Optimal scheduling for agile Earth observation satellites

Jochim Maene

Delft University of Technology

TU Delft

OHB

Cover Image: Artist impression of GeoEye 2 orbiting the Earth.
Courtesy: DigitalGlobe/Lockheed Martin

Optimal scheduling for agile Earth observation satellites

by

Jochim Maene

to obtain the degree of Master of Science
at the Delft University of Technology

to be defended publicly on Thursday August 22, 2019 at 2:00 PM.

Student number:	4371275	
Project duration:	January 7, 2019 – August 22, 2019	
Supervisors:	ir. R. Noomen,	TU Delft
	Dr.ir. F.J. De Bruijn,	OHB System AG
	ir. C. Palm	OHB System AG
Thesis committee:	Dr.ir. E.J.O Schrama,	TU Delft (chair)
	ir. R. Noomen,	TU Delft
	Dr.ir. F.J. De Bruijn,	OHB System AG
	Dr.ir. M. Snellen	TU Delft

Preface

Before you lies the MSc. thesis *Optimal scheduling for agile Earth observation satellites*. This thesis was written as final milestone to obtain the degree of Master of Science in Aerospace Engineering at Delft University of Technology. A novel approach to generate an optimized schedule of image requests will be introduced in this work. The presented method and results can be of interest for space mission designers and operators of agile Earth observation satellites.

This research was carried out at the department for Mission Analysis and Flight Dynamics of OHB System AG Bremen, in collaboration with the Astrodynamics and Space Missions section of Delft University of Technology. I want to thank OHB for providing me with an office space and financial support for the thesis project.

Of course, the successful completion of this research would not have been possible without the people who surrounded me the past seven months. At foremost, I like to thank my supervisors who guided me in the right direction throughout this research project. Ron, thank you for not sparing any time to review this work and your patience during our weekly meetings. Also, your thoughtful comments always made me critical of my own work. Ferdi, my sincere gratitude goes out to you for offering this research opportunity. When introducing me to the topic, it was certainly not a exaggerated that the research topic would be challenging. Nevertheless, this only made it a more rewarding experience. Also thank you for helping out with any other technical question I had. Chris, thank you for always looking at my results with a critical eye. My thesis definitely become more practically applicable because of you.

Finally, my gratitude goes out to my parents for their unconditional support for all my endeavours far away from home.

Jochim Maene
Bremen, July 2019

Abstract

The past decade has seen a continuous increase of Earth observation satellite missions. Even more, the market is foreseen to boom in the future since Earth observation is widely recognised as an important tool to address key global problems such as climate change, resource exploitation and disaster mitigation. Furthermore, applying data analysis on Earth observation data is also of commercial concern for agriculture and resource management. In general, a commercial trend exists now towards higher resolution imagery, which drives the use of agile satellites. These satellites perform observation attitude manoeuvres by rotating around all three satellite axes to specifically point the observation sensor to the target only. Nevertheless, a disadvantage of agile satellites is the increased computational complexity to compute the activity schedule of the satellite. In fact, the problem can be interpreted as a selective travelling salesman problem with time windows for which the travel between the cities is also an optimal control problem with dynamics and actuator constraints. The solution of the optimal control problem is disregarded and approximated by heuristics in literature due to its high complexity for implementation in the scheduling algorithm. Overall, this scheduling problem is proven to be unsolvable in polynomial time, and is therefore a key problem in state-of-the art satellite management. Designing an optimal scheduling method such that as many observations are collected as possible has a major impact on the profits for the satellite operators. Therefore, the scheduling problem of agile Earth observation satellites in not only the challenging subject of this research, but is also a problem with commercial value.

Considering the simplifying assumptions in previous research, this MSc thesis presents a novel method which can consider the optimization of the manoeuvres in between the targets. This optimal control problem is solved using pseudospectral collocation and an interior point optimizer. Since the optimal control problem is computationally expensive, the solution must be approximated for implementation into the scheduling algorithm. It was chosen to generate a database of solutions of the optimal control problem and using an artificial neural network to obtain the approximated optimal attitude manoeuvres. Furthermore, several algorithmic approaches to solve the highly combinatorial scheduling problem have been designed and evaluated. A dynamic programming algorithm for presorted targets and a simulated annealing algorithm were found to provide the best quality solution for larger problem sizes. Additionally, a new exact scheduling algorithm was developed based on dynamic programming logic. The algorithm is shown to be able to plan up to one hundred targets for a given restricted number of visible targets at each epoch, while previous research was only able to solve problem sizes of up to twelve targets. Furthermore, the new exact scheduling algorithm is also shown to have unmatched performance for cases of up to twelve targets, such that this should not be considered a difficult problem anymore. In addition, the established scheduling approach provides a significant improvement over any previous research due to the introduction of the near time-optimal manoeuvres between the targets. The expected increase in scheduling performance compared to the state-of-the-art is shown to be around 10%. Considering the high cost of agile Earth observation satellites, these advancements can offer important profit increases for the satellite operators.

List of acronyms

AEOS	Agile Earth Observation Satellite
BVP	Boundary Value Problem
CGL	Chebychev-Gauss-Lobatto
CMG	Control Moment Gyro
CNES	Centre National d'Études Spatiales
DP	Dynamic programming
ECEF	Earth Centred Earth Fixed
ECI	Earth Centred Inertial
ENU	East, North, Up
EO	Earth Observation
FFT	Fast Fourier Transform
FLASH	FLight dynamics and Analysis Software Hub
GA	Greedy algorithm
GCRF	Geocentric Celestial Reference Frame
GPR	Gaussian Process Regression
ICRF	International Celestial Reference Frame
IMPULSE	IMage Planning Utility for agiLe SatEllites
IP	Interior Point
IPOPT	Interior Point OPTimizer
IVP	Initial Value Problem
JPL	Jet Propulsion Laboratory
KKT	Karush Kuhn-Tucker
LG	Legendre-Gauss
LGL	Legendre-Gauss-Lobatto
LGR	Legendre-Gauss-Radau
MILP	Mixed Integer Linear Programming
MS	Multiple shooting
MSE	Mean squared error
NLP	Non-Linear Programming
NP	Non-deterministic Polynomial time
OCP	Optimal Control Problem
ODE	Ordinary Differential Equation
PSM	Pseudo-Spectral Method
QP	Quadratic Programming
RK	Runge-Kutta
SA	Simulated annealing
SNOPT	Sparse Nonlinear OPTimizer
SQP	Sequential Quadratic Programming
STK	Systems Tool Kit
SVM	Support Vector Machine
TPBVP	Two Point Boundary Value Problem
TSP	Travelling Salesman Problem
WORHP	We Optimise Really Huge Problems

List of symbols

Roman symbol	Description	Unit
a	Semi-major axis	m
$\mathbf{a}$	Non-linear programming equality constraint vector	-
$\mathbf{a}_d$	Perturbing acceleration	$\mathrm{m/s^2}$
b	Bias of artificial neuron	-
$\mathbf{b}$	Non-linear programming inequality constraint vector	-
$\mathbf{c}$	Constraint defect vector	-
$\mathbf{C}$	Path constraints for the optimal control problem	-
$d_{\max}$	Maximum possible needed time for attitude between two targets	
$d_{\min}(i,j)$	Minimum attitude manoeuvre duration between target i and j	s
$\mathbf{D}$	Spectral differentiation matrix	-
$\mathbf{e}$	Error vector	-
$\hat{\mathbf{e}}_i$	Standard basis vector element indicating axis direction	-
$\mathbf{f}$	First-order derivatives for dynamical equations	-
$\mathbf{g}$	Gradient vector	-
$\mathbf{G}$	Jacobian matrix	-
h	Step size	s
H	Hamiltonian	-
$\mathbf{H}$	Hessian matrix	-
$\mathbf{I}$	Inertia matrix	$\mathrm{kg \cdot m^2}$
j	Array to store objective function values for scheduling algorithms	-
J	Objective function	-
J_2	Second-order gravity parameter	-
$\mathbf{k}$	Slope for Runge-Kutta integration	-
ℓ	Lagrange basis polynomial	-
l	Line rate of scanner	1/s
L	Integral term in objective function	-
L	Lagrange interpolant	-
$\mathcal{L}$	Lagrangian	-
m	Array to store index values for scheduling algorithms	-
n_u	Number of control variables	-
n_x	Number of state variables	-
n_T	Number of targets in a set	-
p_i	Priority parameter for target observation	-
P	Probability distribution	-
$\mathbf{q}$	Quaternion vector	-
r	Radial distance	m
$\mathbf{r}$	Position vector	m
R	Pearson correlation coefficient	-
$\mathbf{R}$	Rotation matrix	-
$R_\oplus$	Radius of the Earth	m
$\mathbb{R}$	Set of real numbers	-
s	Binary variable for scanning direction	-
S	Subset of targets	-
t	Time	s
$\mathbf{T}$	Torque	$\mathrm{N \cdot m}$
T	Temperature for simulated annealing	K
T	Set of observation requests	-
$\mathbf{u}$	Control vector	-

U_p	Perturbing potential	m^2/s^2
$\mathbf{v}$	Velocity vector	m/s
v	Maximum number of visible targets during scheduling	-
$\mathbf{W}$	Weight matrix	-
x_i	Binary decision variable for target selection	-
$\mathbf{x}$	State vector	-
y_{ij}	Binary decision variable for target sequence	-
$\mathbf{y}$	Vector of optimization variables	-

Greek symbol	**Description**	**Unit**
α_s	Azimuth angle of scanning direction	deg
α_t	Azimuth angle of target location	deg
α	Neural net training regularization parameter	-
β	Neural net training regularization parameter	-
γ	Effective number of parameters in neural network	-
$\boldsymbol{\delta}$	Target tracking state vector	-
ϵ	Elevation angle	deg
$\boldsymbol{\zeta}$	Numerical integration defect	-
η	Off-nadir angle	deg
θ	Angular resolution	deg
θ	Pitch angle	deg
θ	True anomaly	deg
θ	Cooling rate in simulated annealing algorithm	-
λ	Longitude	deg
$\boldsymbol{\lambda}$	Lagrange multiplier for equality constraints	-
μ	Gravitational constant	m^3/s^2
μ	Barrier parameter	-
$\boldsymbol{\mu}$	Lagrange multiplier for inequality constraints	-
ν	Argument of latitude	deg
σ	Activation function	-
τ	Non-dimensional time	-
ϕ	Terminal cost function	-
ϕ	Roll angle	deg
ϕ	Basis function for interpolation	
$\boldsymbol{\Phi}$	Integration increment function	-
φ	Latitude	deg
ψ	Yaw angle	deg
$\boldsymbol{\psi}$	Boundary conditions for the optimal control problem	-
$\boldsymbol{\omega}$	Angular velocity vector of the satellite	deg/s

Contents

1 Introduction

This introduction will present the framework and outline of this MSc thesis research project. A brief introduction to the subject and existing literature is given in Section 1.1. Next, Section 1.2 continues by defining the scope of research and the structure of this thesis.

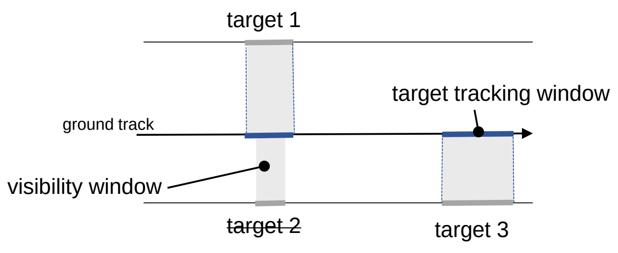
1.1. Research framework

The research framework gives the motivation and heritage related to this research in Sections 1.1.1 and 1.1.2 respectively. Furthermore, the research gaps are identified in Section 1.1.3.

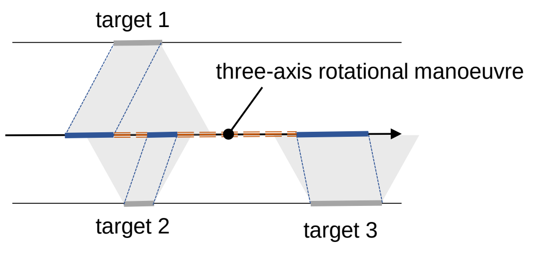
1.1.1. Research motivation

During the past decade, new commercial initiatives from companies such as Planet have demonstrated the business potential of Earth Observation (EO). These companies use satellite constellations to sell medium-resolution imagery with short revisit times. Typically, a large imagery database is provided, from which the customer can buy the desired observation data. However, a new commercial trend exists towards high-resolution imagery, below 1 m sampling (Denis et al., 2017). As a result of the higher resolution, a smaller area can be imaged at once. This problem justifies a business model of 'imagery-on-demand', where the customer orders custom images of a targeted area. Therefore, instead of scanning the entire area using just one degree of freedom, as is conventional, it is now more efficient for satellite operators to specifically point the camera to the custom area of interest only. This is realized by using three-axis attitude control which rotates the satellite to point at the scheduled targets over the area. The added benefit of this extra freedom is illustrated in Figure 1.1. For the non-agile satellite, the only possibility for target acquisition is by using the roll axis to point to scanner to the target. On the other hand, an infinite number of possibilities exist to acquire a target using the agile satellite, since the acquisition time can be chosen from a time window. Therefore, in this example the conflict between target 1 and 2 can be resolved using three-axes attitude control, such that more observations can be collected. Satellites that use three-axis attitude control to increase the operational efficiency are referred to as agile Earth observation satellites, or AEOS (Lemaître et al., 2002).

In order to be competitive and maximize the revenues of the imagery-on-demand business model, it is essential for the satellite operator that as many customer requests can be satisfied as possible. This



(a) Non-agile satellite.



(b) Agile satellite.

Figure 1.1: EO scheduling using **(a)** non-agile and **(b)** agile satellites (top view on Earth surface).

can be stated as a scheduling problem, with the additional challenge of generating optimal three-axis rotational manoeuvres between the target observations. The AEOS scheduling problem is proven to be NP-hard by Lemaître et al. (2002), making it a key problem in state-of-the-art Earth observation satellite management. Therefore, the scheduling of agile observation satellites is not only the challenging subject of this research, but is also a problem of significant commercial interest

1.1.2. State-of-the-art Earth observation scheduling methods

The Earth observation scheduling research field has a large heritage from decades of Earth observation missions. Several works such as by Globus et al. (2004) performed an extensive comparison of scheduling methods. However, most of this research is only applicable to non-agile satellites, which is a problem with significantly reduced complexity. Specific research about the AEOS scheduling problem is rather limited. Lemaître et al. (2002) were the first authors who treated the AEOS scheduling problem, in the context of the Pleiades Earth observation constellation of CNES. To date, this is still the most cited work about AEOS scheduling, as it gives a comprehensive overview of the problem complexity and most successful algorithms to solve this problem. Given the complexity of the problem, the authors also organized a public challenge, which opened up competition for the best scheduling algorithms. The problem was tackled by experts in the operations research field, who found that highly customized heuristic algorithms have the best performance for the given data set. The approach of the second-best algorithm of the competition is described by Cordeau and Laporte (2005). After the competition, Habet et al. (2010) suggested an improved heuristic algorithm for which the upper bounds of the optimal solution are calculated using dynamic programming.

Due to the increasing commercial interest in space, there is currently also a renewed attention for the AEOS scheduling problem. Several new attempts use branch-and-bound approaches (Chu et al., 2017) or integer programming (She et al., 2018). These algorithms provide good performance when scheduling up to 50 targets. On the other hand, several papers use improved heuristics methods, such as (Lu et al., 2018) and (Xu et al., 2016). Furthermore, also some algorithms considering the practical implementations have been published. Examples are found for the Terra Bella constellation (Augenstein et al., 2016) and in the framework of the MUSIS project (Grasset-Bourdel et al., 2011).

1.1.3. Current research gap

Most research about the AEOS scheduling problem was executed by the operations research community, with the disregard of attitude dynamics or mechanical satellite constraints. That is, the focus has been primarily on the optimization of the actual schedule, while the attitude manoeuvre in between the targets is often assumed as a constant or based on a conservative estimate. The manoeuvre is typically not optimized since it requires the solution of a complex optimal control problem, which is too computationally expensive for implementation in a scheduling algorithm. For example, Lemaître et al. (2002) predict the manoeuvre time based on the ground distant between two targets and Nag et al. (2017) use an eigenaxis rotation as time-optimal manoeuvre between the targets. However, the eigenaxis manoeuvres are typically sub-optimal and can not model any complex boundary conditions or angular velocity constraints. The attempt made by Beaumet et al. (2007) to obtain a high-quality estimate of optimal manoeuvre time using constraint programming was unsuccessful. The authors report that this approach is not robust and the quality of the bounds of the optimal time are poor. Clearly, optimization of the attitude trajectory between the targets would outperform previous methods, such that the reduction in attitude manoeuvre duration results in an improved schedule. Additionally, the manoeuvre time in between the targets will depend on the time at which the manoeuvre is initiated. This time-dependency has only been investigated by Liu et al. (2017) and Peng et al. (2018). Other research, such as by Lemaître et al. (2002), neglect this time-dependency and repair the schedule if a violation occurs due to this assumption.

Next to the attitude manoeuvres, also the scheduling algorithms have room for improvement. While many heuristic algorithms have been proposed with very efficient performance for large problem sizes, no known algorithm exists which can solve instances of more than 12 targets exactly on a normal desktop computer. These exact solutions are usually applied to evaluate the performance of the heuristic methods and are generated by formulating the scheduling problem as a Mixed Integer Linear Programming (MILP) problem. The MILP problem is then solved using a solver such as CPLEX. However, for more than 12 targets, the RAM memory is typically insufficient to generate an exact solution, as is reported by Liu et al. (2017), Grasset-Bourdel et al. (2011) and Oguz et al. (2010).

1.2. Thesis research scope

This section formulates the research questions, gives an overview of the methodology and presents the structure of this report in Sections 1.2.1, 1.2.2 and 1.2.3 respectively.

1.2.1. Research questions

The state-of-the-art literature review and identified research gaps lead to the following research questions for this thesis:

1. How can the time-optimal slew manoeuvres between targets be approximated, such that it is suitable for implementation in scheduling algorithms, while considering the time-dependency of the manoeuvres?
    - 1.1. How can the attitude manoeuvres be optimized for minimal slew time, while considering all satellite constraints and the time-dependent final boundary condition, such that the problem can be solved in a fast and robust way?
    - 1.2. Which technique and input parameters can be used to approximate the optimal manoeuvre time, given a database of optimal solutions?
    - 1.3. What is the efficiency of the approximation, in terms of quality and required computational time?
    - 1.4. What is the improvement that can be offered by using this approximation instead of the approximating manoeuvres as in state-of-the-art literature?
2. How can the approximated time-optimal manoeuvres be combined with exact scheduling algorithms to locally optimize the observation of 12 image requests during a single pass over the area of interest?
    - 2.1. If not possible with exact algorithms, which heuristics provide the best performance?
3. Given the approximated time-optimal manoeuvres, which heuristics have the best performance to schedule larger problem sizes of up to 500 image requests, distributed over a single satellite orbit?

1.2.2. Research methodology

Given the research questions from Section 1.2.1, this section explains the followed research methodology. First, a solution to the time-optimal manoeuvres between the targets is developed. The required kinematics for the boundary conditions of this optimal control problem are stated as an ordinary differential equation. Next, a trade-off is performed to select the best method to transcribe the optimal control problem into a non-linear programming (NLP) problem. Furthermore, several numerical techniques are included to improve the robustness of the solution. Using the robust algorithm, a database of solutions to the optimal control problem is constructed. This database is then used to get an approximation of the time-optimal solution, which can be implemented in the scheduling algorithms. The specific technique to get the approximate manoeuvre time is the result of a trade-off.

As next topic of this research, the scheduling problem is tackled. Two simplifying assumptions are proposed, such that the search space of the optimal solution is greatly reduced without a big impact on the scheduling performance. Next, the scheduling problem is formulated as an MILP problem. Furthermore, a novel approach to obtain the exact solution for a restricted number of targets is established using dynamic programming principles. Also, three more heuristic algorithms are developed and investigated for larger problem sizes.

1.2.3. Report structure

The technical content of this report is structured in the following seven chapters:

Chapter 2 establishes a description of the problem at hand and discusses the complexities that have to be considered. Also the mission for which the problem will be tackled is introduced.

Chapter 3 provides the necessary dynamic models and reference frames to describe the motion of the satellite for the six degrees of freedom. Furthermore, a kinematic relation is derived in this chapter to compute the attitude profiles of the satellite while a target is being scanned.

Chapter 4 gives the background of the non-linear programming fundamentals and optimal control theory, which form the basis to calculate the optimal attitude manoeuvres. Furthermore, a numerical study is performed to choose the transcription method and the non-linear programming solver. Also three example solutions of the optimal attitude manoeuvres are provided.

Chapter 5 continues by explaining the methodology to approximate the optimal manoeuvres using a database. Several methods are explained and compared.

Chapter 6 treats the main scheduling problem. The inputs to the scheduling methods are described, as well as the assumptions. Furthermore, a selection is made of relevant algorithms that are developed for this research. Also all algorithms are described in detail, together with their pseudocode. This chapter is concluded by a trade-off about the performance for the scheduling algorithms for different problem sizes.

Chapter 7 present how the simulations are implemented. The simulation architecture and numerical tools are treated.

Chapter 8 reports the performed verification efforts for the optimal attitude manoeuvres and the scheduling algorithms.

Supplementary theory and results are collected in Appendices A, B and C.

2 Problem and mission description

Before starting to answer the research questions, this chapter will give a comprehensive overview of the problem at hand. An overview of the problem and the reasons for its complexity is given in Section 2.1. Next, a satellite mission is described in Section 2.2 for which the AEOS scheduling problem will be applied.

2.1. Problem complexity characterization

As was stated in Chapter 1, an AEOS uses all three satellite axes to point a body-fixed camera to a target. Compared to a non-agile satellite, the advantage is the increased degree of freedom for planning the target observations. Both the starting time and the direction in which the target is scanned can now be chosen, which gives rise to an infinitely large number of possibilities to acquire a target. This large search space is also directly one of the most challenging aspects of this problem. The objective is to find the activity plan which maximizes the profits of the satellite operators. This implies an optimization problem that is twofold:

1. The attitude control inputs should be optimized such that the satellite points in minimum time to a predefined target while considering the motion of the satellite with respect to the target. This is an optimal control problem, with the added complexity that the end state is constrained depending on the final time. As constraints, the limitations of the actuators need to be taken into account. Details on how to solve this kind of optimization problem are given in Chapter 4.
2. The optimal activity plan should be found such that the collection of observations is maximized. This means that certain targets and their order need to be selected in an optimal way. Possibly not all targets can be visited, such that this problem is a combination of the travelling salesman problem and the knapsack problem (Grasset-Bourdel et al., 2011). More specifically, it is an example of oversubscribed scheduling. The observation scheduling is discussed in detail in Chapter 6.

Because the optimal control problem and scheduling problem are inextricably linked, the lower-level optimal control problem has to be solved within the upper-level scheduling problem. This nested type of optimization is often referred to as bi-level optimization. A similar problem exists in highly automated production plants, where robots need to visit multiple moving targets. A paper by Palagachev and Gerdts (2017) explains different ways in which this scheduling problem can be approached, when the robots need to be optimally controlled. As shortly introduced in Section 1.2.2, this research will find an approximation of the low-level optimization problem, such that it can be efficiently implemented in the high-level scheduling problem.

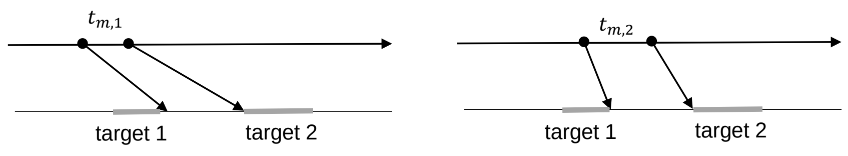


Figure 2.1: Illustration of how the manoeuvre time depends on the angular distance between two subsequent targets (view perpendicular to Earth surface).

Due to the agility of the satellite, the problem is also complicated in various other ways. The most important implications are:

- As shown in Figure 2.1, the necessary time to manoeuvre between two targets depends on the time at which this manoeuvre is started. For this two-dimensional case, it can be easily inferred that a smaller angular distance between the targets viewed from the satellite implies a faster required manoeuvre, such that $t_{m,1} < t_{m,2}$. Next to the angular distance, also the difference in the required angular velocity to scan the targets will influence the manoeuvre times. Therefore, the optimal manoeuvre time between two targets is affected by the order in which these targets are visited. If a target is added, removed or swapped with another target in a scheduling sequence, this means all subsequent manoeuvre times must be re-evaluated.
- Since the lower-level optimal control problem aims to find the minimum-time manoeuvres, the final state constraints will depend on the minimum time. The final state needs to be fixed so that the satellite points at the moving target. This complex optimal control problem shows similarities with minimum-time interception problems.
- The image strips can be acquired by scanning in two different directions (see Section 2.2). Furthermore, the target can be scanned any scanning direction angle w.r.t the local North. This again greatly increases the search space for the optimal schedule.

From above statements it is clear that the problem is especially complicated due to the time dependent manoeuvres and the highly combinatorial search space. In fact, the number of possible ways to acquire the targets equals

$$2n_{az}n_T \sum_{i=1}^{n} i!\binom{n}{i} = 2n_{az}n_T \sum_{i=1}^{n} \frac{n!}{(n-i)!} \tag{2.1}$$

where n is the number of targets that can be acquired, and the azimuth scanning angle and the time epochs are discretized by n_{az} and n_T points respectively. For example, if the time window is discretized into steps of 1 second and the time window has an average duration of 60 seconds, there are 61 possible epochs for a single image acquisition. Also when taking $n_{az} = 36$, this would result in already $5.6 \cdot 10^{12}$ possible ways to acquire 12 targets.

2.2. Mission case study

In order to measure the performance of the novel method of AEOS scheduling as developed during the thesis, an example mission is specified. The underlying mission concerns an agile small satellite in a polar orbit at 500 km altitude. The exact initial state of the orbit is given in Table 2.2. The payload consists of a small camera, which is fixed to the body of the satellite. The camera concentrates the incoming light on an array of photo-diodes, such that the instantaneous observation is in fact a single, one-dimensional stroke of pixels. By sweeping the camera over the area of observation using the three-axes attitude control, an image can be formed. This process is illustrated in Figure 2.2. The number of strokes that can be imaged per second is referred to as the line rate, while the angular resolution determines the size of the pixels. The advantage of these sensors is the high spatial resolution, which is the reason that these scanners are used for most agile Earth observation satellites. For further analysis, the requested images will always be given as a single rectangular image strip. This image strip can be scanned in two directions using the line scanner. In case the satellite scans in flight direction, this will be indicated as the forward scanning direction. Vice versa, the backward scanning direction means the scanner moves against the flight direction. Also, the image acquisition duration is taken as 2 seconds. As shown in Table 2.1, the line rate is 5000 pixels per seconds, such that an image has a length of 10,000 pixels. A stabilization time of 1 second before acquiring the target must guarantee high image quality.

Furthermore, the satellite has three-axes attitude control, capable of fast angular manoeuvres. The performance is given in Table 2.1, together with other mission constraints for the observation scheduling. Due to the high torque requirement of an agile satellite, Control Moment Gyros (CMGs) are a common choice. Therefore, these actuators are also adopted for this research. Often, four CMGs in

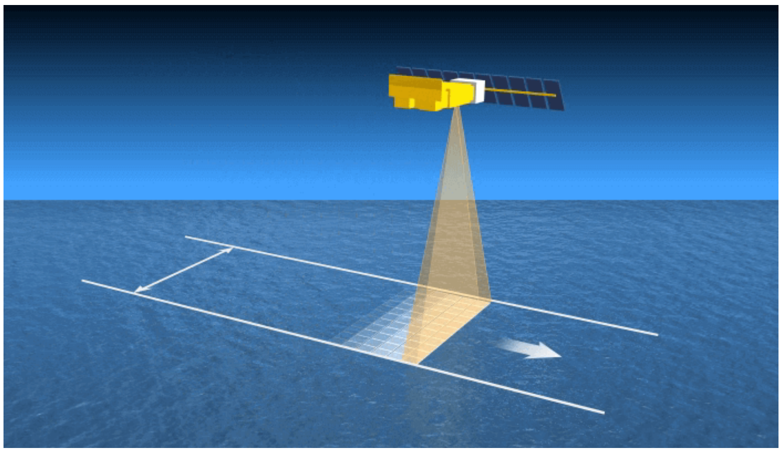


Figure 2.2: Illustration of an image acquisition using a line scanner (Krishnamurti et al., 2013).

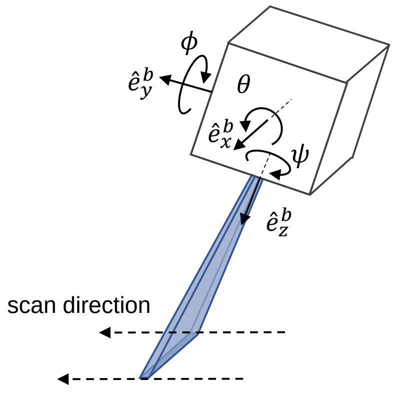


Figure 2.3: Satellite body axes and line scanner alignment.

pyramid configuration are chosen for redundancy. This configuration has the additional advantage that the momentum envelope is nearly spherical as shown in Figure 2.4, resulting in an almost equal momentum capability for all three axes (Tekinalp et al., 2009). From the momentum envelope, the angular velocity and torque span can be found, which is out of the scope of this research. The angular velocity and torque ellipsoidal constraints have the following form:

$$\begin{aligned} \frac{\omega_x^2}{\omega_{\max,y}^2} + \frac{\omega_y^2}{\omega_{\max,y}^2} + \frac{\omega_z^2}{\omega_{\max,z}^2} \leq 1 \\ \frac{T_x^2}{T_{\max,x}^2} + \frac{T_y^2}{T_{\max,y}^2} + \frac{T_z^2}{T_{\max,z}^2} \leq 1 \end{aligned} \tag{2.2}$$

where the values for $\boldsymbol{\omega}_{\max}$ and $\mathbf{T}_{max}$ are specified in Table 2.1.

Furthermore, the axes on the satellite body are defined as illustrated in Figure 2.3 and go through the center of mass of the satellite. The z-axis is aligned with the pointing direction of the camera. The camera is fixed to the satellite, such that the instantaneous scanning line is parallel to the x-axis. The y-axis completes this right-handed frame and such that it is positive in flight direction.

Table 2.1: Satellite hardware and mission properties.

Parameter	Value	Unit
Maximum torque $[T_x, T_y, T_z]_{\max}$	[0.05, 0.03, 0.04]	Nm
Maximum angular velocity of the satellite $[\omega_x, \omega_y, \omega_z]_{\max}$	[2.5, 3, 2.8]	deg/s
Maximum off-nadir angle	30	deg
Scanner line rate	5000	1/s
Scanner angular resolution	$2 \cdot 10^{-6}$	rad
Image acquisition duration	2	s
Stabilization duration	1	s
Satellite inertia matrix	$\begin{bmatrix} 2.0 & 0.25 & 0.2 \\ 0.25 & 3.0 & 0.1 \\ 0.2 & 0.1 & 4.0 \end{bmatrix}$	kg m^2

Table 2.2: Orbital parameters of the initial state for the reference orbit as used for all scheduling cases. The orbit is propagated using the initial epoch at January 1st, 2019.

a [m]	e [-]	inc [deg]	RAAN [deg]	AOP [deg]	TA [deg]
$6.878 \cdot 10^6$	0	90	0	0	0

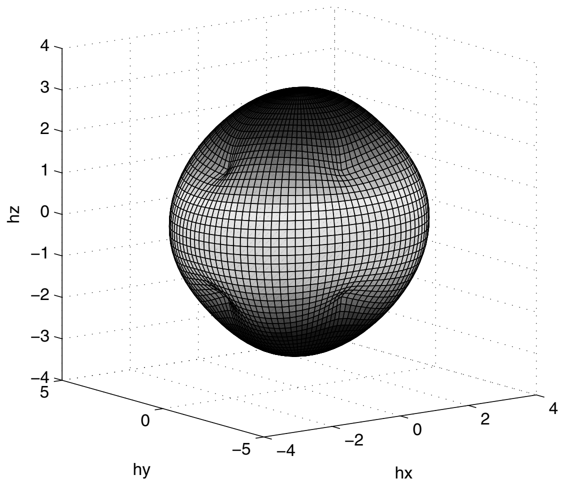


Figure 2.4: CMG momentum envelope for a pyramid configuration (Tekinalp et al., 2009).

3 Orbital mechanics

The flight dynamics and kinematics presented in this chapter contribute to the required tools to describe the behaviour of the six degrees of freedom of the satellite in orbit. Firstly, the three-dimensional axes used as reference are defined in Section 3.1. After choosing a set of three-dimensional axes, the translational motion of the center of mass of the satellite is described in Section 3.2. Similarly, Section 3.3 explains the dynamics of the angular orientation w.r.t. the reference axes. Lastly, the satellite rotation is related to the tracking of a target on the Earth surface in Section 3.4.

3.1. Reference frames

In order to properly define the position and attitude of a satellite at a certain moment of time, a reference frame needs to be defined. The reference frames which do not experience any acceleration, or inertial reference frames, are described in Section 3.1.1. On the other hand, the used non-inertial reference frames are given in Section 3.1.2.

3.1.1. Inertial reference frames

The current fundamental reference frame for astronomical studies is the International Celestial Reference Frame (ICRF), which is a quasi-inertial frame that is centered at the barycenter of the Solar System. The ICRF axes directions are defined w.r.t. quasars using Very Long Baseline Interferometry (Montenbruck and Gill, 2001). This makes the ICRF the most accurate reference frame currently available. The reference has known several updates, including better models and more reference sources. The most recent version, dating from 2009 (ICRF-2), will be used. Nevertheless, since the origin of this reference is not convenient for orbit propagation around the Earth, another reference frame should be defined.

For Earth applications, a more practical counterpart of the ICRF is the Geocentric Celestial Reference Frame or GCRF. This is an Earth-centered inertial frame (ECI), meaning that the center of mass of the Earth is the center of the reference frame and the xy-plane is coplanar with the equator. The positive direction of the x-axis is pointing towards the vernal equinox and the positive z-axis is directed to Earth's celestial north pole. The y-axis completes the right-handed reference frame. If no reference frame is mentioned in a superscript for further discussions, the state is expressed in the ECI frame, e.g. the position vector $\mathbf{r} = \mathbf{r}^{eci}$.

3.1.2. Non-inertial reference frames

When representing the targets and ground station, it is often more practical to define their position in an Earth centered Earth Fixed (ECEF) reference frame, since their coordinates will be fixed as the reference frame rotates in space. The positive x-axis now points to the Greenwich Meridian. The z-axis is parallel to the instantaneous Earth's rotation axis and the y-axis completes the right-handed reference frame. It is important to notice that the Earth's axis is not stationary in inertial space due to several perturbations such as nutation, precession and polar motion. Montenbruck and Gill (2001) explain the transformation of the ECEF frame to the ECI, whilst considering these perturbations.

