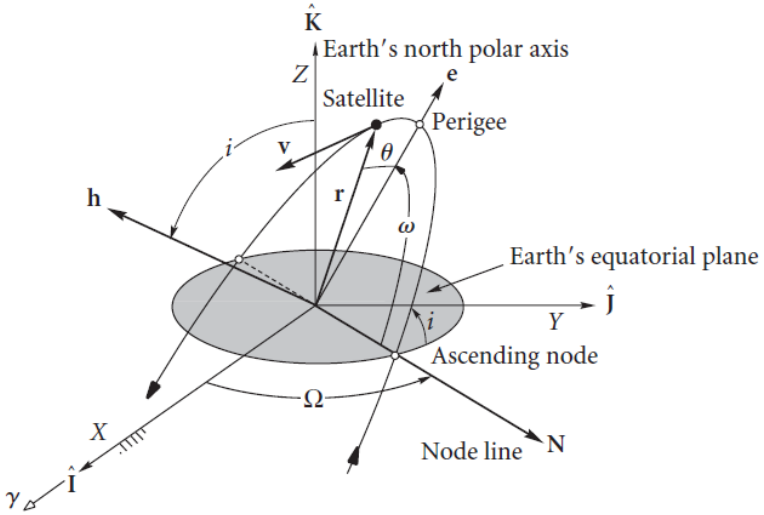


Figure 2.5: Classical orbital elements [43]

The six classical orbital elements defined above are not exhaustive; in certain special cases one or more elements become indeterminate. In those situations, alternate angular parameters are introduced:

- If the inclination $i = 0$, then Ω and ω cannot be determined. Instead, one defines the *longitude of periapsis*, Π, measured from the reference direction $\hat{\mathbf{I}}$ to periapsis.
- If the eccentricity $e = 0$, then ω and ν become undefined. In this case, one uses the *argument of latitude*, θ, which is the angle from the ascending node to the satellite's current position measured in the direction of motion.
- If both $e = 0$ and $i = 0$, then Ω, ω, and ν are all indeterminate. An alternate parameter, the *true longitude at epoch* ℓ, is used to specify the satellite's position along the circular, equatorial orbit.

2.1.9 Orbital elements determination from r and v

If a ground station on Earth provides the position vector $\mathbf{r}$ and velocity vector $\mathbf{v}$ of a satellite in the geocentric-equatorial frame at a given instant, one can compute the eccentricity vector via:

$$\frac{\mathbf{v} \times \mathbf{h}}{\mu} - \frac{\mathbf{r}}{r} = \mathbf{e}, \tag{2.18}$$

where $\mathbf{h} = \mathbf{r} \times \mathbf{v}$ is the specific angular momentum, $r = \|\mathbf{r}\|$, and μ is the standard gravitational parameter of Earth. Once $\mathbf{e}$ is known, its magnitude $e = \|\mathbf{e}\|$ gives the

orbit's eccentricity. From **r** and **v** one also obtains **h** and the specific mechanical energy $\mathcal{E} = \frac{v^2}{2} - \frac{\mu}{r}$, from which the semi-major axis a and semilatus rectum p can be derived in the usual way.

The remaining orientation angles follow from the following relations, where **I**, **J**, and **K** are the unit axes of the geocentric-equatorial frame and **P** is the unit vector along periapsis (i.e., $\mathbf{e}/e$):

$$\Omega \;=\; \arccos\!\Big(\frac{\mathbf{I}\cdot(\mathbf{K}\times\mathbf{h})}{\|\mathbf{K}\times\mathbf{h}\|}\Big), \qquad \omega \;=\; \arccos\!\Big(\frac{(\mathbf{K}\times\mathbf{h})\cdot\mathbf{P}}{\|\mathbf{K}\times\mathbf{h}\|}\Big),$$
$$i \;=\; \arccos\!\Big(\frac{\mathbf{K}\cdot\mathbf{h}}{\|\mathbf{h}\|}\Big), \qquad \nu \;=\; \arccos\!\Big(\frac{\mathbf{r}\cdot\mathbf{P}}{r}\Big).$$

2.1.10 Determining r and v from orbital elements

Assume that all six orbital elements are specified and we wish to compute the satellite's position and velocity. In the perifocal frame, the position vector can be written as:

$$\mathbf{r} \;=\; r\,\cos\nu\,\hat{\mathbf{p}} \;+\; r\,\sin\nu\,\hat{\mathbf{q}}, \tag{2.19}$$

where the magnitude of r is given by:

$$r \;=\; \frac{p}{1+e\,\cos\nu}, \tag{2.20}$$

with $p = a\,(1-e^2)$.

To find the velocity vector, differentiate **r** with respect to time, treating the perifocal axes as fixed:

$$\dot{\mathbf{r}} = \mathbf{v} \;=\; \Big(\dot{r}\,\cos\nu - r\,\dot{\nu}\,\sin\nu\Big)\,\hat{\mathbf{p}} \;+\; \Big(\dot{r}\,\sin\nu + r\,\dot{\nu}\,\cos\nu\Big)\,\hat{\mathbf{q}}. \tag{2.21}$$

Using the standard relations

$$\dot{r} = \sqrt{\frac{\mu}{p}}\,e\,\sin\nu, \qquad r\,\dot{\nu} = \sqrt{\frac{\mu}{p}}\,\Big(1+e\,\cos\nu\Big),$$

one obtains the compact form:

$$\mathbf{v} \;=\; \sqrt{\frac{\mu}{p}}\,\Big[-\sin\nu\,\hat{\mathbf{p}} \;+\; \Big(e+\cos\nu\Big)\,\hat{\mathbf{q}}\Big]. \tag{2.22}$$

2.2 External disturbances

2.2.1 Atmospheric Drag

Drag is caused by the residual atmosphere in the LEO orbits, where the density of the air is lower than on the ground and the continuum model of fluid mechanics doesn't apply because the interaction is on a molecular level. Four assumptions are made:

- The momentum of molecules arriving at the surface of the spacecraft is totally lost to the surface;
- The thermal motion of the atmosphere is much smaller than the spacecraft speed;
- The spacecraft is nominally non-spinning.

The force is calculated as:

$$\mathbf{F} = \frac{\rho}{2} V^2 S C_D \tag{2.23}$$

ρ was obtained from the JB-2006 model of atmospheric density based on altitude:

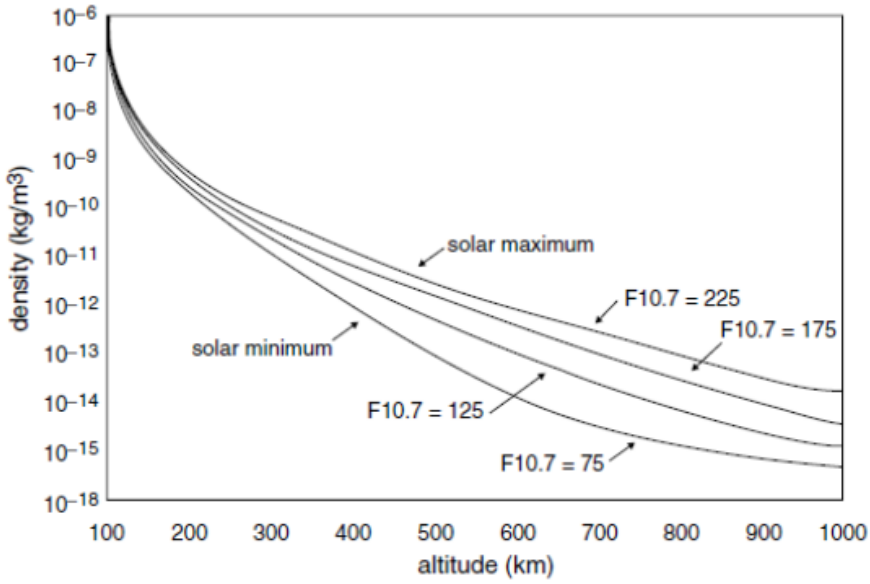


Figure 2.6: JB-2006 model [24]

2.2.2 J_2 Effect

The term J_2 comes from an infinite-series mathematical equation that describes the perturbational effects of oblateness on the gravity of a planet. The coefficients

of each term in this series are denoted J_k, of which J_2, J_3, and J_4 are called "zonal coefficients." However, J_2 is over 1000 times larger than the rest and has the strongest perturbing influence on orbits.

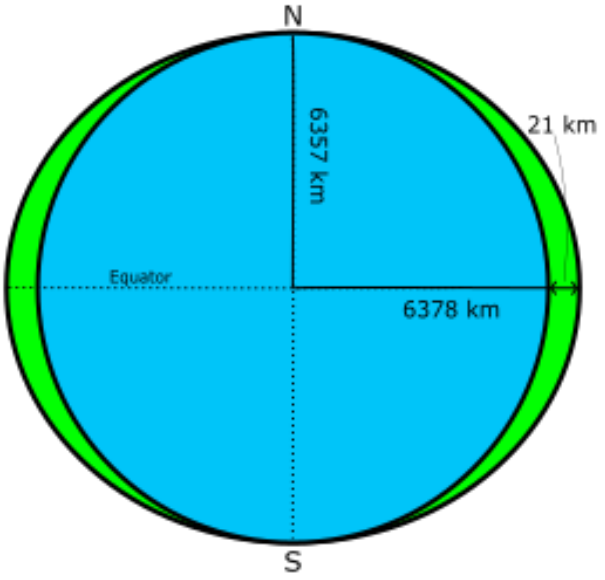


Figure 2.7: Earth's oblateness [24]

It is possible to define the force applied to the spacecraft by this effect as:

$$\mathbf{F}_{J_2} = -\, m_c \, \frac{3\, J_2\, \mu\, R^2}{2\, r^4} \begin{bmatrix} 1 - 3\, \sin^2(i)\, \sin(\nu) \\ 2\, \sin^2(i)\, \sin(\nu)\, \cos(\nu) \\ 2\, \sin(i)\, \cos(i)\, \sin(\nu) \end{bmatrix} \tag{2.24}$$

where:

- i is the orbital inclination,
- ν is the true anomaly,
- r is the orbital radius,
- R is the mean radius of the Earth, and
- $J_2 = 1.08263 \times 10^{-3}$.

2.2.3 Gravity Gradient

The gravity-gradient torque is caused by Earth's gravity force, which decreases quadratically with distance from the Earth's center. In fact, the gravitational

force on a mass farther from the Earth is smaller than the force on a mass that is closer. This difference in gravitational force produces a torque on the spacecraft. In particular, four assumptions were made:

- Only one celestial primary body is considered;
- The celestial body possesses a spherically symmetrical mass distribution;
- The spacecraft is small compared to its distance from the mass center;
- The spacecraft is a single rigid body.

After these assumptions, we obtain:

$$\tilde{\mathbf{T}} \;=\; 3\,\omega^2 \begin{bmatrix} \left(I_{zz} - I_{yy}\right)\phi \\ \left(I_{zz} - I_{xx}\right)\theta \\ \left(I_{xx} - I_{yy}\right) r^2\,\phi\,\theta \end{bmatrix} \tag{2.25}$$

where:

- ω is the orbital rate,
- I_{xx}, I_{yy}, I_{zz} are the principal moments of inertia of the spacecraft,
- ϕ and θ are the small angular displacements about the x and y axes, respectively,
- r is the orbital radius.

2.2.4 Magnetic Torque

The disturbance related to the magnetic field of the Earth acts purely as a torque and is caused by the interaction of the residual dipole of the spacecraft with Earth's magnetic field. Usually, it is considered constant, but more accurate models can be used. The torque can be written as:

$$\mathbf{T}_m \;=\; \mathbf{M} \times \mathbf{B} \tag{2.26}$$

where $\mathbf{M}$ is the residual magnetic dipole of the satellite and $\mathbf{B}$ is the local geomagnetic field.

2.2.5 Solar Radiation

The disturbance related to solar radiation can be considered both as a disturbance force (secondary effect) and as a disturbance torque, which is the primary effect.

In particular, the force magnitude can be written as:

$$F_s \;=\; \Big(1+K\Big)\, P_s\, S \tag{2.27}$$

where:

- S is the frontal area of the spacecraft exposed to the Sun;
- K is the reflectivity index ($K = 1$ in sunlight, $K = 0$ in eclipse);
- $P_s = \dfrac{I_s}{c}$ is the solar radiation pressure constant, with $I_s = 1367$ W/m^2 (solar irradiance at 1 AU) and $c = 3 \times 10^8$ m/s (speed of light).

In this thesis, the force direction is taken along the line between the Sun and the satellite (referred to as the Sun vector), pointing away from the Sun. The resulting torque about the spacecraft's center of mass is given by:

$$\mathbf{T}_s \;=\; \mathbf{r}_s \times \mathbf{F}_s \tag{2.28}$$

where $\mathbf{r}_s$ is the vector from the spacecraft's center of mass to its optical center of pressure, and $\mathbf{F}_s$ is the force vector of magnitude F_s acting in the Sun line direction.

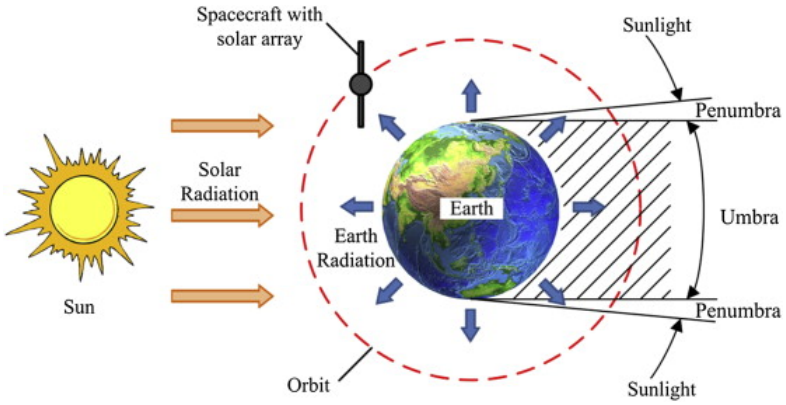


Figure 2.8: Solar radiation [24]

2.3 Orbital manoeuvres

Orbital manoeuvres are performed to change one or more orbital parameters. During manoeuvres, propulsive forces are applied to the spacecraft, implying a change in vehicle mass with the ejection of a mass $\dot{m}_p$. In this section, some impulsive manoeuvres will be reported such as the classic fuel efficient Hohmann transfer and the bi-elliptic version, the phasing manoeuvre and finally the orbital plane change.

2.3.1 Impulsive manoeuvres

During an impulsive manoeuvre, the position of the spacecraft is considered to be fixed; only the velocity changes. The impulsive manoeuvre is an idealization by means of which we can avoid having to solve the equations of motion (Eq. 2.6) with the rocket thrust included. The idealization is satisfactory for those cases in which the position of the spacecraft changes only slightly during the time that the manoeuvering rockets fire. This is true for high-thrust rockets with burn times short compared with the coasting time of the vehicle.

Each impulsive manoeuvre results in a change $\Delta\mathbf{V}$ in the velocity of the spacecraft. $\Delta\mathbf{V}$ can represent a change in the magnitude or the direction of the velocity vector, or both. The magnitude ΔV of the velocity increment is related to Δm, the mass of propellant consumed, by the formula

$$\frac{\Delta m}{m} = 1 - e^{-\frac{\Delta v}{I_{sp} g_0}} \tag{2.29}$$

where m is the mass of the spacecraft before the burn, g_o is the sea-level standard acceleration of gravity, and I_{sp} is the specific impulse of the propellants. Specific impulse is defined as follows:

$$I_{sp} = \frac{\text{thrust}}{\text{sea-level weight rate of fuel consumption}}$$

Specific impulse has units of seconds, and it is a measure of the performance of a rocket propulsion system. I_{sp} for some common propellant combinations are shown in Table 2.1. Figure 2.9 is a graph of Equation 2.29 for a range of specific impulses. Note that for ΔV s on the order of 1 km/s or higher, the required propellant exceeds 25 percent of the spacecraft mass prior to the burn.

There are no refueling stations in space, so a mission's delta-v schedule must be carefully planned to minimize the propellant mass carried aloft in favor of payload.

Propellant	I_{sp} (s)
Cold gas	50
Monopropellant hydrazine	230
Solid propellant	290
Nitric acid / monomethylhydrazine	310
Liquid oxygen / liquid hydrogen	455

Table 2.1: Specific impulse of various propellant types [43]

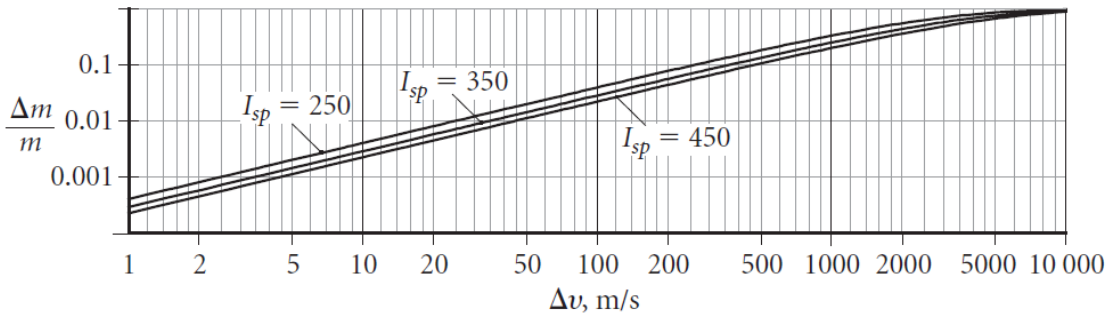


Figure 2.9: Propellant mass fraction versus ΔV for typical specific impulses [43]

Hohmann Transfer

The Hohmann transfer [46], is the minimum-energy, two-impulse manoeuvre for transferring a spacecraft between two coplanar circular orbits sharing a common center. The transfer trajectory is a Keplerian ellipse whose periapsis and apoapsis coincide with the radii of the inner and outer circles, respectively. Only one half of this ellipse is traversed either from the lower to the higher orbit or vice versa by applying two instantaneous velocity changes.

The specific orbital energy of any Keplerian orbit depends solely on its semi-major axis a. For an ellipse, the specific energy ε is given by

$$\varepsilon \;=\; -\frac{\mu}{2a}\,, \tag{2.30}$$

where μ is the gravitational parameter of the central body. A larger semi-major axis corresponds to a less negative (i.e. higher) energy.

Starting on the inner circular orbit at point A, an impulsive increment ΔV_A is applied in the direction of motion to place the vehicle onto the transfer ellipse. After coasting along the ellipse from A to the opposite apsis B, a second impulse ΔV_B circularizes the trajectory onto the outer orbit. Without the second burn, the vehicle would return along the same ellipse back to A. The total propellant cost is quantified by the sum

$$\Delta V_{\text{total}} \;=\; \Delta V_A + \Delta V_B\,. \tag{2.31}$$

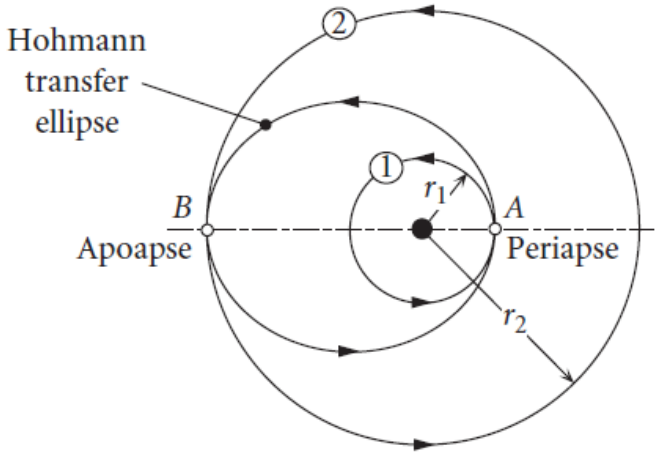


Figure 2.10: Hohmann transfer [43]

The same total ΔV is required if the transfer begins at point B on the outer circular orbit. Since lowering the spacecraft's energy to reach the inner, lower energy circular orbit must be accomplished by retrograde burns, the required ΔV must be provided by retrofires. In a retrofire manoeuvre, the rocket thrust is applied opposite to the velocity vector so as to brake the spacecraft. Because each ΔV corresponds to the same propellant expenditure regardless of thrust orientation, when summing the individual ΔV values we need consider only their magnitudes.

Bi-elliptic Hohmann Transfer

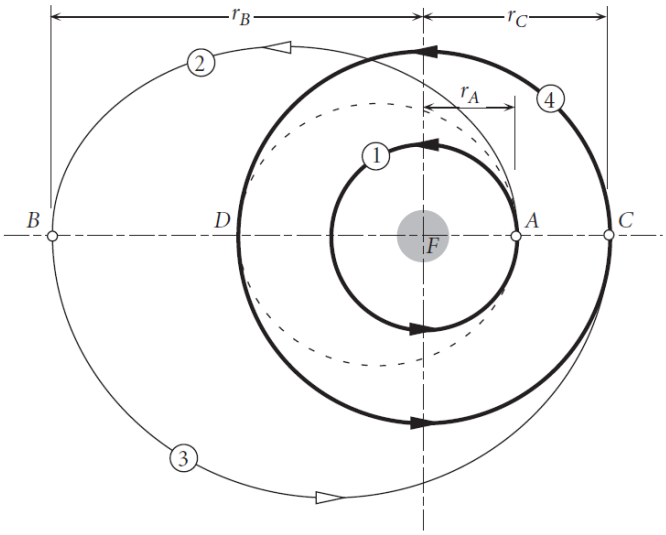


Figure 2.11: Bi-elliptic transfer from inner orbit 1 to outer orbit 4 [43]

A bi–elliptic transfer from a circular orbit at radius r_A to another at r_C (orbit 4 in Fig. 2.11) consists of two coaxial half-ellipses. The first ellipse (arc 2) is tangent to the inner circle at r_A and meets the outer circle at point B. The second ellipse (arc 3) starts at B and is tangent to the outer circle at r_C. Point B is chosen beyond r_C so that the burn ΔV_B there can be made arbitrarily small; indeed, as $r_B \to \infty$, we have $\Delta V_B \to 0$. In order for this bi–elliptic scheme to use less propellant than the single-ellipse (Hohmann) transfer, its total ΔV must satisfy

$$\Delta V_{\text{total, bi-elliptic}} < \Delta V_{\text{total, Hohmann}} \tag{2.32}$$

A straightforward ΔV analysis of the two methods gives

$$\Delta v_{\text{H}} = \left[\frac{1}{\sqrt{\alpha}} - \frac{\sqrt{2\,(1-\alpha)}}{\sqrt{\alpha\,(1+\alpha)}} - 1\right]\sqrt{\frac{\mu}{r_A}}, \tag{2.33}$$

$$\Delta V_{\text{bi}} = \left[\frac{\sqrt{2\,(\alpha+\beta)}}{\alpha\,\beta} - \frac{1+\sqrt{\alpha}}{\sqrt{\alpha}} - \frac{\sqrt{2}}{\beta\sqrt{1+\beta}}\,(1-\beta)\right]\sqrt{\frac{\mu}{r_A}} \tag{2.34}$$

where

$$\alpha = \frac{r_C}{r_A}, \qquad \beta = \frac{r_B}{r_A}. \tag{2.35}$$

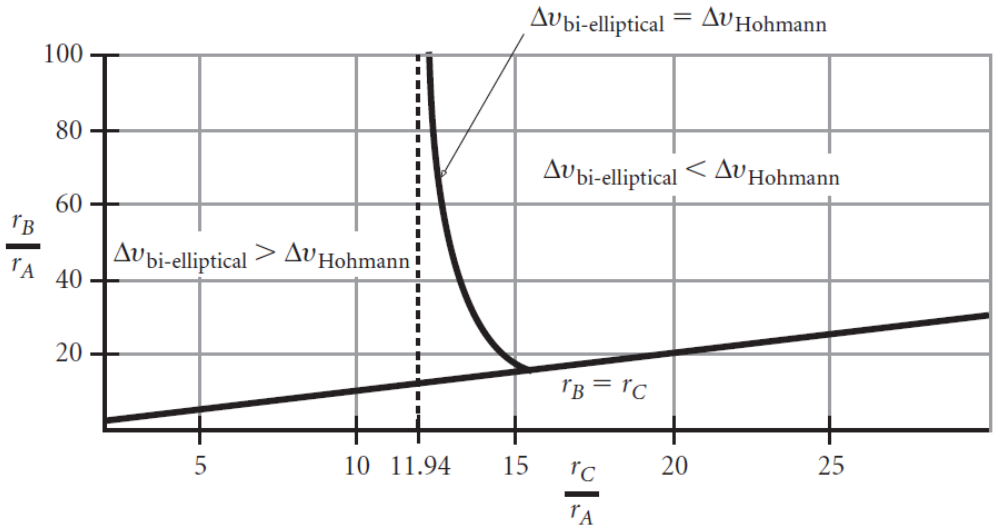


Figure 2.12: Orbits for which the bi-elliptical transfer is either less efficient or more efficient than the Hohmann transfer [43]

Plotting the difference between Hohmann and bi-elliptical ΔV_{total} as a function of the parameters α and β reveals the regions in which this difference is positive, negative, or zero. These regions are shown in Fig. 2.12.

- If the target-orbit radius satisfies $\alpha = r_C/r_A \lesssim 11.9$, the single-ellipse (Hohmann) transfer uses less Δv.
- If $\alpha \gtrsim 15$, the bi-elliptic scheme becomes more efficient.
- For intermediate values $11.9 \lesssim \alpha \lesssim 15$, whether the bi-elliptic transfer wins or loses depends on the apoapsis ratio $\beta = r_B/r_A$: large β favor the bi-elliptic trajectory, while small β favor Hohmann.

However even a modest savings in ΔV can be outweighed by the much longer flight time of the bi-elliptic path compared to the single semi-ellipse of the Hohmann transfer.

Phasing manoeuvres

A phasing manoeuvre is a two-impulse Hohmann transfer that departs from and returns to the same circular orbit, as illustrated in Fig. 2.13. The transfer ellipse called the phasing orbit is chosen so that the spacecraft re-encounters its original orbit after a specified period. Phasing manoeuvres change only the spacecraft's position along the orbit without altering its energy.

When two spacecraft sharing the same orbital radius are at different longitudes, one can perform a phasing manoeuvre to catch up with the other for a rendezvous. Similarly, geostationary communications and weather satellites use phasing manoeuvres to drift to new longitudes above the equator.

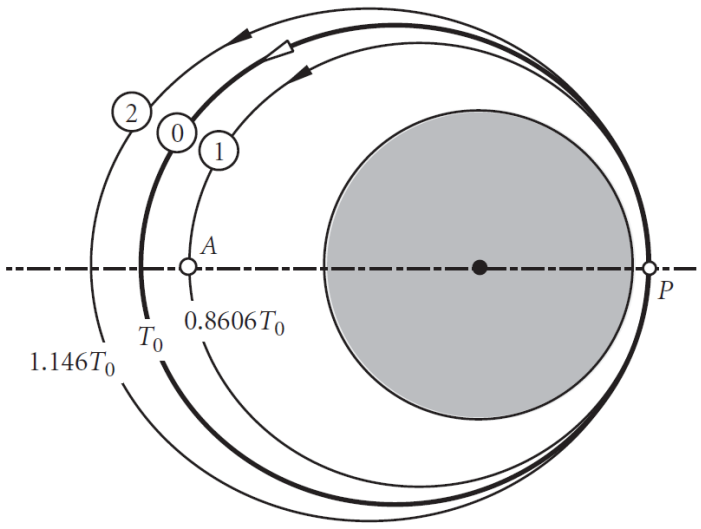


Figure 2.13: Main orbit (0) and two phasing orbits, faster (1) and slower (2). T_0 is the period of the main orbit [43]

A phasing manoeuvre can target a vacant point in space point P rather than a physical object. In Fig. 2.13, one might select phasing orbit 1 so that the spacecraft returns to P in less than one period of the main orbit. This choice is appropriate when the chaser trails the target: a retrograde burn at P slows the spacecraft relative to the original orbit, placing it on the shorter phasing ellipse. If instead the chaser leads the target, phasing orbit 2 with a longer period is preferred, and a prograde burn at P speeds the vehicle up to drop back onto the main orbit behind the target.

Once the desired phasing period T is chosen, the semi-major axis a of the phasing ellipse follows from

$$a \;=\; \left(\frac{T\sqrt{\mu}}{2\pi}\right)^{2/3} \tag{2.36}$$

Knowing that $2a = r_P + r_A$, one then determines whether P serves as periapsis or apoapsis of the ellipse. The eccentricity follows from the orbit equation 2.20, and evaluation of the energy or angular momentum relation at P (or A) completely specifies the phasing trajectory.

Orbital plane change

A plane-change manoeuvre uses a single impulse at constant radius, keeping the orbital energy and angular momentum fixed. Only the direction of the tangential velocity V_t is altered, its magnitude (and that of the radial and total speed) remains the same.

The total cost in ΔV is

$$\Delta V \;=\; 2\,V_t\,\sin\!\left(\frac{\Delta\psi}{2}\right) \tag{2.37}$$

where $\Delta\psi$ is the angle between the tangential velocity directions before and after the burn.

Since ΔV scales with orbital speed, it is most efficient to perform the plane change at apoapsis or combine it with a transfer burn. Fig. 2.14 shows a schematic representation of this manoeuvre.

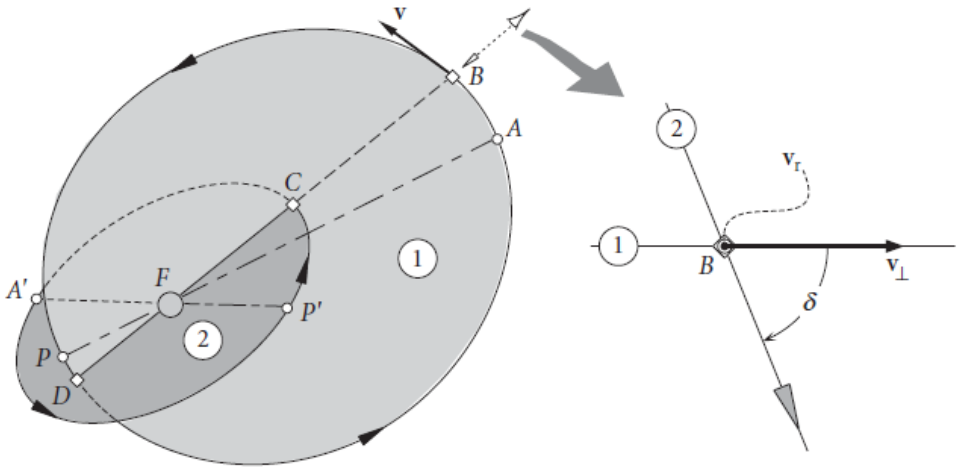


Figure 2.14: Two non-coplanar orbits about F (on the left). A view down the line of intersection of the two orbital planes (on the right) [43]

Chase manoeuvres

Chase trajectories can be found as solutions to *Lambert's problem*, which is the problem of two position vectors and the flight time between them.

According to H. D. Curtis [43] we can synthesize the problem as follows: let's suppose we have two known position vectors, $\mathbf{r}_1$ and $\mathbf{r}_2$, which describe two points P_1 and P_2 of an object orbiting a central body (mass M).