Another non-inertial reference frame, especially useful for describing the scanning direction of the target, is the local tangent to the target location (see Figure 3.1). For this East, North, Up (ENU)

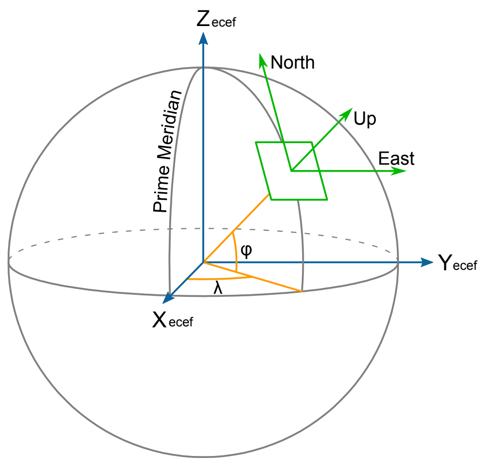


Figure 3.1: The ECEF and ENU reference frame.

reference frame, the positive x- and y-axes point to the local East and North direction respectively (Wakker, 2015, p. 250). The z-axis is positive in the upward direction. The ENU coordinates $\mathbf{r}^{enu}$ are converted to ECEF coordinates $\mathbf{r}^{ecef}$ using the following translation and rotation:

$$\mathbf{r}^{ecef} = \begin{bmatrix} -\sin\lambda & -\cos\lambda\sin\varphi & \cos\lambda\cos\varphi \\ \cos\lambda & -\sin\lambda\sin\varphi & \sin\lambda\cos\varphi \\ 0 & \cos\varphi & \sin\varphi \end{bmatrix} \mathbf{r}^{enu} + \mathbf{r}_c^{ecef} \tag{3.1}$$

where λ is the longitude, φ the latitude of the target point and $\mathbf{r}_c^{ecef}$ the ECEF coordinates of the ENU origin. The longitude and latitude can be expressed as ellipsoidal or spherical coordinates. Here, the WGS84 reference ellipsoid will be used as it is a good approximation of Earth's shape. The transformation from spherical to ellipsoidal coordinates is described in (Montenbruck and Gill, 2001, p. 187).

The last considered non-inertial reference frame is the Hill frame. When defining the rotation angles of the satellite around the axes of this orbital frame, it is easier to gain insight about the orientation of the satellite. The origin of this reference frame is fixed at the center of mass of the satellite and coordinates are expressed in the radial, tangential (along-track) and normal (cross-track) direction. In physical terms, the x-axis is the radial axis that points away from the center of the Earth, while the z-axis is fixed to the angular momentum vector of the orbit. Again, the y-axis completes the right-handed reference frame (Montenbruck and Gill, 2001). The transformation matrix $\mathbf{R}_{hill,eci}$ from the ECI to Hill frame is therefore written as

$$\mathbf{R}_{hill,eci} = \begin{bmatrix} \frac{\mathbf{r}_s^{eci}}{\left\|\mathbf{r}_s^{eci}\right\|} & \frac{\left(\mathbf{r}_s^{eci}\times\mathbf{v}_s^{eci}\right)\times\mathbf{r}_s^{eci}}{\left\|\left(\mathbf{r}_s^{eci}\times\mathbf{v}_s^{eci}\right)\times\mathbf{r}_s^{eci}\right\|} & \frac{\mathbf{r}_s^{eci}\times\mathbf{v}_s^{eci}}{\left\|\mathbf{r}_s^{eci}\times\mathbf{v}_s^{eci}\right\|} \end{bmatrix}^T \tag{3.2}$$

where $\mathbf{r}_s^{eci}$ and $\mathbf{v}_s^{eci}$ are the position and the velocity of the satellite respectively.

3.2. Orbital dynamics

The translational equations of motion of the satellite can be described by the famous equations of motion based on the second law of Newton and Newton's law of universal gravity. For this mission, it is assumed that the only perturbing force is caused by the J_2 gravitational potential, while the other perturbations can be neglected. This perturbation is caused by the flattening of the Earth at the poles. In addition to the central gravity, this perturbing force is known to have the largest magnitude at the considered altitude of 500 km (Montenbruck and Gill, 2001, p. 55). In an inertial reference frame, the motion of the satellite around the Earth can be described by Equation 3.3. Furthermore, the gravity perturbation can be expressed by the potential $U_p(\mathbf{r})$, since gravity is a conservative force (Wakker, 2015).

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} - \nabla U_p(\mathbf{r}) \tag{3.3}$$

In the equation above, **r** is the vector from the center of mass of the central body to the satellite. The gravitational parameter of the central body is denoted as μ and equals $3.98600441 \cdot 10^{14}$ m^3/s^2. Next, the perturbing potential is given by

$$U_p(\mathbf{r}) = U_{2,0}(\mathbf{r}) = \frac{\mu}{r^3} J_2 \left(\frac{R_\oplus}{r}\right)^2 \frac{3z^2 - r^2}{2} \tag{3.4}$$

where $J_2 = 1.08262668 \cdot 10^{-3}$ and the Earth radius $R_\oplus$ is 6378136 m. Finally, the perturbing acceleration $\mathbf{a}_p$ can be obtained by taking the negative gradient of the perturbing potential, i.e. the last term in Equation 3.3.

3.3. Attitude dynamics

The spacecraft attitude is controlled around the three body-fixed axes, in a configuration as shown in Figure 3.2. The actuators are put on the body axes of the satellite. The attitude of a satellite is expressed by quaternions, as is frequently used for attitude representation to avoid singularities (see Section 3.4.1). However, for physical interpretation, the roll (ϕ), the pitch (θ) and yaw (ψ) angle in the Hill frame (see Figure 2.3) will also be used. The satellite angular velocity vector is given by $\boldsymbol{\omega} = [\omega_1 \; \omega_2 \; \omega_3]^T$. The state vector **x** and control vector **u** can now be defined as

$$\mathbf{x} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^T \tag{3.5}$$

$$\mathbf{u} = \begin{bmatrix} T_{u1} & T_{u2} & T_{u3} \end{bmatrix}^T \tag{3.6}$$

where the torque is represented by $\mathbf{T}_u = [T_{u1} \; T_{u2} \; T_{u3}]^T$. Now, the attitude kinematic differential equation in terms of quaternions is given as

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{Q}(\boldsymbol{\omega})\mathbf{q} \tag{3.7}$$

Here, the 4×4 matrix $\mathbf{Q}(\boldsymbol{\omega})$ is defined by Wie (2008) as

$$\mathbf{Q}(\boldsymbol{\omega}) = \begin{bmatrix} -\boldsymbol{\omega}^\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} \tag{3.8}$$

and the skew-symmetric matrix $\boldsymbol{\omega}^\times$ is given by

$$\boldsymbol{\omega}^\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{3.9}$$

Furthermore, the dynamic differential equation can be expressed as (Wie, 2008):

$$\dot{\boldsymbol{\omega}} = \mathbf{I}_S^{-1}\left(-\boldsymbol{\omega}^\times \mathbf{I}_S \boldsymbol{\omega} + \mathbf{T}_u + \mathbf{T}_d\right) \tag{3.10}$$

The disturbance torque $\mathbf{T}_d$ is modelled to be zero since its magnitude can be neglected compared to the control torque $\mathbf{T}_u$. Finally, the inertia matrix $\mathbf{I}_S$ of the satellite is written as

$$\mathbf{I}_S = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{3.11}$$

3.4. Rotational kinematics

The kinematics treated in this section describe the orientation of a satellite that is in translational and rotational motion w.r.t. a target which is expressed in a fixed reference frame. The relative orientation will be expressed using quaternions, as introduced in Section 3.4.1. However, the orientation will first be described by direction cosine matrices, which is a more straightforward method to express orientations, as discussed in Section 3.4.2. The kinematic relationship which formulates the relative movement of the satellite and the ground targets is derived in Section 3.4.3.

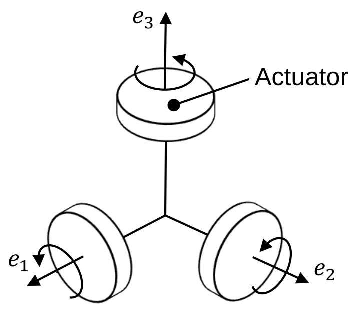


Figure 3.2: Configuration of the actuator axis w.r.t. the principal satellite axes.

3.4.1. Quaternions

Quaternions are essentially the four-dimensional extension of complex numbers. The quaternion number system can be used to facilitate the mathematical operations of three-dimensional orientations and avoids any singulary in the attitude dynamics. This concept is analogous to the way complex numbers can be used for rotations in two dimensions. However, instead of a single complex number, quaternions use three complex numbers which relate as follows (Wie, 2008):

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3.12}$$

Hamilton recognized that the imaginary numbers i, j and k can also represent the cartesian unit vectors $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$ (Wie, 2008). Using this property, a general quaternion q as used in this thesis can be written as:

$$q = x\hat{\mathbf{i}} + y\hat{\mathbf{j}} + z\hat{\mathbf{k}} + s \quad x, y, z, s \in \mathbb{R} \tag{3.13}$$

Note that the scalar s is taken as the fourth element of the quaternion. This is the Hamilton convention, but the scalar can also be the first element of the quaternion, as is the case for the JPL convention. For convenience, the quaternion q will further also be written as its vector ($\mathbf{q}_v$) and scalar (q_s) component:

$$q = \begin{bmatrix} \mathbf{q}_v \\ q_s \end{bmatrix} \tag{3.14}$$

This notation is used to define the conjugate q^* of the quaternion q

$$q^* = \begin{bmatrix} -\mathbf{q}_v \\ q_s \end{bmatrix} \tag{3.15}$$

Furthermore, another representation of the quaternion that will be used is based on Euler's rotational theorem, which states that any rotation or quaternion is defined by a rotational axis and a rotation angle:

$$q = \begin{bmatrix} \hat{\mathbf{e}} \sin \frac{\theta}{2} \\ \cos \frac{\theta}{2} \end{bmatrix} \tag{3.16}$$

Here, $\hat{\mathbf{e}}$ is a normalized rotational axis and θ the rotational angle.

The product of two quaternions is non-commutative and defined for two quaternions q and p as

$$q \otimes p = \begin{bmatrix} \mathbf{q}_v \times \mathbf{p}_v + p_s \mathbf{q}_v + q_s \mathbf{p}_v \\ q_s p_s - \mathbf{q}_v \cdot \mathbf{p}_v \end{bmatrix} \tag{3.17}$$

On the other hand, the sum of two quaternions q and p is found through elementwise addition:

$$q + p = \begin{bmatrix} \mathbf{q}_v + \mathbf{p}_v \\ q_s + p_s \end{bmatrix} \tag{3.18}$$

Another important property of quaternions is that q represents the same rotation as $-q$. This can cause numerical issues as discussed in Section 4.6.2. The equivalence of q and $-q$ is quickly proven by taking the quaternion definition from Equation 3.16. The negative alternative of q is:

$$-q = \begin{bmatrix} -\hat{\mathbf{e}} \sin \frac{\theta}{2} \\ -\cos \frac{\theta}{2} \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{e}} \sin(\pi - \frac{\theta}{2}) \\ \cos(\pi - \frac{\theta}{2}) \end{bmatrix} \tag{3.19}$$

Physically, this is a rotation over the negative axis, with an opposite angle compared to q, which is therefore the same rotation.

3.4.2. Direction cosine matrices

When considering a reference frame A and another reference frame B which are both represented by the set of orthogonal right-hand unit vectors **a** and **b** respectively, the set of basis vectors **a** can be expressed as

$$\mathbf{a} = \mathbf{R}_{A,B}\mathbf{b} \tag{3.20}$$

where $\mathbf{R}_{A,B}$ is the rotation or direction cosine matrix. This matrix can be used to define the orientation of a satellite. For example, the matrix $\mathbf{R}_{b,eci}$ can describe the satellite attitude w.r.t. inertial space. The advantage of using direction cosine matrices instead of quaternions becomes clear in Section 3.4.3, where the kinematics of the problem are formulated. While this is an impractical task in quaternions, it is more straightforward when using direction cosine matrices. However, after the kinematics are formulated in direction cosine matrices, the orientations are converted to quaternions, since these are more efficient for the further calculations. Given the direction cosine matrix, the corresponding quaternion can be obtained as

$$\begin{aligned} q_s &= \pm\frac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}} \\ \mathbf{q}_v &= \frac{1}{4q_s}\begin{bmatrix} R_{23} - R_{32} \\ R_{31} - R_{13} \\ R_{12} - R_{21} \end{bmatrix} \end{aligned} \tag{3.21}$$

Notice that whenever the trace of the direction cosine matrix becomes -1, a singularity occurs in this conversion. A method to handle this issue is described by Klumpp (2002).

3.4.3. Target tracking attitude profile determination

Due to the rotation of the Earth and the orbital velocity of the satellite, the targets and the satellite are continuously moving with respect to each other. Furthermore, the satellite scanner needs to precisely follow the scanning trajectory. Therefore, this section will establish the geometry and kinematic equations to find the necessary attitude profiles of the satellite to scan a single image request.

In order to find the attitude profiles for image acquisition, the attitude is expressed as the following direction cosine matrix $\mathbf{R}_{b,eci}(t)$, which defines the orientation of the satellite body w.r.t. inertial space:

$$\mathbf{R}_{b,eci}(t) = \begin{bmatrix} \hat{\mathbf{e}}_{b,x}^{eci}(t) & \hat{\mathbf{e}}_{b,y}^{eci}(t) & \hat{\mathbf{e}}_{b,z}^{eci}(t) \end{bmatrix} \tag{3.22}$$

where the vectors $\mathbf{e}_{b,i}^{eci}(t)$ express the orientation of each body axis as a function of the ECI standard basis vectors. Firstly, the satellite pointing direction $\mathbf{e}_{b,z}^{eci}(t)$ is found by considering the relative position vector $\mathbf{r}_{st/s}^{eci}$, which points from the satellite to the targeted position in the ECI frame:

$$\mathbf{r}_{st}^{eci}(t) = \mathbf{r}_{s}^{eci}(t) + \mathbf{r}_{st/s}^{eci}(t) \tag{3.23}$$

where $\mathbf{r}_{st}^{eci}(t)$ is the position of the scanning track and $\mathbf{r}_{s}^{eci}(t)$ the position of the satellite. As explained in Section 3.1.2, the z-axis is aligned with the camera pointing direction such that unit vector $\hat{\mathbf{e}}_{b,z}^{eci}(t)$ is obtained as

$$\hat{\mathbf{e}}_{b,z}^{eci}(t) = \frac{\mathbf{r}_{st/s}^{eci}(t)}{\left\|\mathbf{r}_{st/s}^{eci}(t)\right\|} \tag{3.24}$$

To ensure high image quality of the line scanner, the velocity vector of the scanning ground track should be perpendicular to the scanning line. This can be stated as follows:

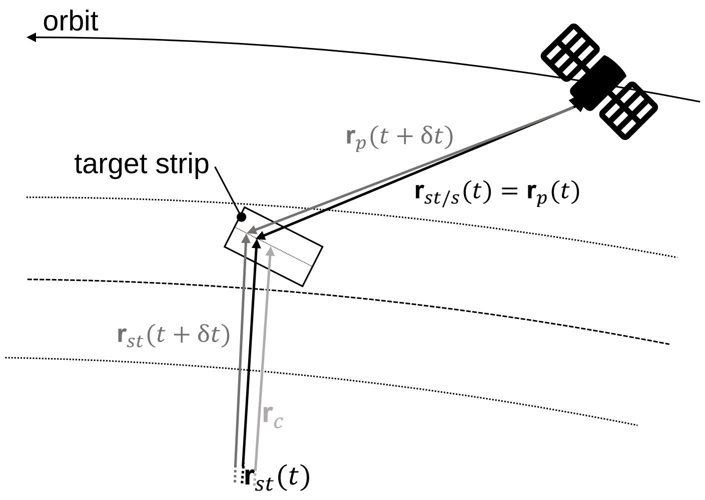

orbit
target strip
$\mathbf{r}_p(t+\delta t)$
$\mathbf{r}_{st/s}(t) = \mathbf{r}_p(t)$
$\mathbf{r}_{st}(t+\delta t)$
$\mathbf{r}_c$
$\mathbf{r}_{st}(t)$

Figure 3.3: Geometry of a single image acquisition after an infinitesimal rotation of the satellite.

$$\hat{\mathbf{e}}_{b,x}^{eci}(t) = \frac{\hat{\mathbf{e}}_{v,st}^{eci}(t) \times \hat{\mathbf{e}}_{b,z}^{eci}(t)}{\left\| \hat{\mathbf{e}}_{v,st}^{eci}(t) \times \hat{\mathbf{e}}_{b,z}^{eci}(t) \right\|} \tag{3.25}$$

Finally, the y-axis simply completes the right-handed satellite body orientation frame as

$$\hat{\mathbf{e}}_{b,y}^{eci}(t) = \hat{\mathbf{e}}_{b,z}^{eci}(t) \times \hat{\mathbf{e}}_{b,x}^{eci}(t) \tag{3.26}$$

The remainder of this section is dedicated to the derivation of the unit vector $\hat{\mathbf{e}}_{v,st}^{eci}$ as presented in Equation 3.25. For convenience, all vectors will be expressed in the ECEF frame, but the vectors can be transformed to the ECI frame as discussed in Section 3.1. A single image strip is defined by its center point $\mathbf{r}_c^{ecef}$ and a scanning direction, which is taken as the angle α_s w.r.t. the local North. The center point of the image is defined by the latitude φ, longitude λ and altitude of the target on the WGS84 ellipsoid. The altitude h of the target is assumed to be constant during the scan, which is a valid approximation for real applications. In the case of scanning a steep terrain on the Earth surface, this assumption should be reconsidered. Using Equation 3.1, the unit vectors to the local East and the local North of the image center point can be expressed in the ECEF frame as:

$$\hat{\mathbf{e}}_E^{ecef} = \begin{bmatrix} -\sin\lambda \\ \cos\lambda \\ 0 \end{bmatrix} \tag{3.27}$$

and

$$\hat{\mathbf{e}}_N^{ecef} = \begin{bmatrix} -\cos\lambda\sin\varphi \\ -\sin\lambda\sin\varphi \\ \cos\varphi \end{bmatrix} \tag{3.28}$$

The scanning azimuth angle α_s, which determines the image acquisition direction, can now be expressed as follows:

$$\hat{\mathbf{e}}_{sd}^{ecef} = \hat{\mathbf{e}}_N^{ecef} \cos\alpha_s + \hat{\mathbf{e}}_E^{ecef} \sin\alpha_s \tag{3.29}$$

Also, perpendicular to the scanning direction and position vector of the mid-point of the target, the following unit vector can be determined:

$$\hat{\mathbf{e}}_{rot}^{ecef} = \frac{\hat{\mathbf{e}}_{sd}^{ecef} \times \mathbf{r}_c^{ecef}}{\left\| \hat{\mathbf{e}}_{sd}^{ecef} \times \mathbf{r}_c^{ecef} \right\|} \tag{3.30}$$

Using the above-defined unit vector and assuming a spherical Earth, the unit vector of the scanning pointing velocity during scanning at time t can found to be:

$$\hat{\mathbf{e}}^{ecef}_{v,st,circ}(t) = \frac{\mathbf{r}^{ecef}_{st}(t) \times \hat{\mathbf{e}}^{ecef}_{rot}}{\left\| \mathbf{r}^{ecef}_{st}(t) \times \hat{\mathbf{e}}^{ecef}_{rot} \right\|} \tag{3.31}$$

Here, the scanning track is assumed to be on a great circle. However this can be corrected for the WGS84 ellipsoid by converting the unit vector in the ENU frame and defining $\hat{\mathbf{e}}^{enu}_{v,gt}(t)$ as:

$$\hat{\mathbf{e}}^{enu}_{v,scan}(t) = \frac{1}{\sqrt{\left(\hat{\mathbf{e}}^{enu}_{v,st,circ,x}(t)\right)^2 + \left(\hat{\mathbf{e}}^{enu}_{v,st,circ,y}(t)\right)^2}} \begin{bmatrix} \hat{\mathbf{e}}^{enu}_{v,st,circ,x}(t) \\ \hat{\mathbf{e}}^{enu}_{v,st,circ,y}(t) \\ 0 \end{bmatrix} \tag{3.32}$$

The unit vector perpendicular to the scanning velocity direction and relative position vector $\mathbf{r}^{ecef}_{t/s}(t)$ is given by:

$$\hat{\mathbf{e}}^{ecef}_{\perp}(t) = \frac{\mathbf{r}^{ecef}_{st/s}(t) \times \hat{\mathbf{e}}^{ecef}_{v,st}(t)}{\left\| \mathbf{r}^{ecef}_{st/s}(t) \times \hat{\mathbf{e}}^{ecef}_{v,st}(t) \right\|} \tag{3.33}$$

A rotation of the satellite about this unit vector $\hat{\mathbf{e}}^{ecef}_{\perp}(t)$ over an angle β can be found using Rodrigues' rotation formula (Wie, 2008, p. 332) and gives the following rotation matrix:

$$\mathbf{A}(t) = \cos\beta \mathbf{I} + (1 - \cos\beta)\hat{\mathbf{e}}^{ecef}_{\perp}(t)\left(\hat{\mathbf{e}}^{ecef}_{\perp}(t)\right)^T + \sin\beta \mathbf{E}(t) \tag{3.34}$$

where $\mathbf{E}(t)$ is a skew-symmetric matrix defined as

$$\mathbf{E}(t) = \begin{bmatrix} 0 & -\hat{\mathbf{e}}^{ecef}_{\perp,z} & \hat{\mathbf{e}}^{ecef}_{\perp,y} \\ \hat{\mathbf{e}}^{ecef}_{\perp,z} & 0 & -\hat{\mathbf{e}}^{ecef}_{\perp,x} \\ -\hat{\mathbf{e}}^{ecef}_{\perp,y} & \hat{\mathbf{e}}^{ecef}_{\perp,x} & 0 \end{bmatrix} \tag{3.35}$$

Furthermore, the angle β is given by:

$$\beta = l\theta\delta t \tag{3.36}$$

where l is the line rate, θ the angular resolution and δt an infinitesimal small time step.

Since the time step and therefore angle β are infinitesimal small, Equation 3.34 can be simplified by using the second-order Taylor expansion around zero for the following trigoniometric functions:

$$\cos\beta = 1 - \frac{\beta^2}{2} + \mathcal{O}(\beta^3), \; \sin\beta = \beta + \mathcal{O}(\beta^3) \tag{3.37}$$

Because β is infinitesimally small, the second-order terms can be neglected compared to the first-order terms and $\mathbf{A}(t)$ simplifies to:

$$\mathbf{A}(t) = \mathbf{I} + \beta\mathbf{E}(t) \tag{3.38}$$

This infinitesimal rotation matrix is used now to find the pointing vector $\mathbf{r}^{ecef}_{p}$ at time $t + \delta t$:

$$\hat{\mathbf{r}}^{ecef}_{p}(t + \delta t) = \mathbf{A}\hat{\mathbf{r}}^{ecef}_{st/s}(t) \tag{3.39}$$

Even more, the infinitesimal difference $\delta\mathbf{r}^{ecef}_{t/s}(t)$ can be described using the outcome from Equation 3.38 as:

$$\delta\hat{\mathbf{r}}^{ecef}_{p}(t) = \hat{\mathbf{r}}^{ecef}_{p}(t + \delta t) - \hat{\mathbf{r}}^{ecef}_{p}(t) = \beta\mathbf{E}(t)\hat{\mathbf{r}}^{ecef}_{st/s}(t) \tag{3.40}$$

The above result can also be written as the following cross product

$$\delta\hat{\mathbf{r}}^{ecef}_{p}(t) = \hat{\mathbf{e}}^{ecef}_{\perp}(t) \times \hat{\mathbf{r}}^{ecef}_{st/s}(t)\beta \tag{3.41}$$

Finally, after using the definition of the infinitesimal angle β the result is:

$$\dot{\hat{\mathbf{r}}}_p^{ecef}(t) = \hat{\mathbf{e}}_\perp^{ecef}(t) \times \hat{\mathbf{r}}_{st/s}^{ecef}(t) l\theta \tag{3.42}$$

The derivative of the satellite pointing direction can now be used to determine the derivative of the scanning track position. Given the pointing vector and satellite position, the ground track is calculated as the intersection of the satellite pointing direction vector and the WGS84 reference ellipsoid. It can be shown that the solution of the intersection takes the following parametric form after an infinitesimal rotation (Qu et al., 2006, p. 58):

$$\mathbf{r}_{st}^{ecef}(t) = \mathbf{r}_s^{ecef} + d(t)\hat{\mathbf{r}}_p^{ecef}(t) \tag{3.43}$$

Here, the satellite position is constant, since a rotation and not a translation of the satellite is considered. Also, the distance d from the satellite to the intersection is the solution of a quadratic equation:

$$d(t) = -\frac{b + \sqrt{b^2 - 4ac}}{2a} \tag{3.44}$$

and the coefficients a, b and c are:

$$a = (R_p + h)^2(\mathbf{r}_{p,x}^2 + \mathbf{r}_{p,y}^2) + R_e^2\mathbf{r}_{p,z}^2 \tag{3.45}$$

$$b = 2\left((R_p + h)^2(\mathbf{r}_{s,x}\mathbf{r}_{p,x} + \mathbf{r}_{s,y}\mathbf{r}_{p,y}) + (R_e + h)^2\mathbf{r}_{s,z}\mathbf{r}_{p,z}\right) \tag{3.46}$$

$$c = (R_p + h)^2(-(R_e + h)^2 + \mathbf{r}_{s,x}^2 + \mathbf{r}_{s,y}^2) + (R_e + h)^2\mathbf{r}_{s,z}^2 \tag{3.47}$$

The derivative of the scanning track position is now

$$\dot{\mathbf{r}}_{st}^{ecef}(t) = d(t)\dot{\hat{\mathbf{r}}}_p^{ecef}(t) + \dot{d}(t)\hat{\mathbf{r}}_p^{ecef}(t) \tag{3.48}$$

Here it is essential to realize that the satellite velocity shall not influence the scanning track velocity, since this velocity vector is only formed by the rotational motion of the image scanner. This is also demonstrated in Equation 3.43, where the satellite position is constant. This principle is counter-intuitive at first, but can be faster understood by considering that the satellite scanner should be scanning the image, and the image is not 'passively' scanned by using the velocity of the satellite. The relative velocity of the satellite and the target is already included in the terms $\dot{\hat{\mathbf{r}}}_p^{ecef}(t)$ and $\dot{d}$. Furthermore, the derivative of the distance $d(t)$ w.r.t time becomes:

$$\dot{d}(t) = -\frac{1}{2a}\left(2\dot{a}d + \dot{b} + \frac{2b\dot{b} - 4c\dot{a}}{\sqrt{b^2 - 4ac}}\right) \tag{3.49}$$

where the coefficient derivatives $\dot{a}$ and $\dot{b}$ are:

$$\dot{a} = 2\left((R_p + h)^2(\mathbf{r}_{p,x}^{ecef}\dot{\mathbf{r}}_{p,x}^{ecef} + \mathbf{r}_{p,y}^{ecef}\dot{\mathbf{r}}_{p,y}^{ecef}) + (R_e + h)^2\mathbf{r}_{p,z}^{ecef}\dot{\mathbf{r}}_{p,z}^{ecef}\right) \tag{3.50}$$

$$\dot{b} = 2\left((R_p + h)^2(\mathbf{r}_{s,x}^{ecef}\dot{\mathbf{r}}_{p,x}^{ecef} + \mathbf{r}_{s,y}^{ecef}\dot{\mathbf{r}}_{p,y}^{ecef}) + (R_e + h)^2\left(\dot{\mathbf{r}}_{s,z}^{ecef}\mathbf{r}_{p,z}^{ecef} + \mathbf{r}_{s,z}^{ecef}\dot{\mathbf{r}}_{p,z}^{ecef}\right)\right) \tag{3.51}$$

The result of the kinematics derivation, Equation 3.48, is an Ordinary Differential Equation (ODE) which gives the change of the scanning track over time. Since no analytical solution was found and the scanning track depends on the satellite position, the ODE is integrated along the orbital dynamic equations as introduced in Section 3.4. However, the satellite state is required in the ECEF frame for this integration, while the orbit is propagated in the inertial ECI frame. The dynamic transition between two reference frames is modelled as:

$$\dot{\mathbf{r}}_s^{ecef} = \dot{\mathbf{r}}_s^{eci} - \omega_{ecef/eci} \times \mathbf{r}_s^{ecef} \tag{3.52}$$

The relative rotation of the non-inertial ECEF frame can be compensated by adding the following fictitious forces to the inertial acceleration:

$$\ddot{\mathbf{r}}_s^{ecef} = \ddot{\mathbf{r}}_s^{eci} - \boldsymbol{\omega}_{ecef/eci} \times \dot{\mathbf{r}}_s^{ecef} - \boldsymbol{\omega}_{ecef/eci} \times \left(\boldsymbol{\omega}_{ecef/eci} \times \mathbf{r}_s^{ecef}\right) \tag{3.53}$$

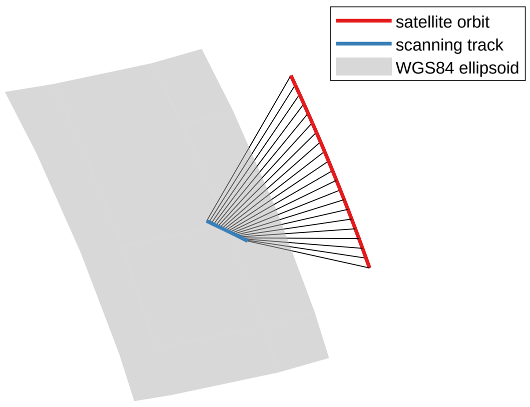


Figure 3.4: Example of the scanning track for a satellite in a polar orbit and a scanning direction of 60 degrees w.r.t the local North.

where $\boldsymbol{\omega}_{eci/ecef} = [0, 0, \omega_e]^T$ is the relative rotation rate of the reference frames. Here, the angular Earth rotation rate is $\omega_e = 7.2921159 \cdot 10^{-5}$ rad/s and assumed to be constant. This rotation rate could also be obtained from a data file of Earth Orientation Parameters (EOP) for higher accuracy (Montenbruck and Gill, 2001). As explained, this is an approximation since perturbations as precession and nutation are here not taken into account. However, since the image acquisition duration is relatively short, these effects are negligible. Finally, an example of the integrated satellite orbit and scanning track is shown in Figure 3.4.

3.4.4. Target tracking angular rate and torque profiles

Section 3.4.3 gives the approach to obtain the attitude profile during tracking of a target. Similarly, the satellite angular rate and torque profiles are needed to accurately scan the image. Instead of making a full kinematic derivation as in Section 3.4.3, the angular rate and torque can be found given the attitude profile expressed in quaternions. First, the angular velocity vector is augmented such that it is written as a pure quaternion $\omega = [\boldsymbol{\omega}, 0]$. Using this notation, the kinematic relation from Equation 3.7 can also be written as:

$$\omega(t) = 2\dot{q}(t) \otimes q(t)^* \tag{3.54}$$

Now, simply taking the derivative of the angular velocity gives the angular acceleration as a function of the quaternions:

$$\dot{\omega}(t) = 2\left(\ddot{q}(t) \otimes q(t)^* + \dot{q}(t) \otimes \dot{q}(t)^*\right) \tag{3.55}$$

Here, the vector component of the term $\dot{q} \otimes \dot{q}^*$ is zero, due to a multiplication property of conjugate quaternions. Finally, the torque can be derived from the attitude dynamic equations:

$$\mathbf{T}_u = \mathbf{I}_S \dot{\boldsymbol{\omega}} + \boldsymbol{\omega}^{\times} \mathbf{I}_S \boldsymbol{\omega} \tag{3.56}$$

In the above equations, the derivatives $\dot{q}(t)$ and $\ddot{q}(t)$ are obtained using a first- and second-order central finite difference of the quaternion profile.

4 Optimal attitude guidance profiles

This chapter gives an overview of the necessary optimal control theory and its application to obtain the optimal attitude guidance profiles, such that the satellite is reoriented between targets in the minimum time possible. Firstly, the background of non-linear programming, necessary for solving the optimal control problem, is explained in Section 4.1. Next, a brief introduction to the required optimal control theory is given in Section 4.2. Typical transcription methods, necessary for the implementation are discussed in Section 4.3. In Section 4.4, the aforementioned theory is used to define the optimal control problem. The implementation details and robustness improvements are documented in Sections 4.5 and 4.6 respectively. Also, the implementation of the selected transcription method is discussed in Section 4.7. Finally, Section 4.8 gives the resulting optimal attitude guidance for some example cases.

4.1. Non-linear programming fundamentals

The non-linear programming (NLP) problem aims to optimize an objective function with a finite number of variables, subjected to an array of constraints. As will be shown, the optimal control problem (OCP) is often discretized, such that it can be expressed as an NLP problem. This discretization is called transcription. In short, NLP adopts a multi-dimensional variant of Newton's secant method to determine a local minimum. A short description of this fundamental optimization technique is given in Section 4.1.1, since the choice of the NLP algorithm will determine the efficiency and robustness of solving the OCP. This technical discussion is based on (Betts, 2009, Ch.1). Afterwards, Sections 4.1.2 and 4.1.3 also discuss two popular NLP algorithms.

4.1.1. Newton's method

Mathematically, the problem is to find the optimal vector $\mathbf{y}^\star$ for the problem

$$\begin{aligned} &\min_{\mathbf{y}\in\mathbb{R}^n} J(\mathbf{y}) \\ \text{subject to: } &\mathbf{a}(\mathbf{y}) = \mathbf{0}, \quad \mathbf{a} \in \mathbb{R}^m \\ &\mathbf{b}(\mathbf{y}) \geq \mathbf{0}, \quad \mathbf{b} \in \mathbb{R}^l \end{aligned} \tag{4.1}$$

Here, J denotes the objective function, **a** is an equality constraint vector and **b** is an inequality constraint vector. Newton's method solves this problem by making use of gradient techniques. These methods approximate the objective function and constraints by polynomial expansions. Firstly, consider the problem without the equality and inequality constraints. Accordingly, the objective function $J(\mathbf{y})$ can be approximated around **y** using a quadratic function. More specifically, a second-order Taylor expansion of the objective function around point $\bar{\mathbf{y}}$ is used:

$$J(\bar{\mathbf{y}}) = J(\mathbf{y}) + \mathbf{g}^T(\mathbf{y})(\bar{\mathbf{y}} - \mathbf{y}) + \frac{1}{2}(\bar{\mathbf{y}} - \mathbf{y})^T\mathbf{H}(\mathbf{y})(\bar{\mathbf{y}} - \mathbf{y}) \tag{4.2}$$

Here, **g** is the gradient vector

$$\mathbf{g}(\mathbf{y}) \equiv \nabla_{\mathbf{y}} J = \begin{bmatrix} \frac{\partial J}{\partial y_1} \\ \frac{\partial J}{\partial y_2} \\ \vdots \\ \frac{\partial J}{\partial y_n} \end{bmatrix} \tag{4.3}$$

and **H** is the Hessian matrix

$$\mathbf{H}(\mathbf{y}) \equiv \nabla^2_{\mathbf{yy}} J = \begin{bmatrix} \frac{\partial^2 J}{\partial y_1^2} & \frac{\partial^2 J}{\partial y_1 y_2} & \cdots & \frac{\partial^2 J}{\partial y_1 y_n} \\ \frac{\partial^2 J}{\partial y_2 y_1} & \frac{\partial^2 J}{\partial y_2^2} & \cdots & \frac{\partial^2 J}{\partial y_2 y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial y_n y_1} & \frac{\partial^2 J}{\partial y_n y_2} & \cdots & \frac{\partial^2 J}{\partial y_n^2} \end{bmatrix} \tag{4.4}$$

Now, assume the expansion is taken around the minimum point $\mathbf{y}^\star$, the solution to the problem. In that case it should hold that $J(\bar{\mathbf{y}}) > J(\mathbf{y}^\star)$. In other words, the slope in all directions needs to be zero when evaluated at $\mathbf{y}^\star$, such that the following condition applies:

$$\mathbf{g}(\mathbf{y}^\star) = \mathbf{0} \tag{4.5}$$

Still, this gradient condition can allow for a local minimum, a local maximum or a saddle point. Therefore, a second condition is imposed on the second derivative. Recalling the Taylor expansion, and substituting the condition for optimality $J(\mathbf{y}^\star) < J(\bar{\mathbf{y}})$ results in

$$\begin{aligned} J(\mathbf{y}^\star) &< J(\mathbf{y}^\star) + \frac{1}{2}(\bar{\mathbf{y}} - \mathbf{y}^\star)^T \mathbf{H}(\mathbf{y}^\star)(\bar{\mathbf{y}} - \mathbf{y}^\star) \\ \Rightarrow \mathbf{0} &< (\bar{\mathbf{y}} - \mathbf{y}^\star)^T \mathbf{H}(\mathbf{y}^\star)(\bar{\mathbf{y}} - \mathbf{y}^\star) \end{aligned} \tag{4.6}$$

These conditions can now be applied to solve the unconstrained problem. This is realized by taking an initial guess and moving along the search direction towards the local minimum in an iterative manner. For a given gradient and Hessian at iteration i, the objective function at the next iteration $i + 1$ is found using the second-order expansions around the current point:

$$J(\mathbf{y}_{i+1}) = J(\mathbf{y}_i) + \mathbf{g}^T(\mathbf{y}_i)\Delta\mathbf{y}_i + \frac{1}{2}(\Delta\mathbf{y}_i)^T \mathbf{H}(\mathbf{y}_i)(\Delta\mathbf{y}_i) \tag{4.7}$$

It can be shown that by imposing the aforementioned conditions for optimality, the search direction becomes (Betts, 2009, p. 10):

$$\mathbf{y}_{i+1} - \mathbf{y}_i = \Delta\mathbf{y}_i = -\mathbf{H}^{-1}(\mathbf{y}_i)\mathbf{g}(\mathbf{y}_i) \tag{4.8}$$

As a next step, the problem is complicated by introducing the equality and inequality constraints. Accordingly, a new augmented cost function needs to be defined that includes the constraints:

$$\mathcal{L}(\mathbf{y}, \boldsymbol{\lambda}, \boldsymbol{\mu}_a) = J(\mathbf{y}) - \boldsymbol{\lambda}^T \mathbf{a}(\mathbf{y}) - \boldsymbol{\mu}_a^T \mathbf{b}(\mathbf{y}) \tag{4.9}$$

Here, the Lagrangian is $\mathcal{L}$, $\boldsymbol{\lambda}$ is the Lagrange multiplier for the equality constraints and $\boldsymbol{\mu}_a$ is the Lagrange multiplier for the active inequality constraints. The inequality constraint is characterized as active when it has become an equality constraint or $b_i(\mathbf{y}^\star) = 0$. On the other hand, for the inactive set of constraints it holds that $b_i(\mathbf{y}^\star) > 0$. A mechanism to detect the active set of constraints is given in (Betts, 2009, p. 19). The condition of optimality is now again reduced to calculating the gradient of the (augmented) objective function:

$$\begin{aligned} \nabla_{\mathbf{y}} \mathcal{L}(\mathbf{y}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}_a^\star) &= \mathbf{0} \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{y}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}_a^\star) &= \mathbf{0} \\ \nabla_{\boldsymbol{\mu}} \mathcal{L}(\mathbf{y}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}_a^\star) &= \mathbf{0} \end{aligned} \tag{4.10}$$

This is the set of first-order conditions similar as for Equation 4.5, but including the equality and in-equality constraints. These equations are also known as the Karush Kuhn-Tucker (KKT) conditions, and these gradients can be determined to be

$$\begin{aligned} \mathbf{g}(\mathbf{y}) - \mathbf{G}_{\boldsymbol{\lambda}}^T(\mathbf{y})\boldsymbol{\lambda} - \mathbf{G}_{\boldsymbol{\mu}_a}^T(\mathbf{y})\boldsymbol{\mu}_a &= \mathbf{0} \\ \mathbf{a}(\mathbf{y}) &= \mathbf{0} \\ \mathbf{b}_a(\mathbf{y}) &= \mathbf{0} \end{aligned} \tag{4.11}$$

Here, the Jacobian matrices of the constraints are

$$\mathbf{G}_{\boldsymbol{\lambda}}(\mathbf{y}) \equiv \frac{\partial \mathbf{a}}{\partial \mathbf{y}}, \quad \mathbf{G}_{\boldsymbol{\mu}_a}(\mathbf{y}) \equiv \frac{\partial \mathbf{b}_a}{\partial \mathbf{y}} \tag{4.12}$$

Again, these first-order KKT conditions are insufficient to guarantee a local minimum. This time the Hessian of the Lagrangian can be calculated as

$$\mathbf{H}_{\mathcal{L}} = \nabla^2_{\mathbf{yy}}\mathcal{L} = \nabla^2_{\mathbf{yy}}J - \sum_{i=1}^{m} \lambda_i \nabla^2_{\mathbf{yy}} a_i - \sum_{i=1}^{l} \mu_{a,i} \nabla^2_{\mathbf{yy}} b_i \tag{4.13}$$

Finally, the second condition for a solution to be a minimum is found using Equation 4.6, while only considering the active constraints. Now, similarly as for the unconstrained optimization, the search direction $\Delta\mathbf{y}$ needs to be found to converge to the local minimum. Unfortunately, this is not a trivial problem, but can be solved through the methods presented in Sections 4.1.2 and 4.1.3.

4.1.2. Sequential quadratic programming

The SQP method basically solves a sequence of sub-problems, for which the objective function is reformulated as a quadratic function (Betts, 2009, p. 18). For each iteration of the solver, the NLP problem from Equation 4.1 is reformulated as the following quadratic programming (QP) problem around the current point:

$$\begin{aligned} &\min\ \mathbf{g}^T\mathbf{y} + \frac{1}{2}\mathbf{y}^T\mathbf{H}\mathbf{y} \\ &\text{subject to: } \mathbf{A}\mathbf{y} = \mathbf{a} \\ &\qquad\qquad\quad \mathbf{B}\mathbf{y} \geq \mathbf{b} \end{aligned} \tag{4.14}$$

Comparing to the original NLP problem statement, the QP sub-problem has a quadratic objective function and linearized constraints. If no constraints are present, this could be solved in a single iteration using Newtow's method. Unfortunately the presence of inequality constraints make this infeasible. The quadratic sub-problem therefore needs an active set method, which treats the active set of constraints as equalities and neglects the inactive ones. A more detailed description of the active set method is given in (Betts, 2009, p. 18). The QP sub-problem defines now a search direction for the solution. The step size in that direction is typically determined using a line search method or interior point method. Finally, the decision variables are updated and a quadratic sub-problem is initiated for this new point. The process is repeated until the convergence criteria are reached.

4.1.3. Interior point methods

The Interior Point (IP) method converts the constrained optimization problem into an unconstrained one. The problem constraints are instead handled by applying a penalty to the objective function when constraints are violated. As a consequence, the solution will avoid moving to the *barriers* or constraints. The general formulation of the NLP problem in Equation 4.1 is now

$$\begin{aligned} &\min_{\mathbf{y}\in\mathbb{R}^n} J_\mu(\mathbf{y}) = J(\mathbf{y}) - \mu \sum_{i}^{n} \ln(b_i(\mathbf{y})) \\ &\text{subject to: } \mathbf{a}(\mathbf{y}) = \mathbf{0},\ \mathbf{a} \in \mathbb{R}^m \end{aligned} \tag{4.15}$$

Here, the logarithmic barrier term replaces the inequality constraints and the barrier parameter μ is larger than zero. The barrier term becomes large when the constraints become active and are therefore close to zero. In this way, the algorithm will tend to move away from these barriers. It is clear that the barrier parameter should be sufficiently small, because the optimal solution can also be at the boundary itself. The IP algorithm tackles this by solving the sub-problem of Equation 4.15 multiple times, such that the parameter μ goes to zero. It can be proven that under standard conditions, the optimal solution from the original problem is reached if μ reaches zero in the limit. It should also be noted that different barrier objective functions $J_\mu(\mathbf{y})$ can be chosen. Here, the specific approach of the solver IPOPT is presented (Wächter and Biegler, 2006, p. 7-10). A more detailed description of various other IP barrier methods can be found in (Betts, 2009, p. 30-36).

4.2. Optimal control theory

Generally two different types of optimal control are distinguished: indirect and direct solutions which are described in Sections 4.2.2 and 4.2.3 respectively. Some introductory elements about optimal control are first given in Section 4.2.1.

4.2.1. Optimal control preliminaries

Take a dynamic system which is governed by an ordinary differential equation (ODE) with the form of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$, where the independent variable t is set to be time. In general, for optimal control problems, a performance index J should be minimized, by steering the state variables $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ using the control variables $\mathbf{u}(t) \in \mathbb{R}^{n_u}$. This can be mathematically summarized in Equation 4.16. Note that the final time t_f can be either free or fixed.

$$\min_{\mathbf{u}(t)} J = \phi\left(\mathbf{x}(t_f), t_f\right) + \int_{t_0}^{t_f} L\left(\mathbf{x}(t), \mathbf{u}(t), t\right) dt \tag{4.16}$$

The above optimal control performance index is written in the Bolza form, where $t \in [t_0, t_f]$ is the time interval, ϕ the terminal cost and L the integral term of the objective function. If the terminal cost is zero, the problem is reduced to a Lagrange problem. This is commonly the case for optimal attitude control problems (Wie, 2008).

In addition to the constraint coming from the dynamics of the systems, also several other physical constraints can be imposed. Firstly, the boundary conditions $\boldsymbol{\psi}$ are imposed at the initial and final time of the interval. On the other hand, the path constraints $\mathbf{C}$ can be imposed over the entire time interval. In mathematical terms:

$$\begin{aligned} \boldsymbol{\psi}(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) &= \mathbf{0} \\ \mathbf{C}(\mathbf{x}(t), \mathbf{u}(t), t) &\leq \mathbf{0} \end{aligned} \tag{4.17}$$

Including these constraints, the optimal control problem is now fully defined. Now, the continuous function $\mathbf{u}^\star(t)$ should be found for which Equation 4.16 is minimized. Analytical solutions exist for some exception cases, however for attitude control no such solution is known when considering multiple constraints (Wie, 2008). Therefore, numerical optimization methods are commonly adopted. Since the control variable vector $\mathbf{u}(t)$ is a function of time, the optimal problem can also be regarded as an infinite-dimensional extension of the NLP problem. Obviously, to practically solve the optimal control problem, a finite set of NLP variables and constraints is required. Therefore, the infinite-dimensional problem is discretized into a finite-dimensional approximation, which is solved by the numerical optimization method. The discretization is referred to as the transcription of the OCP and it should be carefully chosen depending on the problem, since it will greatly determine the performance of the optimization. An overview of several common techniques is presented in Section 4.3. A more detailed technical description of these methods can be found in (Betts, 2009, Ch. 3-4).

Furthermore, a distinction is made between two types of numerical optimization: indirect and direct optimization, which are described in Sections 4.2.2 and 4.2.3 respectively. All transcription methods can be applied to both methods as demonstrated by Betts (1998). In Section 4.3 only details about direct transcription methods will be given as it is the preferred method for this work.

4.2.2. Indirect methods

This is the classical method of optimal control, which is based on Pontryagin's maximum principle and calculus of variation fundamentals. Employing this technique, the goal is to describe the optimal control problem as a two-point boundary value problem (TPBVP). This can be achieved by establishing certain necessary conditions that the profile needs to satisfy to be an optimum (Conway, 2010).

The first step is to define the augmented performance index $\bar{J}$, which is formed by adjoining the system of differential equations to J such that the dynamics are now contained within $\bar{J}$:

$$\bar{J} = \phi(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} \left[L\left[\mathbf{x}(t), \mathbf{u}(t), t\right] + \boldsymbol{\lambda}^T(t) \left\{ \mathbf{f}\left[\mathbf{x}(t), \mathbf{u}(t), t\right] - \dot{\mathbf{x}} \right\} \right] dt \tag{4.18}$$

Here, the Lagrangian multipliers $\boldsymbol{\lambda}$ are also referred to as the costate variables for the continuous constraints. The necessary condition is to set the first-order variation $\delta\bar{J} = 0$ for any $\delta\mathbf{u}(t)$. Now, define the scalar function $\mathcal{H}$, which is the Hamiltonian:

$$\mathcal{H} = L\left[\mathbf{x}(t), \mathbf{u}(t), t\right] + \boldsymbol{\lambda}^T(t)\mathbf{f}\left[\mathbf{x}(t), \mathbf{u}(t), t\right] \tag{4.19}$$

Using these definition, the necessary condition for the control profile $\mathbf{u}(t)$ can be derived to be the Euler-Lagrange equations (Bryson and Ho, 1975, Ch. 2):

$$\dot{\boldsymbol{\lambda}} = -\left[\frac{\partial\mathcal{H}}{\partial\mathbf{x}}\right]^T \tag{4.20}$$

$$\mathbf{0} = \left[\frac{\partial\mathcal{H}}{\partial\mathbf{u}}\right]^T \tag{4.21}$$

$$\boldsymbol{\lambda}^T(t_f) = \frac{\partial\phi}{\partial\mathbf{x}(t_f)} \tag{4.22}$$

The ODE's of the costate variables as given in Equation 4.20 are called the adjoint equations. The control equations and transversality conditions are Equations 4.21 and 4.22 respectively. These equations together specify the TPBVP, which unfortunately can only be solved using numerical methods for optimal attitude control problems. Nevertheless, the indirect method can still provide valuable information on the quality of the solution. A relation that is often used for that purpose is the Legendre-Clebsch condition (Betts, 1998):

$$\frac{\partial^2\mathcal{H}}{\partial\mathbf{u}^2} \geq 0 \tag{4.23}$$

However, this can only be used as indicator for optimality when no constraints are imposed on $\mathbf{u}$.

4.2.3. Direct methods

As opposed to the indirect methods, the control function is parametrized over one or more arcs for the direct method. The adjoint and transversality equations are also no longer necessary. Instead, the optimal control problem is directly transcribed into a parameter optimization problem. The non-augmented performance index J is iteratively reduced by using a grid at different times, such that the control is given through interpolation, or by basis functions. Several popular functions that are used to represent the control are Fourier series, Legendre, Lagrange and Chebyshev polynomials.

The initial guess provided to the direct method is less stringent compared to the indirect methods. Since the optimal control problem needs to be solved many times and not just once, it is of high importance the approach is robust. Therefore the direct method will be used in this work. However, the direct method does not exactly provide the optimal solution itself for most cases because of the discretisation. Therefore, the solutions using indirect method often are more accurate. Nevertheless, with the current computational capabilities and transcription methods, the difference is often very small (Conway, 2010).

4.3. Direct transcription methods

As explained in Section 4.2, the optimal control method has to be transcribed so that it can be solved as an NLP problem. These methods could be used for both the indirect and direct method, but specifically the implementation of the direct method is discussed, as this is the preferred method.

4.3.1. Direct shooting

The direct shooting method discretizes the continuous OCP by choosing the NLP variables as the initial state, final state and a set of control parameters. The NLP variables are contained in the vector

$$\mathbf{y} = \begin{bmatrix} \mathbf{x}(t_0) \\ t_0 \\ \mathbf{x}(t_f) \\ t_f \\ \mathbf{u}_{n_c} \end{bmatrix} \tag{4.24}$$

where $\mathbf{u}_{n_c}$ consists of n_c control components. These control components form a discrete representation of a continuous control function and can be represented by discrete points or coefficients of an interpolation function. For a given initial and final state, which depend on time, the equality constraint vector can be written as

$$\mathbf{a}(\mathbf{y}) = \begin{bmatrix} \boldsymbol{\psi}(\mathbf{x}(t_0), t_0, \mathbf{u}_{n_c}) \\ \boldsymbol{\psi}(\mathbf{x}(t_f), t_f, \mathbf{u}_{n_c}) \end{bmatrix} \tag{4.25}$$

where the notation from Equation 4.1 is used. In general, the single shooting method consists of the following steps:

1. Make a guess for an initial solution.
2. Propagate the differential equations (shoot), resulting in the final state $\hat{\mathbf{x}}(t_f)$ and evaluate the error with the desired final state $\mathbf{x}(t_f)$.
3. Use an NLP algorithm to correct the initial control variables, final time or initial state such that the objective function is minimized and the constraints are satisfied. Repeat steps 2-3 until convergence is reached.

Despite being very simple, this method is usually not applied to solve OCPs because it is not robust. Especially for nonlinear problems, a small change in the initial state or control variables can imply a large change in the final state. Therefore, the OCP cannot always be solved or needs a very good initial guess. This problem is solved by the multiple shooting method as discussed in Section 4.3.2.

4.3.2. Multiple shooting

In order to avoid the sensitivity of the single shooting technique, this method divides the time interval into smaller steps, such that the sub-problem will be 'more' linear. Each segment can be seen as a single shooting problem, with continuity constraints at the boundaries as illustrated in Figure 4.1. The time interval consists now of n_s segments, such that $t_0 < t_1 < ... < t_{n_s} = t_f$. The continuity is guaranteed by adding the defect constraints, which are written as

$$\mathbf{a}(\mathbf{y}) = \begin{bmatrix} \mathbf{x}_1 - \hat{\mathbf{x}}_0 \\ \mathbf{x}_2 - \hat{\mathbf{x}}_1 \\ \vdots \\ \mathbf{x}_n - \hat{\mathbf{x}}_{n-1} \end{bmatrix} \tag{4.26}$$

where $\hat{\mathbf{x}}_i$ is the result of the propagation over segment i. Besides the defect constraints, other constraints such as a fixed initial state constraint can be added. Similar as for the single shooting problem, these constraints have to be zero. Although the multiple shooting method can solve the convergence problem, it is clear that also the problem size will increase. For example, when doubling the number of segments, also the number of NLP variables is doubled. This leads to large Jacobian and Hessian matrices as can be seen from Section 4.1.1. Fortunately, these matrices become very sparse due to the uncoupling between the multiple shooting phases (Betts, 1998, p. 34), which can drastically reduce the computational time. Lastly, multiple shooting also allows to exploit the benefits of parallel computing, since each segment can be propagated independently. A parallel shooting method can greatly improve the performance especially when the propagation of the state equation is time consuming.

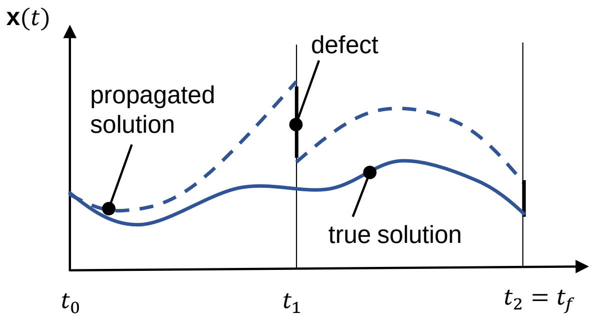


Figure 4.1: Multiple shooting method with two steps.

4.3.3. Collocation

In order to avoid repeating the integration of the ODEs over the segments as is the case for the shooting methods, the collocation method describes the state and control profile as piecewise polynomials over the n_s segments. Commonly splines, Lagrange or Chebyshev polynomials are used that only need to satisfy the equations of motion at the $n_s + 1$ nodes. As a results, both the state and control vector are now discretized over the entire interval and the values between the nodes are found through interpolation. Depending on the interpolation type, the interpolation coefficients depend on the state and state derivative values at the nodes. These are called the collocation conditions and are described for several common interpolation techniques in (Betts, 2009, p. 99). The NLP constraints can therefore be expressed as

$$\mathbf{a}(\mathbf{y}) = \begin{bmatrix} \boldsymbol{\zeta}_1 \\ \vdots \\ \boldsymbol{\zeta}_n \\ \boldsymbol{\psi}(\mathbf{x}(t_0), t_0, u_{n_c}) \\ \boldsymbol{\psi}(\mathbf{x}(t_f), t_f, u_{n_c}) \end{bmatrix} \tag{4.27}$$

where $\boldsymbol{\zeta}$ is the numerical integration defect at each collocation point, which depends on the integration scheme. For a simple Euler integration method, the defect at the nodes is given as

$$\boldsymbol{\zeta}_i = \mathbf{x}_{i+1} - \mathbf{x}_i - h_i \mathbf{f}_i \tag{4.28}$$

Similar expressions exist for more complicated integration methods, which are summarized in (Betts, 2009, p. 133).

4.3.4. Pseudospectral methods

Pseudospectral collocation or global collocation is a collocation method for which the full history of each state and control variable is represented by a single higher-order polynomial. Most existing pseudospectral methods use the Lagrange polynomials as basis functions (Rao, 2010). A Lagrange polynomial can be constructed for order n as

$$L(\tau) = \sum_{i=0}^{n} \ell_i(\tau) x(\tau_i) \tag{4.29}$$

where the basis polynomial $\ell_i(\tau)$ is

$$\ell_i(\tau) = \prod_{j=0, j\neq i}^{n} \frac{\tau - \tau_i}{\tau_j - \tau_i} \tag{4.30}$$

Generally, three different pseudospectral methods using Lagrange polynomials can be distinguished: the Legendre-Gauss (LG), Legendre-Gauss-Lobatto (LGL) and Legendre-Gauss-Radau (LGR) method. The difference between these methods lies in the location of the grid points. An interesting property of choosing these specific grid point schemes is that the approximating solution converges at a spectral or exponential rate. The LG method includes neither of the endpoints, while the LGR method includes

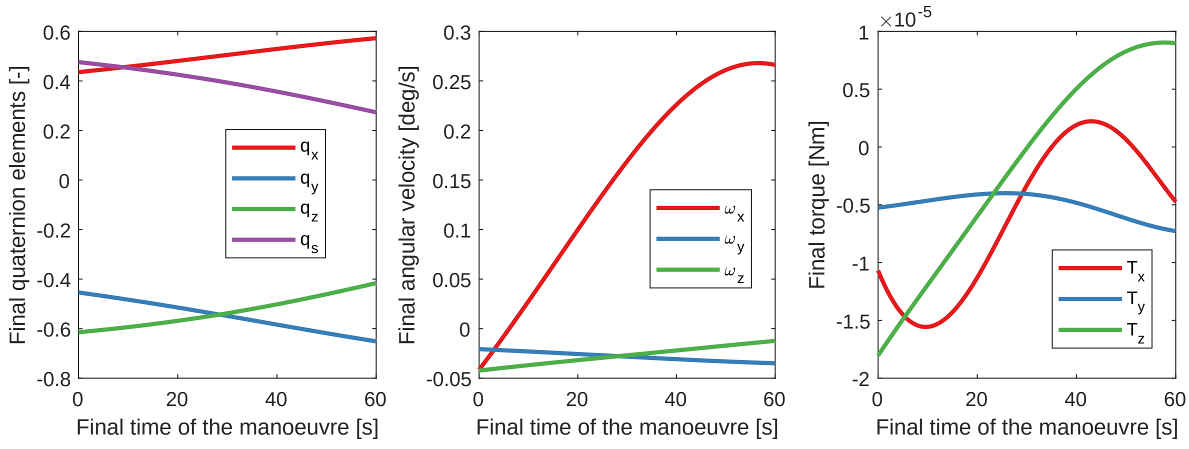


Figure 4.2: The final boundary conditions as a function of the manoeuvre duration for case 1 in Section 8.1.

one of the endpoints and the LGL method includes both of the endpoints. For problems where the final time is free, an endpoint is required such that the LG method is excluded. Next to the Legendre pseudospectral methods, the nodes can also be chosen based on Chebychev or Jacobi polynomials (Garg, 2011).

4.4. Problem formulation for optimal attitude guidance of agile satellites

This section defines the mathematical formulation of the attitude manoeuvre problem, such that the satellite is reoriented between two target observations in the minimum time possible. A complication about this problem, as was introduced in Section 2.1, is that the final orientation of the satellite depends on the final time since the satellite and target are moving w.r.t. each other. Since the OCP requires fixed numerical values for the final boundary conditions (see Equation 4.17), an auxiliary state needs to be introduced. This is the target tracking state, which is defined as

$$\boldsymbol{\delta}(t) = \begin{bmatrix} \mathbf{q}(t) - \mathbf{q}_d(t) \\ \boldsymbol{\omega}(t) - \boldsymbol{\omega}_d(t) \\ \mathbf{T}(t) - \mathbf{T}_d(t) \end{bmatrix} \tag{4.31}$$

where $\mathbf{q}_d(t)$, $\boldsymbol{\omega}_d(t)$ and $\mathbf{T}_d(t)$ are the required attitude, angular velocity and torque vector to observe the final target, depending on the time t. As a consequence, during tracking of this target it holds that $\boldsymbol{\delta}(t) = \mathbf{0}$. The required satellite states and control for this tracking states are obtained by integrating the kinematic ODE as explained in Section 3.4.3. However, this would signify that the ODE should be solved for each iteration in the NLP solver, since the final time is an NLP variable. Clearly, this is inefficient and would be difficult to implement. Therefore, it is chosen to solve the ODE for a range of possible values for the final time. Afterwards, an interpolant is constructed to obtain the final boundary conditions. Fortunately, the final boundary conditions vary smoothly as a function of the manoeuvre, which is illustrated in Figure 4.2. Since the interpolated boundary values are a function of the final time, which is an NLP variable, the interpolant must be given as a symbolic expression to the NLP solver. This is achieved by using the built-in B-spline function of CasADi (see Section 7.2).

The OCP to obtain the optimal attitude manoeuvre between two targets can now be stated as

$$\begin{aligned}
&\min_{\mathbf{u}(t)} J = t_f \\
\text{subject to } &\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \\
&\mathbf{x}(t_0) = [\mathbf{q}(t_0), \boldsymbol{\omega}(t_0)] \\
&\mathbf{u}(t_0) = \mathbf{T}(t_0) \\
&\boldsymbol{\delta}(t_f) = \mathbf{0} \\
&\frac{\omega_x(t)^2}{\omega_{\max,x}^2} + \frac{\omega_y(t)^2}{\omega_{\max,y}^2} + \frac{\omega_z(t)^2}{\omega_{\max,z}^2} \leq 1 \\
&\frac{T_x(t)^2}{T_{\max,x}^2} + \frac{T_y(t)^2}{T_{\max,y}^2} + \frac{T_z(t)^2}{T_{\max,z}^2} \leq 1 \\
&\left|\frac{dT_i}{dt}\right| \leq \left(\frac{dT}{dt}\right)_{\max} \qquad \forall i \in [x, y, z]
\end{aligned} \tag{4.32}$$

Here, the path constraints for the angular velocity and torque are modelled as ellipsoidal constraints for the reasons explained in Section 2.2. Furthermore, a maximum change in the torque is added as constraint for a practical consideration. Since the solution of a time-optimal control problem is often bang-bang control, this would mean the change of the angular acceleration over time as experienced by the satellite is rather high. If these accelerations become too high, vibrations are induced to the satellite that distort the image quality. Therefore, it will be investigated in Section 4.5.3 whether smoothing the attitude profiles can reduce the vibrations. Alternatively, the constraint could be replaced by an increased stabilization time after the manoeuvre.

4.5. Numerical implementation considerations

This section discusses all design choices concerning the implementation of the OCP. First an NLP solver is chosen based on some validation cases in Section 4.5.1. Next, the specific transcription method and the order are chosen in Section 4.5.2. Finally, it is investigated whether adding an additional constraint on the torque profile can reduce satellite vibrations in Section 4.5.3.

4.5.1. Comparison of state-of-the-art NLP solvers

The two common strategies to solve the NLP problem are Sequential Quadratic Programming (SQP) and Interior Point (IP) algorithms, which are introduced in Sections 4.1.2 and 4.1.3 respectively. This section will compare two state-of-the-art implementations of both methods and conclude on the preferred solver for the problem at hand.

Some popular implementations for solving an NLP problem are IPOPT (Wächter and Biegler, 2006), SNOPT (Gill et al., 2005) and WORHP (Büskens and Wassel, 2012). Out of these software packages, IPOPT uses an IP method and is freely available. Since the solver has been widely used for many problems with good results, this will be the chosen NLP solver that uses the IP method. In order to compare the IP method with the SQP algorithm, the WORHP solver is used. Although the SNOPT software package has been more widely used, this solver is only commercially available while WORHP is free for academic use and has been shown to be very competitive or even outperforming SNOPT. The WORHP solver employs an SQP method with an underlying IP method for the quadratic sub-problems and is known to be able to handle several millions of variables and constraints due to efficient handling of the sparsity of large-scale problems (Büskens and Wassel, 2012).

The chosen test case to compare the performance of the SQP and IP method is taken from (Betts, 2009, p. 299). This concerns an OCP that aims to find the minimum time in which a satellite can execute a 150° roll manoeuvre around the x-axis. The angular velocity of the satellite is zero at both the start and end of the manoeuvre and the magnitude of the torque is limited during the manoeuvre by $T \leq 50$ N·m for all three axes. Furthermore, the dynamic equations for this problem are identical to the dynamics introduced in Equation 3.10. The moments of inertia are $I_x = 5621$ kg·m^2, $I_y = 4557$

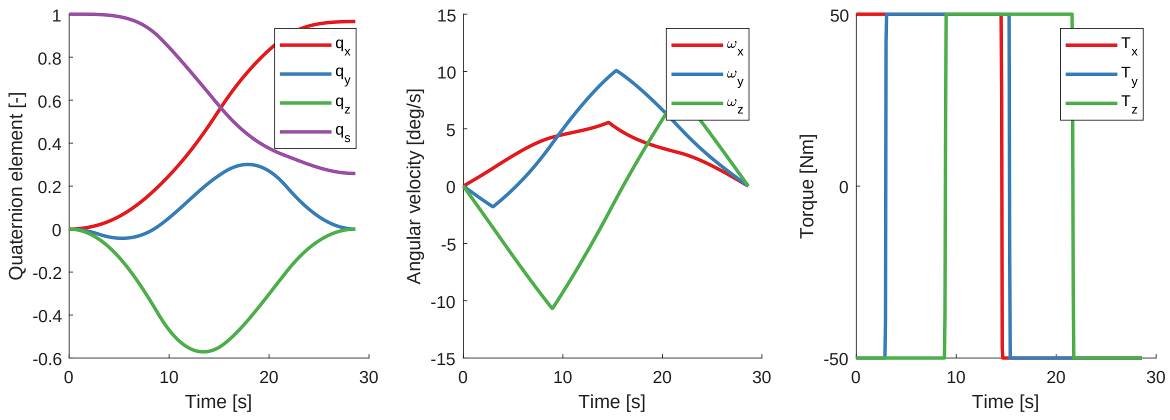


Figure 4.3: Optimal state and control history.

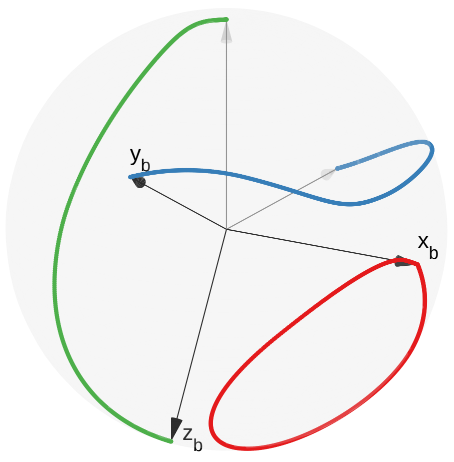


Figure 4.4: Traces of the satellite body frame axes during a time optimal manoeuvre of 150 deg around the x-axis plotted on a unit sphere.

kg·m^2 and $I_z = 2364$ kg·m^2. All off-diagonal components in the inertia matrix are zero. The resulting problem statement is shown below:

$$\begin{aligned} &\min_{\mathbf{u}(t)} J = t_f \\ \text{subject to } &\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \\ &\mathbf{q}(t_0) = [0, 0, 0, 1] \\ &\mathbf{q}(t_f) = [\sin 75°, 0, 0, \cos 75°] \\ &\boldsymbol{\omega}(t_0) = \mathbf{0} \\ &\boldsymbol{\omega}(t_f) = \mathbf{0} \\ &|T_i(t)| \leq T_{\max} \quad \forall i \in [x, y, z] \end{aligned} \tag{4.33}$$

To gain more insight about the performance of both solvers, they are tested for two different transcription methods: multiple shooting and the LGR pseudospectral collocation. These transcription methods are solely chosen to demonstrate the difference in performance for the solvers between shooting and collocation methods. The transcription method that will be used for the problem in this thesis is determined in Section 4.5.2.

The multiple shooting solution for this problem using 500 intervals is shown in Figure 4.3. A visual representation of the slew is also shown in Figure 4.4. It is interesting to note that the optimal slew

Table 4.1: Comparison of NLP solvers IPOPT and WORHP for solving a spacecraft attitude manoeuvre OCP.

Transcription method	J [s]**		CPU time [s]		Iterations	
	IPOPT	WORHP	IPOPT	WORHP	IPOPT	WORHP
LGR PSM, 25 points	28.846	60.487	0.524	0.542	55	11
LGR PSM, 100 points	28.643	30.567	102	35.3	421	63
Multiple shooting, 100 points	-	28.632*	-	2.81*	-	21*
Multiple shooting, 500 points	-	28.630*	-	18.0*	-	29*

*Convergence could only be reached when the solver is initialized using the solution from IPOPT for LGR PSM, 25 points.
**The optimal result as reported by Betts (2009) is $J = 28.630$ s.

is not an eigenaxis rotation around the x-axis. This attitude manoeuvre can by intuition be thought off as the first idea to obtain a time-optimal rotation. However this is generally not the case as shown by Bilimoria and Wie (1993) and confirmed in this example.

The results of the NLP solver comparison for this problem are given in Table 4.1. The initial solution used to generate these profiles is described in Section 4.6.1, except if stated otherwise. The quality of the solution is given by the performance parameter J and the NLP solver efficiency is given by the CPU time. Furthermore, also the number of iterations as necessary for convergence is included. Regarding the quality of the solutions, the WORHP solver was often found to converge towards a local minimum of poor quality. Especially considering the solution of the PSM using 25 collocation points, the quality of the solution using WORHP can become unacceptable. On the other hand, the solution using the IPOPT solver converges consistently to the optimum when using more points. Only if the WORHP solver is initialized with a good initial solution and the multiple shooting transcription method is used, the solver is able to obtain high-quality solutions. Nonetheless, the WORHP solver was found to be able to solve problems with a higher number of decision variables. The IPOPT solver was unable to converge, while the WORHP solver can efficiently cope with these problems, on the condition it was given a good initial guess. Next to that, efficiency of the solvers in CPU time is found to be comparable for this study case. In particular for high-order solutions, the WORHP solver becomes more efficient due to its highly optimized handling of the sparsity of the system. However for lower-order problem formulations, no significant differences were found.

From these results and further experimenting with the algorithm, the following conclusions were derived:

- The optimal attitude guidance problem has many local optimal solutions. Even for the same performance index, multiple attitude profiles may exist.
- The IPOPT solver is robust in finding a high-quality local optimum when using the PSM.
- The IPOPT solver fails to solve problems with many decision variables.
- The WORPH solver is mostly useful to refine a solution which is obtained using another solver. Furthermore it is very sensitive to the provided initial guess.
- The SQP approach is attractive when the objective function and the Hessian are expensive to evaluate, since in general less iterations are required for the same accuracy.

As a result from these conclusions, the IPOPT solver was chosen. An additional consideration was that the low-order PSM is preferred for practical reasons, explained in Section 4.5.2. Furthermore, since the problem will need to be solved many times, it is also of great interest that the solver is robust, which is also a particular strength of IPOPT. If the solver does not converge to a similar local optimum for slightly different boundary conditions, this would impose discontinuities while generating the large database of optimal solutions. As is further explained in Chapter 5, these discontinuities would result in a major issue when interpolation or regression is used on the database.

4.5.2. Transcription method selection

The choice of the transcription method is a trade-off between accuracy, robustness and computational effort. The single shooting method is known for its simple implementation, but commonly fails to converge due to the sensitivity of the initial state. The multiple shooting (MS) methodli solves this problem

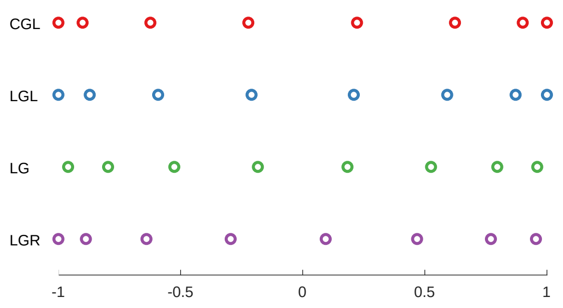


Figure 4.5: Collocation points for a 7th-order polynomial for different PSM methods.

and can achieve high-accuracy solutions, within realistic computation times. However, the collocation methods can be much faster, while still providing good quality solutions. Especially pseudospectral collocation is known for its fast convergence rate. The pseudospectral methods (PSM) that will be tested are the Chebychev-Gauss-Lobotto (CGL) collocation and the three Legendre collocation methods as introduced in Section 4.3.4. The location of the collocation points for these methods is illustrated in Figure 4.5. For the following comparison of transcription methods, the LG and LGL method are not included since they were found to cause convergence issues. A similar conclusion was found by Garg (2011).