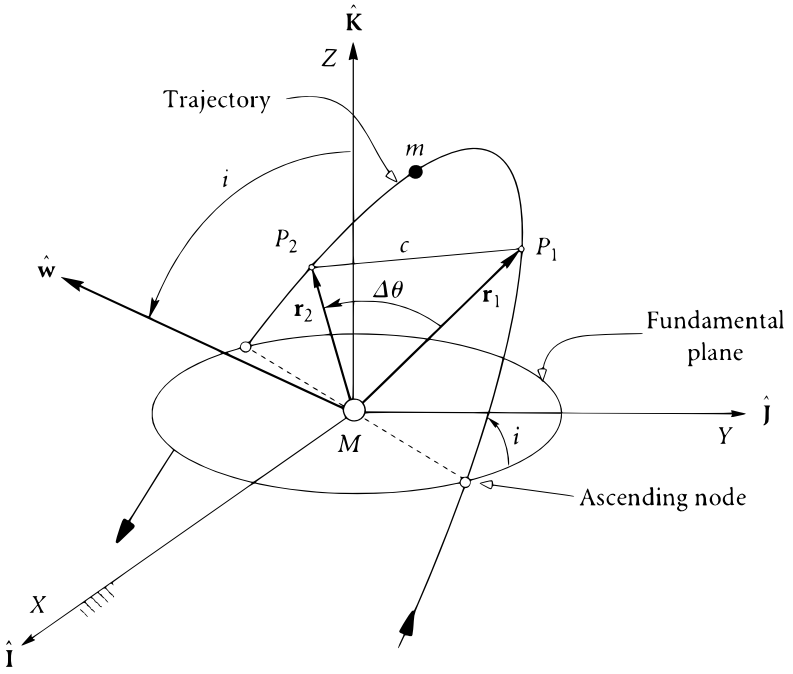


Figure 2.15: Lambert's problem [43]

To solve Lambert's problem, we begin by determining the transfer angle $\Delta\theta$ between the departure and arrival radius vectors. In practice, one computes

$$\cos\Delta\theta = \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{\|\mathbf{r}_1\|\,\|\mathbf{r}_2\|},$$

and then inspects the Z-component of $\mathbf{r}_1 \times \mathbf{r}_2$ to choose the correct branch of the inverse cosine for a prograde transfer. Once $\Delta\theta$ is fixed, we introduce the Lagrange coefficients f and g so that

$$\mathbf{r}_2 = f\,\mathbf{r}_1 + g\,\mathbf{v}_1, \qquad \mathbf{v}_2 = \dot f\,\mathbf{r}_1 + \dot g\,\mathbf{v}_1.$$

Evaluating f and g efficiently uses the universal-variable formalism: we define a parameter χ and an auxiliary variable z in the relation

$$\sqrt{\mu}\,\Delta t = \chi^3\,S(z) + A\,\chi\sqrt{C(z)},$$

where $A = \sin\Delta\theta\sqrt{\frac{r_1 r_2}{1-\cos\Delta\theta}}$, and $C(z)$, $S(z)$ are the Stumpff functions. We then iterate on z via Newton–Raphson, updating

$$y(z) = r_1 + r_2 + A\,\frac{z\,S(z)-1}{\sqrt{C(z)}}, \quad \chi = \sqrt{\frac{y(z)}{C(z)}},$$

until the time-of-flight equation is satisfied. The sign of the final z tells us whether the trajectory is elliptic ($z > 0$), parabolic ($z = 0$), or hyperbolic ($z < 0$). Finally, with f, g, $\dot{f}$, and $\dot{g}$ obtained, the initial and final velocity vectors follow directly from

$$\mathbf{v}_1 = \frac{\mathbf{r}_2 - f\,\mathbf{r}_1}{g}, \qquad \mathbf{v}_2 = \frac{\dot{g}\,\mathbf{r}_2 - \mathbf{r}_1}{g}.$$

This sequence delivers the unique spacecraft conic trajectory connecting the two points in the specified time Δt.

To sum up here follows a flow chart of the Lambert's problem.

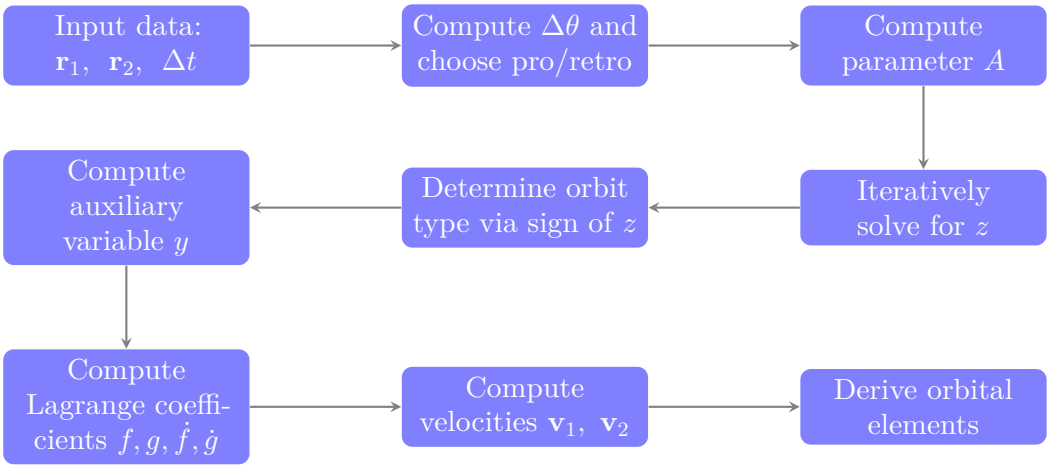


Figure 2.16: Lambert's problem solving logic

Chapter 3

Evolutionary Algorithms

In this work three types of Evolutionary Algorithms were used to achieve the optimal solution of the Lambert's problem in terms of propellant consumption (ΔV) and time of flight: *Particle Swarm Optimization (PSO)*, *Differential Evolution (DE)* and *Covariance Matrix Adaptation Evolution Strategy (CMA-ES)*. In this chapter, after e brief overview of the EAs, the logic of the three pre-mentioned methods are shown together with case studies presented by Pontani [47].

3.1 Overview

The Evolutionary Algorithms are based on the collective learning process within a population of individuals, each of which represents a search point in the space of potential solutions to a given problem. The population is initialized randomly and gradually evolves toward more promising regions of the search space by means of stochastic processes of selection (sometimes deterministic), mutation, and optionally recombination. The environment assigns each individual a fitness value, and the selection mechanism gives those with higher fitness more opportunities to reproduce. When recombination is used, parental information is mixed in the offspring, while mutation introduces new variations and innovations. The following notation is used for a concise description of Evolutionary Algorithms [48].
Let

$$f : \mathbb{R}^n \to \mathbb{R}$$

be the objective function to be minimized. The space of individuals is I, and fitness is given by

$$\Phi : I \to \mathbb{R},$$

of which f is one component. An individual is $\mathbf{a} \in I$ with decision vector $\mathbf{x} \in \mathbb{R}^n$. Parent and offspring population sizes are $\mu \geq 1$ and $\lambda \geq 1$, respectively (typically $\lambda > \mu$ in non-elitist schemes). At generation t, the parent population is

$$P(t) = \{\mathbf{a}_1(t), \ldots, \mathbf{a}_\mu(t)\}.$$

Recombination is modelled by

$$r_{\Theta_r} : I^\mu \to I^\lambda,$$

and mutation by

$$m_{\Theta_m} : I^\lambda \to I^\lambda,$$

where Θ_r and Θ_m collect any additional parameters. These macro-operators can be seen as repeated applications of local operators

$$r' : I^\mu \to I, \quad m' : I \to I,$$

which create single offspring. Selection is defined by

$$s_{\Theta_s} : I^\lambda \cup I^\mu \ \to \ I^\mu,$$

choosing the next parent set from current parents and offspring. The fitness function Φ is evaluated on all individuals in a population, and termination is decided by

$$\iota : I^\mu \to \{\text{true}, \text{false}\}.$$

The resulting algorithmic description is given below

Algorithm 1 Outline of an Evolutionary Algorithm [48]

1: $t := 0$
2: *Initialize* $P(0) := \{\mathbf{a}_1(0), \ldots, \mathbf{a}_\mu(0)\} \in I^\mu$;
3: *Evaluate* $P(0) : \{\Phi(\mathbf{a}_1(0)), \ldots, \Phi(\mathbf{a}_\mu(0))\}$;
4: **while** $(\iota\Big(P(t)\Big) \neq \text{true})$ **do**
5: *Recombine* $P'(t) := r_{\Theta_r}\Big(P(t)\Big)$;
6: *Mutate* $P''(t) := m_{\Theta_m}\Big(P'(t)\Big)$;
7: *Evaluate* $P''(t) : \{\Phi(\mathbf{a}''_1(t)), \ldots, \Phi(\mathbf{a}''_\lambda(t))\}$;
8: *Select* $P(t+1) := s_{\Theta_s}\Big(P''(t) \cup Q\Big)$;
9: $t := t + 1$;
10: **end while**=0

Here $Q \in \{\emptyset, P(t)\}$ is a set of individuals that are additionally taken into account during the selection step. The evaluation stage produces a multiset of fitness values,

which need not coincide with the objective function values. Since the selection operator uses fitness values rather than raw objective values, these fitness values serve as the outcome of evaluation. Nonetheless, objective function values must be computed during fitness assessment so that they are available and can be stored in a suitable data structure.

3.2 Particle Swarm Optimization

In PSO, a swarm of *particles* is initially scattered throughout the search space of an optimization problem. Each particle first evaluates the objective function at its current location. It then updates its motion by considering: its current position, its own best position found so far, and the best position discovered by the entire swarm, all perturbed by random factors. When every particle has been repositioned, the iteration ends. Over successive iterations, the swarm much like a flock of birds searching for food tends to drift toward an optimum of the fitness landscape.

Every particle i is described by three D-dimensional vectors (where D is the dimensionality of the search space):

- current position $\boldsymbol{x}_i$,
- personal best position $\boldsymbol{p}_i$,
- velocity $\boldsymbol{v}_i$.

The vector $\boldsymbol{x}_i$ represents a candidate solution. At each iteration, the objective value at $\boldsymbol{x}_i$ is computed; if it improves upon every previous value obtained by the same particle, then $\boldsymbol{x}_i$ is copied to $\boldsymbol{p}_i$. The best fitness found by the whole swarm, stored in a variable commonly called *gbest* (denoted by $\boldsymbol{g}$ for its position), is updated whenever necessary. New candidate points are produced by adding the velocity $\boldsymbol{v}_i$ to the current position, so PSO's search behavior is governed primarily by the way it adjusts $\boldsymbol{v}_i$, which acts as a step size.

In the version of PSO used in this thesis proposed by Shi and Eberhart [1] [50] (and also used by Pontani [47]) the velocity update for particle i at iteration $t+1$ is

[1] In Kennedy and Eberhart's original PSO [49], the velocity of each particle was updated using only the cognitive and social components: $\boldsymbol{v}_i(t+1) = c_1\, r_1 \left[\boldsymbol{p}_i(t) - \boldsymbol{x}_i(t)\right] + c_2\, r_2 \left[\boldsymbol{g}(t) - \boldsymbol{x}_i(t)\right]$.
Shi and Eberhart introduced the inertia weight ω as a third term to better balance exploration and exploitation.

given by:

$$\boldsymbol{v}_i(t+1) \;=\; \underbrace{\omega\,\boldsymbol{v}_i(t)}_{\text{inertia term}} \;+\; \underbrace{c_1\,r_1\left[\boldsymbol{p}_i(t)-\boldsymbol{x}_i(t)\right]}_{\text{cognitive component}} \;+\; \underbrace{c_2\,r_2\left[\boldsymbol{g}(t)-\boldsymbol{x}_i(t)\right]}_{\text{social component}}, \tag{3.1}$$

where:

- ω is the *inertia weight*, which controls the trade-off between exploration and exploitation;
- c_1 and c_2 are learning coefficients for the cognitive and social components, respectively;
- $r_1, r_2 \sim \mathcal{U}(0,1)$ are independent uniformly distributed random scalars (drawn separately for each dimension and each iteration);
- $\boldsymbol{p}_i(t)$ is the personal best position of particle i up to iteration t;
- $\boldsymbol{g}(t)$ is the global best position (gbest) found by the swarm up to iteration t;
- $\boldsymbol{x}_i(t)$ and $\boldsymbol{v}_i(t)$ are the current position and velocity of particle i at iteration t.

After computing the new velocity, the position is updated according to:

$$\boldsymbol{x}_i(t+1) \;=\; \boldsymbol{x}_i(t) \;+\; \boldsymbol{v}_i(t+1). \tag{3.2}$$

Through these equations, particle i is influenced by:

- its own "momentum" (inertia term),
- attraction toward its personal best solution (cognitive term),
- attraction toward the global best solution found by the swarm (social term).

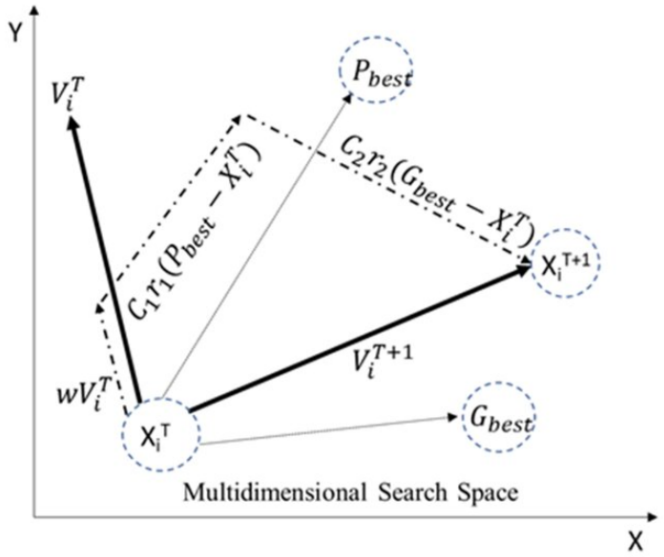


Figure 3.1: PSO strategy [51]

Algorithm 2 PSO pseudo-code

1: ▷ **Random initialization of** p_i^0 **and** v_i^0
2: ▷ Set $p_{\text{best},i} = p_i^0$ and p_{gbest}
3: ▷ Set iter $= 1$
4: **while** $\Gamma\left(p_{\text{gbest}}\right) > \varepsilon$ **and** iter $<$ iter$_{\max}$ **do**
5: ▷ Evaluate $p_{\text{best},i}$ and p_{gbest}
6: ▷ Update p_i and v_i
7: ▷ iter $=$ iter $+ 1$
8: **end while**
9: ▷ **return** p_{gbest} as the optimal solution $=0$

3.3 Differential Evolution

Differential Evolution (DE) is a very powerful stochastic algorithm developed by Storn and Price in 1995 [52], used to perform global optimization of nonlinear problems in a continuous search space. In general, the DE algorithm follows the same computational procedure as a typical Evolutionary Algorithm (EA). Through the successive processes of mutation, crossover and selection, the algorithm then chooses which candidate solutions will comprise the next generation. DE executes these three main steps in a cyclic manner (mutation, crossover and selection) until a termination criterion is met (e.g., reaching a predefined number of generations).

A schematic representation of the DE workflow is shown in Fig 3.2.

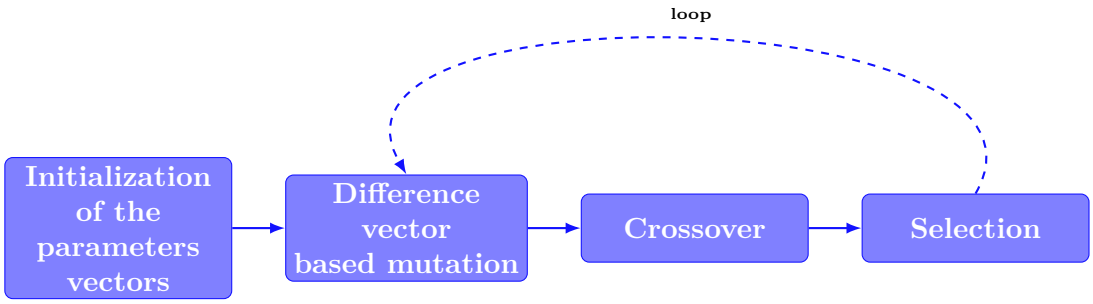


Figure 3.2: Schematic workflow of the Differential Evolution algorithm

Initialization of the parameters vectors

DE searches for a global optimum point in a D-dimensional real parameter space $\Re^D$. It begins with a randomly initiate population of NP D-dimensional real-valued parameter vectors. Each vector, also called a genome or chromosome, represents a candidate solution to the D-dimensional optimization problem. We denote the i-th vector of the population at generation G by

$$\widetilde{\mathbf{x}}_{i,G} \;=\; \Big[x_{1,i,G},\, x_{2,i,G},\, x_{3,i,G},\, \ldots,\, x_{D,i,G}\Big]\,, \quad i = 1,2,\ldots,NP.$$

In many optimization problems each parameter x_j has prescribed lower and upper bounds because it corresponds to a physical quantity (for example, a length or a mass cannot be negative). Let

$$\mathbf{x}_{\min} \;=\; \Big\{\, x_{1,\min},\, x_{2,\min},\, \ldots,\, x_{D,\min}\Big\} \quad \text{and} \quad \mathbf{x}_{\max} \;=\; \Big\{\, x_{1,\max},\, x_{2,\max},\, \ldots,\, x_{D,\max}\Big\}$$

be the vectors of lower and upper bounds for each of the D components. The initial population (i.e., at $G = 0$) should cover these bounds as uniformly as possible. Hence, the j-th component of the i-th vector is initialized according to

$$x_{j,i,0} \;=\; x_{j,\min} \;+\; \mathrm{rand}_{j,i}[0,1]\,\Big(x_{j,\max} - x_{j,\min}\Big), \tag{3.3}$$

where $\mathrm{rand}_{i,j}[0,1] \sim \mathcal{U}(0,1)$ is drawn independently for each component $j = 1,\ldots,D$ of each individual $i = 1,\ldots,NP$. In this way, the initial population uniformly spans the prescribed search space $[\mathbf{x}_{\min},\, \mathbf{x}_{\max}]$.

Difference vector based mutation

Within the Differential Evolution (DE) framework, mutation is interpreted as a perturbation of a parent (target) vector by incorporating information from other

randomly selected individuals in the current population. Specifically, let $\widetilde{\mathbf{x}}_{i,G}$ denote the *target vector* (parent) for the i-th individual at generation G. To create a *donor vector* for each target $\widetilde{\mathbf{x}}_{i,G}$, three distinct parameter vectors denoted $\widetilde{\mathbf{x}}_{r_1,G}$, $\widetilde{\mathbf{x}}_{r_2,G}$, and $\widetilde{\mathbf{x}}_{r_3,G}$ are sampled uniformly at random from the current population. The indices r_1, r_2, and r_3 are chosen such that

$$r_1,\, r_2,\, r_3 \in \{1,2,\ldots,NP\}, \quad r_1 \neq r_2 \neq r_3 \neq i,$$

ensuring they are all distinct and also different from the base index i. Once these three donor candidates have been selected, the difference of any two (for example, $\widetilde{\mathbf{x}}_{r_2,G} - \widetilde{\mathbf{x}}_{r_3,G}$) is scaled by a factor F. Adding this scaled difference to the third vector $\widetilde{\mathbf{x}}_{r_1,G}$ yields the donor vector $\mathbf{v}_{i,G}$. Formally,

$$\mathbf{v}_{i,G} \;=\; \widetilde{\mathbf{x}}_{r_1,G} \;+\; F\Big(\widetilde{\mathbf{x}}_{r_2,G} - \widetilde{\mathbf{x}}_{r_3,G}\Big). \tag{3.4}$$

The resulting donor vector $\mathbf{v}_{i,G}$ is later combined with the target $\widetilde{\mathbf{x}}_{i,G}$ via a crossover operation to form a *trial vector*. Figure 3.3 illustrates this mutation process in a two-dimensional parameter space, with an arbitrary objective function.

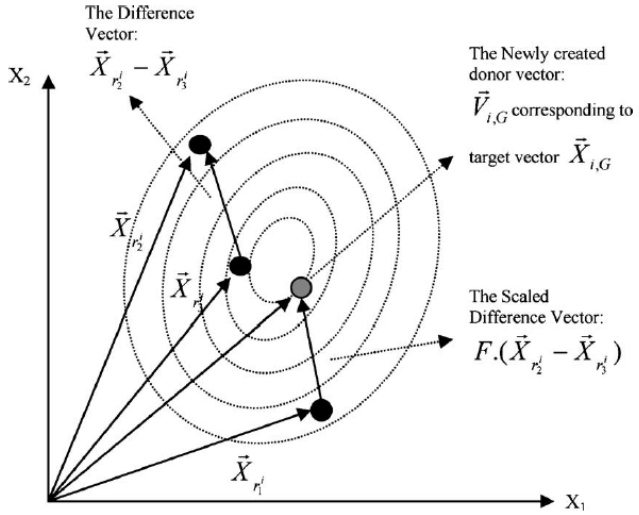


Figure 3.3: Simple DE mutation scheme in 2-D parametric space [53]

Crossover

After generating the donor vector $\mathbf{v}_{i,G}$ via mutation (see Eq. 3.4), Differential Evolution constructs a *trial vector* $\mathbf{u}_{i,G}$ by recombining $\mathbf{v}_{i,G}$ with the original *target vector* $\widetilde{\mathbf{x}}_{i,G}$. In the most common variant, called *binomial crossover*, each component of $\mathbf{u}_{i,G} = [u_{1,i,G}, u_{2,i,G}, \ldots, u_{D,i,G}]$ is defined as

$$u_{j,i,G} \;=\; \begin{cases} v_{j,i,G}, & \text{if } \mathrm{rand}_{j,i}[0,1] \;\leq\; C_r \quad \text{or} \quad j = j_{\mathrm{rand}}, \\ x_{j,i,G}, & \text{otherwise,} \end{cases} \qquad j = 1,2,\ldots,D, \tag{3.5}$$

where:

- $C_r \in [0,1]$ is the *crossover probability* (also called the *recombination rate*);
- $\text{rand}_{j,i}[0,1] \sim \mathcal{U}(0,1)$ is drawn independently for each component j of individual i;
- j_{rand} is a randomly chosen index in $\{1,2,\ldots,D\}$ to ensure that at least one component of $\mathbf{v}_{i,G}$ passes to $\mathbf{u}_{i,G}$, even if all random draws exceed C_r.

Thus, for each component j, if the random number $\text{rand}_{j,i}[0,1]$ does not exceed C_r (or if j equals the preselected index j_{rand}), the trial component $u_{j,i,G}$ is taken from the donor vector; otherwise, it is inherited from the target vector. By controlling C_r, binomial crossover balances exploration (by inheriting more components from $\mathbf{v}_{i,G}$) against exploitation (by preserving more components from $\widetilde{\mathbf{x}}_{i,G}$).

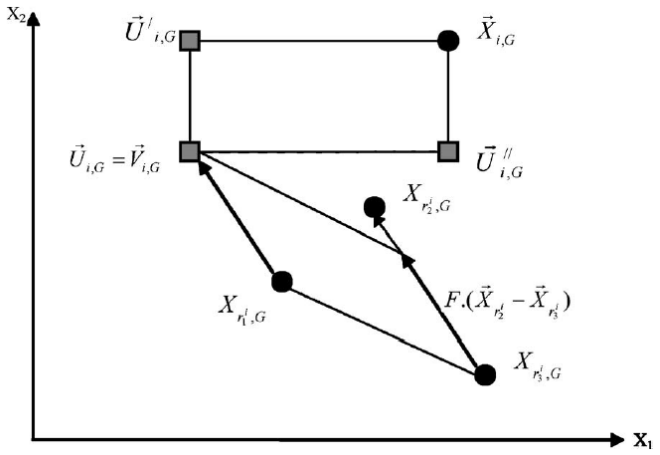


Figure 3.4: Different possible trial vectors formed due to binomial crossover between the target and the mutant vectors in 2-D search space [53]

Selection

Once the trial vector $\mathbf{u}_{i,G}$ has been formed, DE performs a *greedy selection* step to decide whether $\mathbf{u}_{i,G}$ or the original target $\widetilde{\mathbf{x}}_{i,G}$ survives to the next generation $G+1$. Denote by $f(\cdot)$ the objective (fitness) function to be minimized. The selection rule is:

$$\widetilde{\mathbf{x}}_{i,G+1} = \begin{cases} \mathbf{u}_{i,G}, & \text{if } f\big(\mathbf{u}_{i,G}\big) \le f\big(\widetilde{\mathbf{x}}_{i,G}\big), \\ \widetilde{\mathbf{x}}_{i,G}, & \text{otherwise.} \end{cases} \tag{3.6}$$

In other words, if the trial vector yields an equal or lower cost than the target, $\mathbf{u}_{i,G}$ replaces $\widetilde{\mathbf{x}}_{i,G}$ in the next population; otherwise, the algorithm retains the original target. This deterministic, one-to-one replacement ensures that the population's

overall best cost never worsens from one generation to the next. Over repeated mutation, crossover, and selection steps, the DE population gradually converges toward high-quality solutions.

DE Mutation Strategies

Storn and Price identified that it is the mutation operator that fundamentally distinguishes one Differential Evolution (DE) scheme from another. In the simplest DE variant often called DE/rand/1 the donor vector is generated by selecting a base vector $\widetilde{\mathbf{x}}_{r_1,G}$ uniformly at random from the current population and adding to it a single scaled difference of two other randomly chosen vectors:

$$\mathbf{v}_{i,G} \;=\; \widetilde{\mathbf{x}}_{r_1,G} \;+\; F\Big(\widetilde{\mathbf{x}}_{r_2,G} - \widetilde{\mathbf{x}}_{r_3,G}\Big), \tag{3.7}$$

where $r_1, r_2, r_3 \in \{1,2,\ldots,NP\}$ are distinct indices (all different from the target index i), and $F > 0$ is a real-valued scaling factor. When DE/rand/1 is used together with binomial crossover, the full scheme is denoted DE/rand/1/bin.

More generally, DE mutation schemes are named according to the template

$$\mathrm{DE}\Big/\langle\text{base}\rangle\Big/\langle\#\text{differences}\rangle\Big/\langle\text{crossover}\rangle,$$

where

- $\langle$base$\rangle$ indicates which vector is used as the "base" (e.g., rand, best, or target) to which difference vectors are added;
- $\langle$#differences$\rangle$ specifies the number of pairwise differences of population members that are linearly combined;
- $\langle$crossover$\rangle$ denotes the crossover type, either binomial (bin) or exponential (exp).

Altough Storn and Price [52, 54] proposed several mutation strategies in this thesis work the code user will be able to select from three different ones:

1. *DE/rand/1/bin:*
$$\mathbf{v}_{i,G} \;=\; \widetilde{\mathbf{x}}_{r_1,G} \;+\; F\left(\widetilde{\mathbf{x}}_{r_2,G} - \widetilde{\mathbf{x}}_{r_3,G}\right),$$
where $r_1, r_2, r_3 \in \{1,\ldots,NP\}$ are distinct and different from i.
2. *DE/best/1/bin:*
$$\mathbf{v}_{i,G} \;=\; \widetilde{\mathbf{x}}_{\mathrm{best},G} \;+\; F\left(\widetilde{\mathbf{x}}_{r_1,G} - \widetilde{\mathbf{x}}_{r_2,G}\right),$$
where $\widetilde{\mathbf{x}}_{\mathrm{best},G}$ is the current global best, and r_1, r_2 are distinct indices $\neq i$.

3. *DE/current-to-best/1/bin:*

$$\mathbf{v}_{i,G} \;=\; \widetilde{\mathbf{x}}_{i,G} \;+\; F\left(\widetilde{\mathbf{x}}_{\text{best},G} - \widetilde{\mathbf{x}}_{i,G}\right) \;+\; F\left(\widetilde{\mathbf{x}}_{r_1,G} - \widetilde{\mathbf{x}}_{r_2,G}\right),$$

where $\widetilde{\mathbf{x}}_{i,G}$ is the target vector, $\widetilde{\mathbf{x}}_{\text{best},G}$ is the global best, and $r_1, r_2 \neq i$ are distinct.

Algorithm 3 Differential Evolution (DE) pseudo-code

Initialize population $\{\mathbf{X}_i\}_{i=1}^{NP}$ randomly within bounds.
Evaluate each $f(\mathbf{X}_i)$; set best $\mathbf{X}_{\text{best}} \leftarrow \arg\min f(\mathbf{X}_i)$.
for $G = 1$ **to** $G_{\max}$ **and** not converged **do**
  for $i = 1$ **to** NP **do**
    Mutation: pick distinct indices $r_1, r_2, r_3 \neq i$.
    Compute donor $\mathbf{V}_i = \mathbf{X}_{r_1} + F\,(\mathbf{X}_{r_2} - \mathbf{X}_{r_3})$.
    Crossover: for each $j = 1, \ldots, D$ set

$$U_{j,i} = \begin{cases} V_{j,i}, & \text{if } \text{rand}_j \leq C_r \text{ or } j = j_{\text{rand}}, \\ X_{j,i}, & \text{otherwise.} \end{cases}$$

    Selection: if $f(\mathbf{U}_i) \leq f(\mathbf{X}_i)$ then $\mathbf{X}_i \leftarrow \mathbf{U}_i$.
  end for
  Update $\mathbf{X}_{\text{best}} \leftarrow \arg\min_i f(\mathbf{X}_i)$.
end for
return $\mathbf{X}_{\text{best}}$ as the optimal solution. =0

3.4 CMA-ES

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES), introduced by Hansen and Ostermeier [55], is a powerful optimization method that doesn't require derivative information. Like other EAs, it iteratively explores the search space by generating new candidate solutions, evaluating their quality, and selecting the best ones. The main difference is that CMA-ES actively learns the shape and scale of the search distribution, which is modeled as a Gaussian ellipsoid. This distribution gradually adapts, elongating along directions of promising improvement and shrinking in less fruitful directions.

The general procedure of CMA-ES can be summarized in the following steps, also illustrated in Figure 3.5:

1. Generate candidate solutions from the Gaussian distribution.

2. Evaluate each candidate using the objective function.
3. Select the best-performing candidates and update the distribution mean accordingly.
4. Adjust both the size (σ) and shape (covariance $\mathbf{C}$) of the Gaussian ellipsoid to reflect successful directions of search.

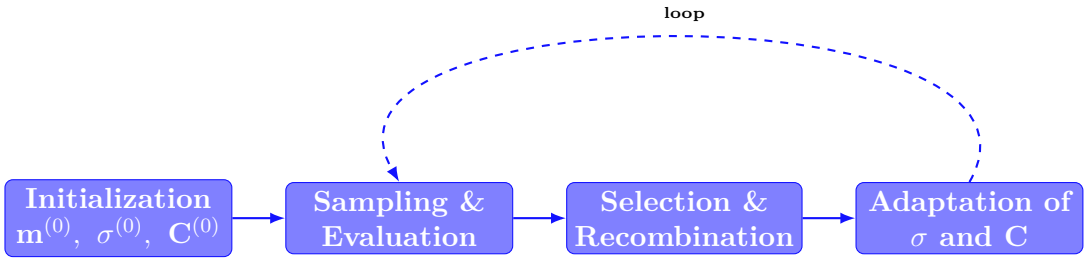


Figure 3.5: Schematic workflow of the CMA–ES algorithm

Initialization

Initially, the algorithm sets up three primary parameters:

- *Mean vector* $\mathbf{m}^{(0)}$: Set at the center of the search domain, this represents the current best guess for the solution.
- *Global step-size* $\sigma^{(0)}$: Determines how broadly the algorithm initially explores the search space.
- *Covariance matrix* $\mathbf{C}^{(0)}$: Initially set as the identity matrix, this defines an initial spherical shape of the search cloud.

The algorithm maintains a population size λ and selects the top half ($\mu = \lfloor \lambda/2 \rfloor$) as parents. These parents influence the next generation proportionally according to their rank-based weights, emphasizing better solutions while maintaining diversity.

Sampling and Evaluation

In each generation, the CMA-ES generates new candidate solutions (offspring) by sampling from the current Gaussian distribution:

$$\mathbf{x}_k^{(g+1)} = \mathbf{m}^{(g)} + \sigma^{(g)}\, \mathbf{B}^{(g)}\mathbf{D}^{(g)}\mathbf{z}_k^{(g)}, \quad \mathbf{z}_k^{(g)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Here, $\mathbf{B}^{(g)}$ and $\mathbf{D}^{(g)}$ represent rotations and scales respectively. Each candidate is evaluated using the objective function, measuring their fitness.