To compare the solution quality and efficiency of the transcription methods, 500 random cases of the OCP described Section 4.4 are solved for a certain number of collocation points. From Figures 4.6 and 4.7 it is clear that increasing the number of collocation points results in a better solution, at the cost of increased computational time. Since a large database of optimal control solutions needs to be constructed, a high accuracy is not as important as computational efficiency for this problem. Furthermore, the LGR and CGL transcription method have the best performance, which are relatively similar. Nonetheless, the CGL has another advantage, since a collocation point is placed directly at the final time of the control interval. Therefore, the accuracy of the final boundary condition will be higher when using the CGL PSM (Rao, 2010). As a conclusion, the CGL method will be used to transcribe the OCP problem.

Now that the transcription method is determined, also the preferred order needs to be chosen. Again, the CPU time, robustness and accuracy are of concern. Looking at Figures 4.6 and 4.7, using 16 points was found to be a good compromise for these trade-off criteria.

4.5.3. Slew profile smoothing

As introduced in Section 4.4, the angular acceleration profile of the satellite over time should be smooth, such that excessive vibrations are avoided and the image quality is high. For this reason, it is investigated whether a constraint on the derivative of the torque w.r.t. time should be imposed. If so, this section will determine a value for the maximum derivative of the torque, based on the satellite design.

The vibrations induced to the satellite by a manoeuvre are analyzed by performing a Fourier transform on the control profile. A Fourier transform converts a signal from the time domain to the frequency domain. In other words, the Fourier transform can decompose the torque input over time into its constituent frequencies. Now, the vibrations experienced by the satellite are a function of the torque input in the frequency domain and the geometry of the satellite. Since making a full analysis of the vibrations would require information about the exact geometry and as well as the damping properties of the satellite, this is out of the scope of this research. Rather, only the frequencies will be identified for which the vibration should be damped. Therefore, only the Fourier transform of the torque input is required.

The results are obtained using the build-in Fast Fourier Transform (FFT) function of Matlab. It was numerically determined that especially short slew manoeuvres should be investigated, since the frequency of the torque input is for these cases the highest. The highest frequency is of interest since

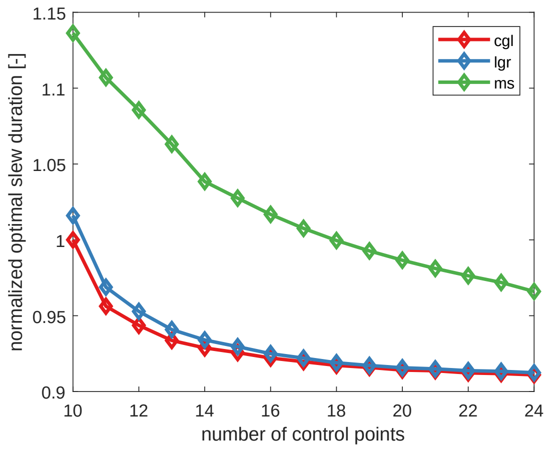


Figure 4.6: Comparison of quality of the optimal solution obtained for different transcription methods. The value for each number of collocation points is the average of 500 test cases, normalized w.r.t. the value for CGL with 10 collocation points.

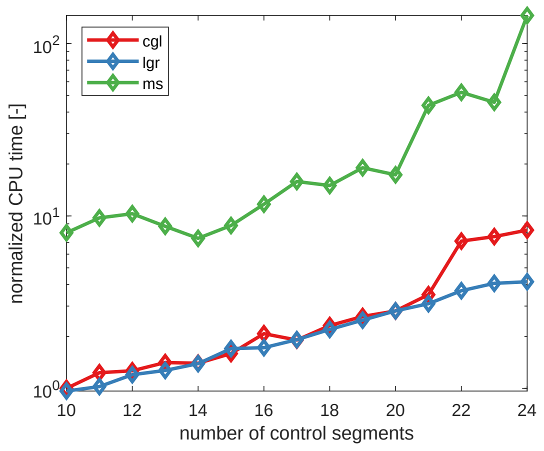


Figure 4.7: Comparison of average CPU time to obtain the optimal solution for different transcription methods. The value for each number of collocation points is the average of 500 test cases, normalized w.r.t. the value for CGL with 10 collocation points.

this frequency should be lower than the lowest eigenfrequency of the satellite. An example of a short manoeuvre without constraint on the torque derivative is shown inn Figure 4.8. It can be seen that the solution is close to a bang-bang solution, such that a peak occurs in the frequency domain around 0.2 Hz and 0.8 Hz. Now, a constraint for the same manoeuvre is applied for the same manoeuvre in Figure 4.9. Clearly, the slew does not resemble a bang-bang manoeuvre. Furthermore, the magnitude of the torque is significantly reduced. The peaks in the frequency domain are now shifted to lower frequencies as well. Therefore, this torque profile is smoother for the satellite. However, the duration of the manoeuvre is more than twice than that of the manoeuvre without smoothing, while this is the optimization objective. Therefore, a trade-off between the optimal time and smoothness of the torque profile is required.

An overview of the frequency of the torque profiles for a range of values for the maximum torque derivative is shown in Figure 4.10, 4.11 and 4.12. These figures display the Fourier transforms of the optimal torque inputs for the x, y and z axis respectively. For the x-axis, a more stringent maximum torque derivative results in a smoother profile as expected. However, for the y-axis and z-axis, this is not always the case. Since the absolute magnitude of the torque for these axes is lower, this issue is not regarded to be critical. Additionally, also the optimal slew time as a function of the maximum derivative constraint is given in Figure 4.13. Clearly, the constraint has a large impact on the optimal slew time. Even more, the maximum frequency of the torque input only decreases significantly when a very stringent constraint on the maximum torque derivative is imposed. In any case, the natural frequencies for the small satellite in this research, the experienced vibrations are well below the typical lowest natural frequency, which ranges between 50 and 100 Hz (Israr, 2014). Therefore, it can be concluded that no limit on the torque derivative shall be imposed. Still, if the satellite would have a higher mass and therefore higher inertia, it should be considered to smoothen the profile.

4.6. Optimal guidance algorithm robustness enhancements

Since the optimal guidance profile algorithm needs to solve a large number of problems, it is important that its solution is obtained in a fast and robust way, avoiding as much as possible convergence issues of the NLP solver.

4.6.1. Initial guess generation

As explained in Section 4.2, the NLP solvers need an initial guess from which the local optimization is initiated. A high-quality initial guess will therefore greatly reduce the number of necessary iterations before convergence. It was found that especially a bad initial guess of the optimal slew time will either

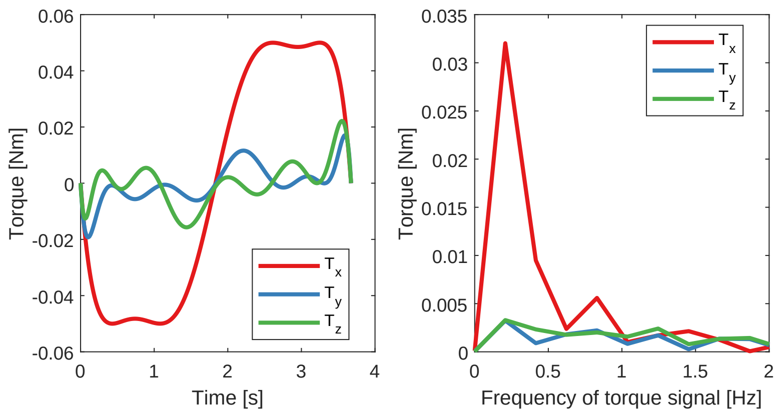


Figure 4.8: Fourier transform of a torque input without a constraint on the torque derivative.

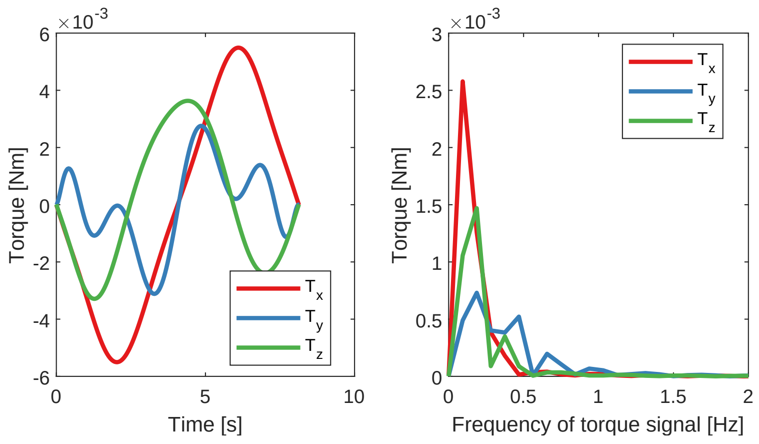
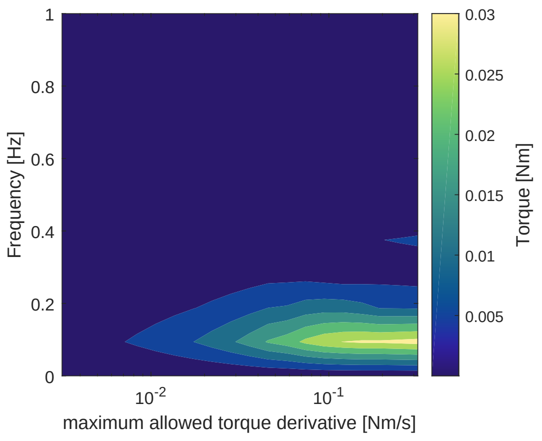


Figure 4.9: Fourier transform of a torque input with a constraint on the torque derivative $(dT/dt)_{\max} = 0.003$Nm/s.



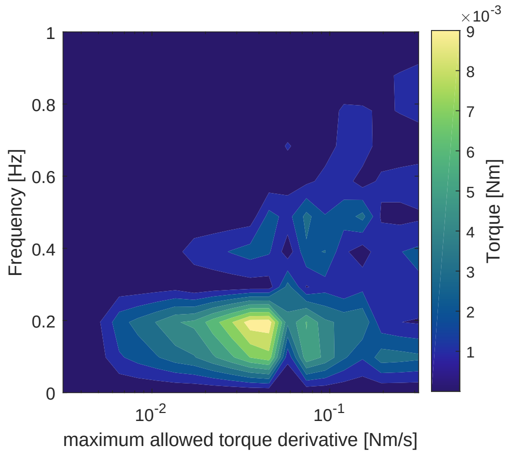


Figure 4.10: Magnitude of the ellipsoidal constraint equations, with constraints at the collocation points.

Figure 4.11: Magnitude of the ellipsoidal constraint equations, with constraints at 100 linear distributed points.

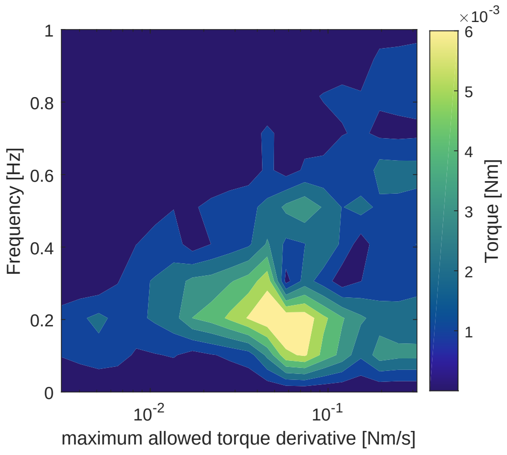


Figure 4.12: Magnitude of the ellipsoidal constraint equations, with constraints at the collocation points.

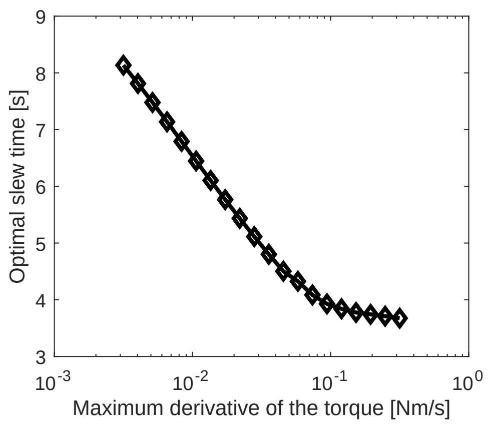


Figure 4.13: Magnitude of the ellipsoidal constraint equations, with constraints at 100 linear distributed points.

cause slow convergence or even failure of the solver. Even more it is also important for an NLP solver that uses the IP method such as IPOPT, that the initial solution is feasible. An infeasible solution requires a *restoration phase* by IPOPT, which tries to reduce the constraint violation and find a feasible solution. However, this regularly results in failure of the solver when no feasible solution is found within a few iterations.

A deterministic solution that provides a feasible initial guess with sufficient quality is the shape-based method as presented by Caubet and Biggs (2013). Here, the quaternion profile is approximated by the shape of a fifth-degree polynomial.

$$\mathbf{q} = \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2 + \mathbf{a}_3 t^3 + \mathbf{a}_4 t^4 + \mathbf{a}_5 t^5 \tag{4.34}$$

The coefficients of the polynomial are determined by taking the first and second quaternion derivatives at the start and the end of the manoeuvre. The derivative can be calculated given the boundary constraint of the angular velocity and torque and using Equations 3.54 and 3.55. Using the derivatives of the quaternion polynomial, the following system of equations can be derived for each quaternion element:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} a_{0,i} \\ a_{1,i} \\ a_{2,i} \\ a_{3,i} \\ a_{4,i} \\ a_5 \end{bmatrix} = \begin{bmatrix} q_{0,i} \\ q_{f,i} \\ \dot{q}_{0,i} \\ \dot{q}_{f,i} \\ \ddot{q}_{0,i} \\ \ddot{q}_{f,i} \end{bmatrix} \quad i \in [x, y, z, s] \tag{4.35}$$

which results in the following polynomial coefficients:

$$\begin{cases} a_{0,i} = q_{0,i} \\ a_{1,i} = \dot{q}_{0,i} \\ a_{2,i} = \frac{1}{2}\ddot{q}_{0,i} \\ a_{3,i} = -\frac{10}{t_f^3}(q_{0,i} - q_{f,i}) - \frac{2}{t_f^2}(3\dot{q}_{0,i} + 2\dot{q}_{f,i}) - \frac{1}{2t_f}(3\ddot{q}_{0,i} - \ddot{q}_{f,i}) \\ a_{4,i} = \frac{15}{t_f^4}(q_{0,i} - q_{f,i}) + \frac{1}{t_f^3}(8\dot{q}_{0,i} + 7\dot{q}_{f,i}) + \frac{1}{2t_f^2}(3\ddot{q}_{0,i} - 2\ddot{q}_{f,i}) \\ a_{5,i} = -\frac{6}{t_f^5}(q_{0,i} - q_{f,i}) - \frac{3}{t_f^4}(\dot{q}_{0,i} + \dot{q}_{f,i}) - \frac{1}{2t_f^3}(\ddot{q}_{0,i} - \ddot{q}_{f,i}) \end{cases} \tag{4.36}$$

Unfortunately, the final time is the objective of this optimization problem and is therefore unknown. In order to still obtain an initial guess, the final time t_f was obtained using a rooting-finding algorithm. The algorithm finds the minimum time t_f for which none of the constraints as presented in Equation 4.32 is violated. First, a polynomial is constructed for each constraint equation for a range of possible

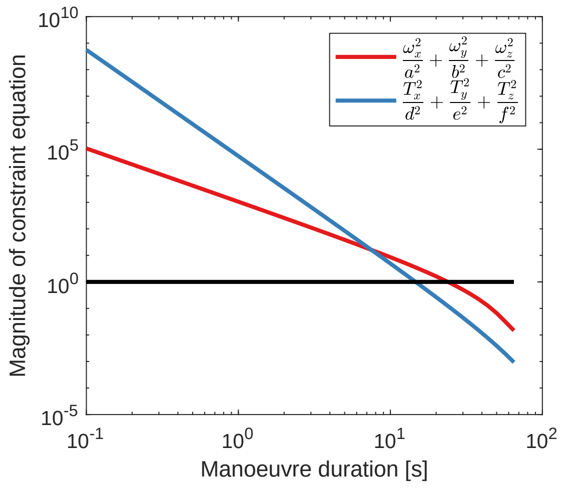


Figure 4.14: Example of an initial guess of time by find the root.

manoeuvre durations, as illustrated in Figure 4.14. Next, the intersections are found with the maximum value of the constraints equation, which equals 1 and is represented by the black line. The polynomials are constructed as Lagrange polynomials and a Matlab built-in root-finding algorithm is used. Also, a minimum value of 0.1 seconds for the manoeuvre duration is assumed, since the constraint equations go to infinity as the manoeuvre time goes to zero. For the presented example in Figure 4.14, the ellipsoidal constraint on the angular velocity dominates and therefore determines the estimated minimum time to be 23.9 s. Finally, using this initial guess of the manoeuvre time, the corresponding attitude and control profiles are obtained using Equation 4.36. The resulting initial guess is shown in Figure 4.15. The torque and angular velocity are found using the inverse dynamics relations from Equations 3.55 and 3.56.

Since the initial guess is found using a shape-based method, the quaternion elements of the resulting manoeuvres are not guaranteed to form a unit quaternion. Therefore, the shape-based method can be further improved by normalizing the polynomials, using the method presented in Section 4.6.3. Since this initial guess does not violate any constraints, a feasbile solution is found at all times, such that the necessary number of iterations for the NLP solver is reduced by 2-3 times. Especially for short slew manoeuvres (<10 seconds) the robustness was also improved.

4.6.2. Quaternion sign flips

It was shown in Equation 3.19 that the quaternions q and $-q$ represent the same orientation. Yet, the trajectory of the rotation between q_0 and q_f is not idential to the trajectory from q_0 to -q_f. One of the paths will be longer, and is referred to as the winding trajectory (Caubet and Biggs, 2013). In order to always select the best sign and converge to the highest quality solution, the following procedure is used. First, the quaternion q_{rot} that defines the rotation between q_0 and q_f is calculated as:

$$q_{rot} = q_0 \otimes q_f^* \tag{4.37}$$

Next, the rotation angle of q_{rot} is obtained by re-arranging Equation 3.16:

$$\theta = 2\arctan\frac{||q_v||}{q_s} \tag{4.38}$$

Whenever this angle $|\theta| > \pi$, the opposite sign should be selected for the boundary quaternion such that the winding trajectory is avoided.

Also, a second issue emerged due the quaternion sign phenomenon. When representing the final boundary constraint of the slew as a function of time, as shown in Figure 4.2, flips in the sign of the

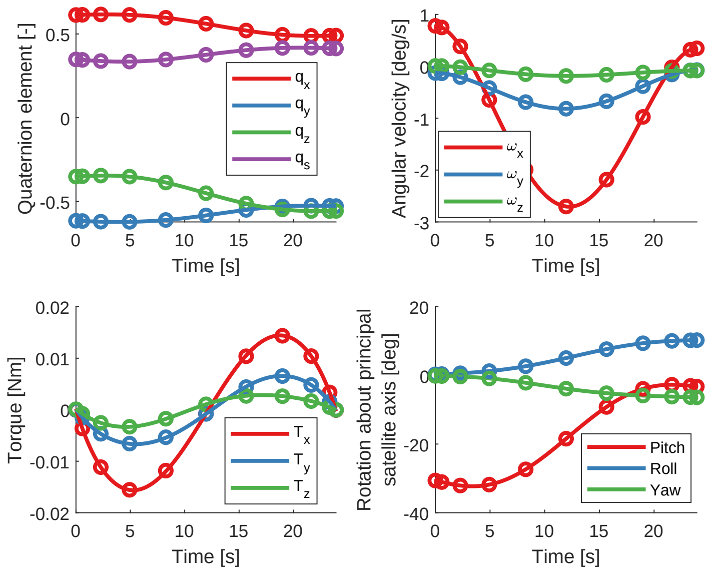


Figure 4.15: Example of an initial guess .

quaternion may occur over time. Also for this issue, a procedure was designed. Consider two quaterions $q(t_i)$ and $q(t_j)$ which represent both the final boundary condition of the quaterion at time t_i and t_j respectively, where $t_i < t_j$. The following rule is proposed to apply on $q(t_i)$ and $q(t_j)$:

$$\begin{cases} \text{flip sign} & \text{if } \|q(t_i)+q(t_j)\| < \|q(t_i)-q(t_j)\| \\ \text{keep sign} & \text{otherwise.} \end{cases} \tag{4.39}$$

This rule is now applied for all epochs at which the final boundary condition is found after integrating the kinematics ODE.

4.6.3. Quaternion interpolation normalization

The final state constraint depends on final time and can be obtained by integration the ODE from Section 3.4.3. Therefore, the final state boundary is expressed as an interpolation which depends on the slew duration, as was explained in Section 4.4. However, because of the interpolation between the calculated nodes, the norm of the final quaternion is not perfectly equal to 1. Since this violates the imposed kinematic constraints, this numerical issue impedes the convergence due to the constraint violation. Typically, the solution will converge to a point that is close to the nodes of the interpolant, such that the interpolation error is smallest. However, this implies a sub-optimal solution is found. Furthermore, a similar issue exists for the shape-based initial guess from Section 4.6.1. A solution as first suggested by Caubet and Biggs (2013) is to normalize the polynomial in the following way. First take a quaternion, which is not necessarily a unit quaternion. The normalized quaternion q_U is then found as:

$$q_{U,i} = \frac{q_i}{\|q\|} \tag{4.40}$$

Furthermore, to obtain the angular velocity as shown in Section 3.4.4, the derivative of the quaternion is necessary as well. The value for $\dot{q}_U$ is found by differentiating the above equation:

$$\dot{q}_{U,i} = \frac{\dot{q}_i}{\|q\|} - \frac{q_i}{\|q\|^3}\sum_{i=1}^{4} q_i\dot{q}_i \tag{4.41}$$

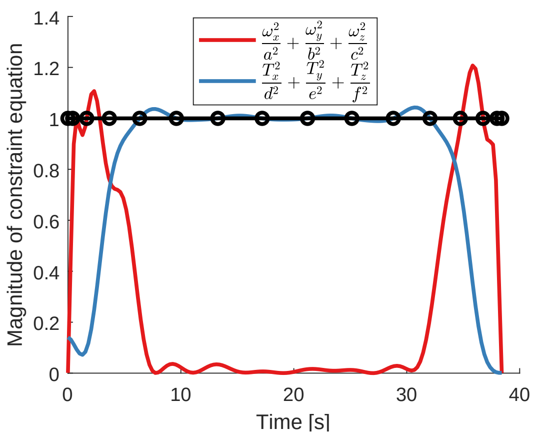


Figure 4.16: Magnitude of the ellipsoidal constraint equations, with constraints at the collocation points.

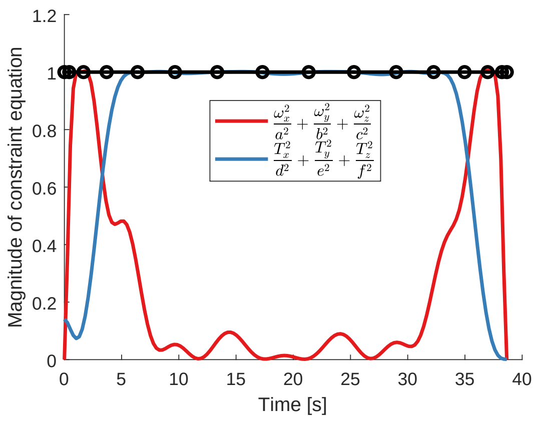


Figure 4.17: Magnitude of the ellipsoidal constraint equations, with constraints at 100 linearly distributed points.

Finally, the second derivative is also required to obtain the torque profile:

$$\ddot{q}_{U,i} = \frac{\ddot{q}_i}{\|q\|} - \frac{\dot{q}_i}{\|q\|^3}\sum_{i=1}^{4} q_i\dot{q}_i + q_i\left[\frac{3}{\|q\|^5}\left(\sum_{i=1}^{4} q_i\dot{q}_i\right)^2 - \frac{1}{\|q\|^3}\left(\sum_{i=1}^{4}\left(\dot{q}_i^2 + q_i\ddot{q}_i\right)\right)\right] \tag{4.42}$$

4.6.4. Constraint violation between collocation points

The path constraints for the angular velocity and torque as introduced in Section 4.4 need to be satisfied over the entire time interval. This is commonly achieved by imposing the constraint at each of the collocation points. Despite, in between the collocation points the path constraint may be violated due to the discretization. For the chosen number of 16 collocation points, still considerable constraint violations occur, as shown in Figure 4.16. This issue can be resolved by adding additional points over the time interval for which that path constraint needs to be satisfied.

As a first trial, the additional points are added in between the collocation points. This method is attempted for 0, 1, 2 and 3 extra points in between the collocation points, which is shown in Table 4.2. As expected, adding more points reduces the path constraint violation. Another strategy is to only evaluate the path constraints at linearly distributed points over the entire manoeuvre. However, the values for the state variables are not known in between the collocation points for both strategies. This can be solved by changing the discretization of the OCP problem, but this approach would imply a significant increase in the number of NLP variables, and therefore computational time. Rather, is was chosen to obtain the state variables by constructing a Lagrange polynomial using the NLP variables at the already existing collocation points.

After comparing both distributions of constraint points, the linearly distributed points are preferred. Overall, the constraint violation was found to be less for this distribution for the same number of constraint points. For example, adding a single point in between the collocation points, such that the constraint is evaluated at 31 points, results in larger constraint violations than using 30 linearly distributed points. Also when using more points for both strategies, the linearly distributed constraint points performed better.

4.7. Pseudospectral collocation using Chebychev-Gauss-Lobotto points

As explained before, the OCP needs to be transcribed into an NLP problem, for which the solving method is presented in Section 4.1. Now, this section will explain the specific procedure to transcribe an OCP using the CGL pseudospectral method, as is used for this thesis. The following theoretical discussion is based on (Trefethen, 2000, p. 51-54) and (El-baghdady and El–Azab, 2016).

Table 4.2: Comparison of the constraint violation due to interpolation using different discretization of points to evaluate the path constraints. In total, 500 random cases are used to evaluate the maximum value of the path constraint equation during the attitude manoeuvre. The worst case value of all 500 cases is displayed.

	Extra points in between collocation points				**Linearly distributed points**		
	0	1	2	3	30	100	200
Path constraint for ω	2.034	1.229	1.113	1.050	1.151	1.015	1.015
Path constraint for T	1.179	1.083	1.028	1.034	1.020	1.024	1.018
normalized CPU time [-]	1.00	1.55	1.59	1.53	1.48	1.50	1.88

First, consider the Chebychev polynomials as defined on the interval $[-1, 1]$. For the polynomial order N, the CGL collocation points $\boldsymbol{\tau}_{N+1} = [\tau_0, \dots, \tau_N]$ are:

$$\tau_i = -\cos\frac{i\pi}{N}, \quad i = 0, 1, \dots, N \tag{4.43}$$

These collocation points can be scaled to the time domain as:

$$t_i = \frac{1}{2}(\tau_i + 1)(t_f - t_0) + t_0 \tag{4.44}$$

For convenience, t_0 is set to zero and t_f is simply the full control interval. Now, the time instances t_i are the points in time where NLP variables are chosen to represent the state and control. In other words, the NLP variable vector is now:

$$\mathbf{y} = \begin{bmatrix} \mathbf{x}(\tau_0) \\ \vdots \\ \mathbf{x}(\tau_N) \\ \mathbf{u}(\tau_0) \\ \vdots \\ \mathbf{u}(\tau_N) \\ t_f \end{bmatrix} \tag{4.45}$$

Since the variable discretization is now established, the NLP constraint vectors **a**(**y**) and **b**(**y**) can be derived. For most of the constraints as presented in Equation 4.32, this is a trivial exercise. Nonetheless, the method to impose the dynamical constraints is more challenging and is discussed below.

It was shown in Section 4.3.4 that the pseudospectral method represents the full history of the states and control variables as a single Lagrange polynomial L:

$$L(\tau) = \sum_{i=0}^{N} \ell_i(\tau) x(\tau_i) \tag{4.46}$$

Simply taking the derivative w.r.t. the non-dimensional time τ results in:

$$\frac{dL(\tau)}{d\tau} = \sum_{i=0}^{N} \frac{d\ell_i(\tau)}{d\tau} x(\tau_i) \tag{4.47}$$

In the above equation, the derivative of the Lagrange basis ℓ_i polynomials can be expressed as:

$$\frac{d\ell_i(\tau)}{d\tau} = \sum_{j=0}^{N} \frac{d\ell_i(\tau_j)}{d\tau} \ell_j(\tau) \tag{4.48}$$

Even more, this sum can be put in matrix form:

$$\frac{d\boldsymbol{\ell}(\tau)}{d\tau} = \boldsymbol{\ell}(\tau)\mathbf{D}_{N+1} \tag{4.49}$$

where **D** is the spectral differentiation matrix and $\boldsymbol{\ell} = [\ell_0, \ldots, \ell_N]$. The entries of the differentation matrix can be computed as

$$D_{i,j} = \begin{cases} -\frac{2N^2+1}{6} & i = j = 0 \\ \frac{c_i}{c_j}\frac{-1^{i+j}}{\tau_i - \tau_j} & i \neq j \\ -\frac{\tau_i}{2(1-\tau_i^2)} & 1 \leq i = j \leq N-1 \\ \frac{2N^2+1}{6} & i = j = N \end{cases} \tag{4.50}$$

where

$$c_i = \begin{cases} 2 & \text{for } i = 0 \text{ or } N, \\ 1 & \text{otherwise} \end{cases} \tag{4.51}$$

Finally, since the NLP state variables $\mathbf{x}(\tau_i)$ are evaluated at the collocation points, one can now simplify Equation 4.47 to obtain $\dot{\mathbf{x}}$ and impose the dynamics constraints at each collocation point τ_i:

$$\mathbf{D}_{N+1}\mathbf{x}(\tau_i) = \frac{t_f}{2}\mathbf{f}(\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), t) \tag{4.52}$$

By re-arranging and setting one side of the equation to zero, it is clear this result is in fact a special case of the integration defect $\boldsymbol{\zeta}$ as used for the general collocation method (see Section 4.3.3).

4.8. Example optimal profiles

This section gives three example solutions of the optimal attitude manoeuvre problem in this thesis. A short, a medium and a longer attitude manoeuvre are chosen to give more insight about the results. The short attitude manoeuvre concerns a manoeuvre such that the same target is scanned twice. The target is situated directly below the satellite, such that the off-nadir angle is zero, and is scanned twice in the same direction. The resulting state and control variables are shown in Figure 4.18. Clearly, the manoeuvre resembles a bang-bang manoeuvre. This is the case since the maximum angular velocity around the x-axis is not reached during this short manoeuvre.

The input for the second example manoeuvre of medium duration is shown in Table 4.3. The results as shown in Figure 4.19 illustrate that the optimal control input is significantly more complex than a bang-bang solution for this case. In fact, all slews which reach the maximum limit of the ellipsoidal constraint of the angular velocity show more complex profiles. A similar behaviour can be observed for the long slew, presented in Figure 4.20. It was found that the long slew manoeuvres typically have a large torque magnitude at the start and end of the manoeuvre. The period in between is a coasting phase with a small magnitude for the torque input. Finally, it is worth to note that these attitude manoeuvres required around 2 seconds CPU time to find the local optimum. This problem is typically solved from thousands up to millions of times within the scheduling algorithm, depending on the algorithm. Therefore, these attitude manoeuvres cannot be directly implemented in the scheduling algorithms, which is further discussed in Chapter 5.

Table 4.3: Input values for the examples cases of optimal slews. The explanation of the input parameters is given in Section 5.1.

Input parameter [deg]	η_0	$\alpha_{t,0}$	$\alpha_{s,0}$	η_f	$\alpha_{t,f}$	$\Delta\alpha_s$	ν_0
short slew	0	0	0	0	0	0	0
medium slew	0	0	0	30	0	10	0
long slew	0	0	0	30	90	45	0

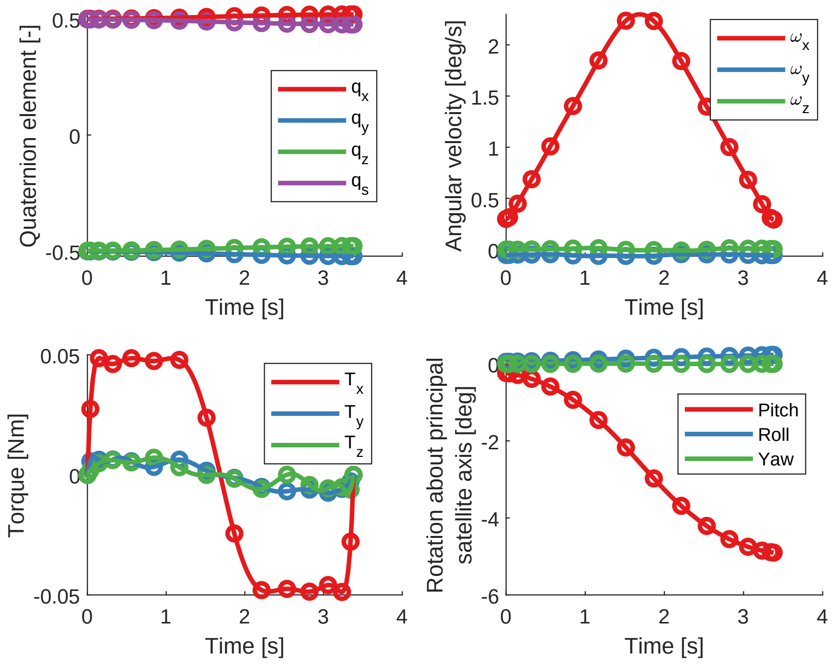


Figure 4.18: Example optimal state and control history for a slew of short duration. The pitch, roll and yaw angle indicate the orientation of the satellite body axes w.r.t. the Hill frame using a z←y←x rotation.

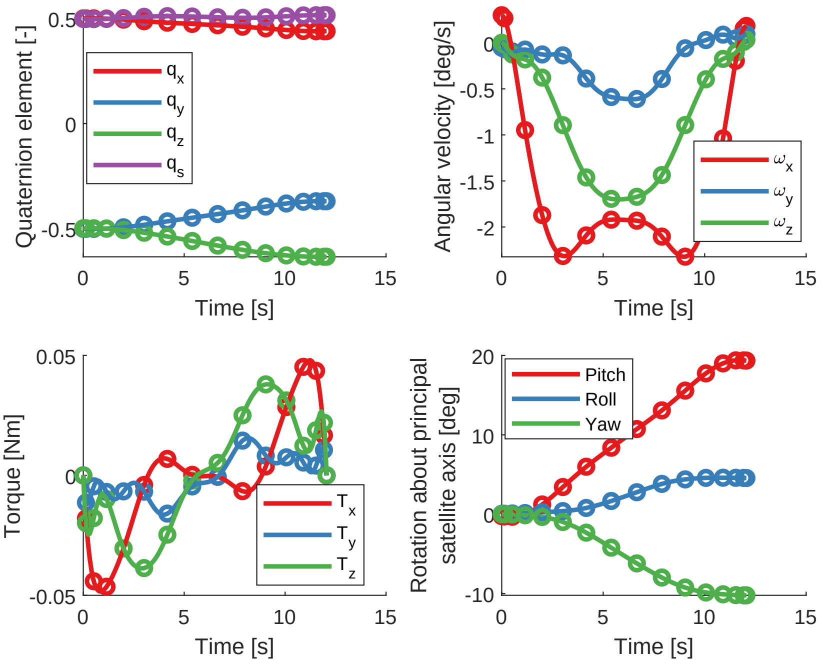


Figure 4.19: Example optimal state and control history for a slew of medium duration. The pitch, roll and yaw angle indicate the orientation of the satellite body axes w.r.t. the Hill frame using a z←y←x rotation.