Selection and Recombination

Candidates are ranked according to fitness. The new mean is computed as a weighted average of the best μ offspring:

$$\mathbf{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}^{(g+1)}$$

This moves the Gaussian cloud towards regions of better solutions.

Adaptation of σ and C

Step-size adaptation

To adjust the global scale of exploration, CMA-ES uses an evolution path $\mathbf{p}_s$ that records recent successful steps. If recent steps are larger than expected by random chance, the step size σ increases; if smaller, it decreases:

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_s}{d_s}\left(\frac{\|\mathbf{p}_s^{(g+1)}\|}{\sqrt{N}} - 1\right)\right).$$

Covariance adaptation

The covariance matrix $\mathbf{C}$ is updated to reflect the direction and spread of successful steps:

$$\mathbf{C}^{(g+1)} = (1 - c_1 - c_\mu)\mathbf{C}^{(g)} + c_1 \mathbf{p}_c^{(g+1)} {\mathbf{p}_c^{(g+1)}}^\top + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}^{(g)} {\mathbf{y}_{i:\lambda}^{(g)}}^\top$$

These updates reshape and reorient the search distribution to better fit the landscape of the optimization problem. Periodically, the covariance matrix is decomposed into its eigenvalues and eigenvectors to keep numerical stability, which helps maintain a clear interpretation of directions and scales for subsequent iterations.

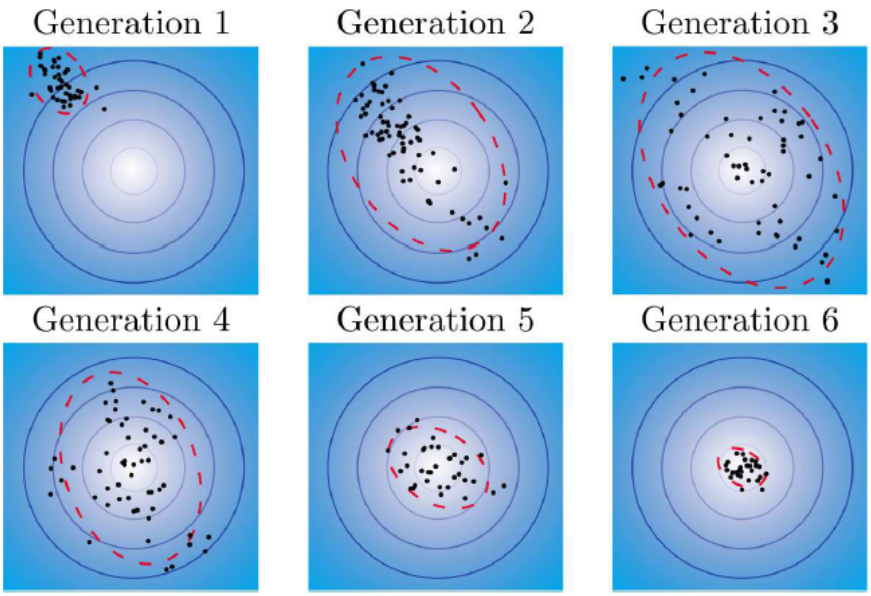


Figure 3.6: Evolution of the CMA-ES search cloud over six generations on a 2D convex objective. Each panel shows background contours of the cost function, sampled offspring (black dots), and the standard deviation contour of the Gaussian distribution (orange dashed). As generations advance, the cloud deforms and shrinks, homing in on the optimum [56]

Algorithm 4 CMA–ES pseudo-code

Initialize: mean $\mathbf{m}$, step-size σ, covariance $\mathbf{C} = \mathbf{I}$, paths $\mathbf{p}_s = \mathbf{0}$, $\mathbf{p}_c = \mathbf{0}$.
Choose population size λ, parent count $\mu = \lfloor \lambda/2 \rfloor$, weights w_i, and learning rates c_s, c_c, c_1, c_μ.
Evaluate $f(\mathbf{m})$; set best $\mathbf{m}_{\text{best}} \leftarrow \mathbf{m}$.
for $g = 1$ **to** $g_{\max}$ **and** stopping criterion not met **do**
    Sample λ candidates:
    Draw each $\mathbf{z}_k \sim \mathcal{N}(0, I)$, set $\mathbf{x}_k = \mathbf{m} + \sigma\, \mathbf{B}\mathbf{D}\, \mathbf{z}_k$.
    Clip $\mathbf{x}_k$ to bounds; compute $f_k = f(\mathbf{x}_k)$.
    Select & Recombine:
    Sort candidates by f_k. Update $\mathbf{m}_{\text{best}}$ if improved.
    Compute new mean $\mathbf{m} \leftarrow \sum_{i=1}^{\mu} w_i\, \mathbf{x}_{i:\lambda}$.
    Adapt step-size: Update $\mathbf{p}_s$ using weighted combination of previous path and new direction; adjust σ up/down according to $\|\mathbf{p}_s\|$ relative to expectation.
    Adapt covariance: Update $\mathbf{p}_c$ similarly; incorporate a rank-one term from $\mathbf{p}_c$ and a rank-μ term from the top μ steps to reshape $\mathbf{C}$.
    Eigen-refresh: Decompose $\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^\top$ for next sampling.
end for
return $\mathbf{m}_{\text{best}}$. =0

3.5 Algorithms performance evaluation

In this section, the performance of the three previously introduced algorithms is evaluated and compared across two distinct case studies. Initially, the common elements shared by the three algorithms, including the objective function, will be briefly described. Subsequently, the comparison will focus on metrics such as the number of generations required for convergence and the computational running time.

3.5.1 Objective Function

The objective function used in this thesis is specifically designed to minimize the total Δv (impulse) required for performing an orbital rendezvous manoeuvre and it's inspired by the one proposed by Pontani [47]. The decision variables optimized by the evolutionary algorithm are:

- Δt: The flight time of the chase manoeuvre, i.e., the time between the initiation of the transfer and the final encounter with the target satellite.

- TA_{dep}: The initial true anomaly (in degrees) of the target satellite at the moment when the manoeuvre begins.

Given these two inputs, the objective function computes the required impulse by performing the following steps:

1. *Initial conditions:* Given the orbital parameters for both the chaser and the target satellite, the state vectors (position $\mathbf{r}_1$ and velocity $\mathbf{v}_1$) of the chaser at the manoeuvre start time are computed. Likewise, the initial true anomaly TA_{dep} allows computation of the target satellite's initial position and velocity.
2. *Calculating Target arrival conditions:* With the given flight time Δt, the function calculates the arrival time. Using Kepler's equation, the target satellite's final position $\mathbf{r}_2$ and velocity $\mathbf{v}_2$ at the arrival time are determined from its orbit.
3. *Lambert's transfer problem* (see Section 2.3.1): With initial chaser position $\mathbf{r}_1$, target arrival position $\mathbf{r}_2$, and flight time Δt, Lambert's equation is solved to find the required initial velocity $\mathbf{v}_{1,\text{lam}}$ and arrival velocity $\mathbf{v}_{2,\text{lam}}$ for the rendezvous.
4. *Delta-V computation:* The manoeuvre's total impulse Δv consists of two velocity changes:

$$\Delta \mathbf{v}_1 = \mathbf{v}_{1,\text{lam}} - \mathbf{v}_1, \tag{3.8}$$
$$\Delta \mathbf{v}_2 = \mathbf{v}_2 - \mathbf{v}_{2,\text{lam}}. \tag{3.9}$$

   Thus, the total Δv required is simply the sum of the magnitudes of these two velocity changes:

$$\Delta v_{\text{total}} = |\Delta \mathbf{v}_1| + |\Delta \mathbf{v}_2|. \tag{3.10}$$

Each iteration of the optimization proposes a pair of values $(\Delta t, \text{TA}_{\text{dep}})$, evaluates this objective function, and selects candidate solutions based on lower Δv_{total} values. Through iterative improvement, the algorithm identifies the optimal flight time and initial true anomaly that result in the lowest fuel consumption for the rendezvous manoeuvre. The code implemented in the work is reported in the Appendix A.

3.5.2 Case study 1 – Pontani 2D "case b"

The first benchmark is the *Pontani 2D case b* transfer [47], a manoeuvre between two coplanar orbits whose orbital elements are reported in Table 3.1, while the

reference ΔV budget and the time of flight (Δt) are given in Table 3.2.[2]

Orbital parameter	Initial orbit	Final orbit
a [km]	18000	35000
e	0.60	0.80
i [°]	0	0
Ω [°]	0	120
ω [°]	0	0
ν [°]	158.94	190.50

Table 3.1: Orbital elements of Pontani's 2D transfer

Transfer detail	Value
ΔV_1 [m/s]	1815
ΔV_2 [m/s]	748
ΔV_{tot} [m/s]	2563
Δt [s]	61482

Table 3.2: Reference solution for Pontani's 2D transfer

The resulting trajectory is shown in Fig. 3.7 and will be used as reference for the evolutionary algorithms presented in the following. All the scripts were executed in MATLAB® on an **Acer Aspire A315-57G** notebook equipped with an **Intel® Core™ i5-1035G1** CPU @ 1.00 GHz (4 physical cores, 8 threads).

[2]In Table 3.1 the true anomaly of the *final* orbit is the value assumed by the target spacecraft at rendezvous.

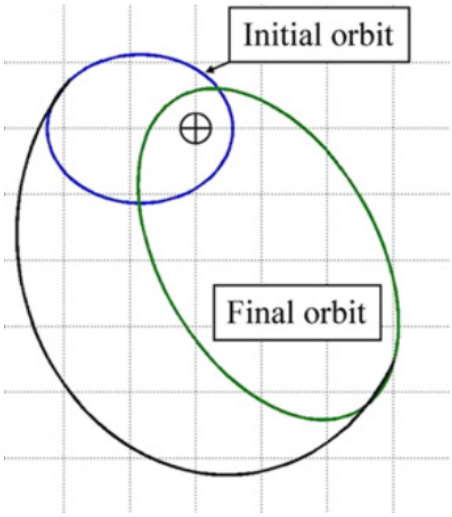


Figure 3.7: Pontani 2D case b reference trajectory [47]

Particle Swarm Optimization (PSO)

The algorithm parameter use in the PSO are listed in Table 3.3; the inertia weight and the learning factors were selected through empirical tuning. A stopping tolerance on the global best cost avoids unnecessary iterations once convergence is achieved, while a parallel implementation exploits all CPU cores and reduces wall–clock time.

PSO parameter	Value
Population size	300
Maximum iterations	200
Inertia weight ω	0.9
Cognitive coefficient c_1	2
Social coefficient c_2	2
Stopping tolerance	10^{-9}

Table 3.3: Parameters of the PSO run

The optimiser returns the transfer reported in Table 3.4; the last column shows the relative error with respect to Pontani's analytical solution. All discrepancies are below 0.15 %, confirming the soundness of the numerical method. Note that the departure true anomaly of the target, TA_{dep}, is optimised in this work, whereas Pontani optimises the chaser's anomalies; hence no direct comparison is possible for that variable.

Transfer detail	Value	Error
ΔV_1 [m/s]	1814.1	0.05 %
ΔV_2 [m/s]	749.3	0.13 %
ΔV_{tot} [m/s]	2563.4	0.02 %
Δt [s]	61486.9	0.008 %
TA_{dep} [°]	195.04	–

Table 3.4: Transfer obtained with PSO (Pontani 2D case)

The computational performance are summarised in the blue box below. Thanks to the stopping tolerance the algorithm converged in 64 iterations, avoiding 136 additional iterations that would have increased the execution time without improving the solution.

Computational performance (PSO - Pontani 2D)

Metric	Value
Execution time [s]	15.67
Iterations	64

Table 3.5: PSO performance (2D case).

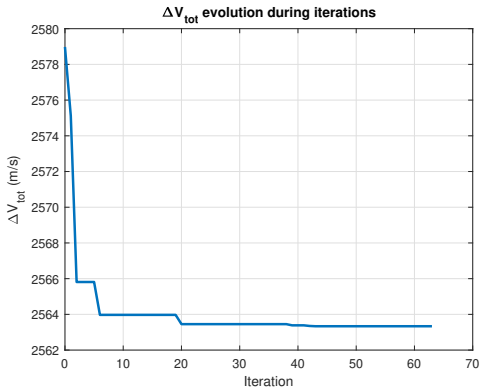


Figure 3.8: PSO cost evolution

Finally, Fig. 3.9 depicts the optimised trajectory, which is practically indistinguishable from the analytical solution in Fig. 3.7.

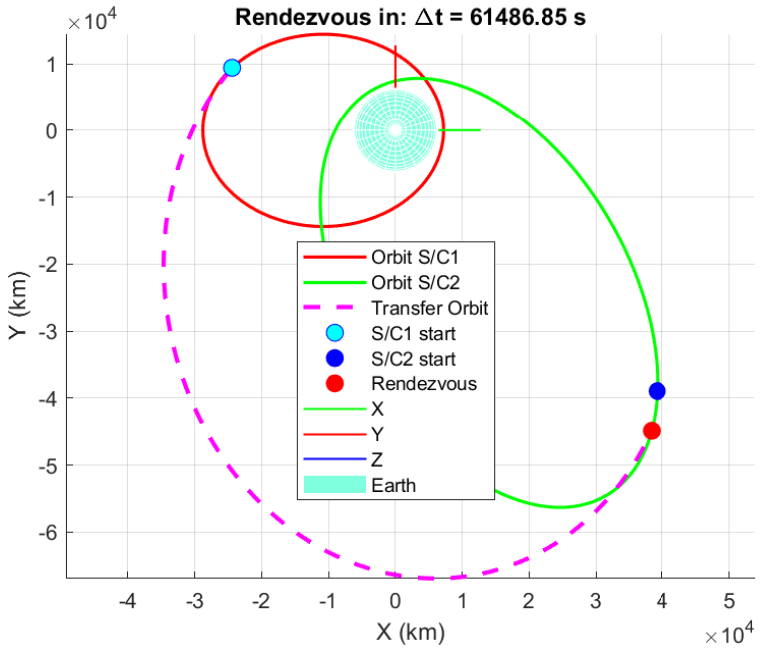


Figure 3.9: Transfer trajectory obtained with PSO (Pontani 2D case)

Differential Evolution (DE)

To ensure a fair comparison with the other methods, we configured the Differential Evolution (DE) run to match the population size and maximum number of iterations used elsewhere. The key settings: population size, maximum generations, crossover rate, and scale factor are summarized in Table 3.6. We adopted the same stopping criterion as in the PSO experiments and leveraged parallel evaluation across cores to accelerate convergence.

DE parameter	Value
Population size	300
Maximum generations	200
Crossover rate C_r	0.9
Scale Factor F	0.2
Stopping tolerance	10^{-9}

Table 3.6: Parameters of the DE run.

We implemented three classical DE strategies (see Section 3.3), and the resulting transfer performance and computational metrics appear below:

1. *DE/rand/1/bin*:

Transfer detail	Value	Error
ΔV_1 [m/s]	1814.8	0.01 %
ΔV_2 [m/s]	748.47	0.06 %
ΔV_{tot} [m/s]	2563.28	0.01 %
Δt [s]	61484.56	0.005 %
TA_{dep} [°]	195.05	–

Table 3.7: Transfer obtained with DE/rand/1/bin (Pontani 2D case)

Computational performance (DE/rand/1/bin - Pontani 2D)

Metric	Value
Execution time [s]	14.60
Iterations	58

Table 3.8: DE/rand/1/bin performance (2D case)

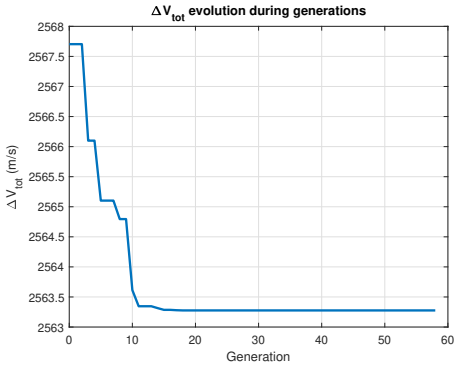


Figure 3.10: DE/rand/1/bin cost evolution

2. *DE/best/1/bin*:

Transfer detail	Value	Error
ΔV_1 [m/s]	1814.8	0.01 %
ΔV_2 [m/s]	748.47	0.06 %
ΔV_{tot} [m/s]	2563.28	0.01 %
Δt [s]	61484.56	0.005 %
TA_{dep} [°]	195.05	–

Table 3.9: Transfer obtained with DE/best/1/bin (Pontani 2D case)

Computational performance (DE/best/1/bin - Pontani 2D)

Metric	Value
Execution time [s]	6.24
Iterations	17

Table 3.10: DE/best/1/bin performance (2D case)

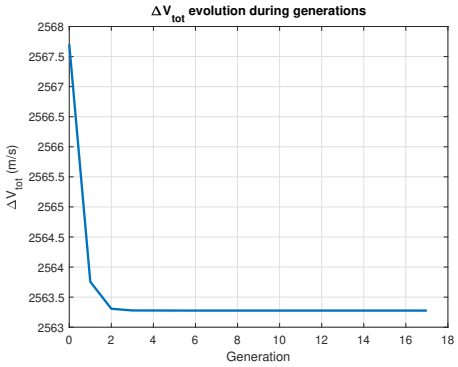


Figure 3.11: DE/best/1/bin cost evolution

3. *DE/current-to-best/1/bin*:

Transfer detail	Value	Error
ΔV_1 [m/s]	1814.80	0.01 %
ΔV_2 [m/s]	748.47	0.06 %
ΔV_{tot} [m/s]	2563.28	0.01 %
Δt [s]	61484.56	0.005 %
TA_{dep} [°]	195.05	–

Table 3.11: Transfer obtained with DE/current-to-best/1/bin (Pontani 2D case)

Computational performance (DE/current-to-best/1/bin - Pontani 2D)

Metric	Value
Execution time [s]	50.57
Iterations	199

Table 3.12: DE/current-to-best/1/bin performance (2D case)

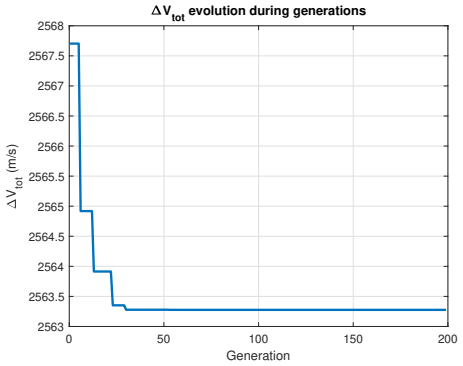


Figure 3.12: DE/current-to-best/1/bin cost evolution

All three strategies achieve a total ΔV within 0.1 % of the reference solution, with *DE/rand/1/bin* and *DE/best/1/bin* offering the fastest convergence. Figure 3.13 shows the trajectory from the *DE/rand/1/bin* run; the other strategies produce virtually identical transfer paths and are omitted for brevity.

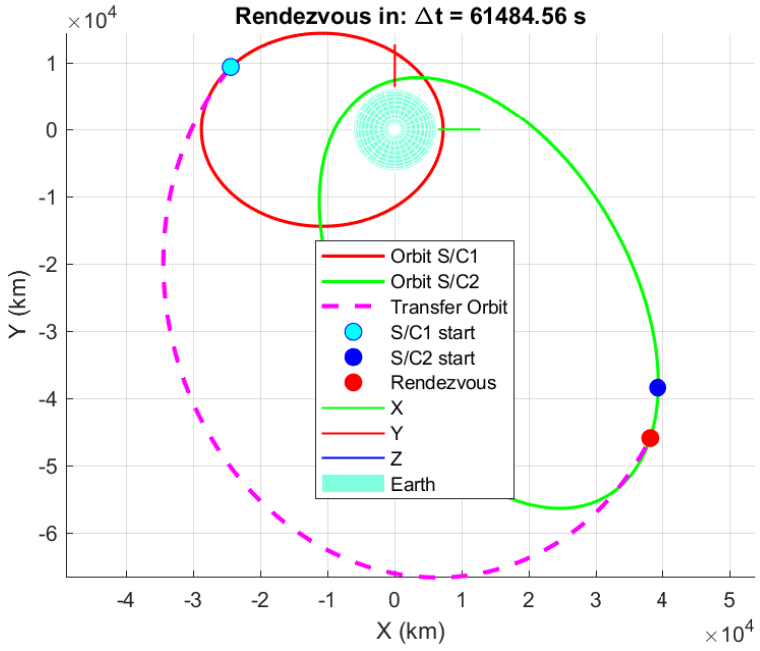


Figure 3.13: Transfer trajectory obtained with DE (Pontani 2D case)

Covariance Matrix Adaptation (CMA-ES)

We applied the CMA-ES algorithm with the default parameter set recommended by Hansen and Ostermeier [55], augmented by a tight termination tolerance to prevent stagnation. Table 3.13 summarizes the configuration used in our 2D rendezvous case.

Parameter	Value / Formula
Population size λ	$\lambda = 4 + \lfloor 3 \log(N) \rfloor$
Max iterations	`opts.maxIter` $= 1000$
Number of parents μ	$\mu = \lfloor \lambda/2 \rfloor$
Initial step-size σ_0	$\sigma_0 = \text{mean}((X_{\max} - X_{\min})/6)$
Termination tolerance `tol`	1×10^{-6}
Effective selection mass μ_{eff}	$\dfrac{(\sum w_i)^2}{\sum w_i^2}$
Step-size learning rate c_s	$c_s = \dfrac{\mu_{\text{eff}} + 2}{N + \mu_{\text{eff}} + 5}$
Step-size damping d_s	$d_s = 1 + c_s + 2\max\left(\sqrt{\frac{\mu_{\text{eff}}-1}{N+1}} - 1{,}0\right)$
Covariance path rate c_c	$c_c = \dfrac{4 + \mu_{\text{eff}}/N}{N + 4 + 2\mu_{\text{eff}}/N}$
Rank-one weight c_1	$c_1 = \dfrac{2}{(N+1.3)^2 + \mu_{\text{eff}}}$
Rank-μ weight c_μ	$c_\mu = \min\left(1 - c_1,\ 2\,\dfrac{\mu_{\text{eff}} - 2 + 1/\mu_{\text{eff}}}{(N+2)^2 + \mu_{\text{eff}}}\right)$

Table 3.13: CMA-ES configuration (Hansen & Ostermeier) [55]

Here, N denotes the number of decision variables, and $X_{\max}$, $X_{\min}$ are their upper and lower bounds.

The optimized transfer results for the Pontani 2D case appear in Table 3.14, with computational performance detailed in the accompanying tcolorbox.

Transfer detail	Value	Error
ΔV_1 [m/s]	1814.8	0.01 %
ΔV_2 [m/s]	748.47	0.06 %
ΔV_{tot} [m/s]	2563.28	0.01 %
Δt [s]	61484.56	0.005 %
TA_{dep} [°]	195.05	–

Table 3.14: Transfer obtained with CMA-ES (Pontani 2D)

Computational performance (CMA-ES – Pontani 2D)

Metric	Value
Execution time [s]	2.54
Iterations	341

Table 3.15: Performance summary

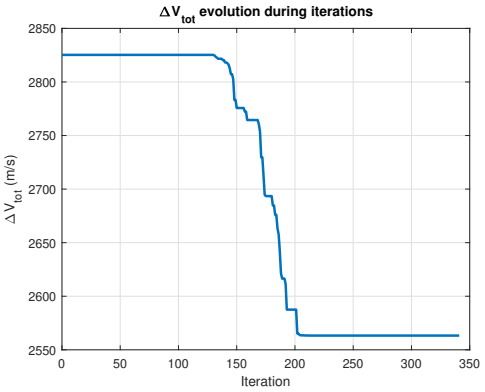


Figure 3.14: CMA-ES cost convergence

Figure 3.25 shows the resulting transfer trajectory, which matches the DE solution.

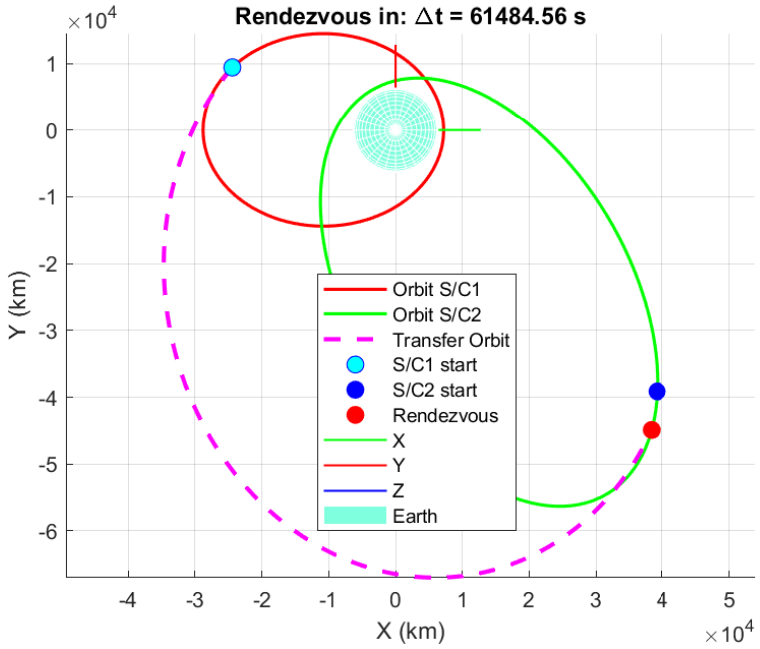


Figure 3.15: Transfer trajectory obtained with CMA-ES (Pontani 2D)

From Figure 3.16, for the case study the CMA-ES delivers the fastest overall runtime (2.54 s) but requires the highest number of iterations (341), reflecting its fine grained covariance adaptation. In contrast, DE/best/1/bin converges in only

17 iterations and completes in 6.24 s, demonstrating rapid progress per iteration. PSO and DE/rand/1/bin show similar behavior around 15 s and 60 iterations each while DE/current-to-best/1/bin is the slowest at 50.57 s despite a moderate 199 iterations.

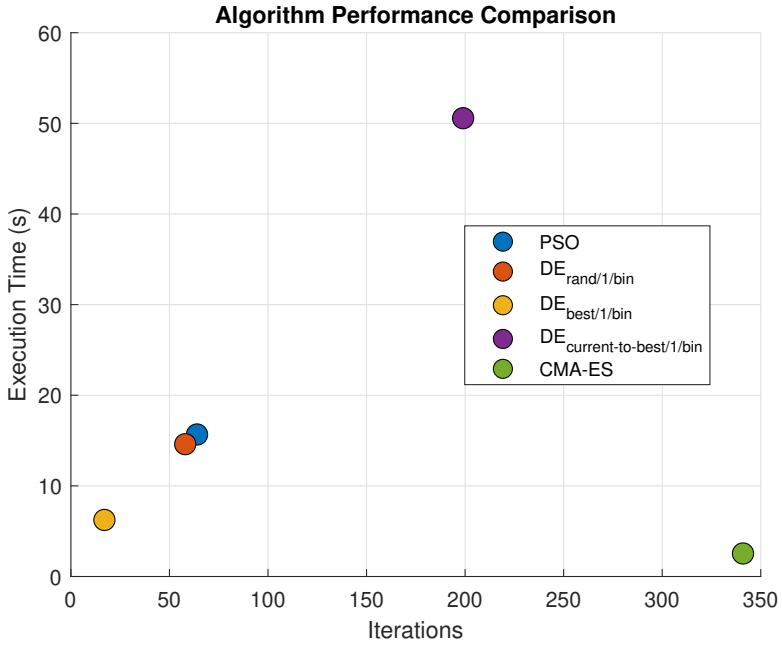


Figure 3.16: Algorithms comparison (Pontani 2D)

3.5.3 Case study 2 – Pontani 3D "case 3"

The second benchmark is the *Pontani 3D case 3* transfer [47], a manoeuvre between two non coplanar orbits, similar to a *GTO* whose orbital elements are reported in Table 3.16 [3], while the reference ΔV budget is given in Table 3.17 [4].

[3] In Table 3.16 the true anomaly of the *final* orbit is the value assumed by the target spacecraft at rendezvous.

[4] In this case the time of flight is not given so the comparison will be executed between the algorithms results.

Orbital parameter	Initial orbit	Final orbit
a [km]	6671.53	42163.95
e	0	0
i [°]	45	0
Ω [°]	0	0
ω [°]	0	0
ν [°]	0	180

Table 3.16: Orbital elements of Pontani's 3D transfer

Transfer detail	Value
ΔV_1 [m/s]	2466.583
ΔV_2 [m/s]	2170.808
ΔV_{tot} [m/s]	4637.391

Table 3.17: Reference solution for Pontani's 3D transfer

The resulting trajectory is shown in Fig. 3.17 and will be used as reference for the evolutionary algorithms presented in the following.

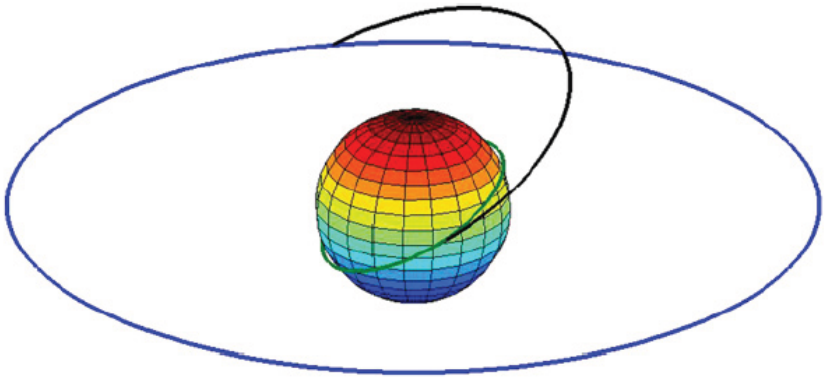


Figure 3.17: Pontani 2D case b reference trajectory [47]

Particle Swarm Optimization (PSO)

For the application of the PSO to that case the same population size and maximum iteration of the previous case were used (see Table 3.3) while the inertia weight and the learning factors were selected through empirical tuning. The algorithm parameter use in are listed in Table 3.18 and the optimal results are reported in Table 3.19.

PSO parameter	Value
Population size	300
Maximum iterations	200
Inertia weight ω	0.7
Cognitive coefficient c_1	1.5
Social coefficient c_2	1.5
Stopping tolerance	10^{-9}

Table 3.18: Parameters of the PSO run (Pontani 3D case)

Transfer detail	Value	Error
ΔV_1 [m/s]	2476.05	0.38 %
ΔV_2 [m/s]	2162.29	0.39 %
ΔV_{tot} [m/s]	4638.35	0.02 %
Δt [s]	18776.95	–
TA_{dep} [°]	101.56	–

Table 3.19: Transfer obtained with PSO (Pontani 3D case)

All the relative error are below 0.4 %, confirming the validty of the PSO also for this case. The computational performance are summarised in the blue box below, as we can see no stopping condition was reached in this case forcing the algorithm to perform all the iterations.

Computational performance (PSO - Pontani 3D)

Metric	Value
Execution time [s]	63.79
Iterations	200

Table 3.20: PSO performance (2D case).

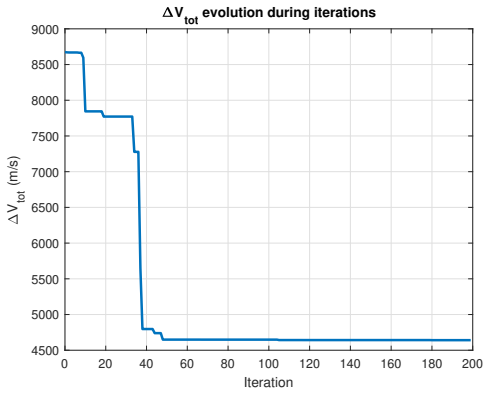


Figure 3.18: PSO cost evolution

Fig. 3.19 shows the optimised trajectory, which is really close to the analytical solution in Fig. 3.17.