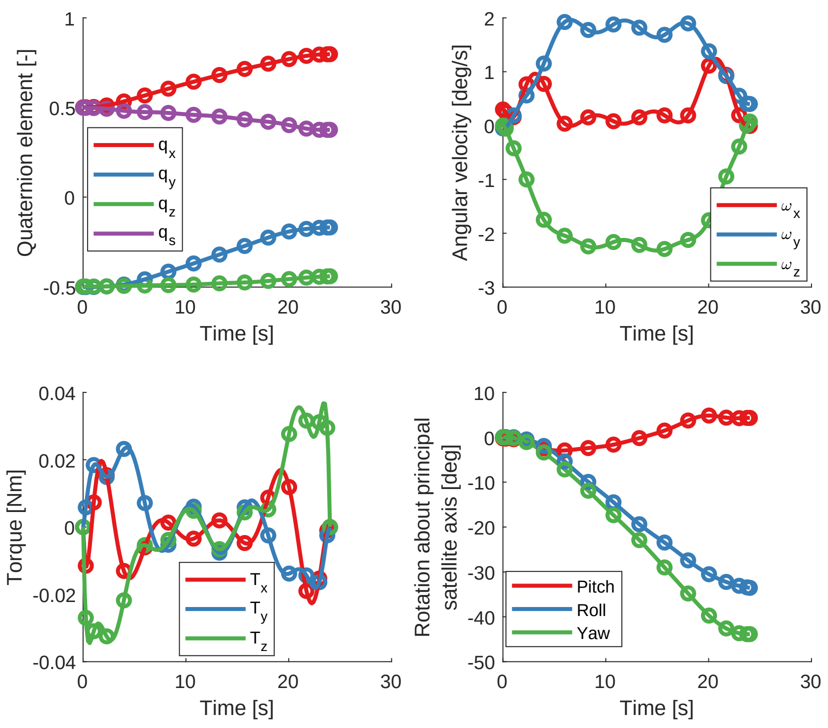


Figure 4.20: Example optimal state and control history for a slew of long duration. The pitch, roll and yaw angle indicate the orientation of the satellite body axes w.r.t. the Hill frame using a z←y←x rotation.

5 Optimal guidance approximation methodology

As explained in Chapter 1 and Section 4.8, the generation of a time-optimal slew manoeuvre is still too computationally expensive for implementation in the scheduling algorithm, which typically requires many evaluations of the optimal manoeuvre times. Therefore, it is decided to obtain the manoeuvre times using a database of optimal manoeuvres. These approximate manoeuvre times are then generated for a specific problem using interpolation or regression techniques. The trade-off and performance of these techniques is the subject of this chapter. Other considered methods to approximate the optimal attitude manoeuvres are the usage of eigenaxis rotations and the analytical shape-based method as presented in Section 4.6.1. However, the eigenaxis rotations are unable to consider all satellite actuator constraints nor the angular velocity boundary conditions and therefore provide an unsatisfactory approximation. On the other hand, the analytical method is able to produce a feasible solution while considering all boundary conditions, but the solutions are still suboptimal and too computationally expensive to obtain. This chapter is structured as follows. Section 5.1 starts by describing the database. Next, a trade-off for the preferred interpolation or regression technique is made in Section 5.2. Sections 5.3 and 5.4 continue with the background theory of the selected method. The results are presented in Section 5.5.

5.1. Database description

The database should be able to model the effect of the inputs on the objective function of the optimal guidance algorithm. Therefore, it was investigated which inputs have significant importance on the optimal manoeuvre time. Given a manoeuvre from target i to target j, the inputs to the guidance algorithm are: the positions of both targets, the scanning directions, and the satellite position over time. Also the line rate of the scanner, the angular resolution, the stabilization time and image acquisition time have an influence on the slew time. However, these parameters are considered to be constant.

After investigation of the dependency of the algorithm inputs on the objective, it was concluded that the following nine parameters will be used for the database:

- The off-nadir angle η_i of the location of target i as seen from the satellite at time t_i
- The off-nadir angle η_j of the location of target j, as seen from the satellite at time t_i
- The azimuth angle $\alpha_{t,i}$ of the location of target i as seen from the satellite at time t_i
- The azimuth angle $\alpha_{t,j}$ of the location of target j as seen from the satellite at time t_i
- The scanning direction angle $\alpha_{s,i}$ of the target i measured as angle w.r.t. the local North of target i
- The difference in scanning direction angle $\Delta\alpha_s = \alpha_{s,j} - \alpha_{s,i}$, where the scanning direction angle of the target j is measured as angle w.r.t. the local North of target j
- A binary variable s_i, indicating the scanning direction of target i
- A binary variable s_j, indicating the scanning direction of target j
- Argument of latitude (ν_i) of the position of the satellite in it's orbit at time of acquiring the center of the image strip of target i

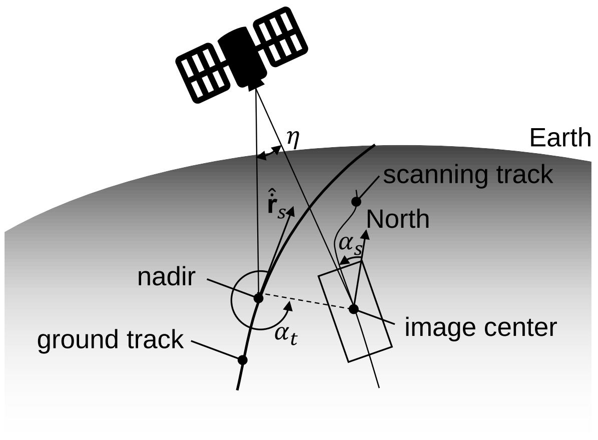


Figure 5.1: Illustration of the database angles η, α_t and α_s.

Here, time t_i indicates the time of scanning the center of target i. Furthermore, the angle η and α_t are defined in the Hill frame, such that the positions of the targets are defined as angles as seen from the satellite. These angles are used since the latitude and longitude of the targets, which is typically the defined input by the customer, are no favorable input parameters. For example at the Earth's poles, the dependency on the longitude is different compared to the situation at the equator, which causes higher non-linear dependency of the objective function on the input parameters. The angular positions defined by η and α_t in the Hill frame are therefore an intuitive better alternative for which the dependency of the objective function on the orbital position is much smaller. Now, the angle η can be found from the normalized relative position vector $\hat{\mathbf{r}}_{c/s}^{hill}$, which gives the position of the center of the target w.r.t the satellite:

$$\eta = \arccos\left(-\hat{x}_{c/s}^{hill}\right) \tag{5.1}$$

and the angle α_t becomes

$$\alpha_t = \arctan\left(\frac{\hat{z}_{c/s}^{hill}}{\hat{y}_{c/s}^{hill}}\right) \tag{5.2}$$

The geometry of the angular parameters η, α_t and α_s is illustrated in Figure 5.1 for a single target. A final consideration about the choice of parameters concerns the time at which the angles η and α_t are computed. In contrast to the latitude and longitude, the angular position of the targets in the Hill frame changes with time. Since the time of the manoeuvres is a function of the angles of the targets at the initial and final time, the final angles in the Hill frame would need to be obtained in an iterative manner. For this reason, the angular position of the targets was computed at a single time. The time t_i was chosen as it is most convenient for implementation in the scheduling algorithm.

Now that the database parameters are set, also their associated upper and lower bounds need to be determined such that the entire attitude manoeuvre design space is captured. Table 5.1 presents the chosen bounds for the angular parameters. The bounds for the parameters η_0, $\alpha_{t,0}$, $\alpha_{s,0}$, η_f, $\alpha_{t,f}$ and ν_0 are directly taken as the minimum and maximum possible value. Recall from Table 2.1 that the maximum off-nadir angle is set to 30 degrees. At time t_i, this is the maximum off-nadir angle of the target i. However, for target j, the maximum off-nadir constraint must be calculated at time t_j, while the database is constructed using angles expressed at time t_i, as was explained before. Since the final time t_j is unknown, the cone of possible target locations is increased such that it includes all possible slew durations. The largest off-nadir angle occurs when the slew duration is maximum. After generating a large database of optimal slew manoeuvres, it was determined that the largest possible slew duration for the used parameter bounds is never more than 60 seconds. When a circular orbit is assumed, the change in true anomaly $\Delta\theta$ after the maximum slew duration $d_{\max}$ is (Wakker, 2015):

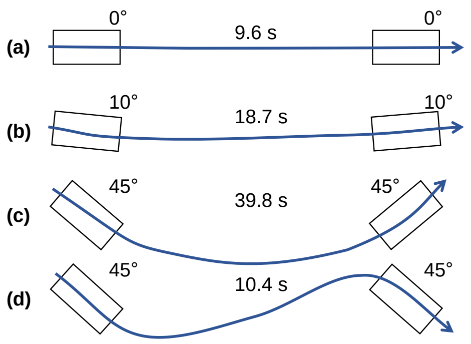


Figure 5.2: Influence of changing the image scanning direction on the minimum attitude manoeuvre duration between two targets (Lemaître et al., 2002), arrows indicate the scanning track of the satellite. The difference in scanning azimuth angle has the biggest impact on the attitude manoeuvre duration.

Table 5.1: List of angular database input parameters and the associated upper and lower bounds.

Database parameter	η_0	$\alpha_{t,0}$	$\alpha_{s,0}$	η_f	$\alpha_{t,f}$	$\Delta\alpha_s$	ν_0
lower bound [deg]	0	0	0	0	0	-45	0
upper bound [deg]	30	360	360	42	360	45	360

$$\Delta\theta = d_{\max}\sqrt{\frac{\mu}{a^3}} \tag{5.3}$$

Using two-dimensional trigonometry, the following relationship is obtained for the maximum final off-nadir angle:

$$\eta_{\max,f} = \arcsin\left(\sin\left(\Delta\theta\right)\frac{R_\oplus}{\sqrt{R_\oplus^2 + a^2 - 2\cos(\Delta\theta)R_\oplus a}}\right) \tag{5.4}$$

Here, a is the semi-major axis of the orbit. Next to the motion of the satellite, also the Earth rotation and orbital perturbances can have a small influence on $\eta_{\max,f}$. These effects are considered by adding a small margin of 1 degree to the maximum off-nadir angle.

The parameter $\Delta\alpha_s$ was taken only between -45 and 45 degrees instead of considering all possible values. The reason is that the difference in scanning azimuth direction between two targets greatly increases the manoeuvres time. As a result, these manoeuvres are never favored in the scheduling algorithm since always faster manoeuvres can be found. An example which demonstrates this rule is shown in Figure 5.2. For that reason, only for small differences in scanning azimuth angles, improvements in the scheduling are possible. Even more, all previous research assumed a constant scanning angle for the scheduling problem (Lemaître et al., 2002). It will be investigated in Section 6.7.4 whether this assumption is also valid for this research, or if it is beneficial to introduce the scanning angle as a decision variable in the scheduling algorithms.

After defining the database parameters, it was attempted to reduce the design space of the attitude manoeuvres by finding symmetries in the database, such that the computational time could be reduced. Nevertheless, no symmetries were found due to the rotation of the Earth and the non-zero stabilization time.

5.2. Database generation method trade-off

A back-of-the-envelope calculation quickly made clear that obtaining the approximate manoeuvre time from the database using interpolation is infeasible due to the nine dimensions of the database and their non-linear dependence on the manoeuvre time (see Section 5.1). The database would have to contain several billion data points to have a reasonably small interpolation error. Therefore, more intelligent methods which can model the database are desired. A more advanced interpolation method is presented in Section 5.2.1. Furthermore, regression of the database using machine-learning techniques is considered in Section 5.2.2.

5.2.1. Grid interpolation techniques

Sparse grid interpolation is the mathematical theory of selecting the minimal number of points, such that a smooth function is optimally approximated and therefore the least number of (expensive) function evaluations are needed. This technique is adopted to counteract the exponential growth of required grid points for increasing dimensions, often referred to as the *curse of dimensionality*. For d dimensions and N grid points in one coordinate direction, the discretization of a sparse grid requires only $\mathcal{O}(N(\log N)^{d-1})$ degrees of freedom, while a full grid would need $\mathcal{O}(N^d)$ degrees of freedom. Even more, the error decay of a full grid interpolation is preserved for sparse grids up to a logarithmic factor. On the other hand, this method requires the multivariate function to have a certain degree of smoothness or periodicity. It must be determined by a numerical experiment whether the function can be sufficiently approximated by sparse grids. A more theoretical background about sparse grids is presented in Appendix B.

The sparse grid method was first tested using the Matlab toolbox Spinterp, version 5.1.1 by Klimke and Wohlmuth (2005). The database was explored for all database parameters, whether their dependence on the optimal manoeuvre time is 'smooth enough' for sparse grids. The sparse grid method using the Clenshaw-Curtis grid (see Appendix B) was found to be reasonably efficient for 2D and 3D interpolation for different combinations of the database parameters, such that the method outperformed any kind of regular interpolation. Unfortunately, when introducing higher dimensional dependency parameters on the optimal manoeuvres, the relations between the parameters become highly non-linear such that the errors consistently exceeded 100% of the manoeuvre time.

An improvement to partially overcome the issues of non-smooth behaviour is the spatially adaptive sparse grid method. This method adds more points in particular regions of the non-linear behaviour. Another experiment was executed using SG++, a C++ toolbox by Pflüger (2010). Unfortunately, the sparse grids were still not able to model the behaviour of the database since the adaptive grid had difficulties to identify the regions which require more points when all dimensions were used.

5.2.2. Machine-learning techniques

Machine-learning is the common name for algorithmic models which can interpret data, improve the model from it, and then make prediction about something. In this research, supervised machine-learning is studied. These are the algorithms which are able to train a target function that can map certain input variables to the output variables as well as possible.

A first machine-learning method is Support Vector Machine (SVM) regression. In principle, an SVM creates the optimal hyperplane which separates the given data in two classes. To construct the hyperplane, several constraints are imposed such that the hyperplane is separated from the data points by a certain minimum value. For SVM regression, this hyperplane is used to perform the regression. For the simple case when a linear function is taken to separate the data, it can be reasoned that the separation of a minimum value to the hyperplane ensures that the resulting regression line is somewhere in the middle of the data points. Furthermore, the SVM method is the only linear model that can also be used for regression of data which is not linearly separable by mapping the data using a non-linear function. For more information, the reader is referred to (Smola and Schölkopf, 2004). An advantage of SVM regression is that the training of the model is typically very fast.

A second considered machine-learning technique is Gaussian Process Regression (GPR), which calculates the probability distribution over the possible fitting functions. Instead of finding the parameters for a fitting function, GPR aims to obtain the maximum likelihood of the best fitting functions for a given dataset. Therefore, the result is not a certain set of optimal weights, but a probability distribution of possible functions. The main advantage of GPR is that the uncertainty of the predicted value is also known. The squared exponential covariance function will be used for the performed comparison, which is widely used (Rasmussen, 2004).

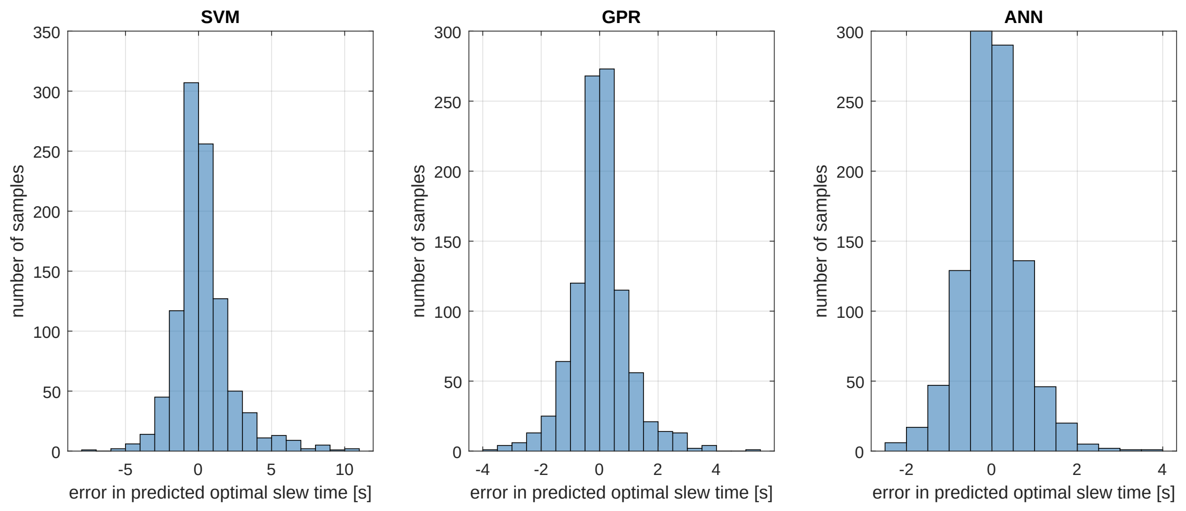


Figure 5.3: Comparison of performance for three machine-learning methods to approximate the optimal slew time, given a dataset of 1000 samples.

The last tested machine-learning method for function fitting is an Artificial Neural Network (ANN). The theoretical background is discussed in detail in Section 5.3. The ANN for this comparison consists of a single hidden layer of 20 neurons. The Levenberg-Marquardt algorithm with bayesian regularization is chosen to train the ANN.

For this comparison, the implementations of the Matlab machine-learning and deep-learning toolbox are used. A small test using 1000 samples of solutions to the OCP is used here for the different machine-learning techniques. The seven non-binary parameters of the database are used, with random input values uniformly distributed over the ranges specified in Section 5.1. The binary variables are chosen such that both the initial and final target are scanned in forward direction. The models are trained using all 1000 samples and the performance is evaluated using another control group of 1000 samples. The performance of the three machine-learning methods is demonstrated in Figure 5.3. The standard deviation of the error in predicated slew times for the different models are the following: $\sigma_{\text{SVM}} = 1.92$ s, $\sigma_{\text{GPR}} = 0.96$ s and $\sigma_{\text{ANN}} = 0.72$ s. The GPR and ANN models provide the best prediction for the control sample group, where the ANN performs slightly better. Even more, when using more samples, the difference between the ANN and GPR is expected to be larger. The ANN can always keep improving when more data is provided, in contrast to GPR which does not benefit as much from additional samples (Foresee and Hagan, 1997). As a consequence, the ANN method will be used to approximate the optimal slew times. A disadvantage however, is that the ANN technique is not able to give an expected uncertainty of the results. Nevertheless, the accuracy of the predicted optimal slew time is of higher importance.

5.3. Artificial neural network preliminaries

The theory of ANNs is inspired by the biological functioning of the brain. The brain operates by using billions of small processing units, called neurons. These neurons are structured in dense networks and communicate through an enormous number of connections between the neurons. The strength of these connections is optimized such that the network can perform a certain task. This process enables the neural network to learn and model complex relations about its environment.

In a similar way to the neural network in the brain, the ANN creates now a structure of neurons for which the strength of the connection between the neurons is represented by a weight factor. These weights are optimized during a process that is called the *training* of the ANN. Finally, the trained model is used to solve a problem. The network is now a mathematical model for which each neuron takes a value based on the connected neurons and an internal processing function. The artificial neurons are

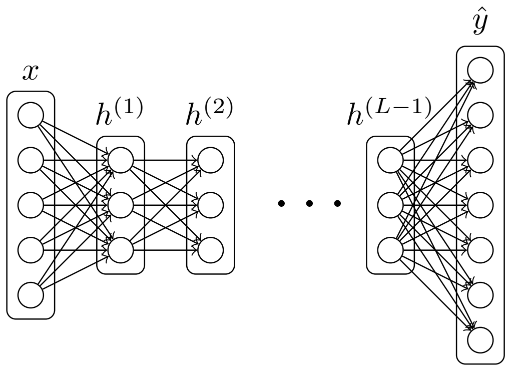


Figure 5.4: Architecture of a deep neural network with five inputs, seven outputs and $L-2$ hidden layers of 3 neurons each.

typically structured into a network as depicted in Figure 5.4, where each node represents a neuron. The neural network has an input level, from which the other neurons compute their according values. The final output layer generates the answer for the given input. This output layer can have any size, but will be a single node for this research. The neurons in between the input and output layer are referred as the *hidden* neurons (h). The traditional neural network has a single hidden layer and neural networks with more hidden layers are the so-called *deep neural networks*. While adding more hidden layers can increase the complexity modelled by the ANN, it also becomes much more computationally intensive to train the network.

Now a closer look is given at the operation within a single artificial neuron. Each individual neuron represents essentially the output of a simple function operation. Given some data input, the artificial neuron takes a weighted sum of all these inputs, adds a bias and finally applies an activation function to the result. The process can be mathematically stated as:

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right) \tag{5.5}$$

Here, the variable a is the value taken by a neuron, w is a weighting factor, b is a bias and σ is an activation function. To keep an overview of the network, the parameters are also indexed in a structured fashion. The index l refers to the level of the neuron in the neural network, j is the index of the neuron in the current layer and k is the index of the connected neuron in the previous layer. Furthermore, the intermediate variable z will be defined for convenience as:

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \tag{5.6}$$

The final concept that needs to be clarified from Equation 5.5 is the activation function σ. It is this function which introduces the non-linearity to the model. The activation function is a monotonically increasing function which maps the weighted sum z_j^l into a certain input domain for the next layer. The desired input ranges is often between 0 and 1 or between -1 and 1. For consistency, the activation function is normally taken to be the same for all artificial neurons in the network. Many activation functions have been proposed, but the hyperbolic tangent function is used for this research:

$$\sigma(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{5.7}$$

There is no general rule for the 'best' choice of activation function. The hyperbolic tangent function is used since it is known to be a robust function and a general accepted function for efficient backpropagation (Hagan et al., 2014, p. 40). Other common activation functions are the sigmoid and Rectified Linear Units (ReLU) functions.

5.4. Artificial neural networks for function fitting

The ANN theory is a powerful framework to solve complex problems. It can been applied to a wide range of problems, which can be divided in three categories. A first one is supervised learning, which trains the ANN by using a database of input data and the corresponding output data. A second category is unsupervised learning. Here, only the input data is given, while the actual output is unknown. This is often used to classify the expected outputs rather than predicting the actual output. Thirdly, the reinforced learning method is not given any data, but trains the ANN in a certain environment, such as the game chess for example. As explained before, a database of optimal attitude manoeuvres is available including both the inputs and outputs, such that supervised learning can be applied. In fact, it was proven by Kurkova (1992) that an ANN with a single hidden layer can approximate any function. This statement is referred to as the universal approximation theorem. Nevertheless, this may require a very large network, which is difficult to train for good performance. Therefore, it can still be beneficial to use less hidden layers instead. The remainder of this section will introduce the method to train the ANN. A general introduction about training of ANNs is given in Section 5.4.1, while specific related theory behind the used training methods in explained in Sections 5.4.2 and 5.4.3.

5.4.1. Training of an artificial neural network

The goal of training an ANN is to optimize the weights and biases of the neural network, such that the model can accurately model the provided dataset. This process is also referred to as *backpropagation*, since the output are traced back in the ANN. In contrast, the *feedforward* process, such as in Figure 5.5, determines the output from the inputs. In practice, the training of the ANN is done by minimizing the sum of the squares of the errors:

$$J(\mathbf{y}) = \sum_i e_i^2 = \mathbf{e}^T\mathbf{e} \tag{5.8}$$

where e_i is the difference between a given data point and the output of ANN, $\mathbf{y}$ is the vector of optimization variables and J is the objective. Considering that the network can have thousands of weights and biases which can be optimized, and the dataset contains thousands of samples, this optimization objective explains why training an ANN is typically computationally expensive. Also, it is not the intention to find the global minimum, since this solution would result in *overfitting*, as explained in the next paragraph. Therefore, the gradient-based methods which can quickly find a local optimum are preferred above meta-heuristics such as a genetic algorithm. Several gradient-based methods popular for back-propagation are the stochastic gradient descent, adaptive momentum estimation and the Levenberg-Marquardt method. The Levenberg-Marquardt algorithm is typically the best choice when the neural network is used as a fitting function (Hagan et al., 2014). The theoretical introduction of the theory behind this algorithm network is given in Section 5.4.2.

Another main concern of designing a neural network is selecting an appropriate number of hidden layers and neurons per layer. If the network does not contain enough neurons, the model will *underfit* the data and can not model the full complexity of the data. On the other hand, *overfitting* can occur if too many neurons or hidden layers are selected. It that case, the model can properly model the training samples, but does not generalize well for new samples. For that reason, ANNs are always compared to validation data, for which the model was not trained. A way to prevent overfitting is making use of regularization algorithm. In this way, the optimal number of neurons requires less tuning, since the algorithm will attempt to reduce the number of neurons in case overfitting occurs. The regularization method used for this research is explained in Section 5.4.3.

5.4.2. Levenberg-Marquardt

The Levenberg-Marquardt algorithm is a modification of Newton's method and is applied to minimize functions that are sums of squares of non-linear functions. As a result, this section shows several similarities with the discussion of Newton's method in the context of NLP and will re-use some concepts as introduced in Section 4.1.1. The new elements are adopted from (Hagan et al., 2014, p. 431-437).

The Levenberg-Marquardt method is a well-known technique for fitting non-linear functions and has therefore also been successfully applied within the ANN framework. The method is particularly well

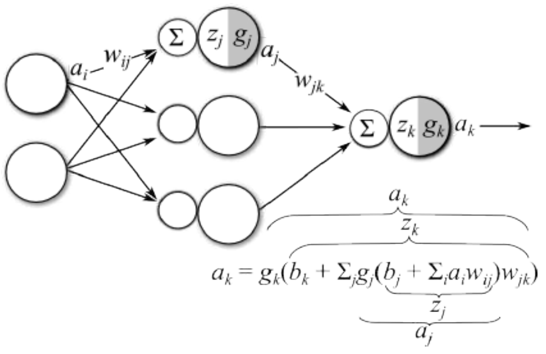


Figure 5.5: Example of the feedforward process of an artificial neural network for two layers.

suited for neural network training, for which the performance index is the sum squared error, as shown in Equation 5.8. Taking the gradient of this equation for parameter j:

$$\nabla J(\mathbf{y})_j = 2\sum_i^N e_i(\mathbf{y})\frac{\partial e_i(\mathbf{y})}{\partial y_j} \tag{5.9}$$

Next, the Jacobian matrix $\mathbf{G}(\mathbf{y})$ is defined as:

$$\mathbf{G}(\mathbf{y}) = \begin{bmatrix} \frac{\partial e_1(\mathbf{y})}{\partial y_1} & \frac{\partial e_1(\mathbf{y})}{\partial y_2} & \cdots & \frac{\partial e_1(\mathbf{y})}{\partial y_n} \\ \frac{\partial e_2(\mathbf{y})}{\partial y_1} & \frac{\partial e_2(\mathbf{y})}{\partial y_2} & \cdots & \frac{\partial e_2(\mathbf{y})}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_N(\mathbf{y})}{\partial y_1} & \frac{\partial e_N(\mathbf{y})}{\partial y_2} & \cdots & \frac{\partial e_N(\mathbf{y})}{\partial y_n} \end{bmatrix} \tag{5.10}$$

such that Equation 5.9 can be written in matrix form:

$$\nabla J(\mathbf{y}) = 2\mathbf{G}^T(\mathbf{y})\mathbf{e}(\mathbf{y}) \tag{5.11}$$

Furthermore, the elements of the Hessian matrix $\mathbf{H}$ can be found to be:

$$\mathbf{H}(\mathbf{y}) = \nabla^2 J(\mathbf{y})_{k,j} = 2\sum_{i=1}^N \left[\frac{\partial e_i(\mathbf{y})}{\partial y_k}\frac{\partial e_i(\mathbf{y})}{\partial y_j} + e_i(\mathbf{y})\frac{\partial^2 e_i(\mathbf{y})}{\partial y_k \partial y_j}\right] \tag{5.12}$$

Again writing this in matrix form gives the following:

$$\nabla^2 J(\mathbf{y}) = 2\mathbf{G}^T(\mathbf{y})\mathbf{G}(\mathbf{y}) + 2\mathbf{S}(\mathbf{y}) \tag{5.13}$$

for which

$$\mathbf{S}(\mathbf{y}) = \sum_{i=1}^N e_i(\mathbf{y})\nabla^2 e_i(\mathbf{y}) \tag{5.14}$$

The second-order term $\mathbf{S}(\mathbf{y})$ is typically small and therefore it is reasonable to approximate the Hessian as

$$\nabla^2 J(\mathbf{y}) \cong 2\mathbf{G}^T(\mathbf{y})\mathbf{G}(\mathbf{y}) \tag{5.15}$$

Now that the gradient and Hessian matrices are obtained, the method for unconstrained optimization using Newton's method can be applied. The resulting algorithm from Section 4.1.1 was:

$$\mathbf{y}_{i+1} = \mathbf{y}_i - \mathbf{H}^{-1}\mathbf{g} \tag{5.16}$$

When substituting the gradient and approximate Hessian, one obtains the following algorithm:

$$\mathbf{y}_{j+1} = \mathbf{y}_i - \left[\mathbf{G}^T(\mathbf{y}_i)\mathbf{G}(\mathbf{y}_i)\right]^{-1}\mathbf{G}^T(\mathbf{y}_i)\mathbf{e}(\mathbf{y}_i) \tag{5.17}$$

This is in fact the *Gauss-Newton* method to find a local optimum. Compared to Newton's method, no second derivative information is required anymore. However, one drawback is that the approximated Hessian may not be invertible. For that reason, the Levenberg-Marquardt method introduces an adjustment to the approximate Hessian:

$$\mathbf{H} \cong 2\mathbf{G}^T(\mathbf{y})\mathbf{G}(\mathbf{y}) + \mu\mathbf{I} \tag{5.18}$$

where **I** is the identity matrix and μ a tuning parameter. Finally, this modification now results in the Levenberg-Marquardt algorithm:

$$\mathbf{y}_{j+1} = \mathbf{y}_i - \left[\mathbf{G}^T(\mathbf{y}_i)\mathbf{G}(\mathbf{y}_i) + \mu_i\mathbf{I}\right]^{-1}\mathbf{G}^T(\mathbf{y}_i)\mathbf{e}(\mathbf{y}_i) \tag{5.19}$$

The particular strength of this algorithm is that it efficiently makes use of both the speed of Newton's method and the convergence properties of the steepest descent method. When the parameter μ goes to zero, it becomes the Gauss-Newton method and if μ is large, it resembles the steepest descent method with small learning rate. Typically, the algorithm starts off using a small value ($\mu = 0.005$ will be used). If a step does not yield a smaller value for the objective function, the step is executed again, while the parameter μ is multiplied by a factor (e.g. 10 for the applied algorithm). Since μ is increased, the new attempt will have better convergence properties, due to the characteristics of the steepest descent method.

5.4.3. Bayesian regularization

The neural network training using the Levenberg-Marquardt algorithm intends to minimize the sum of the squared errors. However, as pointed out in Section 5.4.1, that does not always imply that the network also performs well on new input data. In order to improve the generalization of the network, regularization restricts the magnitude of the weights between the neurons. By minimizing the network weights w_i, the response of the network will be smoother and more generalized. Furthermore, Bayesian methods will be applied to optimize this regularization.

For regularization, the objective function is modified to:

$$J(\mathbf{y}) = \beta\sum_i e_i^2 + \alpha\sum_i w_i^2 \tag{5.20}$$

Compared to Equation 5.8, the performance index is now changed to a cost function with two weighted objectives. The parameters α and β will scale the relative importance of both terms. When $\alpha << \beta$, the mean squared errors are reduced and if $\beta << \alpha$, the weights are reduced and the network output becomes smoother. Of course, the problem arises now to select the appropriate values for α and β, such that a good combination of both objective terms is created. The approach is to use Bayes' theorem to obtain the optimized values. The weights **w** are taken as stochastic variables. A given probability density function of the weights **w** can be updated using new data D as follows (MacKay, 1991):

$$P(\mathbf{w}|D,H) = \frac{P(D|\mathbf{w},H)P(\mathbf{w}|H)}{P(D|H)} \tag{5.21}$$

Here $H(\alpha,\beta)$ is the hypothesis model, being the neural network. The term $P(\mathbf{w}|H)$ is called the prior and represents the probability distribution of the weights without the information of the new data samples. On the other hand, the posterior $P(\mathbf{w}|D,H)$ is the probability distribution of **w** after considering the new data D. $P(D|\mathbf{w},H)$ is the prediction by the model for the new data, given the prior distribution for **w**. Lastly, $P(D|H)$ is a normalization factor, which makes sure that the total posterior probability is 1. The posterior should be maximized, such that the maximum likelihood for the vector **w** if found. This is equal to finding the minimum of the objective function. A derivation to find the regularization parameters from the maximized posterior is given by (Foresee and Hagan, 1997). The optimal parameters are found to be:

$$\alpha^\star = \frac{\gamma}{2\sum_i w_i^{\star 2}} \tag{5.22}$$

$$\beta^\star = \frac{n-\gamma}{2\sum_i(e_i(\mathbf{w}^\star))^2} \tag{5.23}$$

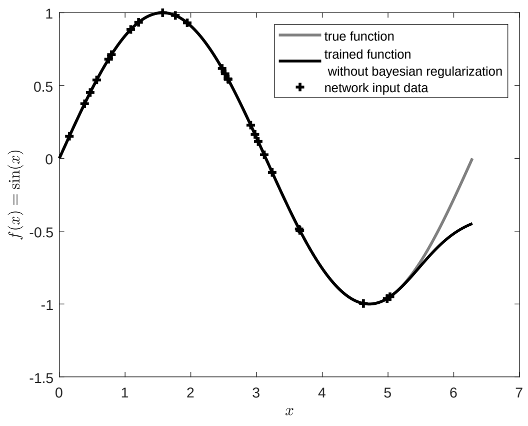


Figure 5.6: ANN function fitting without regularization

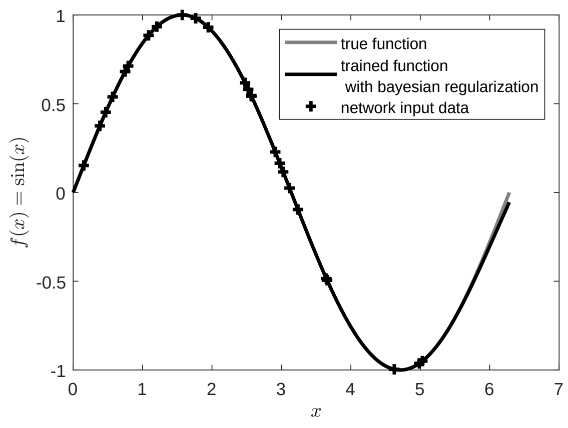


Figure 5.7: ANN function fitting with regularization

where $\mathbf{w}^\star$ is the maximum likelihood estimate of the weights and γ is the effective number of parameters. The latter is an indication of the number of parameters in the network that effectively contribute to the sum of squared errors reduction. If the number γ equals the number of parameters used in the network, it signifies the network may not be large enough to model all interactions. Vice versa, a smaller network may be better if γ is smaller than the number of used parameters. One can find γ as:

$$\gamma = N - 2\alpha^\star \text{trace}\left(\mathbf{H}^\star\right)^{-1} \tag{5.24}$$

Here, N is total number of optimization variables and $\mathbf{H}^\star$ is the Hessian matrix for $\mathbf{w}^\star$. It can be seen that the Bayesian regularization has superior generalization properties when comparing Figures 5.6 and 5.7. This example demonstrates how the regularized network is able to generate a more generalized function and therefore would require less samples to achieve the same performance as the network without regularization. In general, regularization also adds robustness to the training process, since a small change in the number of neurons can have a big impact on the performance of the network if it is trained without regularization. The major downside of adding the Bayesian regularization, is the increased computation time. However, the relative added computational expense becomes small when the network size becomes larger.

5.5. Results

This section contains the results of the approximation of the optimal manoeuvres duration using ANNs. As is established in Section 5.1, the database consists of nine parameters, of which two are binary parameters. However, the input parameters to an ANN for function fitting are not allowed to be binary. Therefore, a separate ANN is generated for each possible combination of the binary parameters. This means that in total four ANNs for the remaining seven input parameters are required. The results for this approach are shown in Section 5.5.1. A second method, which further improved these results by combining 12 ANNs, is documented in Section 5.5.2. The results presented in these sections do not discuss the detailed influence of the nine parameters on the optimal slew time, since this is not relevant for the scheduling algorithms. In order to still give some insight about the influence of the parameters, some visualizations of the nine dimensional database are shown in Appendix A.