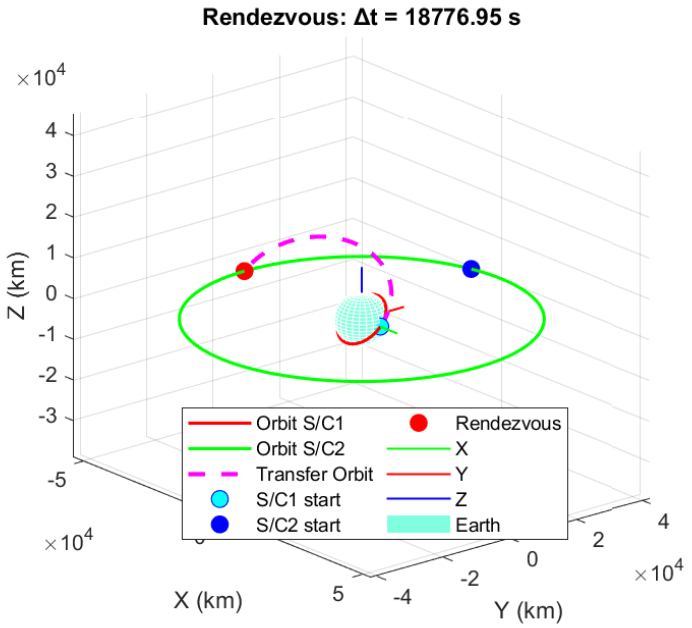


Figure 3.19: Transfer trajectory obtained with PSO (Pontani 3D)

Differential Evolution (DE)

We configured the Differential Evolution (DE) run as previously done in the 2d case. (see Table 3.6).

The resulting transfer performance and computational metrics of the three DE strategies are shown below:

1. *DE/rand/1/bin*:

Transfer detail	Value	Error
ΔV_1 [m/s]	2464.63	0.08 %
ΔV_2 [m/s]	2172.73	0.09 %
ΔV_{tot} [m/s]	4637.36	0.0007 %
Δt [s]	18984.66	–
TA_{dep} [°]	100.68	–

Table 3.21: Transfer obtained with DE/rand/1/bin (Pontani 3D case)

Computational performance (DE/rand/1/bin - Pontani 3D)

Metric	Value
Execution time [s]	49.31
Iterations	199

Table 3.22: DE/rand/1/bin performance (3D case)

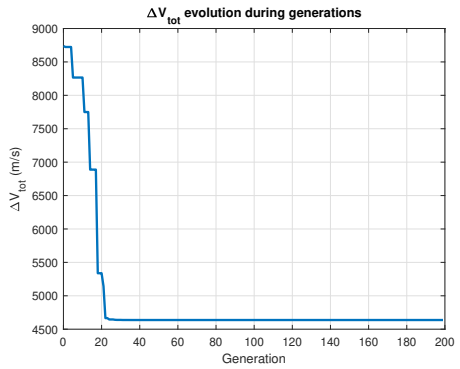


Figure 3.20: DE/rand/1/bin cost evolution

2. *DE/best/1/bin*:

Transfer detail	Value	Error
ΔV_1 [m/s]	2464.63	0.08 %
ΔV_2 [m/s]	2172.73	0.09 %
ΔV_{tot} [m/s]	4637.36	0.0007 %
Δt [s]	18984.45	–
TA_{dep} [°]	100.68	–

Table 3.23: Transfer obtained with DE/best/1/bin (Pontani 3D case)

Computational performance (DE/best/1/bin - Pontani 3D)

Metric	Value
Execution time [s]	33.95
Iterations	142

Table 3.24: DE/best/1/bin performance (3D case)

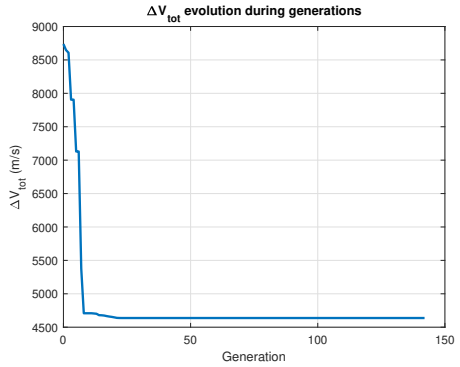


Figure 3.21: DE/best/1/bin cost evolution

3. *DE/current-to-best/1/bin*:

Transfer detail	Value	Error
ΔV_1 [m/s]	2463.17	0.14 %
ΔV_2 [m/s]	2174.21	0.15 %
ΔV_{tot} [m/s]	4637.38	0.01 %
Δt [s]	18973.26	–
TA_{dep} [°]	100.73	–

Table 3.25: Transfer obtained with DE/current-to-best/1/bin (Pontani 2D case)

Computational performance (DE/current-to-best/1/bin - Pontani 3D)

Metric	Value
Execution time [s]	53.96
Iterations	199

Table 3.26: DE/current-to-best/1/bin performance (3D case)

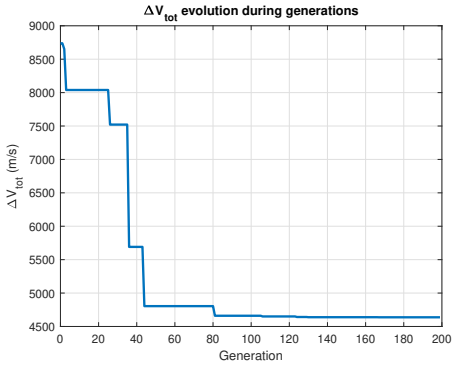


Figure 3.22: DE/current-to-best/1/bin cost evolution

All three strategies achieve acceptable relative errors with *DE/rand/1/bin* and *DE/best/1/bin* offering the lowest ones. Figure 3.23 shows the trajectory from the *DE/rand/1/bin* run; the other strategies produce almost identical transfer paths and are omitted for brevity.

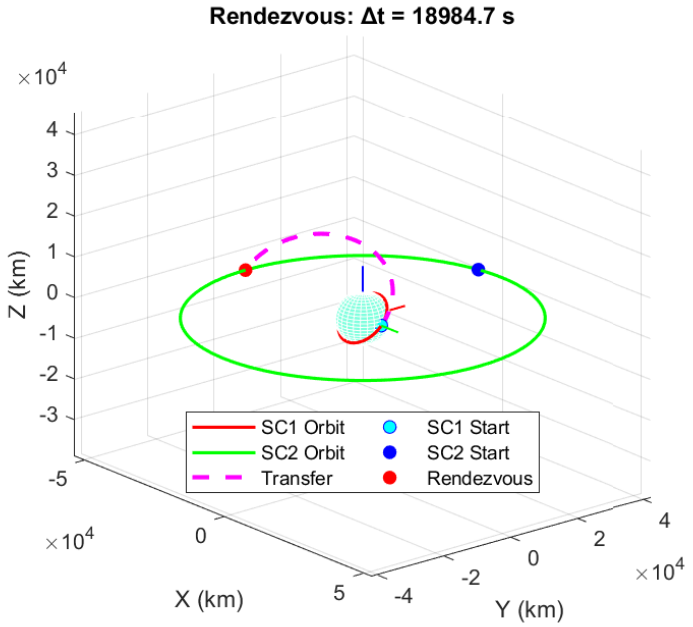


Figure 3.23: Transfer trajectory obtained with DE (Pontani 3D case)

Covariance Matrix Adaptation (CMA-ES)

We applied the CMA-ES algorithm with the default parameter set recommended by Hansen and Ostermeier [55] as previously done in the 2D case incrementing the population size λ after a small tuning process ($\lambda_{2D_{case}} = 4 + \lfloor 3 \log(N) \rfloor \longrightarrow \lambda_{3D_{case}} = 18 + \lfloor 3 \log(N) \rfloor$).

The optimized transfer results for the Pontani 3D case appear in Table 3.27, with computational performance detailed in the accompanying blue box.

Transfer detail	Value	Error
ΔV_1 [m/s]	2464.64	0.08 %
ΔV_2 [m/s]	2172.78	0.09 %
ΔV_{tot} [m/s]	4637.42	0.0006 %
Δt [s]	19055.19	–
TA_{dep} [°]	100.38	–

Table 3.27: Transfer obtained with CMA-ES (Pontani 2D)

Computational performance (CMA-ES – Pontani 3D)

Metric	Value
Execution time [s]	4.41
Iterations	123

Table 3.28: Performance summary

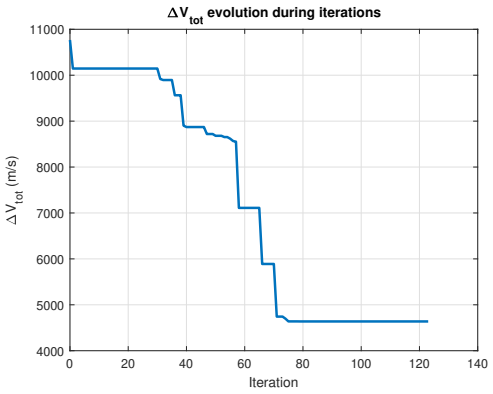


Figure 3.24: CMA-ES cost convergence

Figure 3.25 shows the resulting transfer trajectory, which closely matches the other solutions.

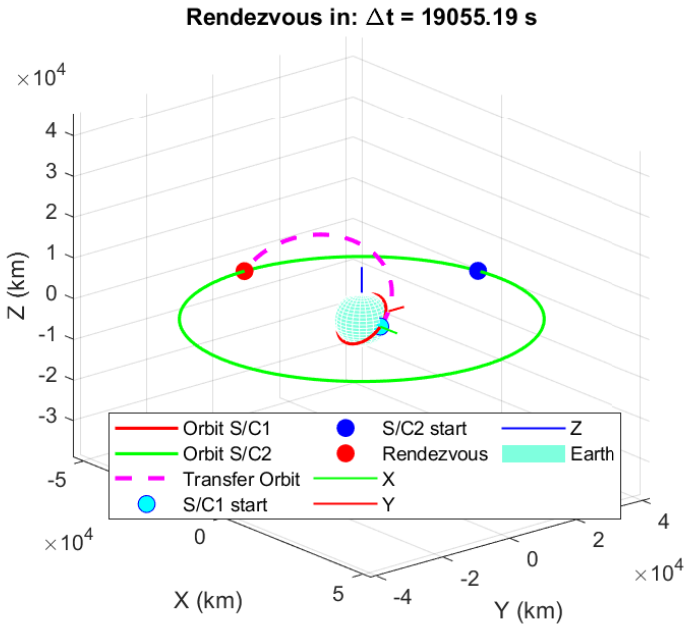


Figure 3.25: Transfer trajectory obtained with CMA-ES (Pontani 3D)

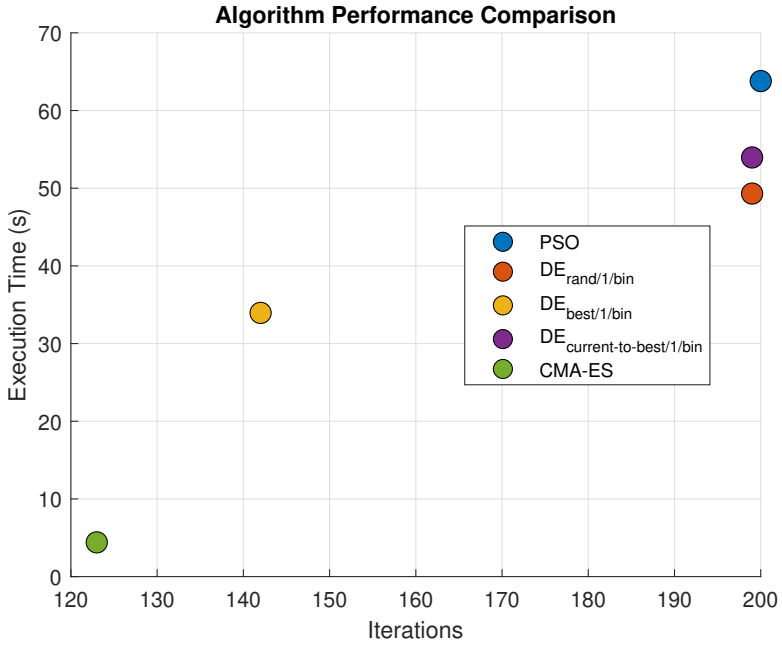


Figure 3.26: Algorithms comparison (Pontani 3D)

From the plotted data in Fig. 3.26, CMA-ES clearly outperforms the others in

wall-clock time (4.41 s) while requiring a moderate number of iterations (123), thanks to its efficient covariance adaptation. DE/best/1/bin achieves the fewest iterations (142) and completes in 33.95 s, demonstrating rapid convergence per iteration. DE/rand/1/bin and DE/current-to-best/1/bin exhibit similar runtimes (49–54 s) at around 199 iterations, reflecting a balanced exploration convergence trade-off. Finally, PSO is the slowest (63.79 s) and executes the full 200 iterations, indicating that, without an early stopping criterion, it is the least efficient for this 3D transfer scenario.

3.5.4 Algorithm Selection

Although CMA-ES demonstrated the shortest runtimes in our benchmarks, we decided to adopt Differential Evolution with both the *rand/1/bin* and *best/1/bin* strategies for two main reasons. First, CMA-ES failed to converge to the true optimum in approximately 1 out of every 100 runs, undermining its reliability in this application. Second, the number of generations required by CMA-ES varied widely from as few as 50 up to nearly 700 making its computational cost unpredictable. By contrast, both *DE/rand/1/bin* and *DE/best/1/bin* consistently reached the optimal solution across all runs, with a stable iteration count and only a modest performance penalty. For these reasons, DE with rand/1 and best/1 strategies (in their parallelized versions) was selected as the primary optimizer in this thesis.

Chapter 4

Travelling Salesman Problem

According to Hou and Misra [57], the classical TSP is one of the most intensively studied problems in computational mathematics due to its broad industrial relevance seeks the permutation of n targets that minimizes the total travel cost. It finds several space mission design applications such as multiple flybys, debris rendezvous tour [58], and asteroid exploration.

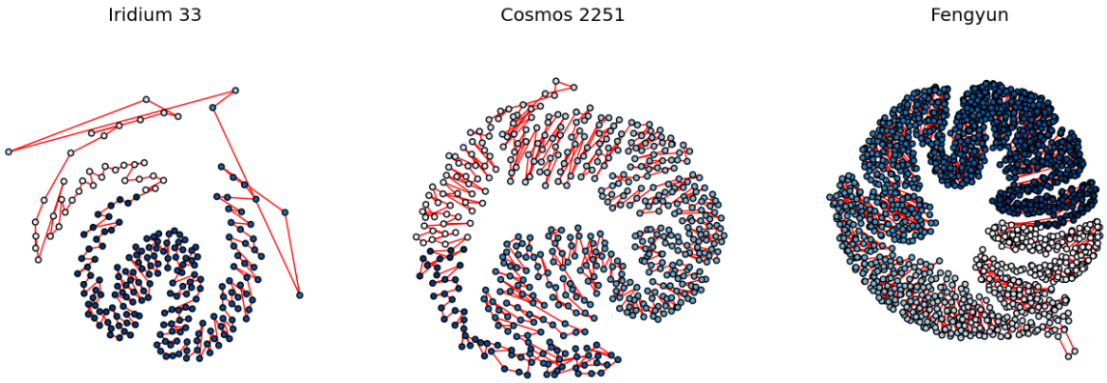


Figure 4.1: Best tours found for the three debris clouds in the classic TSP variant for the fragmentation caused by Iridium 33-Cosmos 2251 collision and the 2007 Fengyun-1C event caused by a Chinese anti-satellite missile test [58]

4.1 TSP formulation

Let $\mathcal{I} = \{1,2,\ldots,N\}$ denote the set of N cities (or nodes). For each ordered pair (i,j), let

$$d_{ij} \geq 0$$

be the cost (e.g. ΔV) of traveling directly from city i to city j. A *tour* is a permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(N))$ of the indices in $\mathcal{I}$. The (open-chain) tour length is

$$J(\pi) \;=\; \sum_{k=1}^{N-1} d_{\pi(k)\,\pi(k+1)}. \tag{4.1}$$

If a closed tour (returning to the start) is required, one adds the term $d_{\pi(N)\,\pi(1)}$.

The *classical TSP* then asks for the permutation π^* that solves

$$\pi^* \;=\; \arg\min_{\pi} \; J(\pi)\,. \tag{4.2}$$

In this thesis work we have combined the TSP with the already illustrated evolutionary algorithms (with the same objective function) in order to solve eq. (4.2).

4.2 TSP logic implementation

Given three different target satellites we want to know the optimal visiting sequence in terms of ΔV. The TSP-EA optimization implemented in MATLAB works as follow:

1. Pre–compute $\Big\{cost_{start}(i),\, \Delta t_{start}(i),\, TA_{dep_{start}}(i)\Big\}_{i=1}^{N}$ from the initial chaser orbit to each of the N targets via an evolutionary algorithm (PSO, DE, CMA-ES).
2. Fill an $N \times N$ cost matrix $cost_{mat}(i,j)$ (and associated $\Delta t_{mat}, TA_{dep_{mat}}$) for each leg $i \to j$, $i \neq j$.
3. Enumerate all $N!$ permutations `permsList`, accumulating for each sequence

$$\texttt{totalCost} = cost_{start}(\pi(1)) + \sum_{k=1}^{N-1} cost_{mat}\Big(\pi(k), \pi(k+1)\Big),$$

   storing also the time of flight Δt and target departure true anomaly TA_{dep} for each leg.
4. Select the minimum-cost sequence.

A more detailed version of the TSP function implemented is depicted in Appendix B

4.2.1 1st Scenario - TSP

The input data of the first case scenario are reported in the Table 4.1

Orbital parameter	Initial orbit	Orbit 1	Orbit 2	Orbit 3
a [km]	18000	35000	40000	45000
e	0.6	0.8	0.7	0.65
i [°]	0	15	25	40
Ω [°]	0	120	90	45
ω [°]	0	10	85	30
ν [°]	158.94	0	45	90

Table 4.1: Orbital elements of the satellites - scen. 1

The algorithm find as best travel cost the sequence

Optimal Visit Sequence

Target **3** $\longrightarrow$ Target **2** $\longrightarrow$ Target **1**

with the following legs details in Table 4.2

Leg	Transfer	Δt [s]	ΔV_{tot} [m/s]	ΔV_1 [m/s]	ΔV_2 [m/s]
1	Chaser $\rightarrow$ Target 3	29227.54	2811.87	1868.65	943.22
2	Target 3 $\rightarrow$ Target 2	91881.79	1922.14	684.50	1237.64
3	Target 2 $\rightarrow$ Target 1	48681.80	1485.32	595.80	889.52

Table 4.2: Summary of transfer legs - scen. 1

and a total cost in terms of ΔV of 6219.32 m/s.
The trajectory combination is illustrated in Figure 4.2

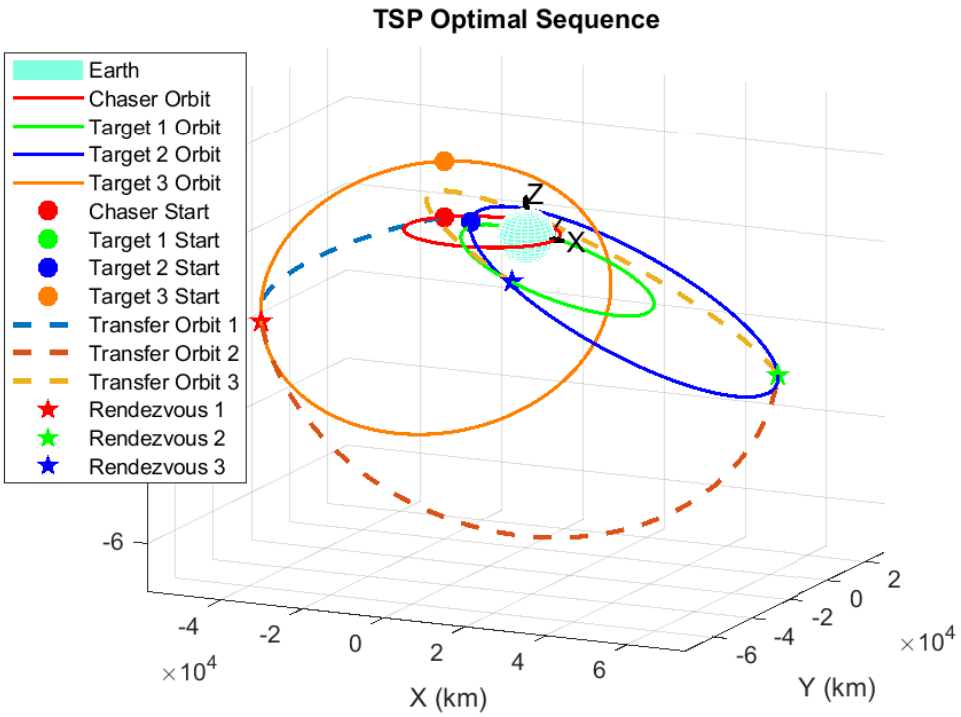


Figure 4.2: TSP trajectory sequence - scen. 1

4.2.2 2nd Scenario - TSP

For the second case scenario more common altitude for the manoeuvres of IOS were chosen. The input data of the second case study are reported in the Table 4.3

Orbital parameter	Initial orbit	Orbit 1	Orbit 2	Orbit 3
a [km]	6928	6778	7078	7168
e	0	0	0	0
i [°]	5	6	4	7
Ω [°]	0	10	5	45
ω [°]	0	0	10	15
ν [°]	10	20	45	0

Table 4.3: Orbital elements of the satellites - scen. 2

The best travel ΔV cost is provided by the sequence

Optimal Visit Sequence

Target **2** ⟶ Target **1** ⟶ Target **3**

with the following legs details in Table 4.4

Leg	Transfer	Δt [s]	ΔV_{tot} [m/s]	ΔV_1 [m/s]	ΔV_2 [m/s]
1	Chaser → Target 2	2322.29	184.88	58.68	126.20
2	Target 2 → Target 1	3671.85	397.92	109.88	288.04
3	Target 1 → Target 3	4207.35	791.54	203.06	588.49

Table 4.4: Summary of transfer legs - scen. 2

Together with a total cost of 1374.34 m/s. The trajectory sequence is shown in Figure 4.3

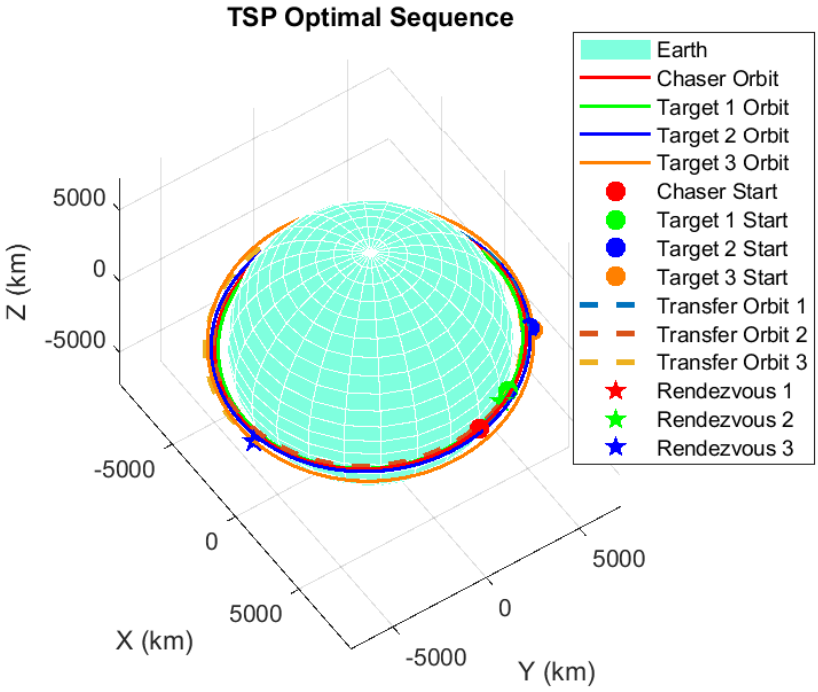


Figure 4.3: TSP trajectory sequence - scen. 2

Chapter 5

3DOF Orbital Simulator

In this chapter we analyze the structure and the workflow of the 3DOF orbital simulator used in this thesis. The aim of the simulator is to perform the two optimizations (TSP + DE) and then execute the orbital propagation via a numerical integrator in which is annidated a GNC function based on the optimization results.

5.1 Orbital Simulator overview

The orbital simulator is structured into five main modules, each implemented as a separate MATLAB function. Figure 5.1 shows the data flow and dependencies between these modules.

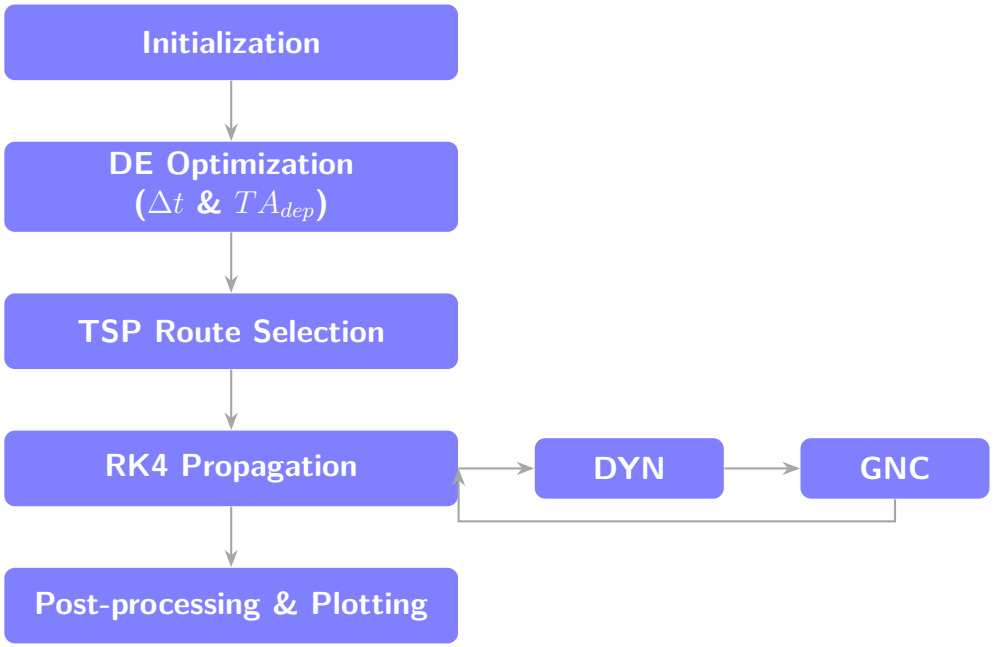


Figure 5.1: Flow chart of the orbital simulator process

The simulator operates as follows:

1. *Initialization and input parameters*
   In the main script, environment constants (`ENV`), chaser spacecraft data (`LV`), target data (`TARGET`), and orbital elements for departure and arrival (`orbit`) are defined. Search bounds (`Xmin`, `Xmax`) and DE settings (`DE.pop_size`, `DE.Cr`, `DE.F`, `DE.eps`, `DE.genmax`, `DE.strategy`) are also configured.

2. *Trajectory Optimization* **(`DE_fun` & `lambert_cost`)**
   The function `DE_fun` implements Differential Evolution (strategies rand/1 or best/1 are used here) to optimize $[\Delta t,\ TA_{\text{dep}}]$. Each candidate is evaluated by `lambert_cost`, which solves the Lambert problem between the initial and target state vectors and returns the total ΔV, the starting ΔV_1 and the final ΔV_2. Once the optimum $[\Delta t^*,\ TA^*_{\text{dep}}]$ is found, the orbital elements of chaser and target are converted to position and velocity vectors via `sv_from_oe` for dynamic simulation.

3. *TSP Route Selection*
   This module works in tandem with the trajectory optimizer to support multi-burn rendezvous scenarios. Given three target satellites, it first computes the optimal ΔV and time-of-flight for each possible transfer leg (see Section 4.2). It then reorders the list of targets so that the visitation sequence minimizes the total mission cost.

4. *Dynamic Propagation* (`RK4` + `DYN` + `GNC`)
   The time evolution of the chaser–target system is integrated using a fourth-order Runge–Kutta scheme (`RK4`), which invokes:
   - `DYN`: computes gravitational, atmospheric drag, and propulsion accelerations based on the current state and throttle/direction command.
   - `GNC`: a guidance module operating in three phases (burn-1, coasting, burn-2), producing at each step the control vector $U = [u_{\text{thr}}, \mathbf{d}_{\text{ECI}}]$.
5. *Post-Processing and Plotting*
   After propagation, key outputs (ΔV expended, final mass, relative velocities, final orbital elements errors) are extracted. The routine `plot simulation results` then generates the cost evolution, 3D trajectory, and time histories of position and velocity.

Detailed descriptions of the RK4 integrator, dynamic model and GNC function are provided in the following sections.

5.2 RK4 numerical integrator

Given an initial value problem for an ordinary differential equation,

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0,$$

we seek a numerical approximation to the true solution $\mathbf{y}(t)$ over an interval $[t_0, t_{\text{end}}]$. In many practical applications such as orbital propagation under complex forces closed-form solutions are unavailable, and we turn to step-by-step integration schemes.

A general one-step method advances the solution from t_n to $t_{n+1} = t_n + dt$ by

$$\mathbf{y}_{n+1} \;=\; \mathbf{y}_n \;+\; \Delta t\, \Phi\Big(t_n,\, \mathbf{y}_n,\, dt\Big),$$

where the increment function Φ depends on values of $\mathbf{f}$ at one or more points. The simplest case, the explicit Euler method,

$$\Phi_{\text{Euler}} = \mathbf{f}\Big(t_n, \mathbf{y}_n\Big),$$

is only first-order accurate (global error $\mathcal{O}(dt)$) and can be unstable for stiff or rapidly varying dynamics.

Runge–Kutta methods improve accuracy by sampling $\mathbf{f}$ at intermediate stages within each step. A general explicit s-stage Runge–Kutta method is given by

$$\mathbf{F}_i = \mathbf{f}\Big(t_n + c_i\, dt,\ \mathbf{y}_n + dt \sum_{j=1}^{i-1} a_{ij}\, \mathbf{F}_j\Big), \quad i = 1, \ldots, s,$$

$$\mathbf{y}_{n+1} = \mathbf{y}_n + dt \sum_{i=1}^{s} b_i\, \mathbf{F}_i,$$

The four-stage method (RK4) achieves fourth-order convergence ($p = 4$) with a minimal number of function evaluations, making it a popular choice for orbit propagation and other problems requiring a good balance of accuracy and efficiency [59].

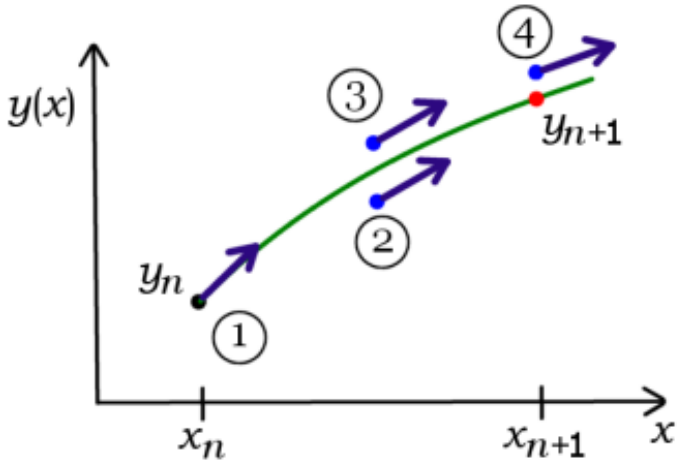


Figure 5.2: The fourth-order Runge–Kutta method evaluates the derivative four times once at the initial point, twice at two intermediate trial points, and once at the final trial point and then combines these four estimates to compute the next value of the unknown y [59]

In this work simulator the numerical integrator advances the state vector $\mathbf{S}(t)$ over a step of size h by computing four increments:

$$\mathbf{F}_1 = f\Big(t_n, \mathbf{S}_n\Big),$$
$$\mathbf{F}_2 = f\Big(t_n + \tfrac{h}{2},\ \mathbf{S}_n + \tfrac{h}{2}\mathbf{F}_1\Big),$$
$$\mathbf{F}_3 = f\Big(t_n + \tfrac{h}{2},\ \mathbf{S}_n + \tfrac{h}{2}\mathbf{F}_2\Big),$$
$$\mathbf{F}_4 = f\Big(t_n + h,\ \mathbf{S}_n + h\,\mathbf{F}_3\Big),$$

and then updates the state as

$$\mathbf{S}_{n+1} \;=\; \mathbf{S}_n \;+\; \frac{h}{6}\Big(\mathbf{F}_1 + 2\mathbf{F}_2 + 2\mathbf{F}_3 + \mathbf{F}_4\Big),$$

where $f(t, \mathbf{S})$ represents the time-derivative provided by the dynamic model explained in the following section.