5.5.1. Single neural network

To generate the ANN results, it was decided after a trial and error that two hidden layers of 50 neurons can appropriately model the complexity of the database, while not leading to overfitting. Furthermore, 150,000 random samples are generated for each scanning direction option. From this, 5% of the samples is used as validation and the remaining samples are given as training data to the ANN. The generation of this input data required approximately five days computational time on an Intel i5 4670 CPU, using all four cores. Furthermore, the training of each ANN was allowed 12 hours CPU time, while using parallel computations on all cores. The forward and backward scanning directions are indicated by indices F and B respectively. An example of the training progress for ANN$_{FF}$ is shown in Figure 5.8.

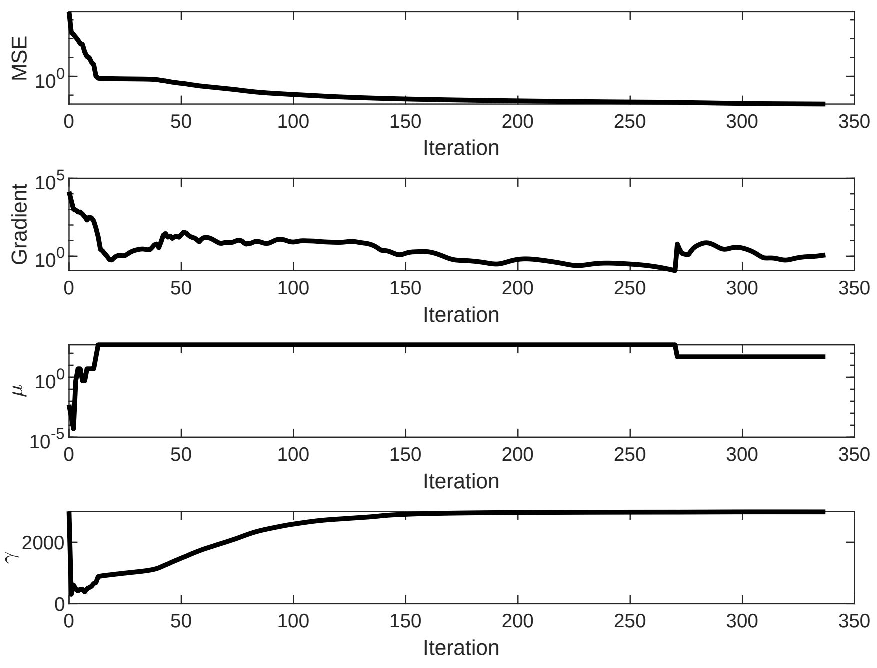


Figure 5.8: Progress of ANN training of ANN$_{FF}$

Furthermore, this example demonstrates that the Bayesian regularization progressively increases the number of effective parameters γ, which contributes to a stable convergence process. Also, μ starts off very small, corresponding to the Gauss-Newton method. When the algorithm continues, μ increases and therefore has the properties of the steepest descent method.

The used performance indicators for the ANN results are the mean squared error (MSE), the standard deviation of the errors σ_e and the Pearson correlation coefficient R. The latter gives an understanding about the correlation between the ANN results and the input data, where a perfect match would result in a value of 1. The coefficient R can be computed as follows (Hagan et al., 2014):

$$R = \frac{n \sum \bar{y}_i y_i - \sum \bar{y}_i \sum y_i}{\sqrt{n \sum \bar{y}_i^2 - \left(\sum \bar{y}_i\right)^2} \sqrt{n \sum y_i^2 - \left(\sum y_i\right)^2}} \tag{5.25}$$

Here, the output from the ANN is $\bar{y}$ and y is the output from the database.

The performance of the resulting ANNs for the validation samples is shown in Table 5.2. As expected, the performance for all four ANNs is comparable. The slight differences can be explained based on the difference in the slew manoeuvres due to the scanning direction and the randomness in the generated data. Also, the back-propagation always solves for a local optimum (see Section 5.3), which starts from a random solution such that the solution is different for each run. The errors are computed as $e = y - \bar{y}$ and follow a Gaussian distribution as can be seen in Figure 5.9. Nevertheless, this result should be interpreted critically as discussed in the next paragraph.

It can be seen in Figure 5.10 that the prediction for longer slew manoeuvres is remarkably better than for short ones. This is caused by the higher non-linear dependency of the optimal slew time on the input parameters for these manoeuvres, making them more difficult to be modelled by the ANN. Unfortunately, the scheduling algorithm often prefers the use of short manoeuvres such that more targets can be observed. An approach to tackle this issue is explained in Section 5.5.2.

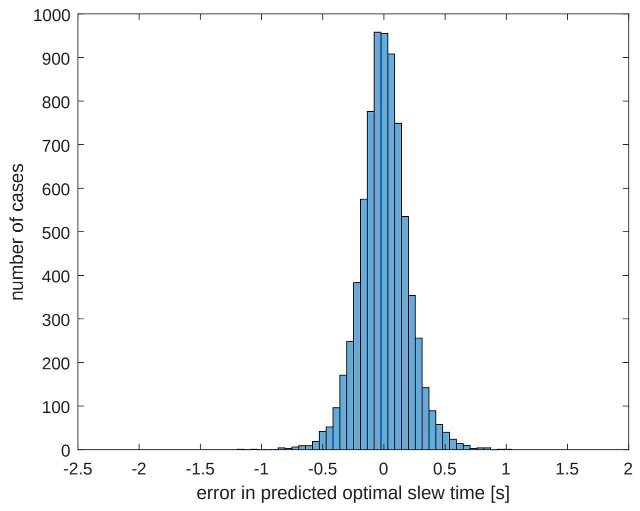


Figure 5.9: Histogram of the errors when using the trained ANN_{FF} model on the validation data.

Table 5.2: Performance indicators for the four trained ANNs, given the validation data.

ANN	MSE [s]	σ_e [s]	R
ANN$_{FF}$	0.0451	0.2126	0.9991
ANN$_{FB}$	0.0332	0.1829	0.9995
ANN$_{BF}$	0.0369	0.1926	0.9995
ANN$_{BB}$	0.0442	0.2099	0.9992

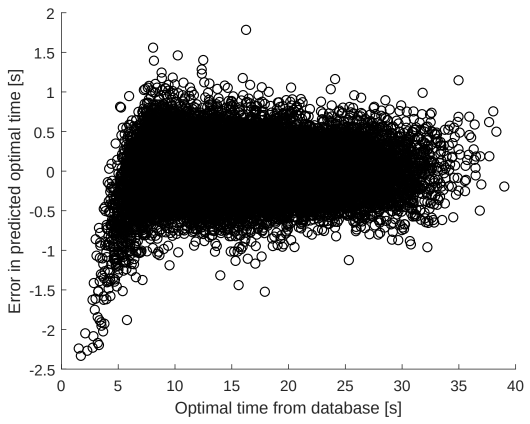


Figure 5.10: Error of the predicted optimal time as obtained with the ANN_{FF} model for the validation data.

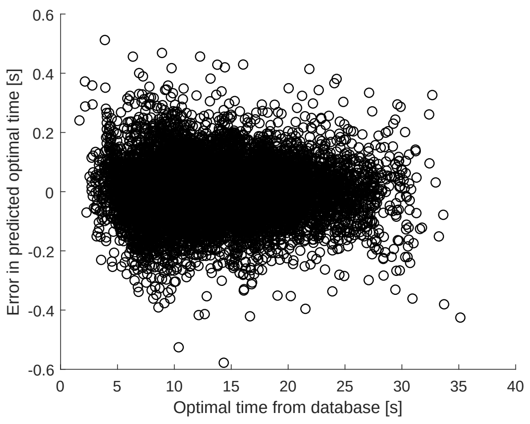


Figure 5.11: Error of the predicted optimal time when combining the three overlapping models $ANN_{FF,1}$, $ANN_{FF,2}$ and $ANN_{FF,3}$ for the validation data.

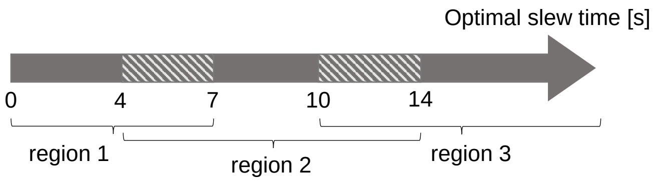
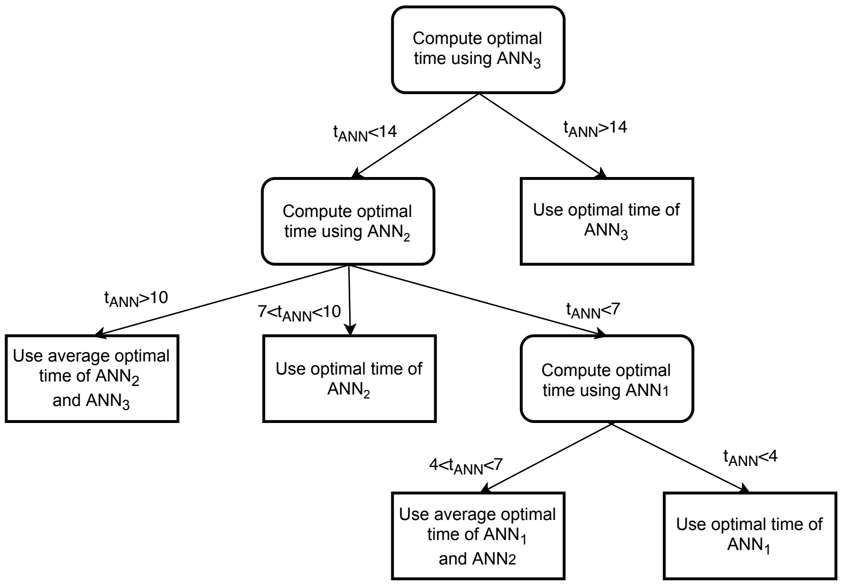


Figure 5.12: Definition of the overlapping regions for separating the ANN training data.



Compute optimal time using ANN3
tANN<14
tANN>14
Compute optimal time using ANN2
Use optimal time of ANN3
tANN>10
7<tANN<10
tANN<7
Use average optimal time of ANN2 and ANN3
Use optimal time of ANN2
Compute optimal time using ANN1
4<tANN<7
tANN<4
Use average optimal time of ANN1 and ANN2
Use optimal time of ANN1

Figure 5.13: Flow diagram to obtain the approximate manoeuvre duration using three ANNs.

5.5.2. Overlapping neural networks

In order to enhance to performance of the fitting using the ANN, it was attempted to combine several ANNs which are trained for specific subsets of the dataset. This offers the advantage that the demonstrated complex modelling of short manoeuvres can be alleviated (see Section 5.5.1). However, the question arises now how the data should be separated and how the ANNs should operate together to give the approximate manoeuvre time. It was decided to simply separate the data based on the optimal time given in the database. In total, three sets of data are defined, which is illustrated in Figure 5.12. Note that the dataset which corresponds to the second region has an overlap with the data from the first and third region.

The second question to answer is how the ANNs should be combined. It was attempted to use classifying machine-learning methods, such as SVMs and ANNs. However, these classifiers could only determine for 98% of the cases correctly in which region the optimal time would be given the input parameters. It was determined to be unacceptable that the optimal time would be determined using the incorrect ANN for the remaining 2% of the cases as this could lead to larger errors than presented in Section 5.5.1. Fortunately, as discussed in Section 5.4.3, the applied Bayesian regularization makes sure that the ANN is properly generalized and can perform well on extrapolation for a limited range outside of the range of the training samples. It was identified that this property is excellent to combine the ANNs using the overlapping regions. More specifically, the optimal times are now obtained through the process as shown in the flow diagram in Figure 5.13. First, an optimal time t_{ANN} is obtained by the ANN$_3$ model which is trained for the data in region 3. The result t_{ANN} is now used to determine whether to use the ANN$_3$ model, or also consider the result of the ANN$_2$ model. If t_{ANN} is in the overlap region or outside the trained region, the slew time corresponding to the ANN$_2$ model is also calculated. For the overlap region, the average result of both models is used. This process is again followed for the result of the ANN$_2$ model.

Table 5.3: Performance indicators for the twelve trained overlapping ANNs, given the validation data for each data subset.

ANN	**MSE[s]**	**σ_e [s]**	**R**
ANN$_{FF,1}$	0.0087	0.0937	0.9959
ANN$_{FF,2}$	0.0160	0.1257	0.9988
ANN$_{FF,3}$	0.0110	0.1044	0.9997
ANN$_{FB,1}$	0.0071	0.0843	0.9967
ANN$_{FB,2}$	0.0147	0.1212	0.9989
ANN$_{FB,3}$	0.0103	0.1013	0.9997
ANN$_{BF,1}$	0.0071	0.0844	0.9967
ANN$_{BF,2}$	0.0144	0.1200	0.9989
ANN$_{BF,3}$	0.0100	0.0999	0.9998
ANN$_{BB,1}$	0.0088	0.0939	0.9959
ANN$_{BB,2}$	0.0156	0.1250	0.9988
ANN$_{BB,3}$	0.0109	0.1043	0.9997

Since three regions are defined for all ANNs, in total 12 ANNs have to be trained now. Again two hidden layers of 50 are used. Furthermore, the computational time limit was set such that all networks are trained within 2 days. For the same dataset as in Section 5.5.1, the results are shown in Table 5.3. The accuracy of all separate ANNs outperforms the results without the overlap regions as presented in Section 5.5.1. More importantly, as shown in Figure 5.11, the larger errors for short attitude manoeuvres have been eliminated.

Now that the optimal attitude manoeuvre time can be reasonably well approximated, the last note of this chapter explains how this approximation is implemented in the scheduling algorithms. From the above results, it was found that all optimal times are obtained within 0.5 second of their true value, with 3σ significance. However, whenever the optimal time is underestimated, this can result in an infeasible schedule. This should be avoided as much as possible, since the scheduling algorithm needs to be executed again for those cases. Therefore, an error margin ϵ of 0.5 seconds is introduced, which is added to the result of the ANN models. Because of this error margin, all optimal times are found within 1 sec, centered around an average of 0.5 second prediction error. Another important conclusion is the required CPU time to obtain this approximate manoeuvre time using the ANNs. The overlapping ANNs are able to perform around 200,000 function evaluations per second, such that the approximate optimal manoeuvre time is obtained about one million times faster than the solution from the OCP. Consequently, the ANN approximation was found to be suitable for implementation in the scheduling algorithms.

6 Optimal satellite observation scheduling

This chapter concerns the selection and scheduling of the observation requests of the customers. This is the high-level optimization problem that needs to be solved for the thesis work. Firstly, the input as given to the schedule optimization algorithm is described in Section 6.1. Next, the assumptions and mathematical formulation of the scheduling problem are given in Section 6.2. The preferred algorithmic concepts to solve that problem are concluded in Section 6.3. The consecutive Sections 6.4, 6.5 and 6.6 present the theory of the developed scheduling algorithms. This chapter is concluded by the results in Section 6.7.

6.1. Input description for the scheduling algorithms

Prior to explaining the scheduling problem at hand, Section 6.1.1 places the activity planning in the context of the ground operations and the customer input. Next, the approach to find the visibility window of the targets is discussed in Section 6.1.2.

6.1.1. Customer input

The Earth observation satellite is operated by a mission operations center. As mentioned in Chapter 1, the image requests are on customer demand. Therefore, the center receives a continuous incoming stream of image requests. All these user requests must be collected and processed such that a precise activity plan for the satellite can be uploaded on a regular basis, e.g. every day. It is assumed that the system does not allow for on-board re-planning of the schedule, which is typically the case in practice. After the observation is performed by the satellite and successfully received by the ground station, the center also processes the data before sending it to the users. This process is illustrated in Figure 6.1.

Furthermore, the planning of the activity plan, which is the central topic here, consists of three major phases:

1. the customer requests are collected, cut in stripes that correspond to the swath width of the satellite and put in a database
2. from the observation requests, an optimization algorithm produces a satellite activity plan that keeps into account all system constraints

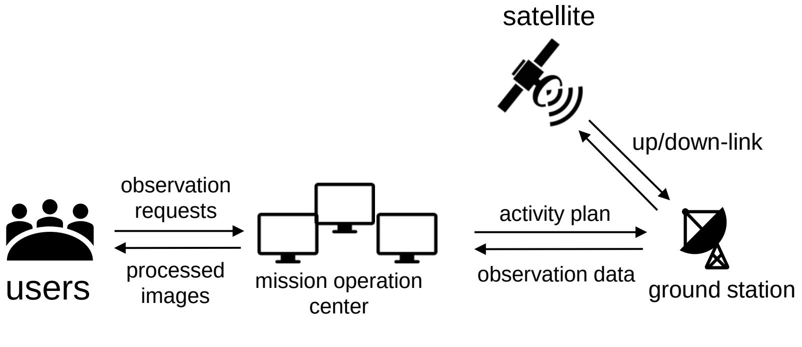


Figure 6.1: Operational architecture of an Earth observation satellite.

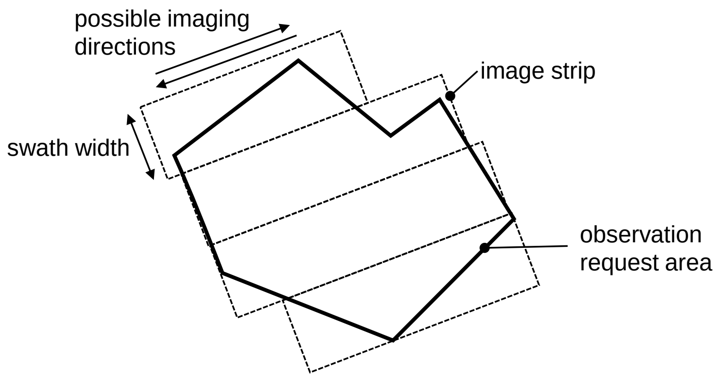


Figure 6.2: Observation request decomposition in image strips of unequal length.

3. the activity plan is validated, eventual schedule conflict are resolved and the plan is uploaded to the satellite

The first phase is discussed in the remainder of this section, while the rest of this chapter continues with phase two. An example of the validation of the activity plan (phase three), is described in Chapter 8.

Now, when the customer requests have been collected, these can consist of different image products, such as custom areas to scan or stereoscopic images. However, the process of cutting the requested area as shown in Figure 6.2 will not be treated. It is assumed for this work that all requests can be covered by a single image strip of a fixed length. The extension of these image strips to images of areas is a geometric problem that is not of interest for this thesis research. Next, also stereoscopic images are not considered, because the time between two stereoscopic scans is considered small, such that no other observations can be scheduled in between. In conclusion, single image strips of fixed length are the only image product to be considered for this research.

Furthermore, before going to the scheduling problem, it is important to define what the input parameters of the observation requests for the image strips should look like. It is determined that the following entries are required for each customer request:

- an identification number
- time window of visibility of the targets
- a geographic area specification, which is defined by a set of coordinates for the center of the image (longitude, latitude and altitude on WGS84 ellipsoid)
- a priority depending on the authority, price paid by the customer, urgency,…

Here, the method to determine the time window for visibility is established in Section 6.1.2.

For the purpose of testing and validating the activity planner, also an observation request simulator was designed. The target requests are simulated to be randomly distributed on the Earth surface. All targets are assigned a priority factor as explained in Section 6.2.2. In 10% of the cases, the target is of high priority, 30% of the cases is considered as medium priority and the remaining targets are of lowest priority. Also, in order to make the target request distribution more realistic, 90% of the targets are on land, while the remaining 10% of the targets is situated in the oceans. Consequently, large gaps exist over the oceans, where the density of targets is far less than over land, as illustrated in Figure 6.3. For these cases, the sequence of targets is interrupted, such that the next sequence can be considered as an independent optimization problem. For all scheduling algorithms, a sequence is defined as the series of subsequent targets which can be seen by the satellite, such that there always exists a slew manoeuvre to each next target that is shorter than the maximum slew duration. The maximum slew duration was determined to be 60 seconds (see Section 5.1).

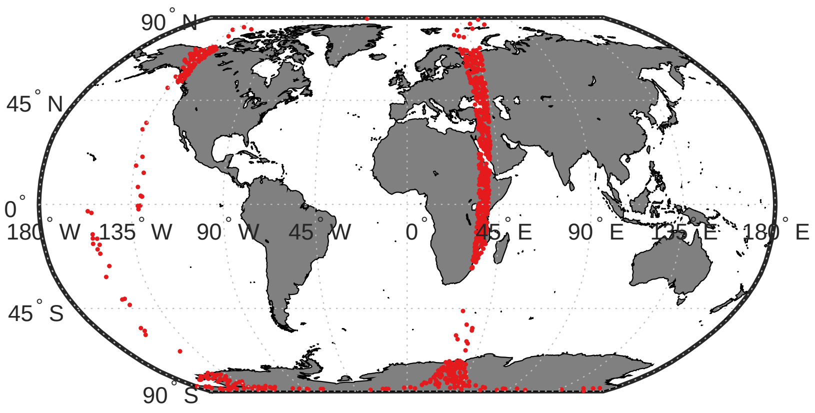


Figure 6.3: Example of randomly generated targets as visible for the satellite during one orbit.

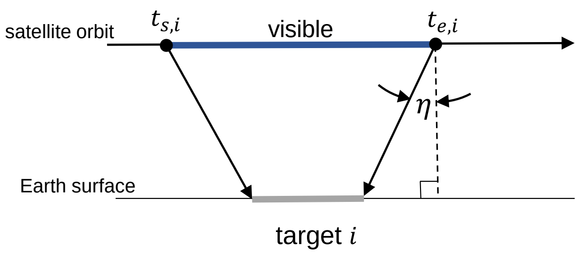


Figure 6.4: Time window based on the maximum off-nadir at which the target is visible.

6.1.2. Visibility window determination

After the customer input is received or generated, the time windows should be determined for which the targets are visible for the satellite. Since the planning horizon is never more than one orbit, the time window can be defined by two parameters: the start epoch t_s and the final epoch t_e at which the target is visible by the satellite. The angular constraints as presented in Section 2.2 will define the time windows when a target can be seen by the satellite. More specifically, the maximum off-nadir angle $\eta_{\max}$ is used, but other constraints on the relative angular position of the target could be imposed as well. The angle η can be obtained using the following dot product:

$$\mathbf{r}_{s/t} \cdot \mathbf{r}_s = \left\|\mathbf{r}_{s/t}\right\| \left\|\mathbf{r}_s\right\| \cos\eta \tag{6.1}$$

Whenever it holds that $\left\|\mathbf{r}_{s/t}^{eci}\right\| < \left\|\mathbf{r}_s^{eci}\right\|$ and the off-nadir angle does not exceed the limit, the target is visible for the satellite. This extra condition is necessary, since the off-nadir can also become smaller than the limit at the opposite side of the Earth.

Now that the condition for visibility is established, the second step is to determine the exact time windows when the target is visible for the satellite. Since the position of the satellite is required, the orbit is propagated for the time interval when the targets are observed. The specific satellite position at each epoch is found through interpolation. Using these positions, the off-nadir angles as a function of time can be found as shown in Figure 6.5. Finally, the intersections with the maximum off-nadir line determine the time window for target visibility. The intersections are obtained by the Matlab built-in root-finding algorithm. Note that it also possible to calculate an analytical solution when a Keplerian orbit is assumed. However, in this research also the J_2 perturbation is included, as mentioned in Section 3.2.

6.2. Scheduling problem modelling

This section provides the simplifying assumptions and mathematical problem statement in Sections 6.2.1 and 6.2.2 respectively.

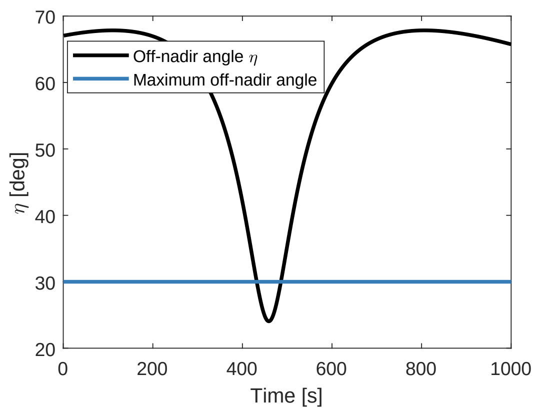


Figure 6.5: Examples of the change over time of the off-nadir angle of a target as seen by the satellite.

6.2.1. Assumptions

Taking in mind the complexities of the problem as explained in Section 2.1, two assumptions are introduced here which considerably reduce the search space without a big impact on the quality of the results. The first assumption concerns the dependency of the optimal slew time on the time of observation, as was demonstrated in Figure 2.1.

Assumption 1. *Take targets T_i and T_j, where target T_i is observed at epoch t_i. The minimum transition time from T_i to T_j is d_{min} and $t_i + d_{\text{min}} \geq t_s(T_j)$. It is assumed that it is always beneficial to observe target T_j at the earliest opportunity time $t_j = t_i + d_{\text{min}}$, instead of waiting for a later observation time.*

This is an important assumption, in the sense that the exact optimal time of scheduling does not need to be determined anymore. That is, the optimal time of observation directly follows from the previous time t_i and the optimal manoeuvre time at t_i to the next target. As a consequence, the first target of the sequence is observed at the start of its visibility window. The remaining scheduling problem only consists of finding the optimal selection and sequence.

This assumption can be justified by a numerical experiment. Consider Figures 6.6 and 6.7, which show the dependency of the slew time on the time of observation in the visibility window for two cases. The first case is the slew between a target T_i which is located at zero off-nadir angle w.r.t. the satellite at time t_i and a target T_j which is also at zero off-nadir angle at time t_j, such that $t_j - t_i = 60$ seconds. Both targets are scanned in the forward direction. From Figure 6.6, it is clear that the possible manoeuvre time does not change quickly with the time t in the visibility window. Going back to the assumption, in fact it would only be beneficial to wait for a later observation time if $\frac{\partial d_{\text{min}}}{\partial t} > 1$. This is never the case, such that the assumption is valid for this example. The second case considers the slew such that the same target is scanned twice in the same direction. In fact, the resulting optimal slew for the case if the first target is observed at zero off-nadir angle was already presented in Section 4.8. Looking at Figure 6.7, another interesting observation is that the result is not a continuous function anymore. This a consequence of using multiple overlapping ANNs for estimating the optimal time, as explained in Section 5.5.2. Even more, discontinuities could also occur, since more than one hidden layer is used for the ANN. These deep neural networks can model discontinuous functions (Hagan et al., 2014), such that the resulting model does not always have a smooth behaviour. A second observation about Figure 6.7 is that the slew time becomes shorter when the target is directly underneath the satellite. This goes against the intuitive idea that a larger angular difference will result in a longer slew duration. Nevertheless, the difference in required angular velocity to scan the target is smaller if the target is at zero off-nadir angle. For this small slew, the angular velocity difference has a larger influence on the optimal slew time than the angular difference itself.

From the above examples, it is clear that Assumption 1 holds for almost all imaginable cases. Nevertheless, this assumption must always be reconsidered carefully. For example, a more complex objective function could make this assumption invalid, as further analyzed in Section 6.7.5. Also, if the

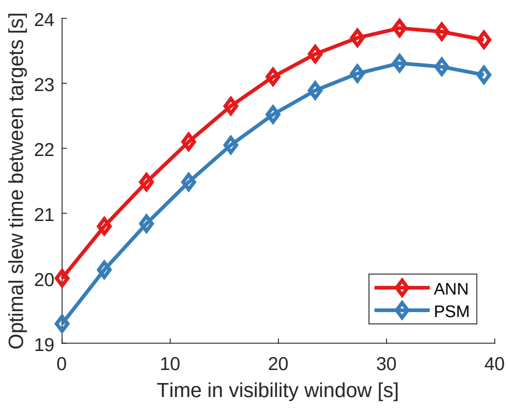


Figure 6.6: Dependency of the optimal slew time, as obtained using the ANN and PSM method, on the time of observation within the availability window, case 1. The time of acquiring the initial time in the visibility window is used.

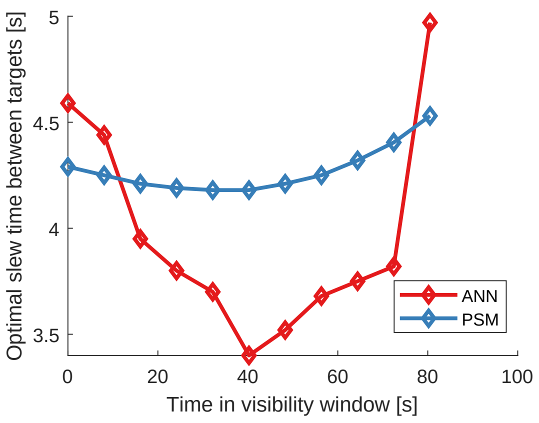


Figure 6.7: Dependency of the optimal slew time, as obtained using the ANN and PSM method, on the time of observation within the availability window, case 2. The time of acquiring the initial time in the visibility window is used.

satellite is less agile than in this research, this assumption could not accurate anymore as the angular difference between the targets will results in a larger difference in slew times.

Assumption 2. *Take a sequence of targets which are observed one after another. It is assumed that it is never beneficial to change the azimuth scanning direction for subsequent observations for this sequence.*

This assumption is justified based on the results in Section 6.7.4. There, it is shown that a non-zero change in azimuth scanning direction improves the scheduling performance by a margin which is too small for the required added computational time of the algorithm. It is important to realize here that the difference of the azimuth scanning direction is zero, but the value $\alpha_{s,0}$ of the scanning direction, which is now constant for the complete sequence, can still take any value. The value of this angle is again a decision variable. However, the angle will be chosen based on the following modes in which the satellite can operate:

1. **Fast acquisition mode:** a mode where the variable $\alpha_{s,0}$ is selected such that the targets are acquired as fast as possible on average. This is achieved by taking $\alpha_{s,0} = 0$. A more detailed analysis of the influence of the angle $\alpha_{s,0}$ on the scheduling performance is shown in Figure 6.9. There it can be seen that if $\alpha_{s,0}$ is approximately zero, more targets can be observed. In this specific example, the most targets could be observed if $\alpha_{s,0} = 9$ degrees. Therefore, it is important to realize that the fast acquisition mode does not guarantee that the most targets are observed. Especially if the sequence of targets is very short, better values for $\alpha_{s,0}$ may exist. This could be solved by considering $\alpha_{s,0}$ as another decision variable in the scheduling algorithm. Nevertheless, the small possible improvement was not considered worth the added computational time.

2. **Energy efficient mode:** a mode where the variable $\alpha_{s,0}$ is selected such that the energy intake during the observation sequence is maximized. This can be done through geometry by finding the angle $\alpha_{s,0}$ such that the incidence angle of the sunlight on the solar panels is optimized. Inevitably, this can lead to lower quality scheduling results. For example, if angle $\alpha_{s,0} = 60$ degrees is energy-wise optimal, almost 10% less targets can be observed in the case presented in Figure 6.9. Do note that the incidence angle of the Sun on the solar panels depends on two angles. One of these angles can be optimized by choosing $\alpha_{s,0}$, but the other angle depends on the pointing angle of the satellite to the target. The energy production could be further optimized by including an additional term in the objective function of the scheduling, but this will not be considered.

Both observation modes are illustrated in Figure 6.8. Note that the slew time is shorter for the energy efficient mode than for the fast acquisition in this example. This demonstrates that the fast acquisition

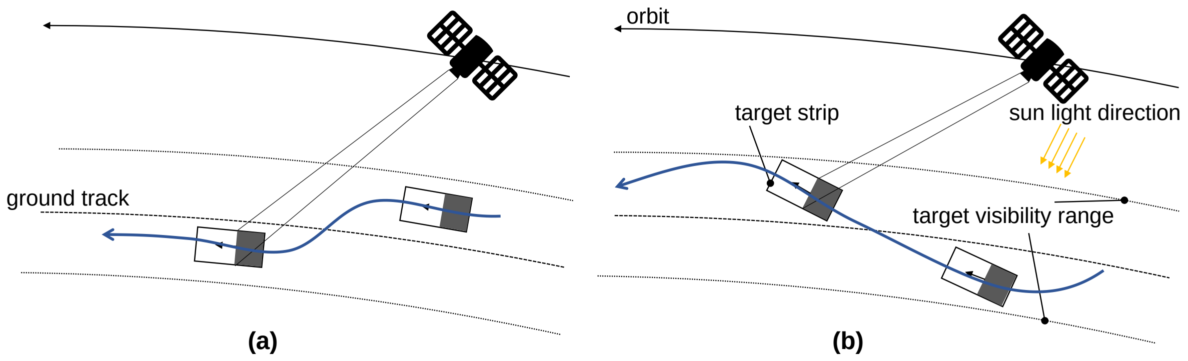


Figure 6.8: Two different satellite observation operational modes: **(a)** fast acquisition mode and **(b)** energy efficient mode

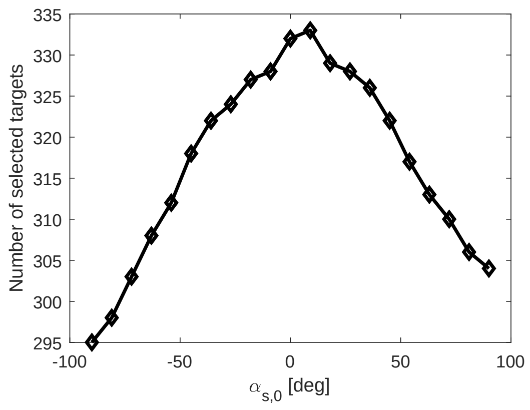


Figure 6.9: Influence of the angle $\alpha_{s,0}$ on the number of targets which be observed. The results are generated using the DP(A) algorithm from Section 6.5.2, with a total of 800 targets randomly distributed over one orbit using the simulator as given in Section 6.1.1

will only result in faster slew on average, but certain slews will be shorter using the energy efficient mode. For this research, the fast acquisition mode was always selected, since the energy constraints were not considered.

6.2.2. Mathematical problem statement

For an agile satellite, both the *selection* and *sequence* determination of the observations needs to be performed. The selection of observations is required since this concerns an oversubscribed scheduling problem, which does not guarantee that all requests will be executed. This leads to the following decision variables for the optimization problem:

$$x_i = \begin{cases} 1, & \text{if target } i \text{ is selected for observation;} \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

$$s_i = \begin{cases} 0, & \text{if target } i \text{ is scanned in forward direction} \\ 1, & \text{otherwise} \end{cases} \tag{6.3}$$

$$y_{ij} = \begin{cases} 1, & \text{if target } j \text{ is selected for observation directly after target } i \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$

The definition of the scanning directions of the decision variable is given in Section 2.1. Furthermore, the exact time of target observation is not a decision variable due to Assumption 1. For each possible image acquisition, a decision variable x_i and s_i exists. Taking n_t as the number of targets, also n^2 different y_{ij} decision variables are introduced.

The objective of the satellite observation scheduling is to plan the image requests such that as many images as possible are observed, while taking into account a priority factor. If several sequences exist with equal objective value, the preference is given to the sequence which is completed first. In more mathematical terms, the objective J is defined as:

$$J = \max\left(\phi(t_f) + \sum_{i=1}^{n_t} x_i p_i\right), \ \forall i \in T \tag{6.5}$$

Here x_i is a decision variable as explained above, p_i is a priority factor for each target and T is the set of all possible targets. In total, three priority levels will be used. The priority factor according to each level are the values 1,2 and 5, where the highest value is attributed to the highest priority. In other words, a target of highest priority has an equal value as five low-priority targets. Also, to give preference to the faster sequence if the total profit of several sequences is the same, the final time objective $\phi(t_f)$ is introduced:

$$\phi(t_f) = 1 - \frac{t_f - \min t_s}{\max t_e - \min t_s} \tag{6.6}$$

where t_f is the time of observation for the last target in the sequence. Also, $\min t_s$ and $\max t_e$ is the minimum and maximum time at which any target in the sequence could be observed. In this way, the variable $\phi(t_f)$ is always smaller than 1, such that it is always preferred to observe more targets than minimizing the final time.