5.3 Dynamic model

The chaser position $\mathbf{r}_C$, chaser velocity $\mathbf{v}_C$, target position $\mathbf{r}_T$, target velocity $\mathbf{v}_T$, and chaser mass m_C are comprised in the state vector $\mathbf{S} = \Big[\mathbf{r}_C, \mathbf{v}_C, \mathbf{r}_T, \mathbf{v}_T, m_C\Big]^T \in \mathbb{R}^{13}$. Almost the same dynamic model was provided for the chaser and target satellite, the only difference lies in the absence of propulsion acceleration for the target which makes it *passive*.

Forces and Accelerations

Gravity

It is possible to select the gravity model in order to take account or not for the J_2 for a high fidelity disturbance model. The auxiliary function `GRAV` provides the gravity acceleration with the J_2 *effect* (see Section 2.2.2).

$$\mathbf{g}(\mathbf{r}) = \begin{cases} \texttt{GRAV}(\mathbf{r}), & \text{if high-fidelity model,} \\ -\dfrac{\mu\,\mathbf{r}}{\|\mathbf{r}\|^3}, & \text{otherwise.} \end{cases}$$

Aerodynamic Drag

The contrbution of this disturbance is stronger at low altitude where the residual atmospheric density is higher (see Section 2.2.1).

$$\mathbf{a}_{\text{drag}} = -\frac{1}{2}\,\rho\, C_D A\, \frac{\mathbf{v}_{\text{rel}}\,\|\mathbf{v}_{\text{rel}}\|}{m}.$$

where $\mathbf{v}_{\text{rel}}$ account for Earth's rotation rate ω_{E}:

$$\mathbf{v}_{\text{rel}} = \mathbf{v} + \omega_{\text{E}}\,\hat{\mathbf{k}} \times \mathbf{r} \quad \Longrightarrow \quad \begin{cases} v_{\text{rel,x}} = v_x + \omega_{\text{E}}\, y, \\ v_{\text{rel,y}} = v_y - \omega_{\text{E}}\, x, \\ v_{\text{rel,z}} = v_z. \end{cases}$$

Propulsion (Chaser only)

Given the throttle $U(1 = \texttt{ON},\ 0 = \texttt{OFF})$ and commanded direction $\hat{\mathbf{d}}$ from the GNC algorithm:

$$\mathbf{a}_{\text{prop}} = \begin{cases} \dfrac{T}{m_C}\,\hat{\mathbf{d}}, & u = 1, \\ \mathbf{0}, & \text{otherwise} \end{cases} \qquad \dot{m}_C = \begin{cases} -\,\dot{m}_{\text{prop}}, & u = 1, \\ 0, & \text{otherwise} \end{cases}$$

Collecting all contributions, the time-derivative $\dot{\mathbf{S}} = f(t, \mathbf{S}, U)$ is

$$\begin{aligned} \dot{\mathbf{r}}_C &= \mathbf{v}_C, \\ \dot{\mathbf{v}}_C &= \mathbf{g}(\mathbf{r}_C) + \mathbf{a}_{\text{prop}} + \mathbf{a}_{\text{drag,C}}, \\ \dot{m}_C &= -\,\dot{m}_{\text{prop}_{\{u=1\}}}, \\ \dot{\mathbf{r}}_T &= \mathbf{v}_T, \\ \dot{\mathbf{v}}_T &= \mathbf{g}(\mathbf{r}_T) + \mathbf{a}_{\text{drag,T}}. \end{aligned}$$

5.4 Guidance algorithm

This section describes the MATLAB function `GNC`, which implements a multi-phase guidance algorithm for an orbital rendezvous manoeuvre. The function computes at each time t:

$$[\mathbf{U},\ \texttt{GNC\_DATA},\ \mathbf{v}_{\text{target\_now}}] \ =\ \texttt{GNC}\Big(t, \mathbf{S}, \texttt{GNC\_DATA}, \texttt{LV}, \texttt{ENV}, \texttt{orbit}\Big),$$

where:

- $\mathbf{S} = [\mathbf{r}_C;\ \mathbf{v}_C;\ \mathbf{r}_T;\ \mathbf{v}_T;\ m_C]$ is the 13-element state (chaser/target positions and velocities, chaser mass).
- $\mathbf{U} = [U_{\text{throttle}};\ \hat{\mathbf{d}}_{\text{ECI}}]$ is the control vector: a binary throttle command and a thrust direction in ECI.
- $\mathbf{v}_{\text{target\_now}}$ is the desired "target" velocity here used for diagnostics.
- `GNC_DATA` holds persistent data (phase, time of flight Δt, time-step dt, target *argument of latitude* θ, target ΔVs from the optimization process, accumulated ΔVs).
- `LV`, `ENV`, `orbit` provide vehicle, environment, and orbital parameters.

Multi-phase structure

The algorithm proceeds through four sequential phases here summarized:

Phase 1 – First Burn

1. Initialized the required $\Delta\mathbf{v}_1$ given by the optimization ($\Delta\mathbf{v}_{1_{target}}$).
2. Compute remaining $\mathbf{\Delta v}_1 = \Delta\mathbf{v}_1 - \Delta\mathbf{v}_{applied}$ and set $dt = 1s$ for a small integration step during burns [1].
3. If $\|\mathbf{\Delta v}_1\| \leq \|\mathbf{\Delta v}_{1_{target}}\|$, set $\hat{\mathbf{d}} = \mathbf{\Delta v}_1/\|\mathbf{\Delta v}_1\|$, throttle on ($U = 1$), and increment the applied ΔV by $\frac{T}{m_c}\,dt$.
4. Else switch throttle off, mark burn complete, set $\Delta t = 10s$, advance to Phase 2.

Phase 2 – Coasting

- Throttle off, $dt = 10s$ (bigger integration step for coasting phase).
- Compute current argument of latitude θ from orbital elements.
- When $|\theta - \theta_{\text{target}}| < 0.5°$, switch to Phase 3.

Phase 3 – Second Burn

1. Initialize required $\Delta\mathbf{v}_2$ given by the optimization ($\Delta\mathbf{v}_{2_{target}}$) and set $dt = 1s$.
2. Compute the velocity error $\mathbf{v}_{error} = \mathbf{v}_T - \mathbf{v}_C$ for chasing the target velocity.
3. If $\|\mathbf{\Delta v}_2\| \leq \|\mathbf{\Delta v}_{2_{target}}\|$ set $\hat{\mathbf{d}} = \mathbf{v}_{error}/\|\mathbf{v}_{error}\|$, throttle on ($U = 1$), and increment the applied ΔV by $\frac{T}{m_c}\,dt$.
4. Else throttle off, mark burn complete, set $dt = 10s$, advance to Phase 4.

Phase 4 – Idle

- Throttle off, coasting with $dt = 10s$ until end of manoeuvre (t $\geq \Delta t$) .

A better view provided from the MATLAB code can be found in Appindix C.2.

[1] Initially the logic of the first burn were different, in fact the early trials were conducted by calling the Lambert equation at each step but due to numerical issues this method was abandoned. A short in-depth analysis about it can be found in Appendix C.1

5.4.1 Flowchart of the GNC algorithm

The following diagram summarizes the phase-based workflow of the `GNC` function:

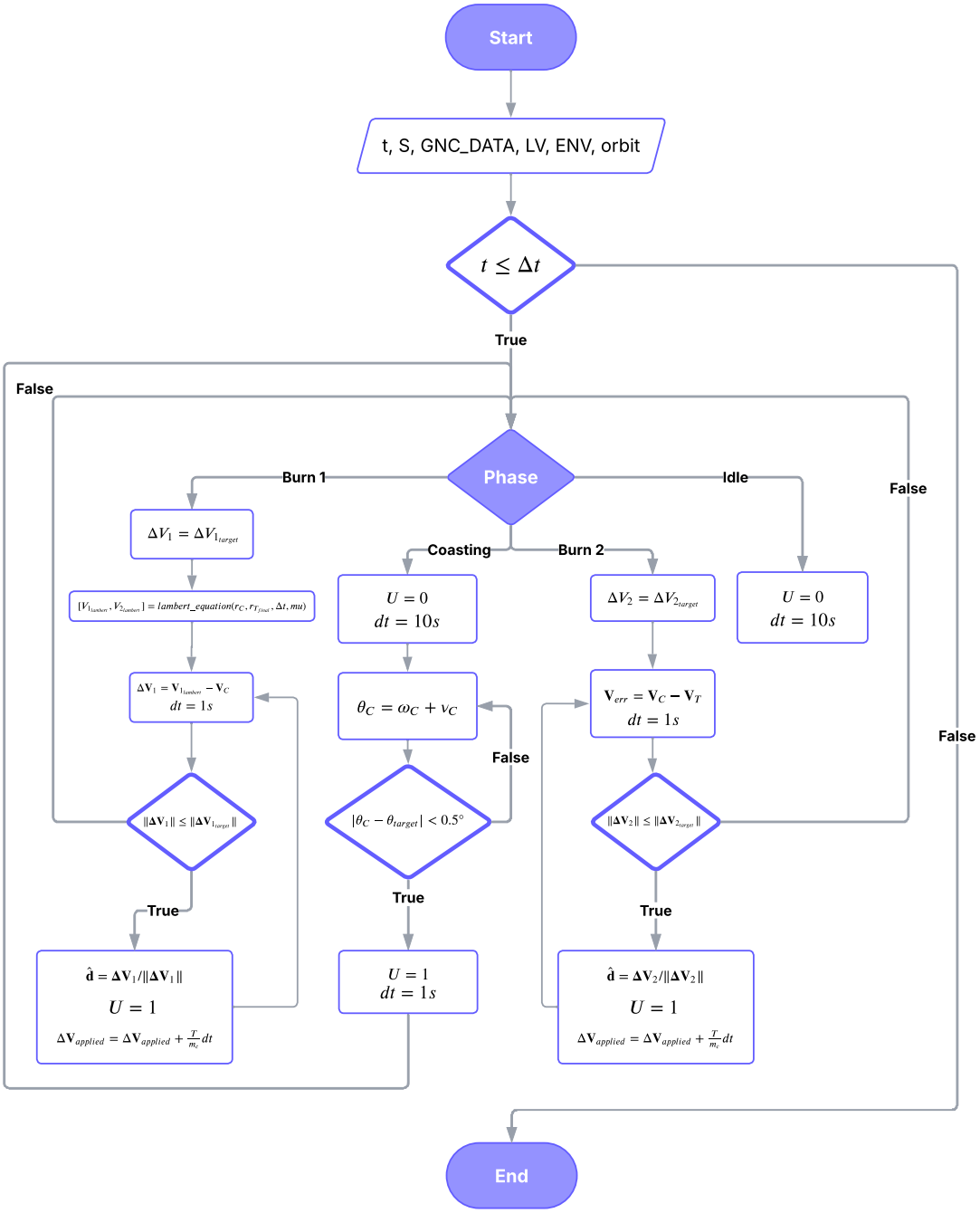


Figure 5.3: Guidance flowchart

5.5 Auxiliary functions

In the previous chapters and sections we have illustrated the main function modules of the propagator used in this work. For the sake of completeness we briefly describes in Table 5.1 all the auxiliary functions that were used.

Function	Description
`DE_fun`	execute the Differential Evolution optimization in combination with the *objective function*
`DYN_target`	contains the dynamic model of the Target satellite and together with `RK4_target` provide a ground propagation of the Target
`GRAV`	contains the gravity model for considering the J_2 effect
`kepler_equation`	solve the Kepler Equation
`lambert_equation`	solve the Lambert problem
`oe_from_sv`	provide the orbital elements given the state vector
`plot_simulation_results`	contains all the useful propagation plots
`RK4_target`	numerical integrator that together with `DYN_target` provide a ground propagation of the Target
`sv_from_oe`	provide the state vector given the orbital parameters

Table 5.1: Auxiliary functions overview

Chapter 6

Simulations results

After having introduced in the previous chapters the structure of the thesis work and its focus on the creation of a reliable orbital simulator which could perform typical IOS manoeuvres optimizing both the both and the cost in terms of ΔV_{tot}, we present in this section a series of possible IOS scenarios tested in the simulator. For each test will be described:

- Chaser and Target spacecraft details and starting orbital parameters;
- Optimization results: TA_{dep}, ΔVs, $time\ of\ flight$. These last two variables, together with the Argument of Latitude provided by an ideal ground propagation of the Target, could be subjected to a simple tuning process for compensating the disturbances effects. For the propagation that follows a TSP optimization will be also showed the *optimal visiting sequence*;
- Manoeuvre results and comparison between actual and desired variable (e.g. *final relative velocity and final orbital parameters*);
- Plot comments (trajectory, velocity evolution, orbital elements evolution, etc.).

The enviroment in which all the test cases are performed in characterized by the following data:

Enviroment data	Value
R_e (Earth radius)	6378137 m
ω_{earth} (Earth's angular velocity)	$7.292115 \times 10^{-5}\,\text{rad/s}$
μ (Earth's gravitational parameter)	$3.986005 \times 10^{14}\,\text{m}^3/\text{s}^2$
ρ (atmospheric density)	$1 \times 10^{-12}\,\text{kg/m}^3$
g_0 (standard gravity)	$9.80665\,\text{m/s}^2$

Table 6.1: Environmental constants used in the simulation

In the majority of the simuation the following approximation of the *AVIO AVUM* data have been used:

Spacecraft data	Value
Initial mass m_0 [kg]	2500
Propellant mass m_{prop} [kg]	580
Thrust [N]	400
Specific impulse I_{sp} [s]	210
Aerodynamic coefficient $(S * C_D)$	7.6

Table 6.2: Approximation of AVUM vehicle (AVIO)

For all the scenarios considered the mass and the aerodynamic coefficient of the Target satellite remains constants and equal to the Chaser one.

6.1 Test case I

In this first case a simple semimajor axis a raise is performed (or energy raise $\mathcal{E} = -\frac{\mu}{2a}$) for bringing the Chaser to a correct rendezvous with the Target.

Orbital element	Chaser Orbit	Target Orbit
a [km]	7278.173	7378.173
e	0.1	0.1
i [°]	0	0
RAAN [°]	0	0
ω [°]	0	0
ν [°]	0	TBO

Table 6.3: Starting orbital elements of Chaser and Target

In Table 6.3[1] are shown the starting orbital parameters of Chaser and Target. In this scenario the Chaser will perform the manoeuvre at the periapsis, where this kind of manoeuvre is more convenient because it is the fastest point on the orbit [60], for raising the apoapsis and arriving to a higher semimajor axis.

Optimization results	Value
ToF (s)	3108.75
Corrected ToF (s)	3108.75 (+0.0%)
TA_{dep} (°)	2.00
ΔV_{tot} (m/s)	50.06
ΔV_1 (m/s)	25.08
ΔV_2 (m/s)	24.98
Corrected ΔV_1 (m/s)	26.06 (+3.91%)
Corrected ΔV_2 (m/s)	26.35 (+5.48%)
θ (°)	179.2323
Corrected θ (°)	172.2422 (–3.9%)

Table 6.4: Optimization results for the manoeuvre

As previously said the Argument of Latitude θ in Table 6.4, which determines the starting of the second burn in the guidance algorithm, is not provided by the optimization but with an ideal ground propagation of the Target. The correction of this parameter together with the one of the time of flight and the ΔVs is crucial for partially compensate the disturbances of J_2 and drag.

Simulation results	Value
Simulated final time (s)	3105.10 s
Estimated ToF (s)	3108.75
Final chaser mass (kg)	2462.12
Final relative velocity (m/s)	0.437
Total ΔV expended (m/s)	52.39
Estimated ΔV (m/s)	50.06

Table 6.5: Simulation performance results

Analysing the Table 6.5 it is possible to notice that the ΔV_{tot} erogated during the simulation is close to value expected by the optimization with a relative error of *4.65%*. The propellant mass used in this manoeuvre is about *37.88 kg* with a remaing propellant of *542.12 kg*.

[1]Here with *TBO* we meant every parameter that will be provided by the optimization

Orbital parameter	Desired	Achieved	Error
a [km]	7378.1726	7378.2394	0.0668
e	0.1000	0.0973	0.0027
i [°]	0.0001	0.0001	0.0000
RAAN [°]	0.0000	359.7738	0.2262
ω [°]	360.0000	0.7754	0.7755
ν [°]	179.2323	179.0035	0.2289
θ [°]	179.2323	179.7789	0.5466

Table 6.6: Orbital parameters – desired vs achieved

In Table 6.6 are depicted the achieved orbital parameters against the desired one. It is clear that the apoapsis raise manoeuvre is permformed correctly, in fact the errors on a and e which determine the geometry of the orbit are small. Even though the error on the Argument of Latitudine is less then 1 degree (0.5466°) a rendezvous manoeuvre feels the effect of this. A low value of relative velocity indicates the reliability of the the second burn logic in the guidance algorithm with a better view provided in Figure 6.1 and 6.2.

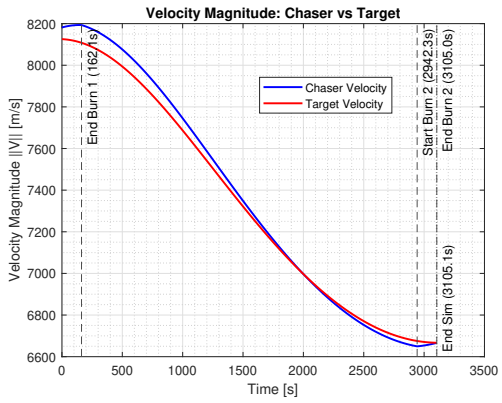


Figure 6.1: Propagation velocity evolution

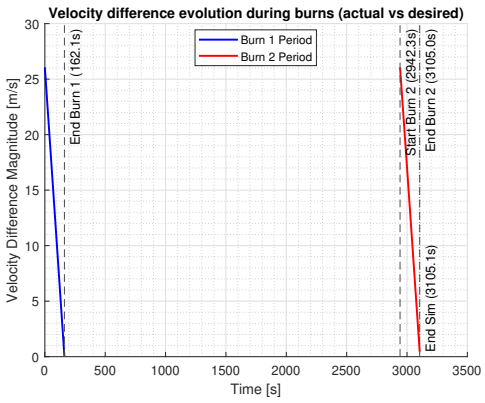


Figure 6.2: Propagation velocity chasing

For a first performance evaluation, looking at the ΔVs (*propellant consumption*) and ToF values obtained this manoeuvre could be considered strongly affordable for our spacecraft. In the Figures below are illustrated the rendezvous trajectory and the evolution of the orbital elements. From a geometric point of view it is important to notice in Figure 6.4 that the transfer orbit eccentricity is higher than the eccentricity of the two main orbits with clear similarity with the Hohmann transfer. In fact when two orbits share the same apsis line the solution is called a *near-Hohmann transfer* [61].

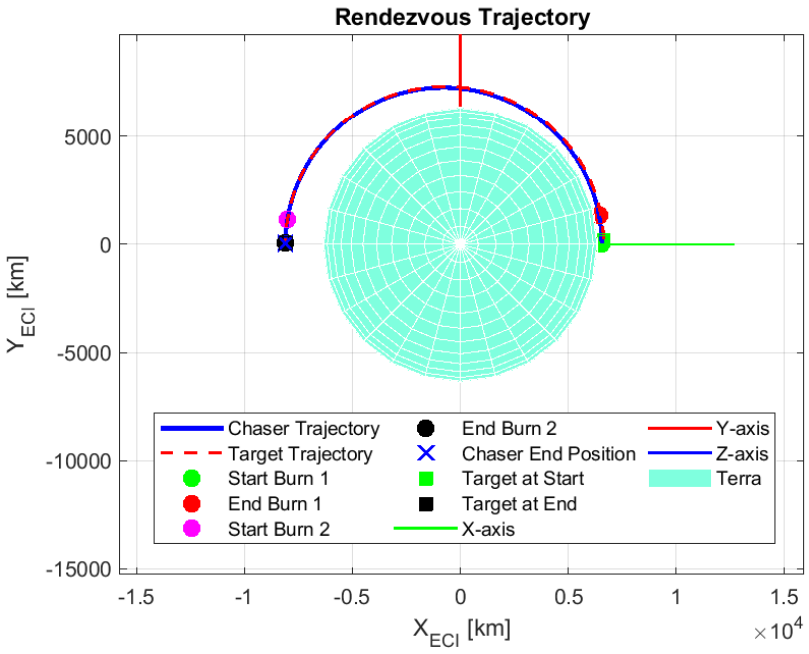


Figure 6.3: Rendezvous trajectory

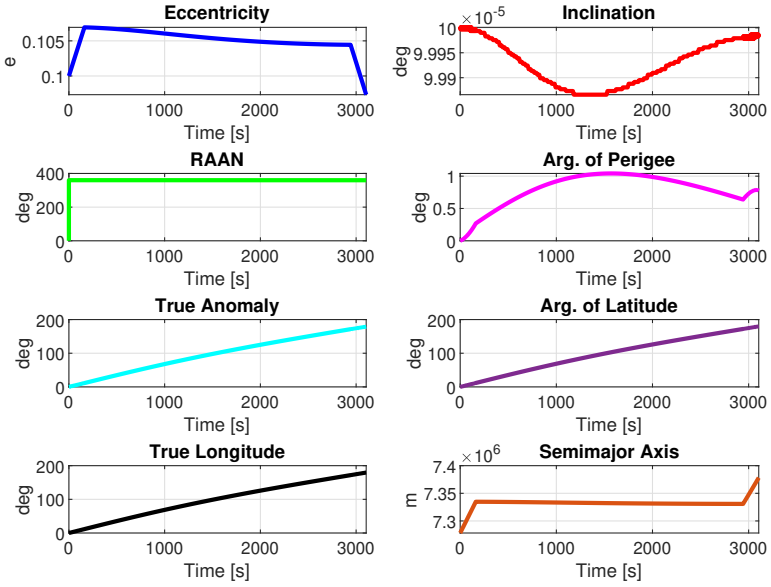


Figure 6.4: Orbital elements evolution

6.2 Test case II

As second case a classical Hohmann transfer is performed at high LEO altitude.

Orbital element	Chaser Orbit	Target Orbit
a [km]	8328.173	8448.173
e	0.0001	0.0001
i [°]	0	0
RAAN [°]	0	0
ω [°]	0	0
ν [°]	0	TBO

Table 6.7: Starting orbital elements of Chaser and Target

In Table 6.7 are shown the starting orbital parameters of Chaser and Target. The eccentricity is set to a very small value (*0.001*) and not zero to avoid numerical issues in the Lambert solver. From literature [46] (see Section 2.3.1) we expect the following ΔV results to represent the most consumption efficient combination:

$$\text{Initial circular velocity:} \quad v_1 = \sqrt{\frac{\mu}{r_1}},$$

$$\text{Final circular velocity:} \quad v_2 = \sqrt{\frac{\mu}{r_2}}.$$

The semi–major axis of the transfer ellipse is

$$a_t = \frac{r_1 + r_2}{2}.$$

The velocities on the transfer ellipse are:

$$\text{At perigee (injection):} \quad v_p = \sqrt{\mu\left(\frac{2}{r_1} - \frac{1}{a_t}\right)},$$

$$\text{At apogee (circularization):} \quad v_a = \sqrt{\mu\left(\frac{2}{r_2} - \frac{1}{a_t}\right)}.$$

The required impulses are

$$\Delta V_1 = v_p - v_1,$$

$$\Delta V_2 = v_2 - v_a,$$

and the total

$$\Delta V_{\text{tot}} = \Delta V_1 + \Delta V_2.$$

with this numerical values

$$\begin{aligned} \Delta V_1 &= 24.70 \text{ m/s}, \\ \Delta V_2 &= 24.61 \text{ m/s}, \\ \Delta V_{\text{tot}} &= 24.70 + 24.61 = 49.31 \text{ m/s}. \end{aligned}$$

This analytical results has been almost totally reached as we can see in Table 6.8

Optimization results	**Value**
ToF (s)	3805.59
Corrected ToF (s)	3805.59 (+0.0%)
TA_{dep} (°)	1.90
ΔV_{tot} (m/s)	49.32
ΔV_1 (m/s)	24.71
ΔV_2 (m/s)	24.61
Corrected ΔV_1 (m/s)	24.83 (+0.5%)
Corrected ΔV_2 (m/s)	25.28 (+2.7%)
θ (°)	179.1795
Corrected θ (°)	172.1915 (–3.9%)

Table 6.8: Optimization results for the manoeuvre

Looking at Table 6.9 it is possible to note that also for this manoeuvre the ΔV_{tot} used in the numerical propagation is close to the value expected with a relative error of *0.63%* and the propellant consumption is about *59.53 kg*, demonstrating again the affordability of this kind of manoeuvre for the spacecraft.

Simulation results	**Value**
Simulated final time (s)	3796.60
Estimated ToF (s)	3805.59
Final chaser mass (kg)	2440.47
Final relative velocity (m/s)	0.064
Total ΔV expended (m/s)	49.63
Estimated ΔV (m/s)	49.32

Table 6.9: Simulation performance results

In the table below are reported the residual errors on the orbital elements. Even though the errors seems to be acceptable we must remember that for a perfect rendezvous the errors on the Argument of Latitude and Eccentricity should be

decidedly lower but our aim is to have a first glance performance evaluation that aligns with the expected behaviour, so we consider this values acceptables.

Orbital parameter	Desired	Achieved	Error
a [km]	8448.1724	8448.0930	0.0795
e	0.0001	0.0021	0.0020
i [°]	0.0001	0.0001	0.0000
RAAN [°]	0.0000	359.8310	0.1690
ω [°]	359.9756	145.8141	145.8385
ν [°]	179.2039	33.5108	145.6931
θ [°]	179.1795	179.3249	0.1453

Table 6.10: Orbital parameters – desired vs achieved

Figure 6.5 and 6.6 together with a low value of final relative velocity, show the consistency of the GNC logic.

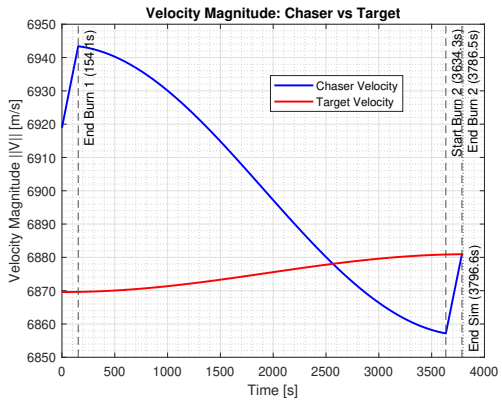


Figure 6.5: Propagation velocity evolution

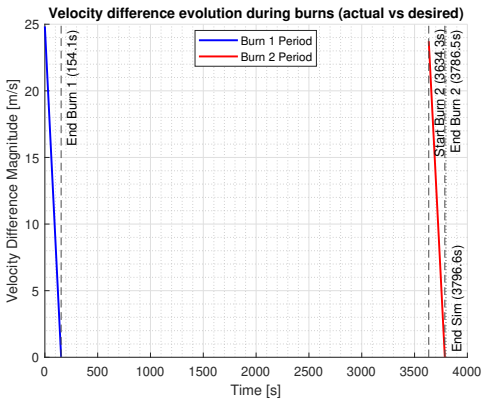


Figure 6.6: Propagation velocity chasing

In the figures below are depicted the rendezvous trajectory and the evolution of the orbital elements. From a geometric point of view the trajectory perfectly matches the expected Hohmann [46].

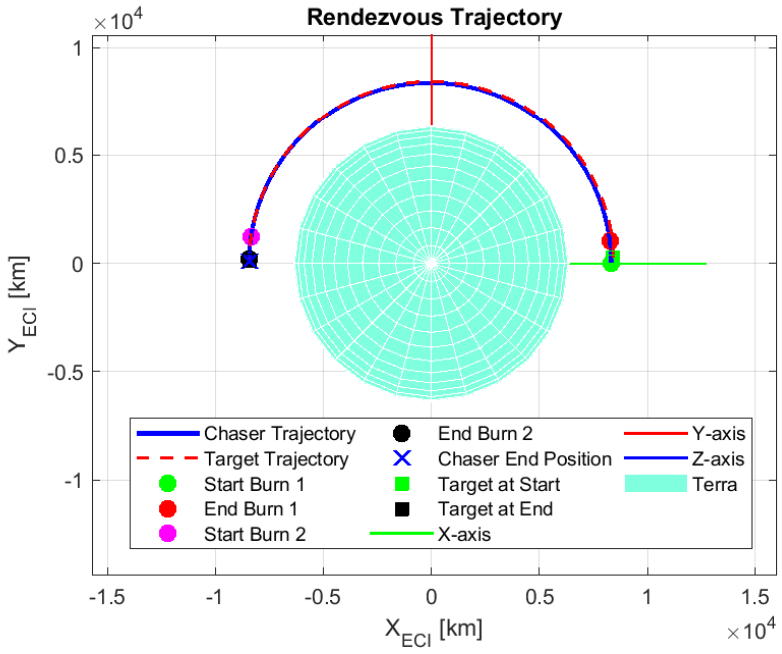


Figure 6.7: Rendezvous trajectory

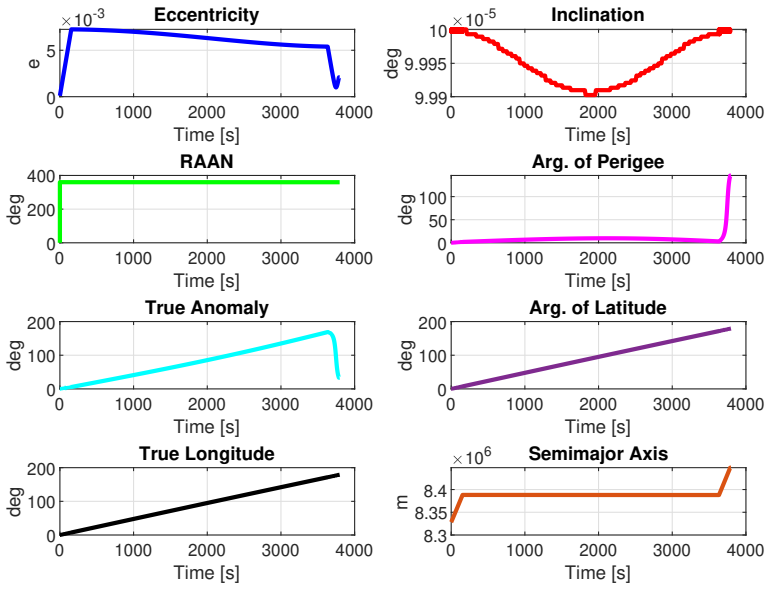


Figure 6.8: Orbital elements evolution

6.3 Test case III

In the third test case, a combined plane-change and apoapsis-raise manoeuvre is executed. Theory [60] indicates that the most energy-efficient way to raise the apogee and change the orbital plane simultaneously is to split the plane rotation between the first and second burns, so that the transfer occurs in an intermediate plane; consequently, the second burn typically requires a larger ΔV than the first. Therefore, we expect the optimization to yield $\Delta V_2 > \Delta V_1$.