Meanwhile, the scheduled observation must satisfy the constraints of the imaging time window and the minimum manoeuvre time between the targets. The first step to state the constraints mathematically is to denote a pair of subsequent observations with the indices (i, j). Values for the time visibility window and the selected scanning direction are attributed to each of these indices. Now for all possible combinations of these pairs, a minimum manoeuvre duration $d_{\min}(i, j)$ can be calculated using results from the ANNs as presented in Chapter 5:

$$d_{\min}(i,j) = d_{slew_{\min}}(i,j) + t_{\text{stab}} + t_{\text{acq}} \tag{6.7}$$

Here, $d_{slew_{\min}}(i, j)$ is the minimum slew time between the targets, t_{stab} is the required stabilization time and t_{acq} is the duration of the actual image scanning. The values for these constants are introduced in Table 2.1. Now, the constraints for target acquisition can be summarized as (Lemaître et al., 2002):

$$t_{s,i} \leq t_{0,i} < t_{f,i} \leq t_{e,i}, \ \forall x_i = 1 \tag{6.8}$$

$$t_i + d_{\min}(i,j) \leq t_j, \ \forall y_{ij} = 1 \tag{6.9}$$

$$x_i = \sum_{j=1}^{n_t+1} y_{ij} = \sum_{j=0}^{n_t} y_{ji} \tag{6.10}$$

Here, $t_{0,i}$ and $t_{f,i}$ are the initial and final time of acquiring target i. Also, epoch t_i is the time of acquiring the center of target i. Equations 6.8 and 6.9 are simply the constraints of the observation window and manoeuvre time respectively. The last constraint from Equation 6.10 regulates that any observation can only have one previous and one consecutive observation in the schedule, such that the resulting sequence is a directed path of selected targets. The number total number of targets is indicated by n_t. In order to use this constraint, also a virtual target must be introduced that starts and ends the sequence of selected targets (Lemaître et al., 2002).

Concluding, it can be said that the problem is formulated as a Mixed Integer Linear Programming (MILP) problem. Specific solvers such as CPLEX exist which can solve the MILP problem for a guaranteed optimality or estimate the bounds in which the optimum solution is situated. Unfortunately, for satellite scheduling, the problem is typically too large to solve using this approach. For one day of satellite scheduling, this problem can result in almost a million binary variables and millions of constraint equations, which are impossible to solve in reasonable time (Nag et al., 2017). Still, this approach is usually applied as verification for a very short time span.

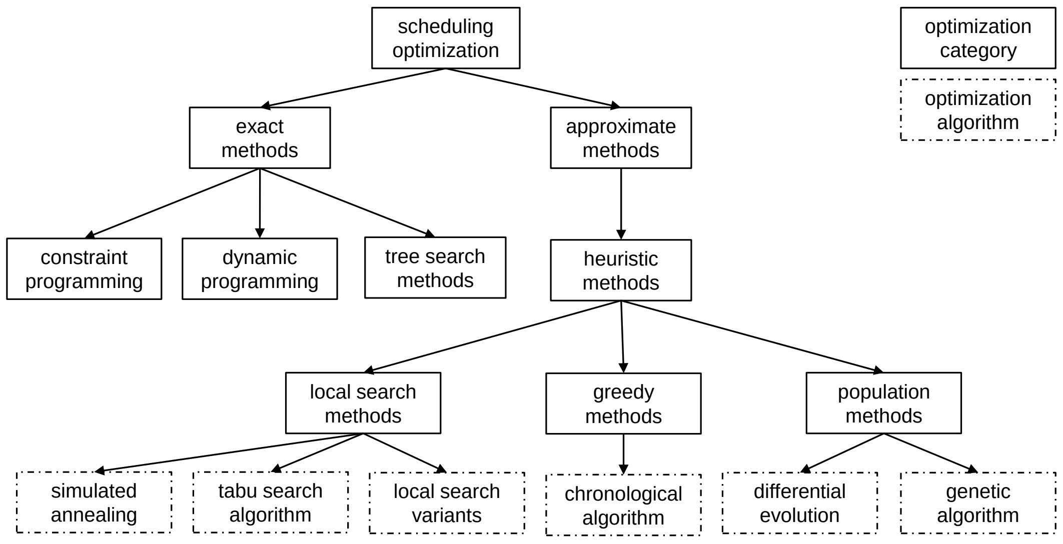


Figure 6.10: Overview of optimization methods for large combinatorial problems, based on Talbi (2009).

6.3. Selection of implemented scheduling algorithms

A wide range of algorithmic approaches for highly combinatorial problems such as the AEOS scheduling problem have been developed in the past. A short overview of the most common methods is given in Figure 6.10. The exact algorithms are only able to solve small problem sizes. Therefore, combinatorial problems are commonly solved using meta-heuristics for larger instance sizes. Especially local search methods are popular for scheduling problems. As for example shown by Globus et al. (2004), other meta-heuristics such as population-based methods (genetic algorithms, differential evolution,...) are known to have poorer performance for this type of problem. This behaviour is caused by the phenomenon that the scheduling problem prefers exploitation over exploration of the solution space. In other words, it is better to improve a current solution than further exploring the whole solution space. This happens for optimization problems where many local optima exist of (almost) equal quality. Furthermore, since the scheduling problem includes a large number of constraints, it is difficult to obtain a feasible solution using heuristics such as cross-over or mutation. A single cross-over operation can cause a feasible solution to become infeasible, while it is more difficult to obtain a feasible solution from an infeasible one. Therefore, heuristics which explore the neighbouring solutions such as local search methods are preferred.

Local search methods come in many forms, but the algorithms which are used most often in operations research are simulated annealing, tabu search and hill climbing. All these algorithms have a different methodology for neighbourhood exploration. Also for AEOS scheduling, several other local search methods have been examined in the past. Examples of these algorithms are an iterative local search by Peng et al. (2018), a large neighbourhood search algorithm by He et al. (2018), and a local search algorithm with custom insertion by Lemaître et al. (2002). The local search method which is chosen for this research is simulated annealing (SA). This method is known for its robustness against convergence to a local optimum. Furthermore, it was shown by Cordeau and Laporte (2005) that SA was the best performing algorithm to solve the public ROADEF challenge (see Section 1.1.2).

Next to the local search methods, also dynamic programming has demonstrated promising results in (Lemaître et al., 2002). This programming framework is typically very efficient when several subproblems have to be solved multiple times. As shown in Figure 6.10, this is an exact method. However, dynamic programming can also be combined with heuristic methods. Finally, also a greedy algorithm will be implemented. This algorithm does not tend to find the global optimum, but it is typically very fast.

6.4. Greedy algorithm

A greedy algorithm (GA) solves an optimization problem by making a locally optimal choice at sequential stages in order to approximate the global optimum. For AEOS scheduling, the sequence of targets is progressively constructed along the track of the satellite. In this way the GA chooses at each step a new target which is not observed yet using a heuristic rule. The chosen criterion that decides which target is observed next, is the minimum transition time between the last imaged target and the possible consecutive ones. This heuristic rule was found to provide the best performance. Other possible criteria are the remaining time until the end of the visibility of a target, as proposed by Verfaillie and Lemaître (2001), or methods which use pre-sorting of the targets. A possible explanation for the good performance of the proposed heuristic rule in this research is that a minimization in transition time automatically implies a maximization of the number of observed targets. This implies that this GA heuristic makes a local optimal choice between the consecutive steps, while other heuristics are not inherently optimal on this local level.

Algorithm 1 shows the pseudocode of the GA scheduling algorithm. The variables used are the following:

- $S^\star$: the best set of selected targets found by the algorithm
- t: epoch of observing the last target of the current solution for $S^\star$
- T_i: a possible target to be acquired

The function *getTargetsInView* determines all targets that will be in view from the start of the previous image until the maximum slew duration. Therefore, it is possible that a certain target is not in view yet if the slew time is relatively short. In that case, the first epoch at which the target is visible, is taken as the observation time.

Algorithm 1: Pseudocode for the greedy algorithm.

```
1  $t \leftarrow \min(t_s)$
2  $T_{\text{first}} \leftarrow$ first visible target in sequence
3  $S^\star \leftarrow \{T_{\text{first}}\}$                      ▷Start sequence with first target in view
4  while $t < \max(t_e)$ do
5      $S_{\text{try}} \leftarrow$ getTargetsInView($t$)
6      $t_{\text{next}} \leftarrow \infty$
7      for $\forall T_{try} \in S_{try} \backslash S^\star$ where $p_{T_{try}} = \max p_{\{S_{try} \backslash S^\star\}}$ do   ▷All possible targets with highest priority
8          $t_{\text{try}} \leftarrow t + d_{\min}$
9          if $t_{try} < t_s(T_{try})$ then
10             $t_{\text{try}} \leftarrow t_s(T_{\text{try}})$
11         if $t_{try} < t_{next}$ then                   ▷Find target with shortest slew
12             $t_{\text{next}} \leftarrow t_{\text{try}}$
13             $T_{\text{next}} \leftarrow T_{\text{try}}$
14     $t \leftarrow t_{\text{next}}$
15     $S^\star \leftarrow S^\star \cup \{T_{\text{next}}\}$
```

6.5. Dynamic programming

Dynamic programming (DP) divides the main problem recursively into smaller subproblems. This method is based on Bellman's principle, which states that each part of an optimal solution is optimal as well. In other words: an optimal solution can be found by solving for optimal subproblems. In that sense, dynamic programming is just an optimized method to trace over all possible paths. The algorithm is organized according to a bottom-up approach, which firstly treats the smallest subproblems (Talbi, 2009). The total enumeration of the search space is avoided by memorizing the solutions of the subproblems, such that the DP algorithm avoids solving the same subproblem multiple times. However, as the solutions to all subproblems are saved, the required computer memory can increase exponentially with the number of targets. This is one of the main drawbacks of DP. The DP principles will be applied to construct an exact scheduling method and an approximate algorithm in Sections 6.5.1 and 6.5.2 respectively. Correspondingly, the methods are referred to as the DP(E) and DP(A) algorithms in further discussions.

6.5.1. Exact scheduling algorithm

The exact scheduling DP algorithm is directly linked to research question 2, which requests to find the optimal schedule for 12 targets. While this is still barely possible using a brute-force method, it already requires several hours of CPU time. Therefore, the question came up whether a better approach can be found using the DP principles. The presented method uses the concepts of the DP solution for the travelling salesman problem (TSP) combined with four new propositions, which adapt the solution to AEOS scheduling problem. First, a short introduction of the DP approach for the TSP is given in the next paragraphs.

The exact solution of the TSP using DP is found by calculating the solutions of all subproblems starting with the smallest (bottom-up approach). If a larger subproblem requires the solution of smaller subproblems, this solution is taken from the collection of already computed subproblems. The DP implementation of the TSP is called the Bellman-Held-Karp algorithm. In order to demonstrate the working principles, the algorithm is applied to a subproblem of the AEOS scheduling problem. The subproblem concerns the optimization problem which aims to find the minimum time required to observe n targets, such that only the sequence and not the selection of the targets needs to be determined. In other words, it is assumed here that all time window constraints can be removed, while the slew time between the targets still needs to be satisfied.

Algorithm 2: Pseudocode for the Bellman-Held-Karp algorithm, applied to the AEOS scheduling problem without time constraints.

```
1  $j(i_1, i_2) \leftarrow \infty, \forall i_1 \in [1, \dots, 2^n - 1], \forall i_2 \in [1, \dots, n]$
2  $m(i_1, i_2) \leftarrow 0, \forall i_1 \in [1, \dots, 2^n - 1], \forall i_2 \in [1, \dots, n]$
3  for $T_i \in T$ do
4      $j(\{T_i\}, T_i) \leftarrow t_s(T_i)$                          ▷Start sequence when first visible
5      $m(\{T_i\}, T_i) \leftarrow -1$                               ▷Indicate start of sequence
6  for $n_T \leftarrow 2$ to $|T|$ do                                ▷For all subset sizes larger than 1
7      for $\forall S \subseteq T$ where $|S| = n_T$ do              ▷For all possible subsets of size $n_T$
8          for $\forall T_i \in S$ do
9              for $\forall T_j \in S \backslash \{T_i\}$ do
10                 $d \leftarrow j(S \backslash \{T_i\}, T_j) + d_{\min}(T_j, T_i)$
11                 if $d < j(S, T_i)$ then                           ▷Find shortest manoeuvre time
12                     $j(S, T_i) \leftarrow d$
13                     $m(S, T_i) \leftarrow T_j$
```

The pseudocode for the algorithm is presented in Algorithm 2 and is inspired by the formulation of Held and Karp (1962). The notation for the variables is as follows:

- T: set of all possible targets
- S: subset of possible targets
- j: array which stores the value of the objective function associated to each subset
- m: array which stores the best target to image prior to the current one. This parameter is used for backtracking the optimal sequence

Furthermore, the cardinality of a set S is indicated as $|S|$.

The algorithm starts by creating the solutions of the smallest subsets, that is, when only a single target is imaged. Note that the previous best job in the array m is a fictitious job, indicated by the in-

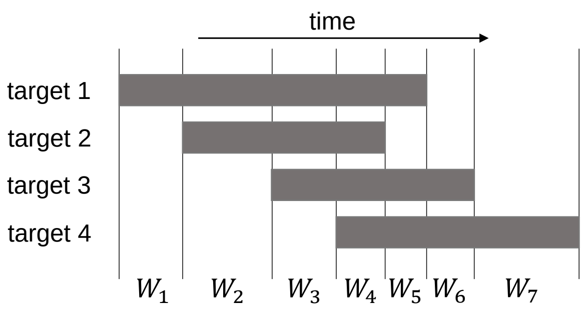


Figure 6.11: Illustration of the scheduling time windows W_i, where the time availability of each target is shown as a horizontal bar.

dex -1. For larger subsets, all possible combinations are enumerated for a certain number of targets. The possible subsets are found in Line 7 of the algorithm. For a subproblem with n_T targets, it can be easily reasoned that the number of possible combinations is $n_T \binom{|T|}{n_T}$. After taking the sum over all subproblem sizes n_T, it can be proven that in total $n2^{n-1}$, or $\mathcal{O}(n2^n)$ subproblems exist. Furthermore, it is determined for each subset which insertion position results in the highest profit gain, such that again $\mathcal{O}(n)$ smaller subproblems need to be solved. As a result, this algorithm has the time complexity of $\mathcal{O}(n^2 2^n)$, which determines how the CPU time scales with the number of targets. The space complexity, which shows how the required computational memory scales, can be derived to be $\mathcal{O}(n2^n)$ from the sizes of the arrays p and m (Bouman et al., 2018). A practical example of all the steps taken by the DP algorithm for four targets is given in Appendix C.

Now that this unconstrained subproblem is solved, one can extend this theory to the AEOS scheduling problem by adding the time window constraints. In order to efficiently apply the DP concept for the TSP here, the scheduling problem is cut up in time windows. By cutting up the problem in windows, the smaller subproblems are created, which can be solved faster. The idea is to find the solution of a local TSP for each of these time windows and efficiently combine these over the entire time interval. The time windows will be defined as the periods of time W_i, such that during each period the same targets are observed. Therefore, each time a target enters or leaves the field of view of the satellite, a new time window is initiated. The set of targets which are visible in time window W_i is denoted as T_{W_i} and the starting time of window W_i is written as t_{W_i}. For n targets, at most $2n - 1$ such time windows can be found. The concept of time windows is also illustrated in Figure 6.11.

Due to the introduction of time-window constraints, it is not always possible anymore to observe all targets, such that also a selection is necessary. Therefore, the DP solution for the TSP is adjusted by introducing the following four proposed principles:

Proposition 1. *Take S_i and $S_j \subseteq T$, where $S_i = \{T_1, ..., T_i\}$ is a feasible solution and the feasibility of inserting target T_j in $S_j = \{S_i, T_j\}$ needs to be determined. For the optimal solution of the subproblem S_i such that target T_i is chronologically the last target of the sequence, it holds that the insertion is feasible if $t(T_i) + d_{\min}(T_i, T_j) < t_e(T_j)$.*

The above proposition avoids that subproblems are re-evaluated within a certain time window. Therefore this rule introduces the idea of how the dynamic programming principle is interpreted for the exact algorithm. This proposition is based on the fact that the slew time to observe a newly inserted target does not depend on the entire sequence, but only on the previous target.

Proposition 2. *Take S_i and $S_j \subseteq T$, where $S_i = \{T_1, ..., T_i\}$ and target T_j is being inserted, resulting in subset $S_j = \{S_i, T_j\}$. If the solution of S_i is infeasible, it directly follows that the solution of S_j is also infeasible.*

Implicitly from Proposition 2, it follows that if a subproblem is infeasible, the larger sequences which consists of the infeasible subproblem do not need to be considered anymore, such that the enumera-

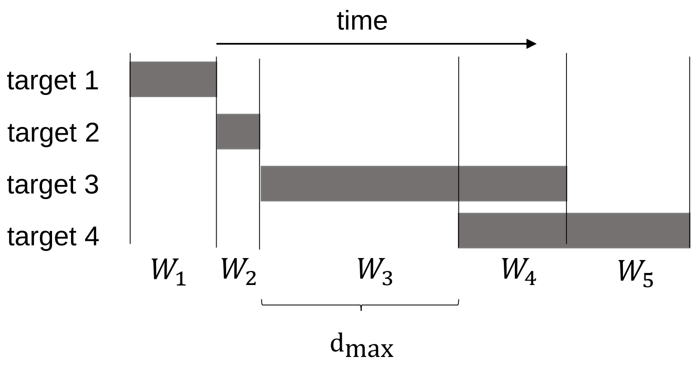


Figure 6.12: Illustration in support of Proposition 4. In this case, target 1 and target 3 can not be seen as 'detached' of each other.

tion of the search space is greatly reduced.

Proposition 3. *Take $S_i \subseteq T_{W_i}$ and $S_j \subseteq T_{W_j}$, such that $i < j$. If and only if it holds that $S_i \cap S_j = \emptyset$, all possible subsets in all time windows which include the targets in S_i are enumerated.*

The proposition above will be used to connect the solutions over the different time windows.

Proposition 4. *Take $S_i \subseteq T_{W_i}$ and $S_j \subseteq T_{W_j}$, such that $i < j$ and $S_i \cap S_j = \emptyset$. Furthermore, the last target T_i of the optimal sequence S_i is acquired at epoch t_i and the start time of window W_j is t_{W_j}. If and only if $t_{W,i} + T_{W,\max} + d_{\max} < t_{W_j}$, the connection between sets S_i and S_j does not need to be considered.*

In the last proposition, $d_{\max}$ is the maximum needed slew time and $T_{W,max}$ is the maximum total visibility duration of a target. This proposition can be introduced because the scheduling problem is subdivided in sequences, based on the maximum slew time as is also explained in Section 6.1.1. As a first intuition, one could reason that a gap of $d_{\max}$ between the two windows i and j is sufficient such that there is always another set $S_k \subseteq T_{W_k}$, where $i < k < j$, which will lead to a better solution. However, a gap of $T_{W,\max} + d_{\max}$ is required as will be explained based on Figure 6.12. Consider the case when target 2 is observed at the end of window W_2. Now, assume that the slew from target 2 to target 3 requires a long time, such that target 3 can be observed just at the end of window W_3. Also, the slew from target 3 to target 4 is again a long manoeuvre. Now, if $d_{\max}$ is taken as limit to consider the connection between two sets of targets, window W_4 can only continue with the solution from window W_3. In this case, the best solution from W_3 is to observe both targets 2 and 3, such that the solutions from window W_4 are further constructed on this previous best solution. Since the slew from target 3 to target 4 is a long slew, target 4 can not be observed at the start of W_4. However, this can lead to a sub-optimal solution since in the case when several other targets follow closely after target 4. In that case, it could be beneficial to not observe target 3, such that target 4 is observed earlier. However, the sets S_i and S_j will be completely detached if the gap in between is $T_{W,\max} + d_{\max}$.

The implementation of the exact algorithm, considering the propositions above is given in Algorithm 3. Clearly, the bottom-up approach of the Bellman-Held-Karp algorithm is applied. Also, the following variables are introduced:

- t: array stores the final time of the optimal sequence associated to each subset
- s_i: binary variable which indicates the scanning direction of target i
- p_i: priority factor of target i (see Section 6.2.2)

Furthermore, a new function *insertTargetInSequence*(T_i, S, T_j) is created. This function checks whether the time windows constraints are not violated when inserting target T_i in sequence S after target T_j, which is the last target in the sequence of set S. If the insertion is infeasible, the function returns the value -1. Otherwise, the epoch of acquiring target T_j is returned. Here, it is assumed that the time is relative to the starting time, and therefore always positive.

Next, some considerations about the algorithm will be discussed. In the first part of the algorithm, between lines 6 and 20, the sequences are initiated for a certain time window. From Line 10 to 20, it is checked whether it is better to initiate a new sequence for the particular window, or to connect with a sequence from previous time windows. Here, it important that the sequences from previous time windows do not include the targets which are added in the current window as introduced in Proposition 3. If the opposite it true, certain targets could be scheduled twice, or sub-optimal solutions could be found. Next, the remaining code is used to find the optimal solutions for all bigger subsets. The order of the for-loops is important as it defines which sequences are created first. The for-loops are structured such that the smallest subproblems are solved first and no subproblem is solved twice. After evaluating all permutations of the possible subsets, the actual sequence solution is found using backtracking. The array m stores now both the indices of the previous time window and the target in that window.

In order to facilitate the implementation, the sets are encoded into the bits of an integer, such that the sets are represented using the binary number system. As an example, the set S of targets $\{T_2, T_3, T_5\}$ becomes the binary number 101100, which corresponds to the integer $2^2 + 2^3 + 2^5 = 44$. If the indexing of the targets starts at T_1 instead of T_0, the target indices should be simply reduced by 1 to obtain the binary representation. It can be proven that each binary number results in a unique integer and vice versa, such that each set is now assigned its own number. These numbers are the indices used in the arrays p, t and m. In this way, the binary representation of the sets make it easier and more efficient to find the solution of the subsets which are already solved before.

The worst-case time complexity of this algorithm is $\mathcal{O}(n\mathcal{v}^2 2^{\mathcal{v}})$, where $\mathcal{v}$ is the maximum number of targets in a single time window and n the total number of targets. Furthermore, the space complexity is $\mathcal{O}(n\mathcal{v}2^{\mathcal{v}})$. Those complexities are as expected, since this algorithm solves the TSP using DP over multiple time windows. The time complexity of the exact solution is therefore exponential, however it is significantly reduced compared to the brute-force solution, which scales with a factorial term. From the time complexity, it can be seen that the number of time windows, and therefore the number of targets, influences the time complexity only linearly. This is a particular strength of this exact solution, since hundreds of targets could be exactly solved, for a restricted number of targets in each time window. Still, this algorithm is not practical if the number of targets per time windows grows too large and can therefore not be used as a general scheduling algorithm. Nevertheless, this can serve as a comparison for approximate scheduling methods for a limited number of targets.

6.5.2. Approximate scheduling algorithm

The approximate DP method is inspired by the weight job interval scheduling problem and the implementation of Lemaître et al. (2002). Here, the jobs are first ordered according to a certain heuristic rule, after which the optimal selection of targets is determined using DP logic. In this sense, the exact DP algorithm as presented in Section 6.5.1 is simplified from a *sequence* and *selection* problem to a *selection* problem.

Of course, the question remains how to determine the sequence of the targets. It is chosen to sort the targets based on the center time of their availability windows. This simple heuristic is based on the intuitive idea that targets can be scheduled chronologically according to the satellite track. It was also found that the optimal solution regularly adheres to this rule. Nevertheless, the assumption results in a sub-optimal solution, which is accepted for this approximate algorithm. Another attempt was made to find the sequence of the targets using heuristics such as local search methods which are commonly used to solve a TSP problem. This was done by reformulating the SA algorithm from Section 6.6 such that it solves the AEOS scheduling problem without the time-window constraints. This subproblem was already discussed in Section 6.5.1 in the context of the Bellman-Held-Karp algorithm. The disadvantage of this heuristic is the increased computational time, which can make the DP(A) algorithm up to five times slower. Now, for a small number of targets, the sequence as given by the SA heuristic outperforms the chronologically sorted sequence. Nevertheless, when increasing the number of targets, the improvement disappears, while for a small number of target the exact solution can be found within reasonable time using other algorithms as shown in Section 6.7.1, such that there is no good incentive to keep this approach.

Algorithm 3: Pseudocode for exact dynamic programming (DP(E)) algorithm.

```
$j(i_1, i_2, i_3, i_4) \leftarrow 0, \forall i_1 \in [1, \dots, |W|], \forall i_2 \in [1, \dots, 2^{\upsilon} - 1], \forall i_3 \in [1, \dots, \upsilon], \forall i_4 \in [1, 2]$
$t(i_1, i_2, i_3, i_4) \leftarrow \infty, \forall i_1 \in [1, \dots, |W|], \forall i_2 \in [1, \dots, 2^{\upsilon} - 1], \forall i_3 \in [1, \dots, \upsilon], \forall i_4 \in [1, 2]$
$m(i_1, i_2, i_3, i_4, i_5) \leftarrow 0, \forall i_1 \in [1, \dots, |W|], \forall i_2 \in [1, \dots, 2^{\upsilon} - 1], \forall i_3 \in [1, \dots, \upsilon], \forall i_4 \in [1, 2], \forall i_5 \in$
    $[1, 2, 3]$
for $W_i \leftarrow 1$ to $|W|$ do
    for $T_i \in T_{W_i}$ do        ▷Start of new sequence or connect to sequence of previous window
        for $s_i \leftarrow 1$ to 2 do
            $j(W_i, \{T_i\}, T_i, s_i) \leftarrow p_i$        ▷Start a new sequence
            $t(W_i, \{T_i\}, T_i, s_i) \leftarrow t_s(T_i)$
            $m(W_i, \{T_i\}, T_i, s_i) \leftarrow [-1, -1, -1]$
            for $W_j \leftarrow 1$ to $W_i - 1$ where $t_{W_i} + T_{W,\max} + d_{\max} > t_{W_j}$ do        ▷See proposition 4
                for $\forall S \subseteq T_{W_j} \backslash T_{W_i}$ where $S \neq \emptyset$ do        ▷See proposition 3
                    $T_j \leftarrow$ last target in optimal sequence for $S$
                    for $s_j \leftarrow 1$ to 2 do
                        if $j(W_j, S, T_j, s_j) \neq 0$ then
                            $t_{\{S,T_i\}} \leftarrow$ insertTargetInSequence$(T_i, S, T_j)$
                            if $t_{\{S,T_i\}} \neq -1$ then
                                if ($t_{\{S,T_i\}} < t(W_i, \{T_i\}, T_i, s_i)$ and $j(W_j, S, T_j, s_j) + p_i = j(W_i, \{T_i\}, T_i, s_i)$)
                                 or $j(W_j, S, T_j, s_j) + p_i > j(W_i, \{T_i\}, T_i, s_i)$ then
                                    $j(W_i, \{T_i\}, T_i, s_i) \leftarrow j(W_j, S, T_j, s_j) + p_i$        ▷Connect to sequence of
                                     previous window
                                    $t(W_i, \{T_i\}, T_i, s_i) \leftarrow t_{\{S,T_i\}}$
                                    $m(W_i, \{T_i\}, T_i, s_i) \leftarrow [W_j, T_j, s_j]$

    for $n_T \leftarrow 2$ to $|T_{W_i}|$ do        ▷Build on the initiated sequences of window $W_i$, similar to Algorithm 2
        for $\forall S \subseteq T$ where $|S| = n_T$ do
            for $\forall T_i \in S$ do
                for $\forall T_j \in S \backslash \{T_i\}$ do
                    for $s_i \leftarrow 1$ to 2 do
                        for $s_j \leftarrow 1$ to 2 do
                            if $j(W_j, S, T_j, s_j) \neq 0$ then        ▷See proposition 2
                                $t_{\{S,T_i\}} \leftarrow$ insertTargetInSequence$(T_i, S, T_j)$
                                if $t_{\{S,T_i\}} \neq -1$ then
                                    if ($t_{\{S,T_i\}} < t(W_i, S, T_i, s_i)$ and
                                     $j(W_i, S \backslash \{T_i\}, T_j, s_j) + p_i = j(W_i, S, T_i, s_i)$) or
                                     $j(W_i, S \backslash \{T_i\}, T_j, s_j) + p_i > j(W_i, S, T_i, s_i)$) then
                                        $j(W_i, S, T_i, s_i) \leftarrow j(W_i, S \backslash \{T_i\}, T_j, s_j) + p_i$
                                        $t(W_i, S, T_i, s_i) \leftarrow t_{\{S,T_i\}}$
                                        $m(W_i, S, T_i, s_i) \leftarrow [W_j, T_j, s_j]$
```

The pseudocode for the approximate DP algorithm is shown in Algorithm 4. Compared to Section 6.5.1, the following variables are added:

- a_s: set of possible azimuth scanning angles α_s
- τ_{T_i}: set of possible times to acquire target T_i. It was decided to discretize the total visibility window of each target with steps of 1 second.

The algorithm basically considers for all smaller subproblems whether a certain new target T_i could be added or not. The solution is updated if the new target can be added, while the best solution of the subproblem is kept if the target can not be added. Furthermore, the algorithm simply loops over all the possibilities of scanning directions, azimuth scanning directions and observation times. Note that the azimuth angle was assumed to be constant for the complete target sequence (Section 6.2.1). However, the angle is included here for the analysis in Section 6.7.4. If the scanning azimuth angle is constant, the set a_s simply consists of a single element. The same holds for the time of target acquisition: this variable does not necessarily need to be included given Assumption 1. Apart for the analysis in Section 6.7.4, this variable can be obtained given the sequence of the smaller subproblems. Finally, the time complexity can be said to be $\mathcal{O}(n^2|a_s|^2|\tau_{T_i}|)$ and therefore the approximate DP algorithm runs in polynomial time.

Algorithm 4: Pseudocode for approximate dynamic programming (DP(A)) algorithm.

```
1  $j(i_1,i_2,i_3,i_4) \in, \forall i_1 \in [1,\dots,|T|], \forall i_2 \in [1,2], \forall i_3 \in [1,\dots,|a_s|], \forall i_4 \in [1,\dots,\max_{T_i}\tau_{T_i}]$
2  $m(i_1,i_2,i_3,i_4,i_5) \in, \forall i_1 \in [1,\dots,|T|], \forall i_2 \leftarrow [1,2], \forall i_3 \in [1,\dots,|a_s|], \forall i_4 \in [1,\dots,\max_{T_i}\tau_{T_i}], \forall i_5 \in$
     $[1,\dots,4]$
3  $T \leftarrow$ sortTargets($T$)        ▷Determine the sequence of the set according to a heuristic
4  for $T_i \leftarrow 1$ to $|T|$ do
5      for $s_i \leftarrow 1$ to $2$ do
6          for $a_{s,i} \leftarrow 1$ to $|a_s|$ do
7              for $\tau_i \leftarrow 1$ to $|\tau_{T_i}|$ do
8                  if $\tau_i = 1$ then
9                      $j(T_i,s_i,a_i,1) \leftarrow p_i$        ▷Start of a new sequence
10                     $m(T_i,s_i,a_i,1) \leftarrow [-1,-1,-1,-1]$
11                 else
12                     $j(T_i,s_i,a_i,\tau_i) \leftarrow j(T_i,s_i,a_i,\tau_i-1)$ ▷Profit is at least equal to previous time step
13                     $m(T_i,s_i,a_i,\tau_i) \leftarrow [T_i,s_i,a_i,\tau_i-1]$
14                 for $T_j \leftarrow 1$ to $T_i - 1$ do        ▷Loop over previous targets
15                     for $s_j \leftarrow 1$ to $2$ do
16                         for $a_{s,j} \leftarrow 1$ to $|a_s|$ do
17                             $t_{\tau_i} \leftarrow$ element in set $\tau_{T_j}$ associated to index $\tau_i$
18                             $\tau_j \leftarrow upperIndex(t_{\tau_i} - d_{\min}(T_j,T_i))$ ▷Round up to nearest element in set
                                 $\tau_{T_j}$ and get the index of this element
19                             if $\tau_j < 1$ then        ▷If before start of visibility window of $T_j$
20                                 $\tau_j \leftarrow 1$
21                             if $\tau_j \leq |\tau_{T_j}|$ then        ▷If not after end of visibility window of $T_j$
22                                 if $j(T_j,s_j,a_j,\tau_j) + p_i > j(T_i,s_i,a_i,\tau_i)$ then
23                                     $j(T_i,s_i,a_i,\tau_i) \leftarrow j(T_j,s_j,a_j,\tau_j) + p_i$
24                                     $m(T_i,s_i,a_i,\tau_i) \leftarrow [T_j,s_j,a_j,\tau_j]$
```

6.6. Simulated annealing

Simulated annealing (SA) is a metaheuristic global optimization method which is often used for highly combinatorial problems. The name refers to a process in thermodynamics, particularly the way in which

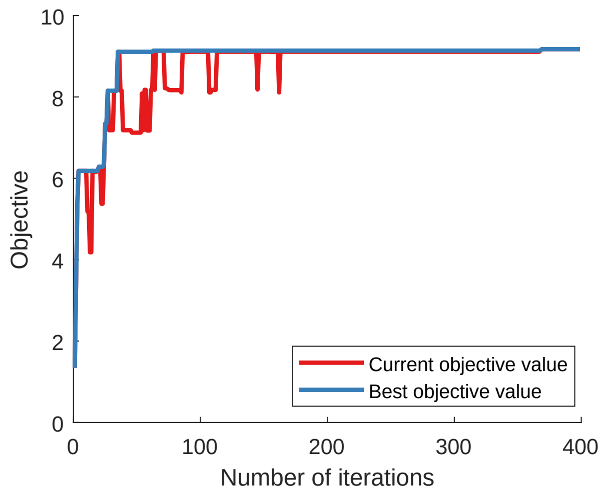


Figure 6.13: Convergence to the optimal solution of case 3 in Section 6.7.1 using the SA algorithm.

a metal cools down. At high temperatures, the atoms move rapidly around. When cooling down the system, the metal will form a structured crystal such that the overall system energy is minimized, which is called the annealing process. When cooling down the metal too rapidly, the minimum energy state will not be reached, since a local optimal structure with defects will be obtained. The SA algorithm is based on this principle as at the start of the algorithm, the solution is allowed to change rapidly, covering the entire search space. As the algorithm progresses, only a few local changes are allowed. This approach makes the SA algorithm less likely than other local search method to be stuck in a local optimum (Kirkpatrick et al., 1983). Furthermore, the SA algorithm can be both applied to minimization and maximization of an objective.

The SA algorithm is a probabilistic algorithm as it makes random changes to a solution, similar to a particle which moves in the metal, and compares the resulting solution J_{try} to the previous solution J. Next, the change in the objective function $\Delta J = J_{\text{try}} - J$ is calculated. The new solution is always adopted in the case when $\Delta J > 0$. However, if $\Delta J < 0$ and therefore the new solution is worse, this new solution is still accepted with a certain probability P_a. This probability depends on the temperature and the difference ΔJ. Summarized, a new solution is accepted with the following probability:

$$P_a = \begin{cases} 1 & \text{if } \Delta J > 0 \\ e^{-\Delta J/T} & \text{if } \Delta J \leq 0 \end{cases} \tag{6.11}$$

where T is the current temperature. After the new solution is accepted or rejected, the temperature is reduced according to a cooling schedule. An often-used multiplicative cooling scheduling determines the temperature T for the next iteration $i+1$ as:

$$T_{i+1} = \theta T_i \tag{6.12}$$

Here, θ is the cooling rate factor, which is a tuning parameter.