The two orbits data are reported in the table below

Orbital element	Chaser Orbit	Target Orbit
a [km]	7378.173	7478.173
e	0.1	0.1
i [°]	5.2	5.8
RAAN [°]	0	0
ω [°]	0	0
ν [°]	0	TBO

Table 6.11: Starting orbital elements of Chaser and Target

The theoretical prediction made in the introduction to this case is confirmed in Table 6.12. The second impulse in 50% more expensive in terms of consumption then the first one due to the plane change manoeuvre even though the Δi is relatively small.

Optimization results	Value
ToF (s)	3188.71
Corrected ToF (s)	3188.71 (+0.0%)
TA_{dep} (°)	2.00
ΔV_{tot} (m/s)	89.62
ΔV_1 (m/s)	35.51
ΔV_2 (m/s)	54.10
Corrected ΔV_1 (m/s)	36.66 (+3.22%)
Corrected ΔV_2 (m/s)	55.36 (+2.33%)
θ (°)	179.9952
Corrected θ (°)	164.6956 (–8.5%)

Table 6.12: Optimization results for the manoeuvre

From Tables 6.13 and 6.14 it is possible to appreciate the propagation results. The predicted ΔV_{tot} was almost the same as the actual one with a discrepancy of just 2.41% and the final relative velocity is near zero. The propellant consumption was

about *108.98 kg* and the two key parameters of this manoeuvre, a and i, present relative errors respectively of 0.0002% and 0.012%, making this kind of scenario affordable for the spacecraft.

Simulation results	Value
Simulated final time (s)	3181.20
Estimated ToF (s)	3188.71
Final chaser mass (kg)	2391.02
Final relative velocity (m/s)	0.004
Total ΔV expended (m/s)	91.78
Estimated ΔV (m/s)	89.62

Table 6.13: Simulation performance results

Orbital parameter	Desired	Achieved	Error
a [km]	7478.1725	7478.1556	0.0170
e	0.1000	0.0974	0.0026
i [°]	5.8000	5.7993	0.0007
RAAN [°]	0.0000	359.5033	0.4967
ω [°]	360.0000	1.5046	1.5046
ν [°]	179.9952	179.1636	0.8317
θ [°]	179.9952	180.6682	0.6730

Table 6.14: Orbital parameters – desired vs achieved

As previously done, here are reported the evolutions of the velocity vector magnitude of Chaser and Target and the velocity difference between actual and desired provided by the guidance algorithm

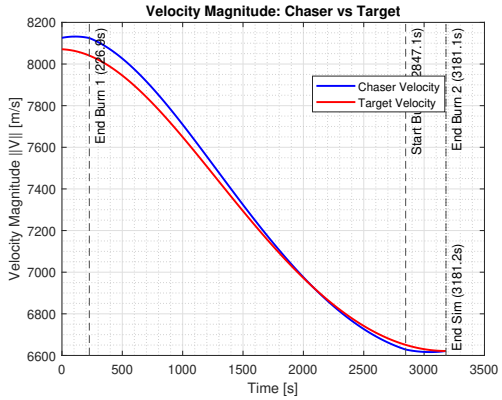


Figure 6.9: Propagation velocity evolution

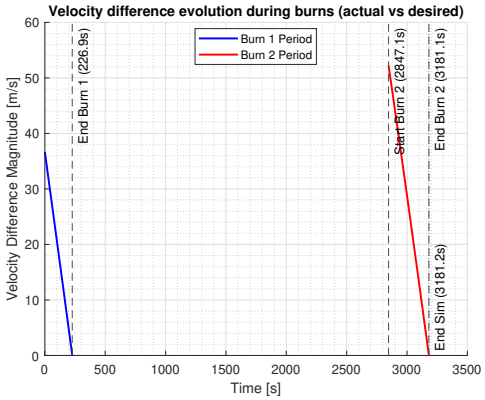


Figure 6.10: Propagation velocity chasing

Altough it's difficult to appreciate the plane change form the trajectory plot (Fig. 6.11) the evolution of the orbital elements provided in the last figure illustrates the physics described previously. In fact it is possible to see the inclination varying mostly in the second burn with a small increase in the first one.

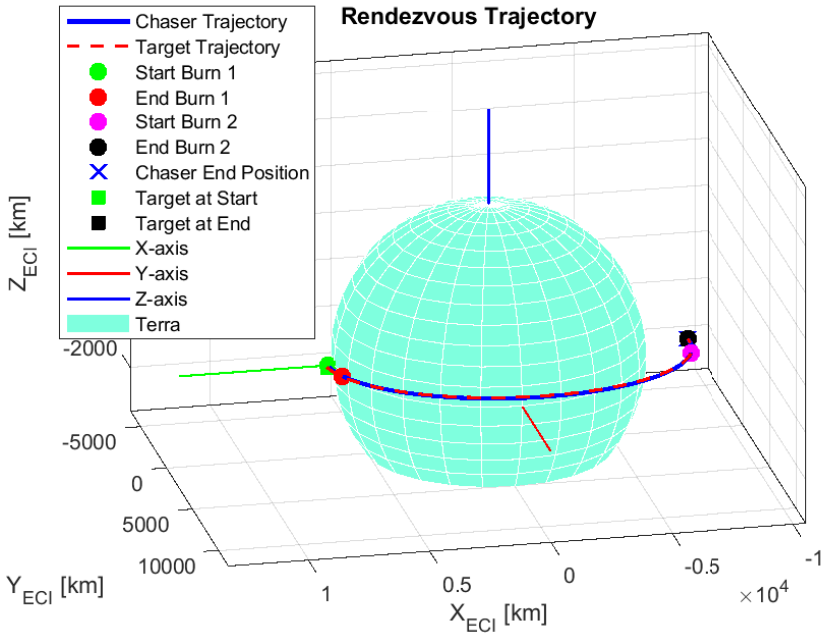


Figure 6.11: Rendezvous trajectory

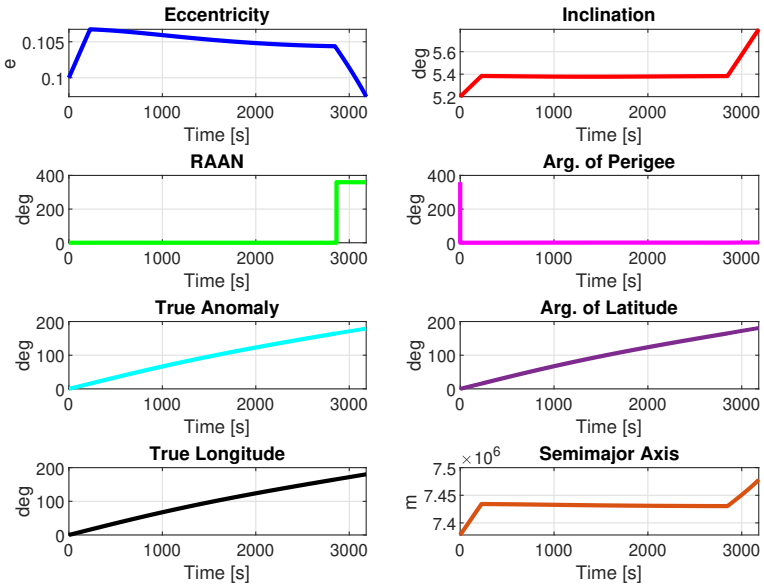


Figure 6.12: Orbital elements evolution

6.4 Test case IV

For the fourth test case a *Sun-synchronous orbit* (SSO) scenario is presented. A (SSO) is defined as one whose orbital plane precesses about the Earth's spin axis at the same rate that the Earth revolves around the Sun. Equivalently, the orbit's ascending-node longitude Ω must advance by approximately

$$n_E \;=\; \frac{2\pi}{T_E} \;\approx\; 1.991 \times 10^{-7}\,\mathrm{rad/s},$$

where $T_E \approx 365.25\,\mathrm{days}$ is the Earth's sidereal year. For a circular orbit this condition guarantees that the local solar time of the ascending node and hence the Sun's elevation angle at any given ground point remains constant from one pass to the next.

Natural two-body motion cannot produce this steady nodal precession, but the Earth's oblateness (the J_2 perturbation) imparts a secular drift of the ascending node given by

$$\dot{\Omega} \;=\; -\tfrac{3}{2}\,\frac{R_E^2}{a^2(1-e^2)^2}\; n\; J_2\; \cos i,$$

By selecting a and i so that $\dot{\Omega} \simeq +n_E$, one obtains the classic Sun-synchronous

condition. That feature is widely used for Earth observation and remote sensing because it ensure:

- Consistent illumination: imaging instruments always see the ground under nearly the same Sun-elevation and shadowing conditions.
- Regular revisit times: the repeat ground-track pattern is highly repeatable, simplifying mission planning and data analysis.
- Temperature stability:constant solar geometry reduces thermal cycling on the spacecraft.

These properties make SSOs ideal for multispectral imaging, radar mapping, and any application requiring uniform lighting over long time spans [62].

Orbital element	**Chaser Orbit**	**Target Orbit**
a [km]	7078.173	7178.173
e	0.003	0.003
i [°]	98.2	98.6
RAAN [°]	0	0
ω [°]	0	0
ν [°]	0	TBO

Table 6.15: Starting orbital elements of Chaser and Target

In the presented scenario the objective is to test a transfer from a Sun-synchronous orbit at 700 km, $i = 98.2°$, to one at 800 km, $i = 98.6°$, driven by several operational and engineering considerations. For example raising the orbit from 700 km/98.2° to 800 km/98.6° greatly reduces atmospheric drag cutting the ΔV needed for station-keeping and aligns the nodal regression (≈ 0.9856 °/day) exactly with Sun-synchronous requirements. The higher altitude also widens the ground swath by about 8% and lengthens the period slightly (100.9 min vs. 98.8 min), offering more flexible revisit and downlink scheduling [63]. Finally, this transfer provides the precise phasing and ΔV profile needed for efficient rendezvous operations at 800 km.

Optimization results	Value
ToF (s)	2994.29
Corrected ToF (s)	2994.29 (+0.0%)
TA_{dep} (°)	1.90
ΔV_{tot} (m/s)	74.00
ΔV_1 (m/s)	36.76
ΔV_2 (m/s)	37.24
Corrected ΔV_1 (m/s)	36.86 (+0.25%)
Corrected ΔV_2 (m/s)	39.03 (+4.8%)
θ (°)	179.9850
Corrected θ (°)	166.3061 (–7.6%)

Table 6.16: Optimization results for the manoeuvre

Simulation results	Value
Simulated final time (s)	2991.80
Estimated ToF (s)	2994.29
Final chaser mass (kg)	2410.32
Final relative velocity (m/s)	0.003
Total ΔV expended (m/s)	75.22
Estimated ΔV (m/s)	74.00

Table 6.17: Simulation performance results

The results demonstrate that the optimized transfer requiring 74 m/s of ΔV over a 2994 s flight time holds up in simulation with just 75.2 m/s actually used and a negligible timing error. A final relative speed of 0.003 m/s and a residual mass of 2410 kg (which means a propellant consumption of *89.68 kg*) confirm good propellant control and velocity guidance performance (Fig. 6.13. It is clear that also this manoeuvre lies well within the spacecraft's performance envelope with also an accurate guidance behaviour (Fig. 6.14).

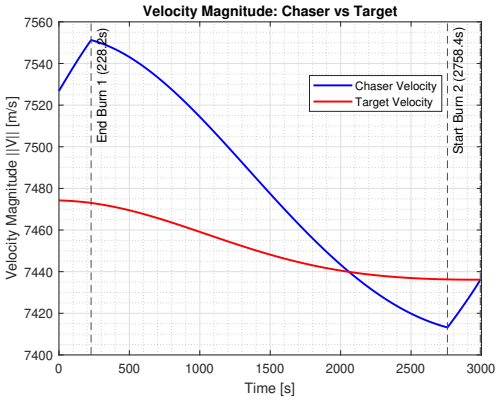


Figure 6.13: Propagation velocity evolution

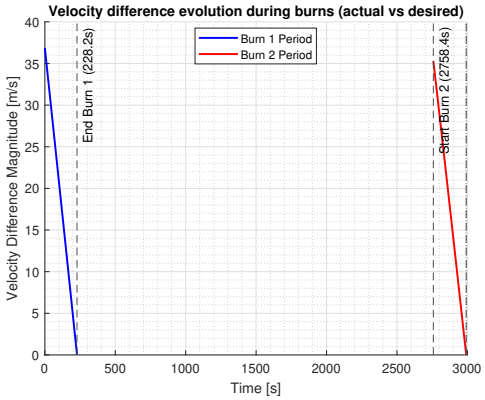


Figure 6.14: Propagation velocity chasing

Orbital parameter	Desired	Achieved	Error
a [km]	7178.1365	7178.1322	0.0043
e	0.0030	0.0029	0.0001
i [°]	98.6000	98.6000	0.0000
RAAN [°]	0.0000	0.0326	0.0326
ω [°]	359.9991	44.0830	44.0838
ν [°]	179.9858	135.9975	43.9883
θ [°]	179.9850	180.0805	0.0955

Table 6.18: Orbital parameters – desired vs achieved

The post-manoeuvre orbital elements align almost exactly with the targets: the semi-major axis error is just *4.3 m*, the eccentricity remains essentially circular with a $1 * 10^{-4}$ deviation, and the inclination matches with a clean *0%* error. These results confirm that the transfer provide both the orbit's size and geometry expected with negligible perturbation. Figure 6.15 and 6.16 shows the rendezvous trajectory evolution from a second point of view demonstrating in a graphic way the reliabily of the results.

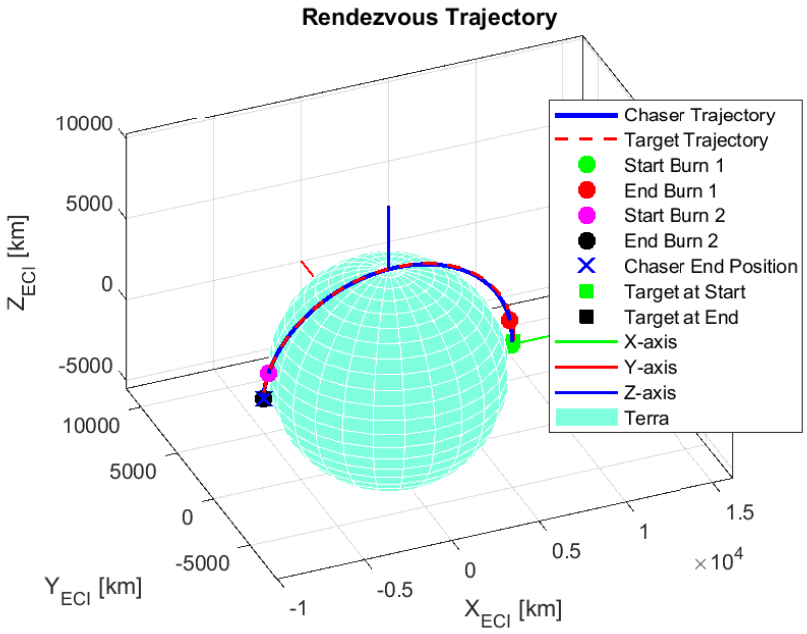


Figure 6.15: Rendezvous trajectory

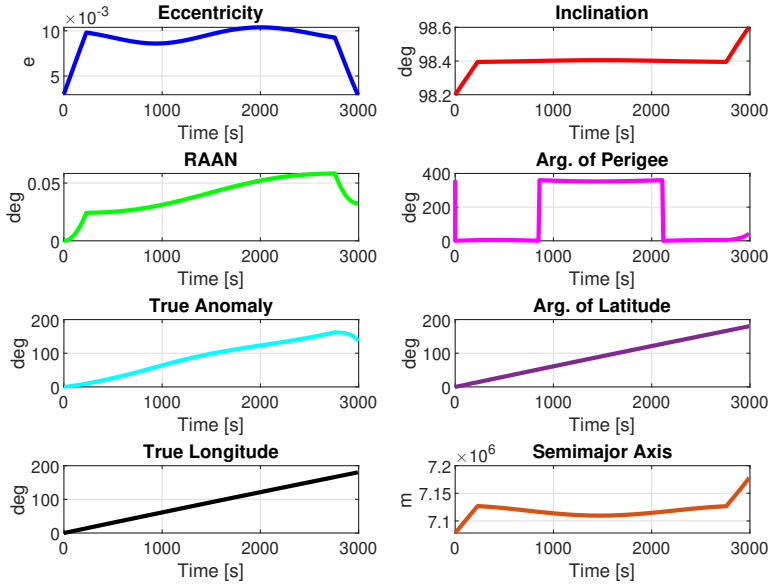


Figure 6.16: Orbital elements evolution

6.5 Test case V

The fifth test case is the Pontani 2D manoeuvre already discussed in Section (1). Knowing the optimal results and the trajectory of the trasnfer we will use this case as a benchmark for the simulator. The starting orbital elements are shown in the table below

Orbital element	Chaser Orbit	Target Orbit
a [km]	18000	35000
e	0.6	0.8
i [°]	0	0
RAAN [°]	0	120
ω [°]	0	0
ν [°]	158.94	TBO

Table 6.19: Starting orbital elements of Chaser and Target

In order to perform this manoeuvre the existent spacecraft data set was modified. The thrust and the specific impulse have been augemented to values comparable with the Rocket Lab *Rutherford Engine* [64] and of course the propellant mass was raised to an acceptable value. The new data set is described in Table 6.20

Spacecraft data	Value
Initial mass m_0 [kg]	10000
Propellant mass m_{prop} [kg]	7000
Thrust [N]	18000
Specific impulse I_{sp} [s]	350
Aerodynamic coefficient $(S * C_D)$	7.6

Table 6.20: Spacecraft data

The optimization results are the same obtained in the previous chapters and for completeness are shown in the table below

Optimization results	Value
ToF (s)	61484.52
Corrected ToF (s)	57672.48 (-6.2%)
TA_{dep} (°)	195.05
ΔV_{tot} (m/s)	2563.28
ΔV_1 (m/s)	1814.80
ΔV_2 (m/s)	748.47
Corrected ΔV_1 (m/s)	1827.51 (+0.7%)
Corrected ΔV_2 (m/s)	55.36 (+2.55%)
θ (°)	186.3694
Corrected θ (°)	188.0094 (+0.88%)

Table 6.21: Optimization results for the manoeuvre

Simulation results	Value
Simulated final time (s)	57671.60
Estimated ToF (s)	57672.48
Final chaser mass (kg)	4695.44
Final relative velocity (m/s)	0.909
Total ΔV expended (m/s)	2594.63
Estimated ΔV (m/s)	2563.28

Table 6.22: Simulation performance results

Table 6.22 illustrates the numerical propagation results. The actual ΔV_{tot} erogated is slightly higher then the one expected by the optimization (+1.22%) with a propellant mass consumption of *5304.56 kg*. The final relative velocity is under the metre per second showing a good convergence of the two velocities (see Fig. 6.17) and the time error between the simulated and the expected one is negligible.

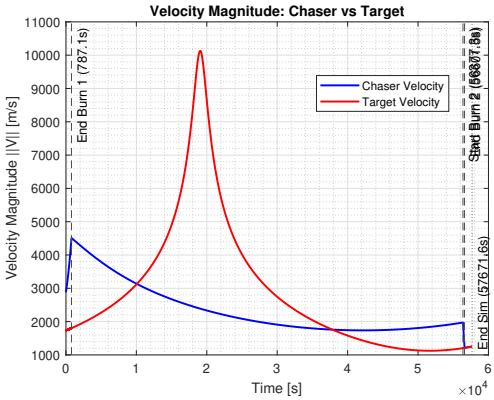


Figure 6.17: Propagation velocity evolution

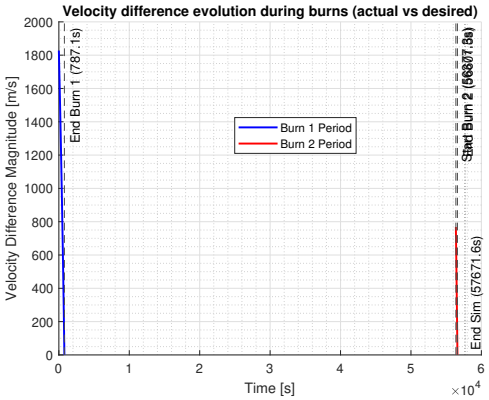


Figure 6.18: Propagation velocity chasing

From Table 6.23 it is possible to notice that the semi-major axis deviates by only 0.0017% from the 35000 km target, while the achieved eccentricity of 0.7995 lies within 5×10^{-4} of the nominal 0.8, confirming preservation of the orbit's scale and shape. The inclination matches exactly at 0.0001°, and the RAAN error of 0.1133° indicates negligible nodal displacement. The pronounced offset in Argument of Latitude (4.1328°) reflect a slight phasing lag rather than any structural transfer error. Such minimal discrepancies are well within acceptable limits for a long-duration manoeuvre and can be readily corrected by a brief phasing adjustment or station-keeping burn.

Orbital parameter	Desired	Achieved	Error
a [km]	34999.9729	34994.0859	5.8869
e	0.8000	0.7995	0.0005
i [°]	0.0001	0.0001	0.0000
RAAN [°]	120.0000	120.1133	0.1133
ω [°]	360.0000	0.1179	0.1179
ν [°]	190.5021	186.2514	4.2507
θ [°]	190.5021	186.3694	4.1328

Table 6.23: Orbital parameters – desired vs achieved

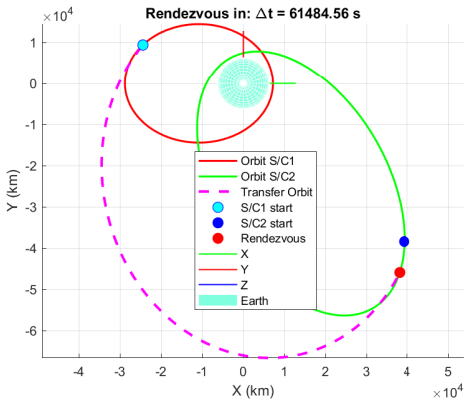


Figure 6.19: Optimal trajectory

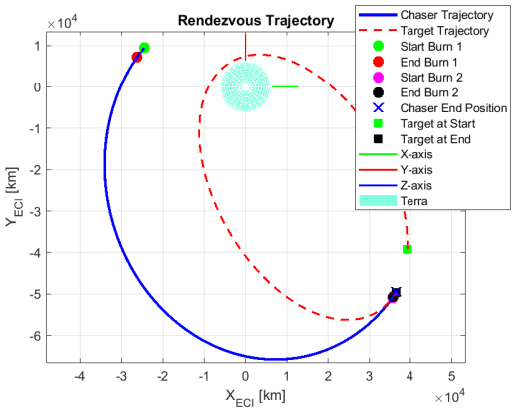


Figure 6.20: Simulated trajectory

Figure 6.19 and 6.20 shows the trajectory plots as graphic benchmark. It is possible to notice a larger gap between the Target initial and final positions: this is due to the time of glight (ToF) correction that happen in the simulation tuning phase. The evolution of the orbital elements follows an expected behaviour demonstrating again the reliability of the model.

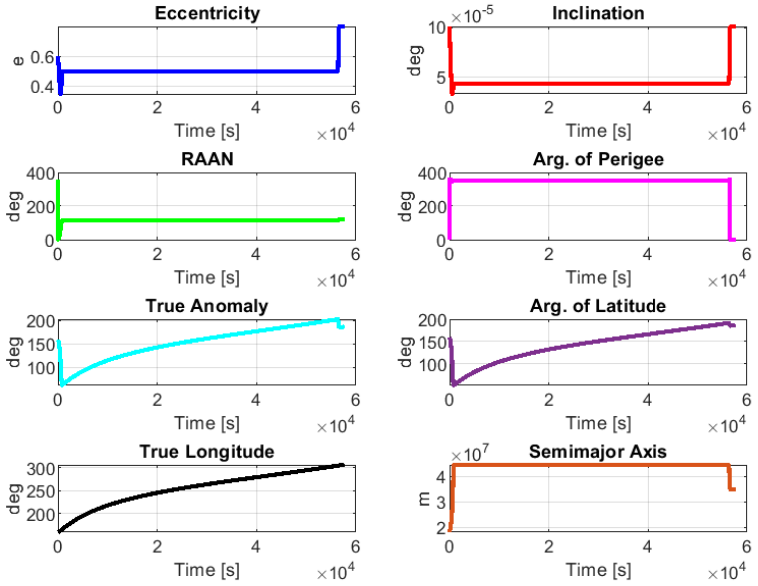


Figure 6.21: Orbital elements evolution

6.6 Test case VI

The second Pontani case, the *3D* one, is performed as sixth test case for comparing it with the optimal solution found in Section (1). The initial orbital parameters are shown in the table below

Orbital element	Chaser Orbit	Target Orbit
a [km]	6671.53	42163.95
e	0	0
i [°]	45	0
RAAN [°]	0	0
ω [°]	0	0
ν [°]	0	TBO

Table 6.24: Starting orbital elements of Chaser and Target

From the starting condition it is possible to see that this is a manoeuvre form a parking orbit in LEO to a Geosynchronous orbit. The propulsion system of the tested spacecraft is not able to provide the necessary thrust for performing this kind of manoeuvre in an impulsive way. The alternative spacecraft parameter are shown in Table 6.25

Spacecraft data	Value
Initial mass m_0 [kg]	10000
Propellant mass m_{prop} [kg]	8000
Thrust [N]	150000
Specific impulse I_{sp} [s]	350
Aerodynamic coefficient $(S * C_D)$	7.6

Table 6.25: Spacecraft data

The values have been assigned considering the ratio $\frac{Thrust}{Weight} \simeq 1 \div 3$ for an instantaneous impulse [65]. So we suppose to erogate the thrust of six *Ruthuerford Engine* aligned. [2] The optimization results obtained are the same described in the previous chapter and here are reported for completeness.

[2] In a real case it is mandatory to consider the momentum caused by this kind of propulsive configuration, but for this 3DOF simulator it is not necessary

Optimization results	Value
ToF (s)	18986.39
Corrected ToF (s)	18986.39 (+0.0%)
TA_{dep} (°)	100.67
ΔV_{tot} (m/s)	4637.32
ΔV_1 (m/s)	2464.50
ΔV_2 (m/s)	2172.82
Corrected ΔV_1 (m/s)	2475.59 (+0.45%)
Corrected ΔV_2 (m/s)	2166.30 (-0.3%)
θ (°)	179.9996
Corrected θ (°)	180.4316 (+0.24%)

Table 6.26: Optimization results for the manoeuvre

Simulation results	Value
Simulated final time (s)	18979.70
Estimated ToF (s)	18986.39
Final chaser mass (kg)	2588.12
Final relative velocity (m/s)	16.392
Total ΔV expended (m/s)	4636.42
Estimated ΔV (m/s)	4637.32

Table 6.27: Simulation performance results

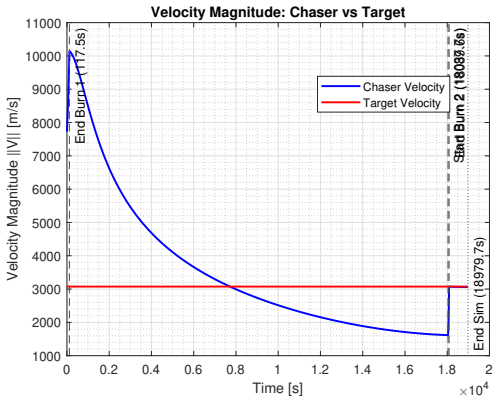


Figure 6.22: Propagation velocity evolution

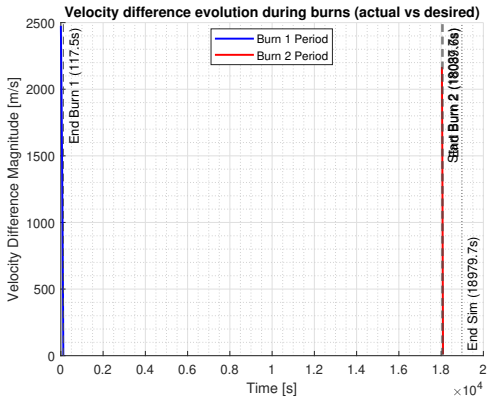


Figure 6.23: Propagation velocity chasing

In Table 6.27 the simulation delivers a final time of 18979.70 s, just 6.69 s shorter than the estimated 18986.39 s, and expends a total ΔV of 4636.42 m/s, only 0.02% below the predicted 4637.32 m/s. The Chaser's final mass of 2588.12 kg confirms that the propellant budget was respected, while a residual relative velocity of

16.39 m/s is critically high for a rendezvous operation, indicating the need of a more accurate/rigid closing approach to achieve the precision required for safe proximity operations.

Orbital parameter	Desired	Achieved	Error
a [km]	42163.9500	42064.2427	99.7073
e	0.0000	0.0606	0.0606
i [°]	0.0001	0.0453	0.0452
RAAN [°]	0.0000	349.7655	10.2345
ω [°]	0.0000	94.1017	94.1017
ν [°]	179.9996	99.5497	80.4499
argLat [°]	179.9996	193.6515	13.6519

Table 6.28: Orbital parameters – desired vs achieved

Examining the errors on the final orbital parameters from Table 6.28, the semi-major axis is undershot by 0.24%, yielding an achieved eccentricity of 0.0606 rather than the nominal circular orbit. The inclination error remains negligible at 0.0452°, and the RAAN offset of 10.2345° is modest, but the Argument of Latitude exhibit a larger deviation of 13.6519°. This phase-angle discrepancies highlight a significant timing offset that would necessitate substantial correction burns to fully align the spacecraft with the target orbital slot.

As done for the test case **IV** here in Figure 6.24 and 6.25 are depicted the trajectory plots as graphic benchmark. It is possible to notice the final phase-angle misalignment but also an overall deep similarity in the two trajectories. The evolution of the orbital elements in Figure 6.26 shows an overall correct trend with a final discrepancy that immediately catches the eye.

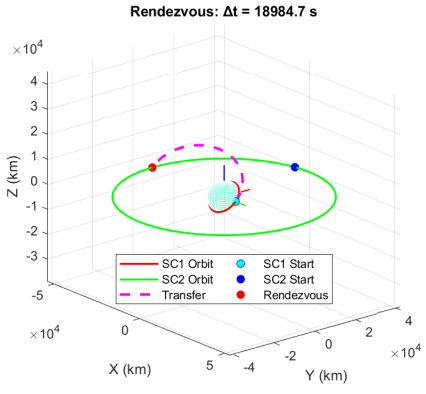


Figure 6.24: Optimal trajectory

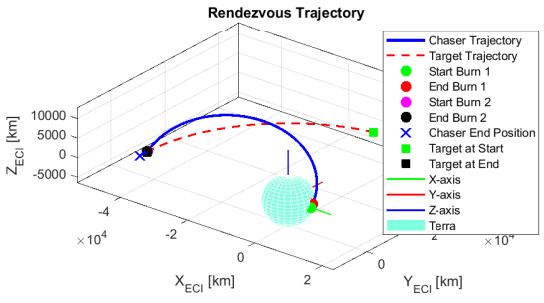


Figure 6.25: Simulated trajectory

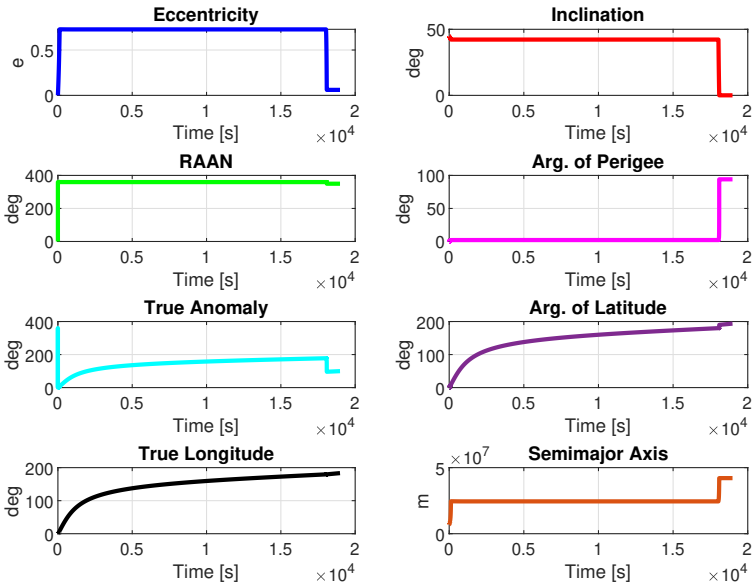


Figure 6.26: Orbital elements evolution

6.7 Test case VII

The seventh test case provides information about the performance of the TSP integration in the simulator. For for simplicity's sake the Chaser will perform three near-Hohmann transfers for raising the apoapis to $r_{apoapis} \simeq 8335\ km$. The starting orbit details are shown in Table 6.29

Parameter	Chaser	Target 1	Target 2	Target 3
a [km]	7278.137	7578.137	7378.137	7478.137
e	0.1000	0.1000	0.1000	0.1000
i [°]	0.0001	0.0001	0.0001	0.0001
RAAN [°]	0.0000	0.0000	0.0000	0.0000
ω [°]	0.0000	0.0000	0.0000	0.0000
ν [°]	0.0100	TBD	TBD	TBD

Table 6.29: Orbital elements for the chaser and three targets

After running the *TSP-DE algorithm* (see Section 4) the best visiting sequence was the following

Optimal Visit Sequence

Target **2** ⟶ Target **3** ⟶ Target **1**

with an esitamed ΔV_{tot} of 147.21 m/s

For each leg performed the optimal reults provided by the simulation are listed in Table 6.34. It is possible no notice that each leg has its owns ΔV and θ corrections. This last thing combined with the fact that each successive leg is influenced by the previous one makes the fine tuning process longer and harder.

Optimization results	**Leg 1**	**Leg 2**	**Leg 3**
ToF (s)	3108.72	3054.02	3027.48
Corrected ToF (s)	3108.72 (+0.0%)	3054.02 (+0.0%)	3027.48 (+0.0%)
Optimal TA_{dep} (°)	2.0	181.2	352.4
ΔV_1 (m/s)	25.08	19.02	20.41
ΔV_2 (m/s)	24.98	29.15	28.36
Corrected ΔV_1 (m/s)	26.00 (+3.68%)	18.70 (-1.71%)	21.20 (+3.89%)
Corrected ΔV_2 (m/s)	26.35 (+5.48%)	30.18 (+3.53%)	30.91 (+9.00%)
θ (°)	179.2334	350.5392	163.3488
Corrected θ (°)	172.2432	338.2702	145.691

Table 6.30: Optimal manoeuvre results for each leg

Simulation results	**Leg 1**	**Leg 2**	**Leg 3**
Simulated final time (s)	3103.10	3045.20	3019.60
Estimated ToF (s)	3108.72	3054.02	3027.48
Final chaser mass (kg)	2437.26	2381.89	2323.72
Final relative velocity (m/s)	0.256	0.158	0.131
Total ΔV expended (m/s)	52.33	47.32	50.91
Estimated ΔV (m/s)	50.06	48.18	48.76

Table 6.31: Simulation results for each leg

In Table 6.31 the simulation results show that the chaser completes the three transfer consuming 176.28 kg of propellant (see Fig. 6.33). The ΔV_{tot} received an overall increase of 2.42% with respect to the TSP-DE estimation. For each leg the final relative velocities are under the metre per second indicating the effective guidance control (see also from Fig. 6.27 to 6.32). These figures demonstrate that each leg's time of flight and propellant use remain within a few percent of the design values, while the low residual velocities indicate effective guidance control.

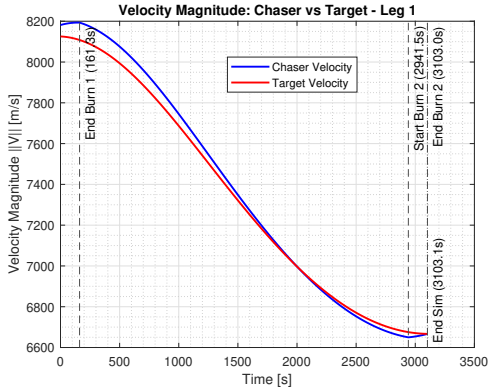


Figure 6.27: Propagation velocity evolution – leg 1

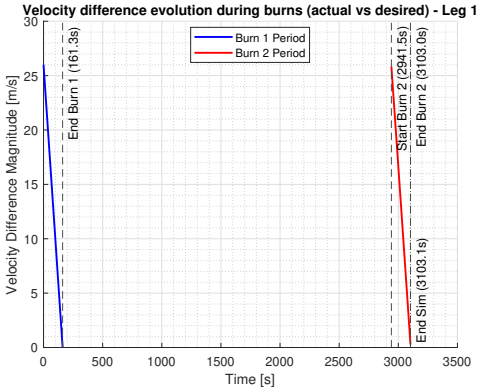


Figure 6.28: Propagation velocity chasing – leg 1

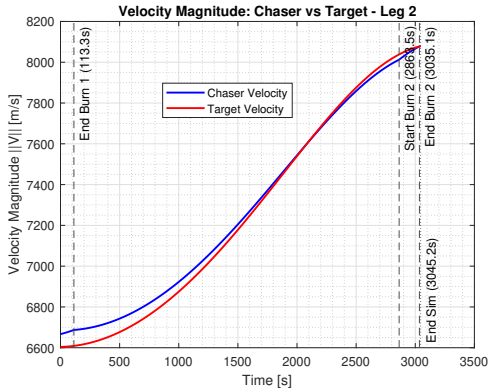


Figure 6.29: Propagation velocity evolution – leg 2

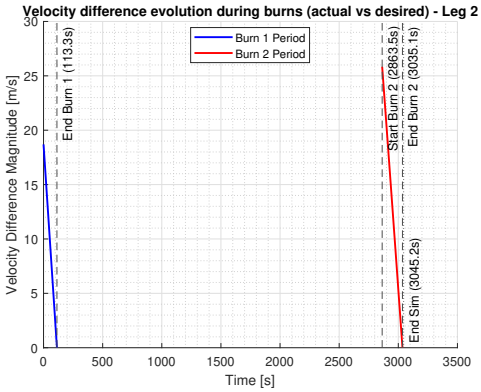


Figure 6.30: Propagation velocity chasing – leg 2

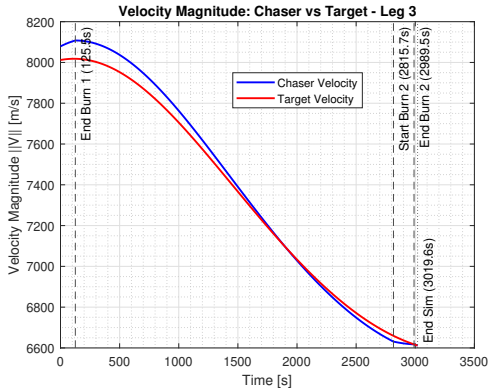


Figure 6.31: Propagation velocity evolution – leg 3

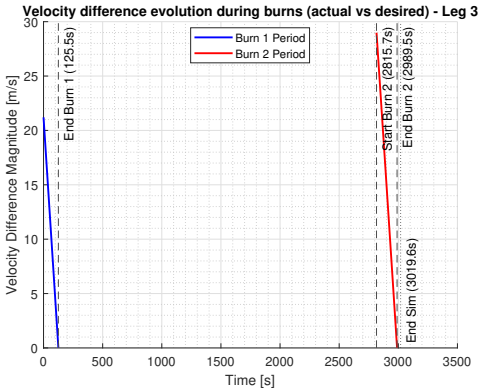


Figure 6.32: Propagation velocity chasing – leg 3

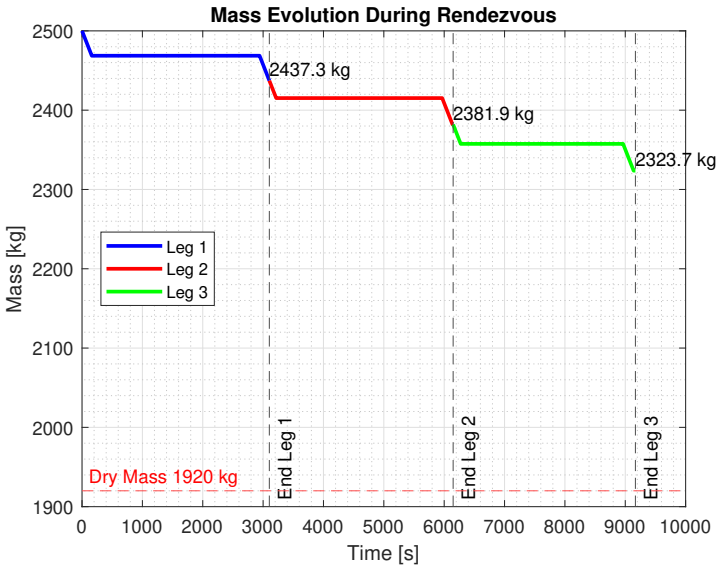


Figure 6.33: Mass evolution for each leg

The orbital element residual errors in Table 6.32 confirms that the semi-major axis errors between chaser and targets stays under 0.0007% and the eccentricity errors stay below 3×10^{-3}. Inclination errors are negligible ($\leq 10^{-4}$ deg).The Argument of Latitude exhibit a phase offset of 0.4563°. As already said. these small geometric and timing discrepancies can be corrected with brief phasing burns to ensure precise final rendezvous geometry.

	Leg 1			Leg 2			Leg 3		
Parameter	**Desired**	**Achieved**	**Error**	**Desired**	**Achieved**	**Error**	**Desired**	**Achieved**	**Error**
a [km]	7378.1326	7378.1305	0.0021	7478.1327	7478.1730	0.0403	7578.1324	7578.1838	0.0514
e	0.1000	0.0973	0.0027	0.1000	0.1019	0.0019	0.1000	0.0977	0.0023
i [°]	0.0001	0.0001	0.0000	0.0001	0.0001	0.0000	0.0001	0.0001	0.0000
RAAN [°]	0.0000	359.7738	0.2262	360.0000	0.0058	0.0058	0.0000	359.7666	0.2334
ω [°]	359.9998	0.8170	0.8172	0.0002	359.1641	0.8361	359.9998	1.0751	1.0753
ν [°]	179.2336	178.8727	0.3609	350.5390	351.2276	0.6886	163.3490	162.5784	0.7706
θ [°]	179.2334	179.6897	0.4563	350.5392	350.3916	0.1476	163.3488	163.6535	0.3047

Table 6.32: Orbital parameters for each leg – desired vs achieved

Figure 6.34 and 6.35 shows the trajectory of each leg in the $X_{ECI} - Y_{ECI}$ plane since the orbits are coplanar. As expected from the physics od the problem the chaser tour start from the lower orbit and continue to the higher one in order to consume less propellant.

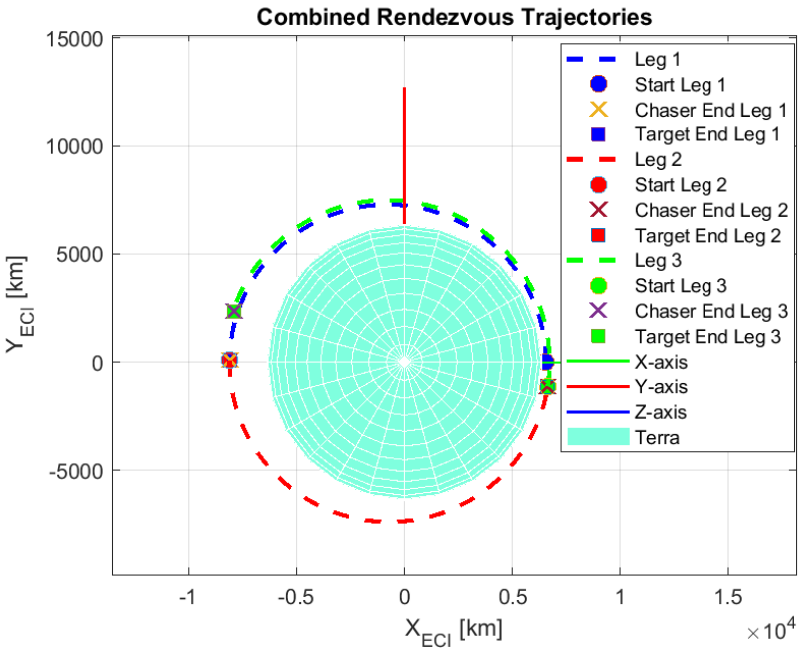


Figure 6.34: Combined rendezvous trajectories

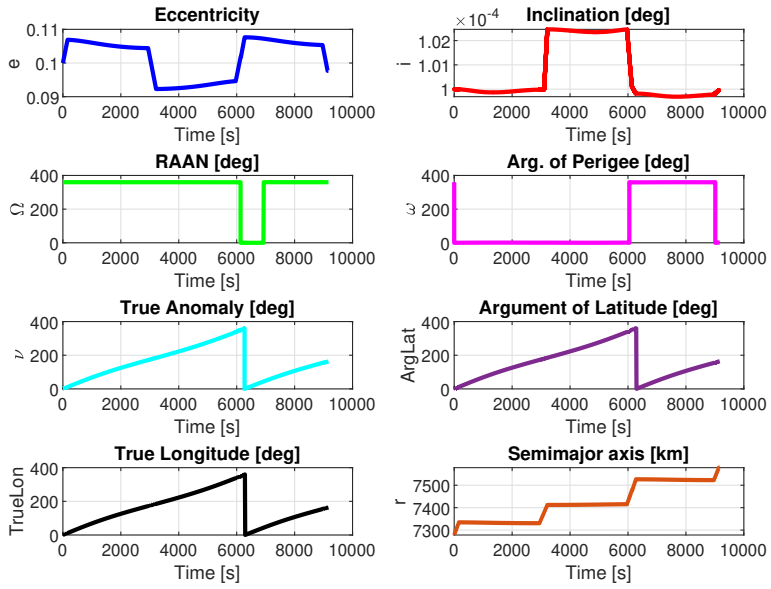


Figure 6.35: Orbital elements evolution

6.8 Test case VIII

For the final test case, a three-leg mission across distinct Sun-synchronous orbits is selected to replicate actual constellation deployments and multi-site servicing operations, thereby showcasing the guidance solution's robustness and flexibility under demanding conditions. Table 6.33 illustrates the starting orbits of Chaser and Targets.

Parameter	Chaser	Target 1	Target 2	Target 3
a [km]	6978.173	7178.173	7078.137	7278.137
e	0.003	0.003	0.003	0.003
i [°]	97.8	98.6	98.2	99.03
RAAN [°]	0.0000	0.0000	0.0000	0.0000
ω [°]	0.0000	0.0000	0.0000	0.0000
ν [°]	0.01	TBD	TBD	TBD

Table 6.33: Orbital elements for the chaser and three targets

The optimization lead to following optimal visit sequence and manoeuvre parameters

Optimal Visit Sequence

Target **2** ⟶ Target **1** ⟶ Target **3**

Optimization results	Leg 1	Leg 2	Leg 3
ToF (s)	2931.94	2858.48	3207.93
Corrected ToF (s)	2931.94 (+0.0%)	2858.48 (+0.0%)	3207.93 (+0.0%)
Optimal TA_{dep} (°)	1.9	181.8	353.3
ΔV_1 (m/s)	37.30	35.12	34.87
ΔV_2 (m/s)	37.77	39.66	44.46
Corrected ΔV_1 (m/s)	37.47 (+0.46%)	34.77 (-1.00%)	33.03 (-5.29%)
Corrected ΔV_2 (m/s)	39.58 (+4.80%)	42.36 (+6.80%)	47.53 (+6.90%)
θ (°)	179.9881	351.7806	180.2270
Corrected θ (°)	165.4905	339.4682	164.3670

Table 6.34: Optimal manoeuvre results for each leg

From Table 6.35 and Figure 6.36, the Chaser's mass decreases from its initial value to 2231.98 kg after the last leg, with a propellant usage of 268.2 kg that indicates a consistent propellant usage in line with the design estimates. The total ΔV expended against the predicted one is higher of just 1.88%. This, together with a consistent guidancr algorithm behaviour (see from Fig.6.37 to 6.42) translates into

residual final relative velocities of only 0.182 m/s, 0.008 m/s and 0.292 m/s, well below the metre per second. Flight durations for each transfer also remain within a few seconds of their optimal values, confirming overall timing fidelity.

Simulation results	**Leg 1**	**Leg 2**	**Leg 3**
Simulated final time (s)	2930.80	2850.50	3198.90
Estimated ToF (s)	2931.94	2858.48	3207.93
Final chaser mass (kg)	2408.58	2321.09	2231.98
Final relative velocity (m/s)	0.182	0.008	0.292
Total ΔV expended (m/s)	76.72	76.18	80.62
Estimated ΔV (m/s)	75.07	74.78	79.34

Table 6.35: Simulation results for each leg

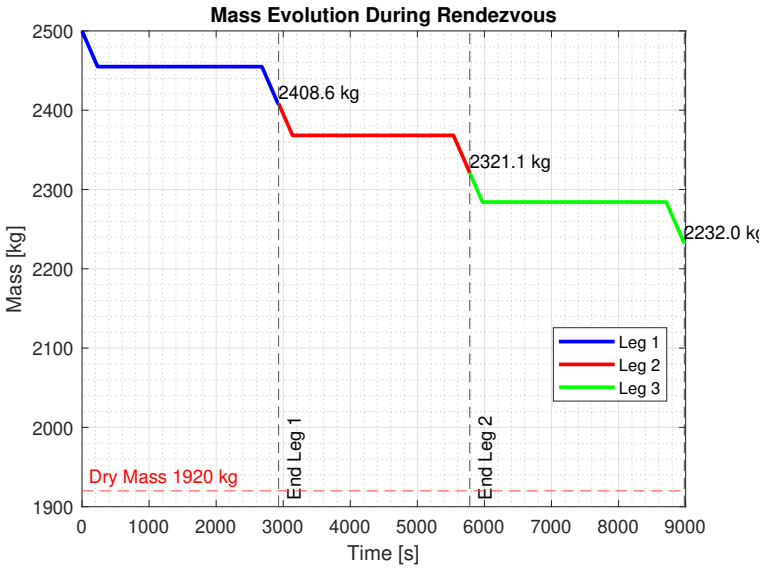


Figure 6.36: Mass evolution for each leg

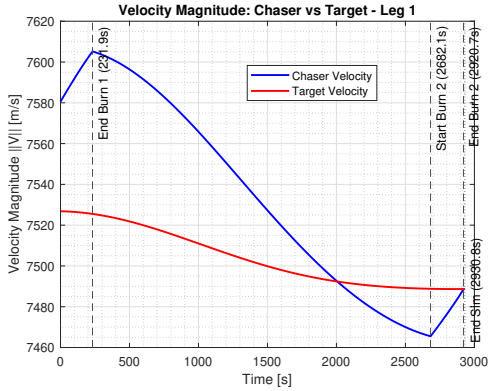


Figure 6.37: Propagation velocity evolution – leg 1

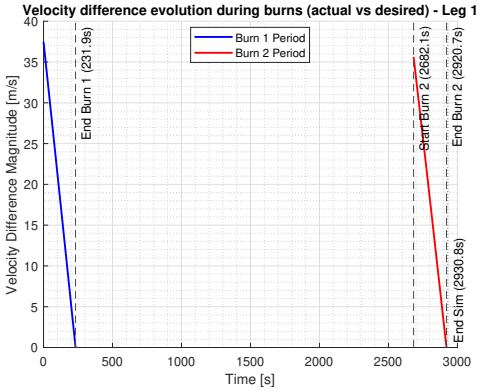


Figure 6.38: Propagation velocity chasing – leg 1

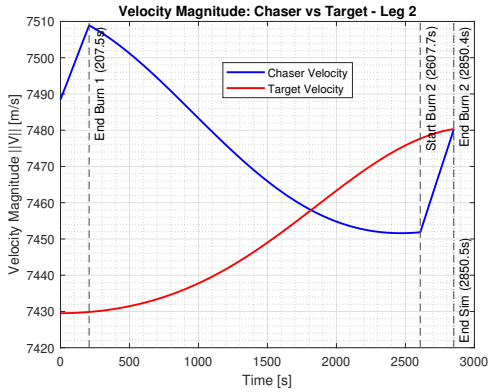


Figure 6.39: Propagation velocity evolution – leg 2

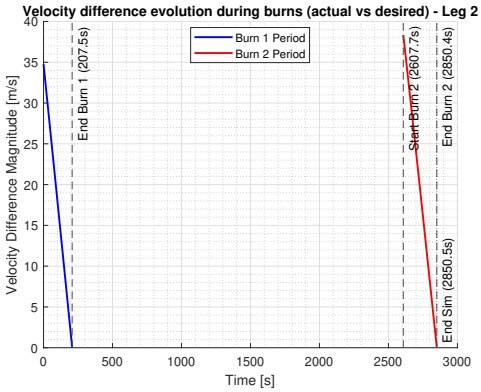


Figure 6.40: Propagation velocity chasing – leg 2

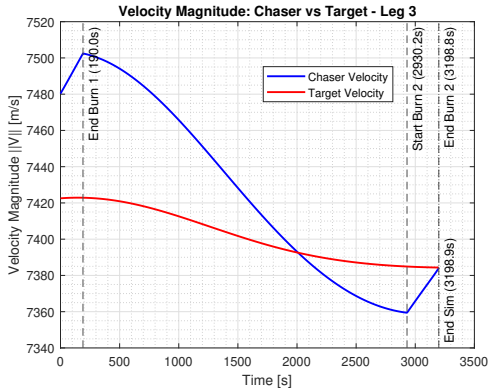


Figure 6.41: Propagation velocity evolution – leg 3

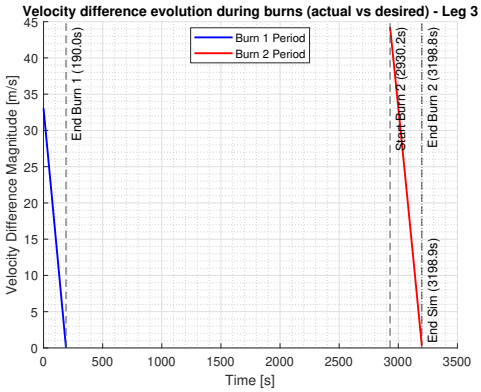


Figure 6.42: Propagation velocity chasing – leg 3

Table 6.36 shows that the orbital geometry is preserved with minimal drift. Semi-major axis errors stay below 300 m (relative errors $\leq 4 \times 10^{-5}$), eccentricity deviations never exceed 1.1×10^{-3}, and inclination remains within 0.0015°. The Argument of latitude errors remain below 0.27°. These small residual can be removed with brief phasing burns to ensure exact rendezvous geometry.

Parameter	Leg 1			Leg 2			Leg 3		
	Desired	Achieved	Error	Desired	Achieved	Error	Desired	Achieved	Error
a [km]	7078.1322	7078.1070	0.0251	7178.1323	7178.1432	0.0109	7278.1316	7278.4237	0.2921
e	0.0030	0.0031	0.0001	0.0030	0.0041	0.0011	0.0030	0.0031	0.0001
i [°]	98.2000	98.2000	0.0000	98.6000	98.5987	0.0013	99.0300	99.0285	0.0015
RAAN [°]	0.0000	0.0297	0.0297	360.0000	0.0251	0.0251	0.0000	0.0144	0.0144
ω [°]	359.9916	46.9442	46.9526	0.0085	331.5073	28.5012	359.9914	44.8804	44.8891
ν [°]	179.9964	133.2294	46.7671	351.7721	20.0051	28.2330	180.2356	135.0896	45.1460
θ [°]	179.9881	180.1736	0.1855	351.7806	351.5124	0.2682	180.2270	179.9700	0.2569

Table 6.36: Orbital parameters for each leg – desired vs achieved

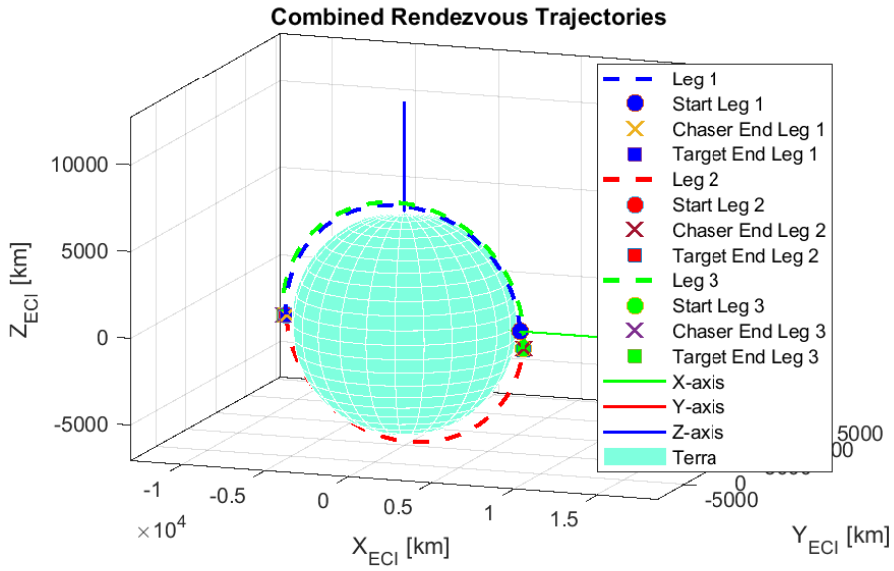


Figure 6.43: Combined rendezvous trajectories

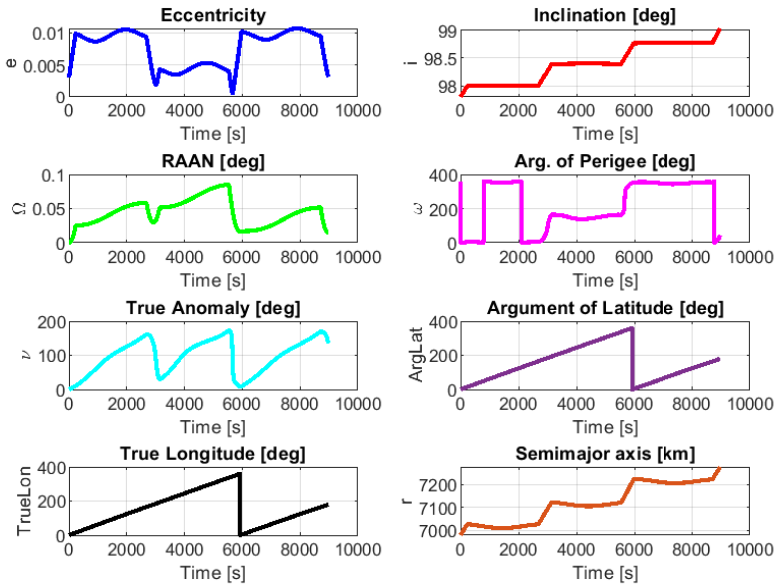


Figure 6.44: Orbital elements evolution

Figure 6.43 illustrate how the chaser departs and arrives on he tthree distinct sun-synchronous orbits, smoothly transitioning altitude and phase in each leg while maintaining tight proximity to the target. Meanwhile, the time histories of the orbital elements in Figure 6.44 confirm that semi-major axis, eccentricity and inclination remain effectively bounded within their expected ranges, and that the small jumps in eccentricity, inclination and semi-major axis coincide precisely with each burn.

Chapter 7

Conclusions and future improvements

The primary objective of this thesis work was to develop and implement a modular MATLAB tool chain that supports the design and validation of impulsive rendezvous manoeuvres for AVIO's In-Orbit-Service (IOS) missions minimizing the propellant consumption (ΔV). In the first chapter, the discussion opens with an hystorical overview of the IOS missions followed by a survey on the current market landscape and a literature review of the optimization and guidance algorithms. After this, the second chapter introduces the theoretical framework of astrodynamics, illustrating the foundamentals laws and detailing the key equations for the classical impulsive manoeuvres. In the third chapter, attention shifts to the choice of evolutionary algorithms, highlighting the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Differential Evolution (DE), and Particle Swarm Optimization (PSO). The performances of each algorithms were investigated with a Lambert's problem based cost function, searching the best combination of *time-of-flight* and Target's satellite departure true anomaly that minimize the total ΔV. Among them the CMA-ES proved the fastest convergence despite the high iteration count, whereas DE and PSO offered a more balanced trade-off between runtime and robustness. The reliability of the optimization results was brought by the Pontani's results presented in literature [1, 47] that work as benchmarks for that purpose. In chapter four, a combination of the evolutionary algorithms and Travelling Salesman was analyzed in order to make the tool chain able to minimise the overall ΔV for multi-target sequences. Chapter five presents a dedicated three-phase guidance law in which the spacecraft first executes a burn along the Lambert solution's target velocity vector until the prescribed $\Delta V_{1_{target}}$ has been delivered. After this initial manoeuvre, the propulsion system is shut down and the vehicle coasts unpowered

until it reaches a target's argument of latitude (θ_{target}) . Ultimately, a second burn is initiated to "chase" the target's actual velocity, until the required $\Delta V_{2_{target}}$ or the optimal time-of-flight is reached. In the last chapter the combined phase flow optimisation-simulation behaved reliably in all eight test runs: the gap between the planned and simulated ΔV stayed below 5 %, and the leftover velocity was always under 0.5 m/s. The tool also worked well in commercial scenarios such as SSO manoeuvres, Hohmann transfers in LEO, and multi-target transfers while keeping within the mass and thrust limits of the AVUM+ stage. It can be stated that the thesis offers a reusable and easy to use framework for early mission analysis that speeds up performances evaluations.

Future improvements

The current guidance architecture still mixes open-loop and closed-loop elements. During the first burn the chaser fires almost exactly along the fixed $\Delta V_{1_{\text{target}}}$ vector delivered by the optimiser, while during the second burn it chases the target velocity, adjusting its thrust direction at every step. This hybrid logic is fast and lightweight, yet it inherits the usual weakness of Lambert-based approaches: the ideal solution assumes perfectly impulsive burns and a disturbance-free environment, so even small modelling errors or perturbations can leave the chaser a few hundred metres from the target. In the thesis this gap is only softened by empirical correction factors on the time-of-flight, the two ΔV magnitudes, and the argument of latitude that triggers the second burn. A clearer path forward is to replace this hybrid scheme with a fully closed-loop controller that reacts to the current relative state either through a state-feedback law based on Clohessy–Wiltshire dynamics or, more ambitiously, through a Model Predictive Controller that embeds the full nonlinear relative motion and jointly minimises position error, velocity error and propellant.

A second improvement becomes relevant as soon as the mission involves more than one target. At present, the simulator starts each transfer at whatever true anomaly happens to arise and initiates the next leg immediately after the previous one, without searching for an optimal waiting time. As a result, the departures can be far from ideal and the total ΔV (or elapsed time) is not truly minimal. Extending the optimiser to include both the initial true anomaly and an adjustable coast phase between legs would let the tool find the best "wait-to-go" strategy, trimming propellant, flight time, or both for multi-target campaigns.

Finally, the simulator itself would benefit from an upgrade from the current 3-DOF translation-only model to a full 6-DOF formulation. A 6-DOF version would incorporate attitude dynamics and its own attitude-control loop, so the tool could

account for thruster torques, solar-radiation pressure, magnetic-torquer interactions, gravity-gradient effects and sensor line-of-sight constraints during docking. Such fidelity is indispensable for precision rendezvous studies, where orientation errors and coupled translation–rotation manoeuvres can make the difference between a clean capture and a missed rendezvous.

Appendix A

Objective function

Objective function MATLAB code

```
function [cost, delta_v1, delta_v2] = lambert_cost(x, orbit)
    % Objective function
    % x(1)=delta_t, x(2)=TA2 (degree)
    delta_t = x(1);
    TA2_deg = x(2);

    mu = 398600;
    deg = pi/180;

    a1 = orbit.a1; e1 = orbit.e1; incl1_deg = orbit.incl1_deg;
    RA1_deg = orbit.RA1_deg; w1_deg = orbit.w1_deg; TA1_deg = orbit.
    TA1_deg;
    a2 = orbit.a2; e2 = orbit.e2; incl2_deg = orbit.incl2_deg;
    RA2_deg = orbit.RA2_deg; w2_deg = orbit.w2_deg;

    % Angular momentum
    h1 = sqrt(a1 * mu * (1 - e1^2));
    h2 = sqrt(a2 * mu * (1 - e2^2));

    % Orbital elements definition
    oe1 = [h1, e1, RA1_deg*deg, incl1_deg*deg, w1_deg*deg, TA1_deg*
    deg];
    oe2 = [h2, e2, RA2_deg*deg, incl2_deg*deg, w2_deg*deg, TA2_deg*
    deg];

    [r1, v1] = sv_from_oe(oe1, mu);
    [r2, v2] = sv_from_oe(oe2, mu);

```

```
26      % Orbital period of the final orbit
27      T2 = 2*pi/mu^2 * (h2/sqrt(1-e2^2))^3;
28
29      factor_initial = sqrt((1-e2)/(1+e2)) * tan(TA2_deg*deg/2);
30      if factor_initial < 0
31          E_initial = 2*(pi + atan(factor_initial));
32      else
33          E_initial = 2*atan(factor_initial);
34      end
35      t_initial = T2/(2*pi) * (E_initial - e2*sin(E_initial));
36
37      t_arr = t_initial + delta_t;
38
39      % Kepler equation with new TA2
40      Me = 2*pi*t_arr/T2;
41      E = kepler_equation(e2, Me);
42      if sqrt((1-e2)/(1+e2)) * tan(E/2) < 0
43          TA2_new = 2*(pi + atan(sqrt((1-e2)/(1+e2))*tan(E/2)));
44      else
45          TA2_new = 2*atan(sqrt((1-e2)/(1+e2))*tan(E/2));
46      end
47
48      oe2_new = [h2, e2, RA2_deg*deg, incl2_deg*deg, w2_deg*deg,
        TA2_new];
49      [r2_new, v2_new] = sv_from_oe(oe2_new, mu);
50
51      [v1_transfer, v2_transfer] = lambert_equation(r1, r2_new, delta_t
        , mu);
52      delta_v1 = v1_transfer - v1;
53      delta_v2 = v2_new - v2_transfer;
54      delta_v_total = norm(delta_v1) + norm(delta_v2);
55
56      cost = 1000 * delta_v_total;
57  end
```

Appendix B

Travelling Salesman Problem

TSP MATLAB code

```
function [bestSeq, totalCost, legCosts, legTOFs, legTAs] = TSP(DE,
    Xmin, Xmax, orbitChaser, orbitTargets)
    % Travelling Salesman Problem
    N = numel(orbitTargets);
    costStart = zeros(1,N);
    tofStart = zeros(1,N);
    TA2Start = zeros(1,N);
    costMat = inf(N,N);
    tofMat = zeros(N,N);
    TA2Mat = zeros(N,N);

    initTA2 = 0;

    % Cost from initial chaser orbit to each target
    for i = 1:N
        orbitLeg = orbitChaser;
        % set target orbit
        orbitLeg.a2 = orbitTargets(i).a2;
        orbitLeg.e2 = orbitTargets(i).e2;
        orbitLeg.incl2_deg = orbitTargets(i).incl2_deg;
        orbitLeg.RA2_deg = orbitTargets(i).RA2_deg;
        orbitLeg.w2_deg = orbitTargets(i).w2_deg;
        orbitLeg.TA2_deg = initTA2;
        [best, f_best, ~, ~, ~] = DE_fun(DE, Xmin, Xmax, orbitLeg);
        costStart(i) = f_best;
        tofStart(i) = best(1);
        TA2Start(i) = best(2);
    end
```

```
28
29      % Cost between each pair of targets
30      for i = 1:N
31          for j = 1:N
32              if i~=j
33                  orbitLeg.a1 = orbitTargets(i).a2;
34                  orbitLeg.e1 = orbitTargets(i).e2;
35                  orbitLeg.incl1_deg = orbitTargets(i).incl2_deg;
36                  orbitLeg.RA1_deg = orbitTargets(i).RA2_deg;
37                  orbitLeg.w1_deg = orbitTargets(i).w2_deg;
38                  orbitLeg.TA1_deg = initTA2;
39                  orbitLeg.a2 = orbitTargets(j).a2;
40                  orbitLeg.e2 = orbitTargets(j).e2;
41                  orbitLeg.incl2_deg = orbitTargets(j).incl2_deg;
42                  orbitLeg.RA2_deg = orbitTargets(j).RA2_deg;
43                  orbitLeg.w2_deg = orbitTargets(j).w2_deg;
44                  orbitLeg.TA2_deg = initTA2;
45                  [best, f_best, ~, ~, ~] = DE_fun(DE, Xmin, Xmax,
    orbitLeg);
46                  costMat(i,j) = f_best;
47                  tofMat(i,j) = best(1);
48                  TA2Mat(i,j) = best(2);
49              end
50          end
51      end
52
53      % Evaluate all visit sequences
54      permsList = perms(1:N);
55      numPerms = size(permsList,1);
56      totalCosts = inf(numPerms,1);
57      totalTOFs = zeros(numPerms,N);
58      totalTAs = zeros(numPerms,N);
59      for k = 1:numPerms
60          seq = permsList(k,:);
61          cost = costStart(seq(1));
62          tofs = zeros(1,N);
63          tas  = zeros(1,N);
64          tofs(1) = tofStart(seq(1));
65          tas(1) = TA2Start(seq(1));
66          cost = cost + costMat(seq(1),seq(2));
67          tofs(2) = tofMat(seq(1),seq(2));
68          tas(2) = TA2Mat(seq(1),seq(2));
69          cost = cost + costMat(seq(2),seq(3));
70          tofs(3) = tofMat(seq(2),seq(3));
71          tas(3) = TA2Mat(seq(2),seq(3));
72          totalCosts(k) = cost;
73          totalTOFs(k,:) = tofs;
74          totalTAs(k,:) = tas;
75      end
```

```
76
77      % Select optimal sequence
78      [~, idx] = min(totalCosts);
79      bestSeq = permsList(idx,:);
80      totalCost = totalCosts(idx);
81      legCosts = [costStart(bestSeq(1)), costMat(bestSeq(1),bestSeq(2))
        , costMat(bestSeq(2),bestSeq(3))];
82      legTOFs = totalTOFs(idx,:);
83      legTAs = totalTAs(idx,:);
84
85  end
```

Appendix C

GNC algorithm

C.1 1st Burn logic

In the early implementation, the first burn logic invoked the Lambert problem solver at every integration step in order to compute the "desired" departure velocity for the next step ($V_{1_{des}}$). However, when executing non-coplanar maneuvers, this approach frequently produced extremely large and physically unrealistic velocity vectors from the Lambert solution (see Fig. C.1). These spurious high velocities then corrupted the state propagation and rendered the downstream guidance logic invalid.

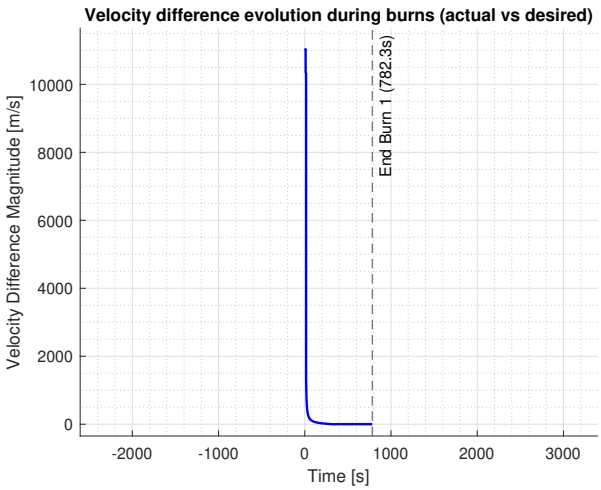


Figure C.1: $|V_{1_{des}} - V_C|$ undesired peek during a simple plane change maneouver of 1°

One might expect to avoid such solutions by constraining the maximum $V_{1_{des}}$, but even this solution was not fruitful and didn't prevent the guidance algorithm from corrupted results. Given this issues the logic was changed in the one already explained in the thesis main body.

C.2 Guidance code

GNC algorithm MATLAB code

```
function [U, GNC_DATA, v_target_now] = GNC(t, S, GNC_DATA, LV, ENV,
    orbit)

    % Spacecraft and target state
    POS_C = S(1:3);    VEL_C = S(4:6);
    POS_T = S(7:9);    VEL_T = S(10:12);
    massC = S(13);

    % U = [throttle, direction_eci(3)]
    U = zeros(4,1);
    dir_eci = [0;0;0];
    dt_step = GNC_DATA.dt;

    % Calculation constants
    accel = LV.thrust / massC;
    tol_dv = 0.01;

    switch GNC_DATA.phase

        %% PHASE 1 - First burn
        case 1
            % 1) Build the required DV vector
            if ~isfield(GNC_DATA,'dv1_vec_cmd')
                GNC_DATA.dv1_vec_cmd = GNC_DATA.dv1_target';
                GNC_DATA.dv1_applied = zeros(3,1);    % accumulated DV
            end

            % 2) Compute residual DV
            dv_res1 = GNC_DATA.dv1_vec_cmd - GNC_DATA.dv1_applied;
            norm_res1 = norm(dv_res1);
            if norm_res1 > tol_dv
                dir_eci = dv_res1 / norm_res1;       % residual unit
    vector
            else
                dir_eci = [0;0;0];
            end
            v_target_now = VEL_C + dv_res1;
```

```
36
37              % 3) Throttle ON/OFF logic and update accumulated DV
38              if norm(dir_eci) > 1e-9
39                  U(1) = 1;   % engine ON
40                  if dt_step > 0
41                      dv_inc = dir_eci * accel * dt_step;
42                      GNC_DATA.dv1_applied = GNC_DATA.dv1_applied +
    dv_inc;
43                      GNC_DATA.dv_accum1  = norm(GNC_DATA.dv1_applied);
44                  end
45              else
46                  U(1) = 0;   % engine OFF
47              end
48
49              % 4) First burn termination criterion
50              if (norm_res1 <= tol_dv) || (GNC_DATA.dv_accum1 >= norm(
    GNC_DATA.dv1_target))
51                  U(1) = 0;
52                  GNC_DATA.burn_active = false;
53                  GNC_DATA.phase = 2;      % switch to coasting
54                  GNC_DATA.dt = 10;
55              else
56                  GNC_DATA.burn_active = (U(1) == 1);
57                  GNC_DATA.dt = 0.1;
58              end
59
60
61          %% PHASE 2 - Coast until desired latitude argument
62          case 2
63              U(1) = 0;
64              GNC_DATA.dt = 10;
65              v_target_now = VEL_C;
66
67              % Compute current latitude argument
68              oe_c = oe_from_sv(POS_C, VEL_C, ENV.mu);
69              arg_lat = mod(oe_c(5) + oe_c(6), 2*pi);
70
71              delta_ang = abs(arg_lat - GNC_DATA.target_arg_lat);
72              delta_ang = min(delta_ang, 2*pi - delta_ang);
73              if delta_ang < deg2rad(0.5)
74                  GNC_DATA.phase = 3;      % start second burn
75                  GNC_DATA.burn_active = true;
76                  GNC_DATA.dt = 0.1;
77              end
78
79          %% PHASE 3 - Second burn
80          case 3
81              % 1) Initialize the required DV2 vector
82              if ~isfield(GNC_DATA,'dv2_vec_cmd')
```

```
83                  GNC_DATA.dv2_vec_cmd   = GNC_DATA.dv2_target';
84                  GNC_DATA.dv2_applied   = zeros(3,1);    % accumulated
    DV
85              end
86
87              % 2) Compute residual DV
88              dv_res2  = VEL_T - VEL_C;
89              norm_res2 = norm(dv_res2);
90              if norm_res2 > tol_dv
91                  dir_eci = dv_res2 / norm_res2;
92              else
93                  dir_eci = [0;0;0];
94              end
95              v_target_now = VEL_C + dv_res2;
96
97              % 3) Update accumulated DV and throttle
98              if norm(dir_eci) > 1e-9
99                  U(1) = 1;
100                 if dt_step > 0
101                     dv_inc = dir_eci * accel * dt_step;
102                     GNC_DATA.dv2_applied = GNC_DATA.dv2_applied +
    dv_inc;
103                     GNC_DATA.dv_accum2   = norm(GNC_DATA.dv2_applied);
104                 end
105             else
106                 U(1) = 0;
107             end
108
109             % 4) Second burn termination criterion
110             if (norm_res2 <= tol_dv) || (GNC_DATA.dv_accum2 >= norm(
    GNC_DATA.dv2_target))
111                 U(1) = 0;
112                 GNC_DATA.burn_active = false;
113                 GNC_DATA.phase = 4;     % maneuver complete
114                 GNC_DATA.dt = 10;
115             else
116                 GNC_DATA.burn_active = (U(1) == 1);
117                 GNC_DATA.dt = 0.1;
118             end
119
120         %% PHASE 4      End of maneuver
121         case 4
122             U(1) = 0;
123             GNC_DATA.dt   = 10;
124             v_target_now = VEL_T;
125     end
126
127     % Update last time and thrust direction
128     GNC_DATA.last_t = t;
```

```
129     if (U(1) > 0.5) && (norm(dir_eci) > 1e-6)
130         U(2:4) = dir_eci;
131     else
132         U(2:4) = [0;0;0];
133         U(1) = 0;
134     end
135 end
```

Bibliography

[1] Mauro Pontani and Bruce A. Conway. «Particle swarm optimization applied to impulsive orbital transfers». In: *Acta Astronautica* 74 (2012), pp. 141–155. DOI: 10.1016/j.actaastro.2011.09.011 (cit. on pp. 1, 134).

[2] Wikipedia contributors. *Skylab*. https://it.wikipedia.org/wiki/Skylab. [online; accessed 02-May-2025]. 2025 (cit. on p. 3).

[3] National Aeronautics and Space Administration. *On-Orbit Satellite Servicing Study: Project Report*. Technical Report. NASA Goddard Space Flight Center, 2010 (cit. on p. 3).

[4] Wikipedia contributors. *Hubble Space Telescope*. https://it.wikipedia.org/wiki/Hubble_Space_Telescope. [online; accessed 02 May 2025]. 2025 (cit. on p. 3).

[5] Wikipedia contributors. *ETS-VII*. https://it.wikipedia.org/wiki/ETS-VII. [online; accessed 02 May 2025]. 2025 (cit. on p. 3).

[6] Wikipedia contributors. *International Space Station*. https://it.wikipedia.org/wiki/International_Space_Station. [online; accessed 03 May 2025]. 2025 (cit. on p. 3).

[7] Wikipedia contributors. *Orbital Express*. https://it.wikipedia.org/wiki/Orbital_Express. [online; accessed 05 May 2025]. 2025 (cit. on p. 4).

[8] NASA Astronomy Picture of the Day. *New Eyes for the Hubble Space Telescope*. https://apod.nasa.gov/apod/ap970221.html. [online; accessed 05 May 2025]. 1997 (cit. on p. 4).

[9] Northrop Grumman. *SpaceLogistics*. https://www.northropgrumman.com/space/space-logistics-services. Accessed: 2025-05-10. 2025 (cit. on p. 4).

[10] Blackerby, Okamoto, Fujimoto and Okada. «ELSA-D: an in-orbit end-of-life demonsration mission». In: *ELSA-1-Conference-IAC-2018-v1* (2018) (cit. on p. 4).

[11] Defense Advanced Research Projects Agency. *RSGS: Robotic Servicing of Geosynchronous Satellites*. https://www.darpa.mil/research/programs/robotic-servicing-of-geosynchronous-satellites. Accessed: 2025-05-10 (cit. on pp. 4, 13).

[12] Wei-Jie Li et al. «On-orbit service (OOS) of spacecraft: A review of engineering developments». In: *Progress in Aerospace Sciences* 108 (2019), pp. 32–120 (cit. on p. 5).

[13] European Space Agency. *ESA costruisce la prima missione di manutenzione in orbita con D-Orbit*. https://www.esa.int/Space_in_Member_States/Italy/ESA_costruisce_la_prima_missione_di_manutenzione_in_orbita_con_D-Orbit. Accessed: 2025-05-10. 2024 (cit. on p. 5).

[14] Giorgio Gasbarrini. *In-Orbit Services: AVIO Vision*. Presentation at Clean Space Industrial Days, ESA Indico. 21 September 2021. 2021 (cit. on pp. 7, 8).

[15] Stefano Gallucci, Roberto Mancini, and Ettore Scardecchia. «Vega Space System». In: *8th European Conference for Aeronautics and Space Sciences (EUCASS)*. 2019. DOI: 10.13009/EUCASS2019-987 (cit. on pp. 7, 8).

[16] Stefano Gallucci and Roberto Mancini. «The AVUM Orbital Module for the Space Rider System». In: *Proceedings of the 8th European Conference for Aeronautics and Space Sciences (EUCASS)*. 2019. DOI: 10.13009/EUCASS2019-0860 (cit. on pp. 7, 8).

[17] European Space Policy Institute. *In-Orbit Services*. Tech. rep. Accessed: 2025-05-09. European Space Policy Institute, 2020. URL: https://www.espi.or.at/reports/in-orbit-services/ (cit. on pp. 8–10).

[18] Precedence Research. *On-Orbit Satellite Servicing Market Size, Share and Trends 2025 to 2034*. https://www.precedenceresearch.com/satellite-market. Accessed: 2025-05-09. 2025 (cit. on pp. 10–13).

[19] Rebecca Reesman and Andrew Rogers. *Getting in Your Space: Learning from Past Rendezvous and Proximity Operations*. Tech. rep. OTR201800593. Approved for public release; distribution unlimited. Center for Space Policy and Strategy, The Aerospace Corporation, May 2018 (cit. on pp. 13, 14).

[20] Ron Ticker. *Restore-L Mission Information: Spacecraft Bus Concepts to Support the Asteroid Redirect Robotic Mission and In-Space Robotic Servicing*. PowerPoint presentation (PDF) via NASA Headquarters. accessed 2025-05-16. 2015. URL: https://www.nasa.gov/wp-content/uploads/2015/05/restore-L-info_nnh15heomd001_r7.pdf (cit. on p. 13).

[21] Rich Burns, Craig A. McLaughlin, Jesse Leitner, and Maurice Martin. «TechSat 21: Formation Design, Control, and Simulation». In: *Proceedings of the IEEE Aerospace Conference*. Air Force Research Laboratory, Space Vehicles Directorate. Kirtland Air Force Base, NM, USA, 2000 (cit. on p. 13).

[22] Owen Brown and Paul Eremenko. *The Value Proposition for Fractionated Space Architectures*. Tech. rep. Unpublished manuscript. Defense Advanced Research Projects Agency and Booz Allen Hamilton, 2011 (cit. on p. 13).

[23] Christophe Bonnal, Jean-Marc Ruault, and Marie-Christine Desjean. «Active debris removal: Recent progress and current trends». In: *Acta Astronautica* 85 (2013). Available online 16 January 2013, pp. 51–60. DOI: 10.1016/j.actaastro.2012.11.009 (cit. on p. 14).

[24] Elisa Capello. *01SRGMT – Dinamica e Controllo di Veicoli Spaziali*. Lecture notes, Politecnico di Torino. mailto:elisa.capello@polito.it. 2023–2024 (cit. on pp. 14, 15, 19, 22, 24, 39, 40, 42).

[25] Davide Celestini. «Navigation and Guidance Algorithms for In-Orbit servicing Rendezvous Mission». Master thesis – supervisors: Prof.ssa E. Capello, Ing. M. Saponara. MA thesis. Torino, Italia: Politecnico di Torino, Corso di Laurea in Ingegneria Aerospaziale, Thales Alenia Space, 2021 (cit. on pp. 15, 24).

[26] Yazhong Luo, Jin Zhang, and Guojin Tang. «Survey of Orbital Dynamics and Control of Space Rendezvous». In: *Chinese Journal of Aeronautics* 27.1 (2014). Available online 1 August 2013, pp. 1–11 (cit. on p. 16).

[27] Jr. Bryson Arthur E. and Yu–Chi Ho. *Applied Optimal Control: Optimization, Estimation and Control*. New York, NY: Taylor & Francis, 1975. ISBN: 0-89116-228-3 (cit. on p. 16).

[28] Pontani M. Romano V. Corallo F. and Teofilatto P. *Minimum-Fuel Orbit Transfers Using Modified Equinoctial Elements via Indirect Heuristic Method*. Technical Report. Rome, Italy: Sapienza Università di Roma and Thales Alenia Space Italia, 2019 (cit. on p. 16).

[29] Fei Ren, Ruichuan Li, Jikang Xu, and Chenyu Feng. «Indirect optimization for finite-thrust orbit transfer and cooperative rendezvous using an initial guess generator». In: *Advances in Space Research* 71 (2023), pp. 2575–2590. DOI: 10.1016/j.asr.2022.11.009. URL: https://www.sciencedirect.com/science/article/pii/S0273117722010328 (cit. on p. 17).

[30] Maozhang Zheng, Jianjun Luo, and Zhaohui Dang. «Optimal Impulsive Rendezvous for Highly Elliptical Orbits Using Linear Primer Vector Theory». In: *Chinese Journal of Aeronautics* 37.3 (2024), pp. 194–207. DOI: 10.1016/j.cja.2023.06.005 (cit. on p. 17).

[31] John T. Betts. «Survey of Numerical Methods for Trajectory Optimization». In: *Journal of Guidance, Control, and Dynamics* 21.2 (Mar. 1998), pp. 193–207. DOI: 10.2514/2.4231 (cit. on p. 18).

[32] Michael A. Patterson and Anil V. Rao. «GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using *hp*-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming». In: *ACM Transactions on Mathematical Software* 39.3 (2013), pp. 1–41. DOI: 10.1145/2484920.2484922 (cit. on p. 18).

[33] Danylo Malyuta, Taylor P. Reynolds, Michael Szmuk, Behcet Acikmese, and Mehran Mesbahi. *Fast Trajectory Optimization via Successive Convexification for Spacecraft Rendezvous with Integer Constraints*. arXiv preprint arXiv:1906.04857. [math.OC]. 2019 (cit. on p. 18).

[34] Zhenbo Wang. «A survey on convex optimization for guidance and control of vehicular systems». In: *Annual Reviews in Control* 57 (2024), p. 100957. DOI: 10.1016/j.arcontrol.2024.100957. URL: https://www.sciencedirect.com/science/article/pii/S1367578823001082 (cit. on p. 18).

[35] John M. Hanson, M. Wade Shrader, and Craig A. Cruzen. «Ascent Guidance Comparisons». In: *Journal of the Astronautical Sciences* 43.3 (July 1995). NASA–TM–112493, pp. 307–326 (cit. on p. 19).

[36] Robert D. Braun, Zachary R. Putnam, Bradley A. Steinfeldt, and Michael J. Grant. «Advances in Inertial Guidance Technology for Aerospace Systems». In: *Proceedings of the AIAA Guidance, Navigation, and Control (GNC) Conference*. Boston, MA, Aug. 2013 (cit. on p. 19).

[37] Robert D. Braun and Robert M. Manning. «Mars Exploration Entry, Descent, and Landing Challenges». In: *Journal of Spacecraft and Rockets* 44.2 (Mar. 2007), pp. 310–323. DOI: 10.2514/1.25116 (cit. on p. 20).

[38] Weilin Wang. «Cooperative augmented proportional navigation and guidance for proximity to uncooperative space targets». In: *Advances in Space Research* 71.3 (2023), pp. 1594–1604. DOI: 10.1016/j.asr.2022.09.026 (cit. on p. 20).

[39] Leone Guarnaccia, Riccardo Bevilacqua, and Stefano P. Pastorelli. «Suboptimal LQR-based spacecraft full motion control: Theory and experimentation». In: *Acta Astronautica* 122 (2016), pp. 114–136. DOI: 10.1016/j.actaastro.2016.01.016 (cit. on p. 21).

[40] Pedro A. Capó-Lugo and Peter M. Bainum. «Digital LQR control scheme to maintain the separation distance of the NASA benchmark tetrahedron constellation». In: *Acta Astronautica* 65.7–8 (2009), pp. 1058–1067. DOI: 10.1016/j.actaastro.2009.02.008 (cit. on p. 22).

[41] Peng Li and Zheng H. Zhu. «Model predictive control for spacecraft rendezvous in elliptical orbit». In: *Acta Astronautica* 146 (2018), pp. 339–348. DOI: 10.1016/j.actaastro.2018.03.025 (cit. on p. 22).

[42] Xiutao Gu, Liaoxue Liu, Lu Wang, Zhaobao Yu, and Yu Guo. «Energy-optimal adaptive artificial potential field method for path planning of free-flying space robots». In: *Aerospace Science and Technology* 143 (2024), p. 108935. DOI: 10.1016/j.ast.2023.108935 (cit. on p. 23).

[43] Howard D. Curtis. *Orbital Mechanics for Engineering Students*. 3rd ed. Oxford, UK: Elsevier Butterworth-Heinemann, 2013. ISBN: 978-0-08-097747-8 (cit. on pp. 24, 35–37, 43–47, 49, 50).

[44] Benjamin A. Stahl and Robert D. Braun. «Low-Thrust Trajectory Optimization Tool to Assess Options for Near-Earth Asteroid Deflection». In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. Aug. 2008. DOI: 10.2514/6.2008-6255. URL: https://doi.org/10.2514/6.2008-6255 (cit. on p. 25).

[45] Davide Conte and David B. Spencer. «Targeting the Martian Moons via Direct Insertion into Mars' Orbit». In: *AIAA/AAS Astrodynamic Specialist Conference*. Vail, CO, Aug. 2015 (cit. on p. 34).

[46] Walter Hohmann. *The Attainability of Heavenly Bodies*. NASA TT F-44. Translation of "Die Erreichbarkeit der Himmelskörper," R. Oldenbourg, Munich–Berlin, 1925. Washington, DC: National Aeronautics and Space Administration, 1960 (cit. on pp. 44, 105, 107).

[47] Mauro Pontani, Pradipto Ghosh, and Bruce A. Conway. «Particle Swarm Optimization of Multiple-Burn Rendezvous Trajectories». In: *Journal of Guidance, Control, and Dynamics* 35.4 (2012), pp. 1190–1203. DOI: 10.2514/1.55592 (cit. on pp. 52, 54, 65, 66, 68, 76, 77, 134).

[48] Thomas Bäck and Hans-Paul Schwefel. *An Overview of Evolutionary Algorithms for Parameter Optimization*. Technical Report. P.O. Box 50 05 00, D–4600 Dortmund 50, Germany: University of Dortmund, Department of Computer Science, Chair of Systems Analysis, 1993 (cit. on pp. 52, 53).

[49] Riccardo Poli, James Kennedy, and Tim Blackwell. «Particle Swarm Optimization: An Overview». In: *Swarm Intelligence* 1.1 (2007). Received: 19 December 2006 / Accepted: 10 May 2007, pp. 33–57. DOI: 10.1007/s11721-007-0002-0 (cit. on p. 54).

[50] Yuhui Shi and Russell Eberhart. «A Modified Particle Swarm Optimizer». In: *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*. Part of the IEEE World Congress on Computational Intelligence. Anchorage, AK, USA: IEEE, 1998, pp. 69–73 (cit. on p. 54).

[51] Hana Rhim. *How Does Particle Swarm Optimization Work?* Baeldung. Reviewed by Grzegorz Piwowarek. May 2024. URL: https://www.baeldung.com/cs/pso (visited on 06/04/2025) (cit. on p. 56).

[52] Rainer Storn and Kenneth Price. *Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces*. Technical Report TR-95-012. International Computer Science Institute, Mar. 1995 (cit. on pp. 56, 60).

[53] Swagatam Das and Ponnuthurai N. Suganthan. «Differential Evolution: A Survey of the State-of-the-Art». In: *IEEE Transactions on Evolutionary Computation* 15.1 (2011), pp. 4–31. DOI: 10.1109/TEVC.2010.2059031 (cit. on pp. 58, 59).

[54] Kenneth Price, Rainer Storn, and Janne Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Heidelberg: Springer, 2005. ISBN: 978-3-540-21268-4 (cit. on p. 60).

[55] Nikolaus Hansen and Andreas Ostermeier. «Completely Derandomized Self-Adaptation in Evolution Strategies». In: *Evolutionary Computation* 9.2 (2001), pp. 159–195. DOI: 10.1162/106365601750190398. URL: https://direct.mit.edu/evco/article/9/2/159/1355/Completely-Derandomized-Self-Adaptation-in-Evolution (cit. on pp. 61, 74, 83).

[56] U. Tan, O. Rabaste, C. Adnet, and J.-P. Ovarlez. «On the Eclipsing Phenomenon with Phase Codes». In: *Transactions of the French Society for Signal Processing* 58.4 (2021), pp. 123–130. DOI: 10.1007/s00034-021-02123-4. URL: https://link.springer.com/article/10.1007/s00034-021-02123-4 (cit. on p. 64).

[57] Liqiang Hou and Arun Misra. «Traveling Salesman Problem of optimal debris removal sequence using non-population gradient search». In: *Acta Astronautica* 215 (2024), pp. 373–386. DOI: 10.1016/j.actaastro.2023.12.045 (cit. on p. 86).

[58] Dario Izzo, Ingmar Getzner, Daniel Hennes, and Luís Felismino Simões. «Evolving Solutions to TSP Variants for Active Space Debris Removal». In: *Proceedings of the 2015 Genetic and Evolutionary Computation Conference*. GECCO '15. New York, NY, USA: ACM, 2015, pp. 1207–1214. ISBN: 978-1-4503-3472-3. DOI: 10.1145/2739480.2754727 (cit. on p. 86).

[59] Agostino De Marco. *Integrazione di sistemi di equazioni differenziali con il metodo di Runge–Kutta*. Appunti delle lezioni di Dinamica del Volo. Ing. Agostino De Marco, agodemar@unina.it. Università degli Studi di Napoli "Federico II", Dipartimento di Progettazione Aeronautica, 2023 (cit. on p. 94).

[60] Lorenzo Casalino. *Space Propulsion*. Lecture notes, Politecnico di Torino. mailto:lorenzo.casalino@polito.it. 2023–2024 (cit. on pp. 102, 109).

[61] Jing Li. «Minimum-fuel two-impulse transfer between coplanar noncoaxial elliptical orbits». In: *Advances in Space Research* 73.1 (Jan. 2024), pp. 143–159. DOI: 10.1016/j.asr.2023.10.008 (cit. on p. 103).

[62] Daniele Mortari, Matthew P. Wilkins, and Christian Bruccoleri. «On Sun-Synchronous Orbits and Associated Constellations». In: *Advances in the Astronautical Sciences* 154 (2015), pp. 1441–1458 (cit. on p. 113).

[63] GlobalSpec. *Chapter 12: Relationships Between Swath Width, Footprint, Integration Time, Sensitivity, Frequency, and Other Parameters for Satellite-Borne, Real Aperture Imaging Systems*. GlobalSpec Reference Library, "Microwave Radiometer Systems: Design and Analysis, Second Edition". Retrieved from https://www.globalspec.com/. 2006 (cit. on p. 113).

[64] Rocket Lab. *Launch – Electron: Rutherford Engine*. https://www.rocketlabcorp.com/launch/electron/. Accessed June 15, 2025. 2025 (cit. on p. 117).

[65] James R. Wertz and Wiley J. Larson. *Space Mission Analysis and Design*. 3rd. El Segundo, CA and Dordrecht, The Netherlands: Microcosm Press and Kluwer Academic Publishers, 1999. ISBN: 079236209X (cit. on p. 121).