The implementation of the SA heuristics is shown in Algorithm 5. In order to avoid starting in a local optimum, the initial solution is an empty set which is the worst possible solution of the problem. Nevertheless, it was identified that increasing the problem size results in a significant increase in necessary iterations to build a good solution. Therefore, if more than 50 targets need to be scheduled, the results from the GA is used as initial solution. This reduced the necessary CPU time by more than 90% in some cases with 500 targets, while the value of the objective function was not affected. Another rule that was used to speed up the algorithm is the variable *nChange* in the algorithm. This variable is the maximum number of targets than will be attempted to remove or insert in the current sequence. The actual number of targets that is inserted follows a discrete uniform distribution between

1 and *nChange*. On the other hand, between 0 and *nChange* targets will be removed for each iteration.

Furthermore, the targets are always removed and inserted in a certain time window W_i. These time windows are constructed such that the same targets are seen by the satellite during this particular time period, as was defined already for the DP(E) algorithm in Section 6.5.1. It was tried to use a random insertion position over the full sequence, but this approach did not deliver satisfactory results. If the random insertion would remove a target which is not in the same time window as the target which is inserted, it is difficult for the algorithm to find an improved sequence. The time windows offer the benefit that a solution is changed more locally, such that the removal and insertion operators are more effective. Also, the considered time window for insertion is a discrete random variable which follows a uniform distribution over the set $\{1, ..., |W|\}$. Similarly, the scanning direction is chosen randomly if a new target is inserted. One disadvantage of the SA method is that many parameters need to be tuned, which is a trial-and-error exercise. The settings used for this research are summarized in Table 6.1. It was found that choosing different parameters depending on how many targets need to be scheduled was required for good performance of the algorithm.

Algorithm 5: Pseudocode for simulated annealing algorithm.

```
$S \leftarrow \emptyset$
$T \leftarrow T_{\text{initial}}$
for $i \leftarrow 1$ to $i_{\max}$ do
    nChange $\leftarrow round(2(i_{\max} - i)/i_{\max} + 0.5)$
    $W_i \leftarrow$ randomInteger$(1 \dots |W|)$
    $S_{\text{try}} \leftarrow$ removeTarget$(W_i, S,$ nChange$)$
    $S_{\text{try}} \leftarrow$ insertTarget$(W_i, S_{\text{try}},$ nChange$)$
    $J_{\text{try}} \leftarrow$ evaluateSequence$(S_{\text{try}})$
    if $J_{try} > J$ then
        $S \leftarrow S_{\text{try}}$
    else
        $P_a = exp(-(J_{\text{try}} - J)/T_i)$
        if $P_a >$ random$(0, 1)$ then
            $S \leftarrow S_{\text{try}}$
        end
    end
    $T \leftarrow \theta T$
end
```

6.7. Scheduling results

This section contains a comparison of the performance of the four developed scheduling algorithms. It was already established before that the number of targets has a significant influence on the computation time due to the exponentially large search space. Still, some algorithms are capable of solving large cases, while others fail. Therefore, the algorithms are tested for three different problem sizes. Firstly, Section 6.7.1 shows the results when the number of targets is limited to 12 instances. Next, the problem size is increased to 50 targets in Section 6.7.2. Finally, Section 6.7.3 treats the results for the largest case of up to 500 targets. All presented scheduling algorithms are written in Matlab and converted to MEX-files, which is a binary executable created from the underlying C, C++ or Fortran code. This

Table 6.1: Tuning parameters for the SA algorithm for different problem sizes.

Problem size	**T_{initial} [K]**	**Cooling rate θ**	**Maximum number of iterations**
< 25	1	0.98	400
25 to 100	2	0.99	500
> 100	5	0.995	1000

typically results in a code that is between 10 and 50 times faster than using Matlab.

6.7.1. Short-term scheduling cases

In order to test the algorithm for smallest scheduling instances, several cases of 12 targets were created that are expected to be difficult to handle by the algorithms and highlight their strengths and weaknesses. In total, five cases of twelve targets are constructed according to the following rationale:

Case 1. A test case for which the targets are randomly distributed. This case is included to demonstrate the quality of the scheduling algorithms without structuring the targets in a certain fashion that is expected to be difficult to handle by some algorithms.

Case 2. The targets are distributed in a regular grid, such that the targets are widely distributed and the slew time is therefore relatively long between the targets. As a consequence, it is important for the algorithm to select the best sequence. The DP and GA algorithm are expected to perform poorly for this case since they do not optimize the sequence, but rather assume a sequence or find a local optimal one.

Case 3. The targets are spread as far as possible from each other in the cross-track direction. Therefore it is more difficult for the satellite to visit all targets, since long slew manoeuvres are necessary between the targets. Like in Case 2, it is expected that pre-sorting the targets (DP) or selecting the best neighbor (GA) will lower the quality of the results.

Case 4. The targets are aligned and put closely together. In this way, all targets can be seen at the same time for a certain epoch. This is challenging for the exact DP algorithm, since this results in the expected worst-case CPU time for 12 targets.

Case 5. The targets of this case are placed in a particular way to demonstrate a weakness of the GA. While the slew time to a certain target might be shorter at a certain decision point, it is possible that a *hotspot*, where many targets are located on a small area, is missed due to the local optimal choice.

All algorithms are tested for each case, with the results shown in Figures 6.14-6.33. Each schedule is represented on a local map where the targets are represented as dots. The targets which are selected for observation are encircled. Furthermore, the sequence is illustrated by the connecting line in between the selected targets. The crossing of the orbit with the equator represents the ascending node of the orbit. In other words, the satellite goes from the bottom to the top of the figures over time. A last element that is indicated in the figures is the time of observation of a selected target. Whenever a target is observed, a line is drawn between the satellite ground track at that time and the relevant target. The SA algorithm is always run for 11 times, since it is a probabilistic algorithm, and the solution with median performance index is presented. The median result is used rather than the best of all runs, to give an idea of the expected performance when a the algorithm is run for a single time. In the results, the CPU time and performance of this median result is reported.

Some specific findings about the results in Table 6.2 for these cases are:

- The results demonstrate that the solution using the DP(E) and SA algorithms are always identical in terms of number of selected targets. However, the SA algorithm sometimes chooses a sub-optimal sequence which takes longer to observe. While the SA could always find the optimal number of selected targets after 11 runs, one target less was imaged in around 10% of the cases. Also, when the scanning direction is removed as a decision variable, the SA algorithm was found to converge more consistently to the global optimum. This is because the scanning direction variable has a relatively small impact on the objective function, such that the algorithm does not always have a clear reason to scan the image in a particular direction. Therefore, the randomness of the algorithm will have a relatively large impact on the determination of this decision variable.
- The DP(A) algorithm and GA approaches were not able to find the optimal solution in any of the cases. Only for case 4, the GA could observe the same number of targets, with a slightly worse final time. As expected, the sorting of the targets for the DP(A) algorithm and the local optimal choice of the GA are especially inaccurate for cases where the targets are widely spread

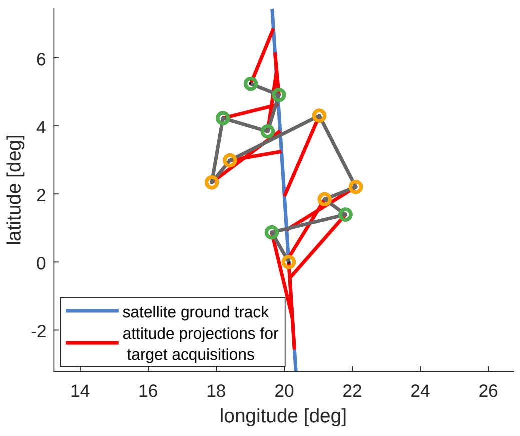


Figure 6.14: Scheduling result case 1 using SA. Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

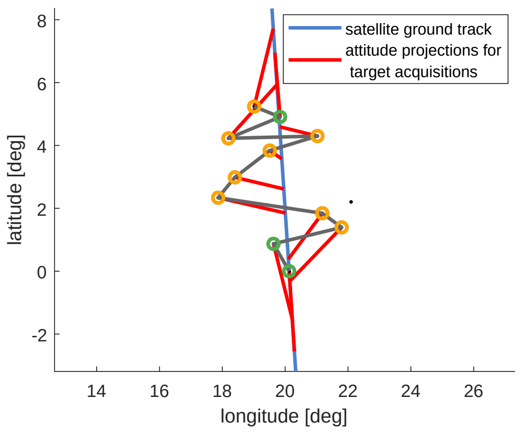


Figure 6.15: Scheduling result case 1 using DP(A). Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

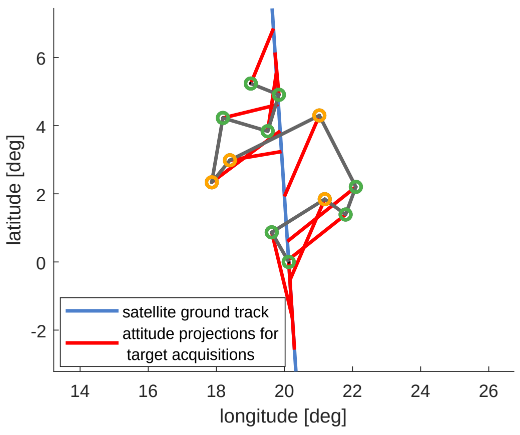


Figure 6.16: Scheduling result case 1 using DP(E). Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

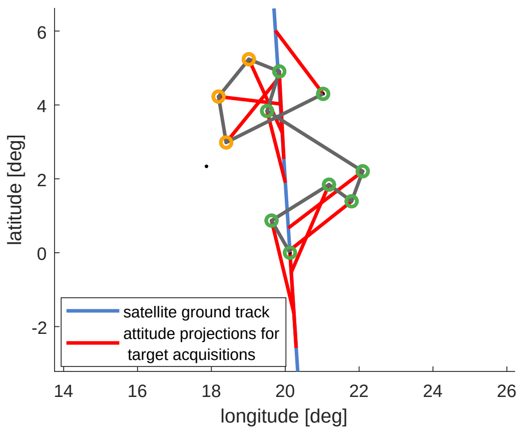


Figure 6.17: Scheduling result case 1 using GA. Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

in cross-track direction (cases 2 and 3).

- GA is by far the fastest algorithm and has a comparable performance to the DP(A) algorithm. The biggest weakness of the algorithm is that making a local optimal choice can result in missing a hotspot of targets, which is demonstrated in case 5.

In conclusion, it can be said that these short-term scheduling cases should not be considered anymore as a difficult problem. Previous literature used a MILP formulation and solvers such as CPLEX to obtain the exact solution, which requires often at least one hour of solving time. The inefficiency of the MILP solvers for this problem is attributed to the poor quality of the possible cutting planes (Lemaître et al., 2002). However, the DP approach can enable significant reduction in computational time and required memory. All cases could be solved under 80 seconds, even for the most difficult cases. When the scanning direction would be excluded, as is mostly the case in literature, the maximum computation time would be as short as 20 seconds. For this problem size, it is therefore advisable to use the DP(E) algorithm. For even shorter computation times, the SA algorithm can provide solutions of almost equal quality. In case the available CPU time is short, the DP(A) algorithm or GA should be considered.

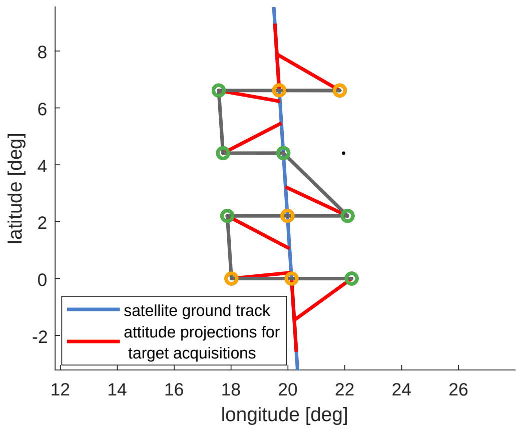


Figure 6.18: Scheduling result case 2 using SA. Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

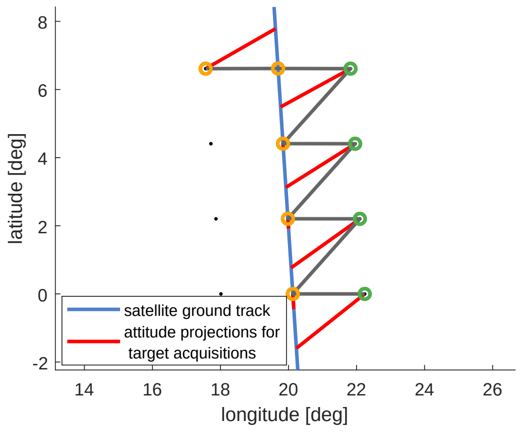


Figure 6.19: Scheduling result case 2 using DP(A). Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

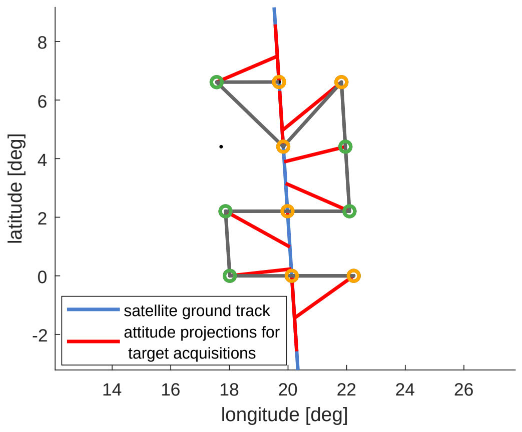


Figure 6.20: Scheduling result case 2 using DP(E). Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

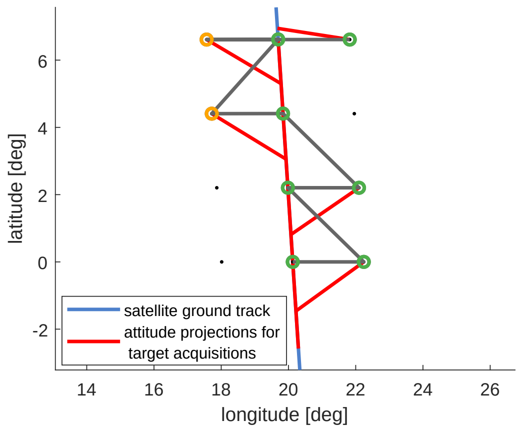


Figure 6.21: Scheduling result case 2 using GA. Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

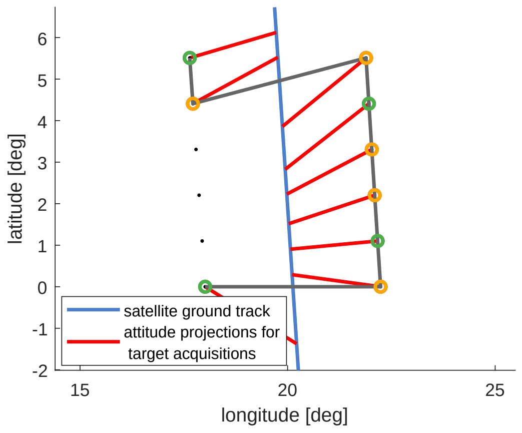


Figure 6.22: Scheduling result case 3 using SA. Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

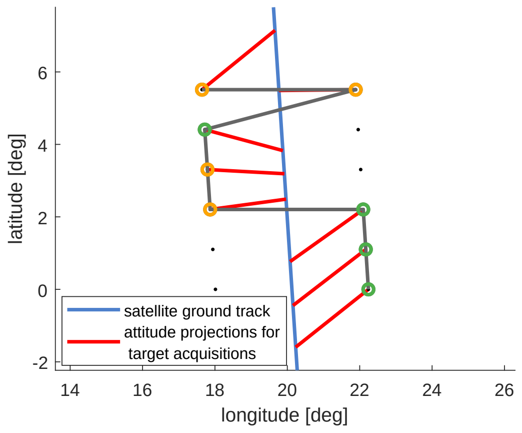


Figure 6.23: Scheduling result case 3 using DP(A). Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

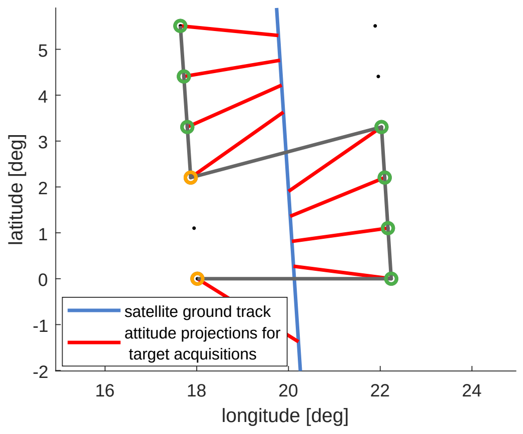


Figure 6.24: Scheduling result case 3 using DP(E). Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

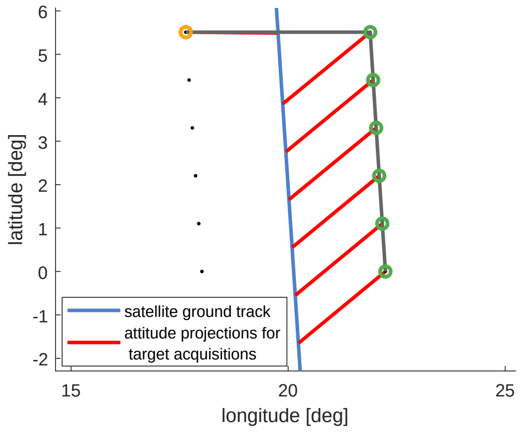


Figure 6.25: Scheduling result case 3 using GA. Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

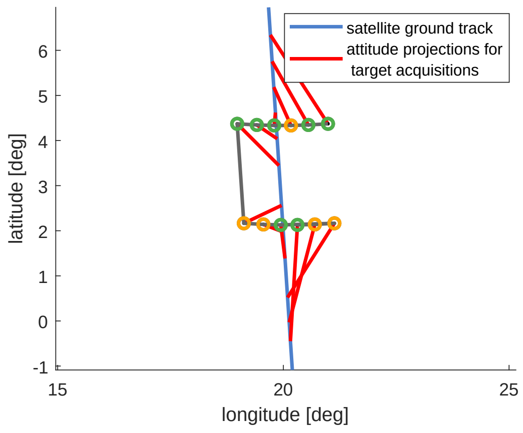


Figure 6.26: Scheduling result case 4 using SA. Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

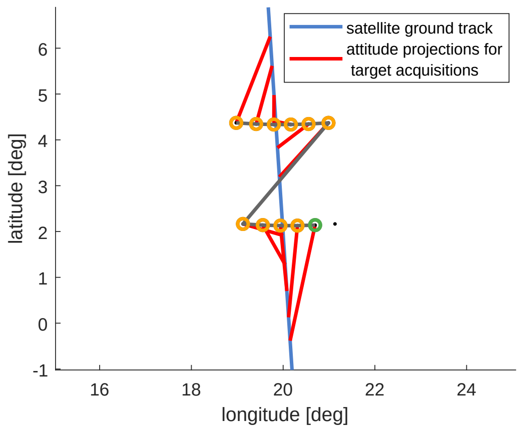


Figure 6.27: Scheduling result case 4 using DP(A). Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

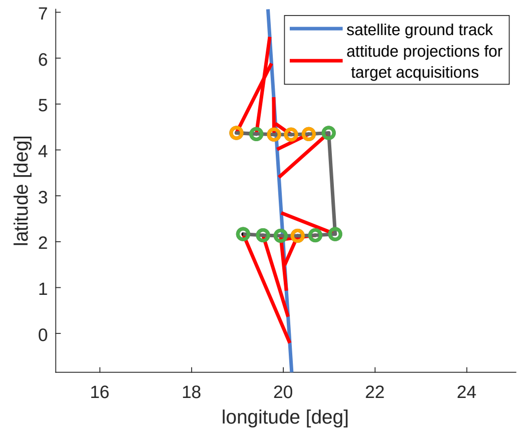


Figure 6.28: Scheduling result case 4 using DP(E). Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

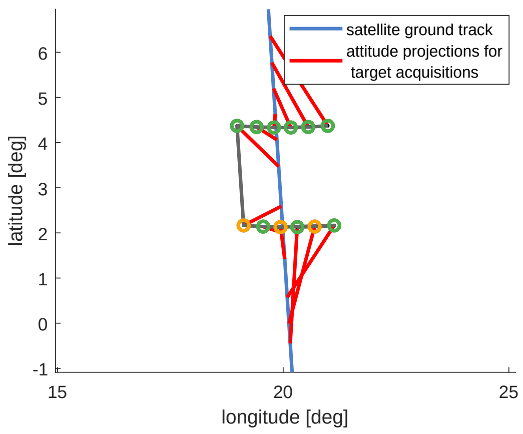


Figure 6.29: Scheduling result case 4 using GA. Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

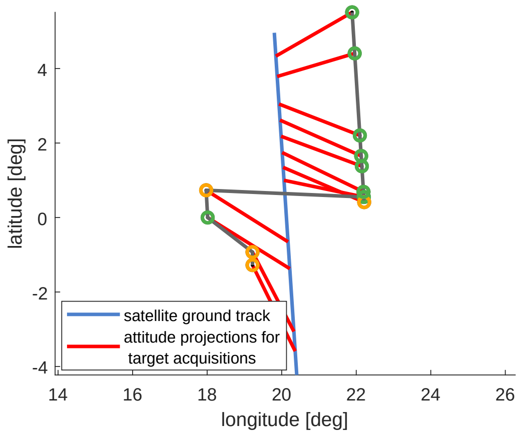


Figure 6.30: Scheduling result case 5 using SA. Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

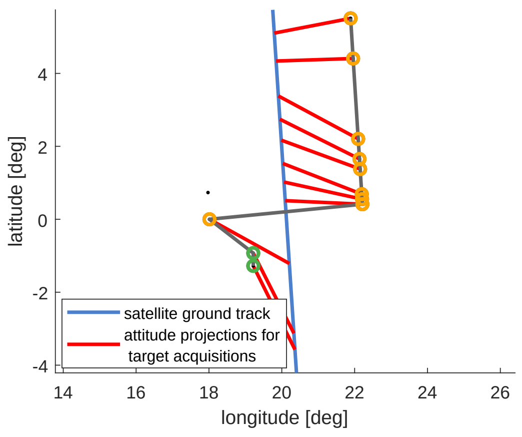


Figure 6.31: Scheduling result case 5 using DP(A). Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

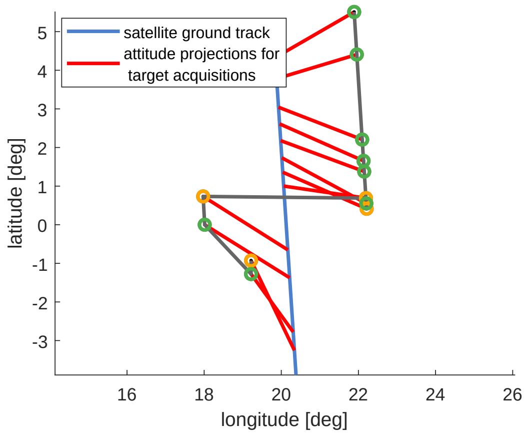


Figure 6.32: Scheduling result case 5 using DP(E). Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

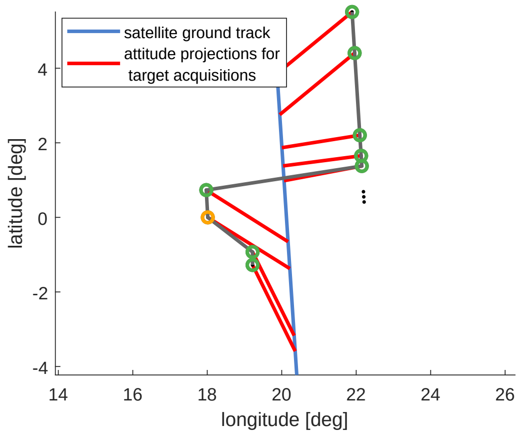


Figure 6.33: Scheduling result case 5 using GA. Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

Table 6.2: Scheduling objective function and required computational time for each short-term scheduling case of 12 targets by the four scheduling algorithms.

	SA		**DP(A)**		**DP(E)**		**GA**	
Case number	J	CPU [s]	J	CPU [s]	J	CPU [s]	J	CPU [s]
1	12.0910	0.32	11.0086	0.124	12.0916	44.0	11.1719	0.002
2	11.0244	0.38	9.1236	0.087	11.0565	3.62	9.1941	0.002
3	9.1233	0.41	8.0088	0.085	9.2152	3.36	7.1940	0.002
4	12.0848	0.60	11.0966	0.175	12.0885	76.8	12.0824	0.002
5	12.2467	0.41	11.1740	0.092	12.2471	22.3	9.2916	0.002

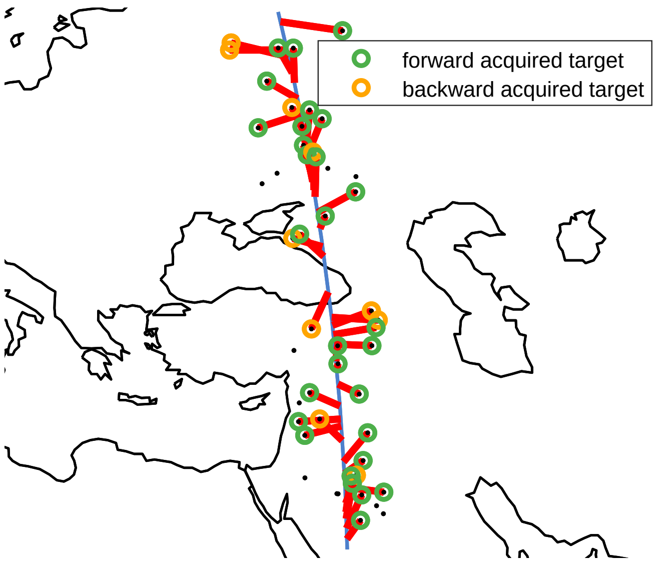


Figure 6.34: Exact schedule solution using DP(E) for 50 targets, case 1. In total 39 out of 50 targets can be acquired.

Table 6.3: Distribution of the medium-term scheduling cases, in terms of the time span in which the targets are visible and the priority of the targets.

Case number	**Time span [s]**	**Low priority targets**	**Medium priority targets**	**High priority targets**
1	475	33	15	2
2	250	30	14	6
3	150	28	17	5

6.7.2. Medium-term scheduling cases

The medium-term scheduling cases studied in this section are three sequences of 50 targets, without any gaps of more than the maximum slew time in between the targets. The difference for the three cases is their distribution over Earth's surface. Each set of possible targets is constructed such that all targets can be seen during a certain time span. For example, a time span of 200 seconds means that the 50 targets are closer together than if the time span is 400 seconds. As shown in Table 6.3, the targets for case 1 are widely distributed, while the targets for case 3 are close together. It is therefore expected that less targets can be imaged for case 3. In contrast to the short-term scheduling cases, also a priority factor is used here. The distribution of the priority is shown in Table 6.3.

The results of the scheduling for the medium-term cases are shown in Table 6.4. The solution of the DP(E) algorithm could not be obtained for cases 2 and 3 due to the limitation of RAM memory. Case 1 could still be solved, since at most 13 targets could be seen for each instance in time, such that arrays of the DP(E) algorithm required about 2 GB of RAM. Note that no sparse matrices were used here, which could improve the memory performance of the algorithm. The exact solution of case 1 is shown in Figure 6.34. The DP(A) algorithm was able to find the same schedule quality when disregarding the final time penalty $\phi(t_f)$. Still, none of the other algorithms could achieve the optimal result, but all heuristics could find results which are remarkably close to the optimum. In fact, the performance of all three approximate algorithms is comparable and within 7% of the optimum. On the other hand, the GA has on average a slightly worse performance, but the solution is still obtained almost instantaneous for this problem size. However, the SA algorithm already requires around one minute of computation time for most cases. The solutions that use the DP(A) algorithm are obtained in less than two seconds, while the performance is best for most of the cases. Therefore, this algorithm is recommended for problem sizes of around 50 targets. Nevertheless, it is still worth to try improving the solution using the SA algorithm if enough computation time is available.

Table 6.4: Scheduling objective function and required computational time for each medium-term scheduling case of 50 targets by the four scheduling algorithms.

	SA		**DP(A)**		**DP(E)**		**GA**	
Case number	J	CPU [s]	J	CPU [s]	J	CPU [s]	J	CPU [s]
1	60.038	57.3	61.014	1.13	61.034	$2.80{\cdot}10^4$	57.045	0.014
2	63.128	43.6	61.057	1.41	out of memory		60.180	0.011
3	49.026	39.4	51.005	1.82	out of memory		45.019	0.016

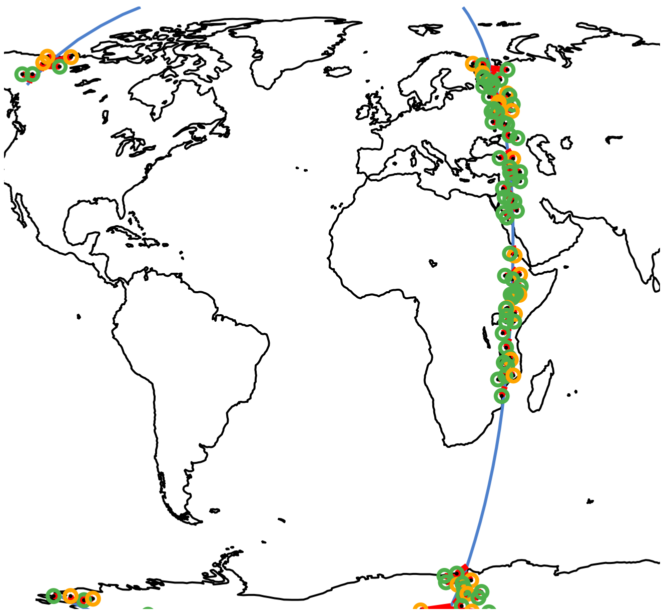


Figure 6.35: Exact schedule solution for 100 targets. In total 99 out of 100 targets can be acquired. Green and yellow circles indicate the selected targets with forward and backward scanning respectively.

6.7.3. Long-term scheduling cases

The section presents the largest considered problem sizes of the AEOS scheduling problem. In total, 100, 300 and 500 targets are distributed randomly over one satellite orbit time span. As a consequence, the oceans will subdivide the problem into several smaller sequences (see Section 6.1.1). Therefore, a smaller percentage of the total number of targets can be observed for the larger problem sizes. It is also expected that the difference in performance between the scheduling algorithms becomes larger when the number of targets increases.

The scheduling results for the long-term cases are shown in Table 6.5. Again, the smallest instance could be solved exactly. Still, it should be noted that for this case 99 out 100 targets could be acquired. Since almost all targets can be acquired, the selection of targets for this problem is not very stringent. Indeed, the performance of the other algorithms is very similar. The SA algorithm could only observe one target less than the optimal solution. However, when increasing the number of targets, the difference in performance in the algorithms becomes significantly larger. In fact, the DP(A) algorithm is able to obtain a 10% higher objective performance than the other algorithms for the largest problem size. This algorithm is also able to solve all cases in under 30 seconds of computation time. Therefore, this is the preferred algorithm for larger problem sizes. Nevertheless, it should be noted that the distribution of the targets is important for the performance of the DP(A) algorithm. The algorithm has good performance for randomly distributed targets, but can have worse performance for cases such as presented in Section 6.7.1.

6.7.4. Choice of decision variables

This section presents the results which support Assumption 2 from Section 6.2.1. In particular, the influence of including the scanning direction and scanning azimuth angle on the scheduling performance

Table 6.5: Scheduling objective function and required computational time by the four scheduling algorithms for the long-term scheduling cases, consisting of 100, 300 and 500 targets distributed over a single orbit.

	SA		**DP(A)**		**DP(E)**		**GA**	
Number of targets	J	CPU [s]	J	CPU [s]	J	CPU [s]	J	CPU [s]
100	173.015	240	172.006	0.950	174.013	$5.30{\cdot}10^3$	170.010	0.009
300	361.003	344	375.000	7.88	out of memory		355.008	0.047
500	461.002	831	512.002	21.1	out of memory		463.001	0.104

Table 6.6: Scheduling objective function and required computational time for each medium-term scheduling case of 50 targets by the four scheduling algorithms.

Case	**Number of acquired targets**	**CPU time [s]**
scanning direction constant and $\Delta\alpha_s = 0$	281	4.65
scanning direction variable, $\Delta\alpha_s = 0$	284	17.3
scanning direction constant, $\Delta\alpha_s \in [-45, -30, ..., 45]$ degrees	285	227
scanning direction constant, $\Delta\alpha_s \in [-45, -36, ..., 45]$ degrees	286	542

is discussed. As an example, the scanning azimuth angle was implemented in the DP(A) algorithm as a decision variable (see Algorithm 4). Next, the schedule results are generated for a case with 500 targets distributed such that these are visible during one satellite orbit. The results for four different cases are shown in Table 6.6. If the scanning direction is always in forward direction and the azimuth scanning angle is taken as zero, 281 targets were observed. When also including the scanning direction as decision variable, three targets more could be acquired. On the other hand, when using a discrete set of azimuth scanning angles, up to four or five targets more are observed, depending on the discretization. However, when including the scanning direction, the CPU time was increased only by a factor four, while the variable $\Delta\alpha_s$ can make the required CPU time up to 100 times longer. The additional performance of the schedule was considered not worth the additional CPU time. However, it enough CPU time is available for the satellite operators for planning, it can be beneficial to still include this decision variable. On the other hand, the added performance by varying the scanning direction is considered to be worth the increase of CPU time.

6.7.5. Off-nadir penalty

An example of the resulting schedule using the DP(A) algorithm for 800 targets distributed over one orbit is shown in Figure 6.36. This example demonstrates that the best found solution often includes many targets which are seen at high off-nadir angles. This makes sense from an intuitive point of view

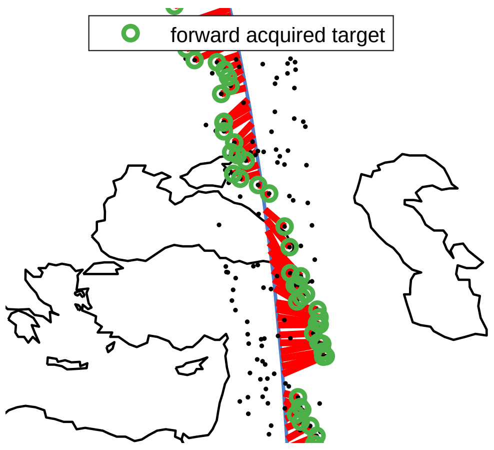


Figure 6.36: Part of the scheduling result using the DP(A) algorithm without off-nadir penalty for 800 targets distributed over a single orbit.

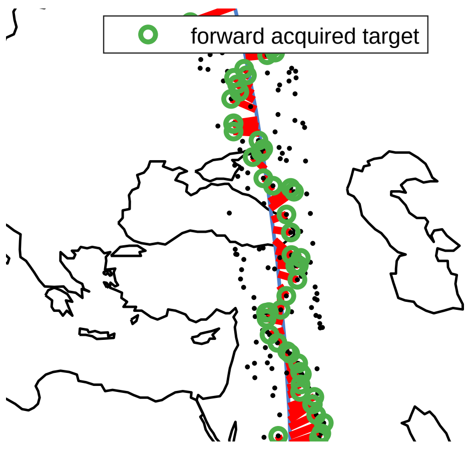


Figure 6.37: Part of the scheduling result using the DP(A) algorithm with off-nadir penalty for 800 targets distributed over a single orbit.

since the relative angles between the targets will be smaller for higher off-nadir angles. As the relative angles are smaller, the slew times between the targets is shorter, such that more targets can be scheduled for these regions. For image quality purposes, this behaviour is unfortunately not desirable, since high off-nadir angles result in poor image quality for the customer. Therefore, this section will explore a possible solution to this problem.

The quality of an image will be measured here by the off-nadir angle at which the image is scanned. In order to include the image quality in the objective function, the off-nadir angle is given as a penalty in the priority factor of the targets:

$$\bar{p} = p(1 - \frac{\eta}{\eta_{\max}}) \tag{6.13}$$

Of course it is trivial to change this adapted priority factor $\bar{p}$ if desirable. As opposed to the objective which is used so far, the exact time at which the target is scanned will influence the objective function. This is caused by the time-dependency of the off-nadir angle. As a consequence, Assumption 1 as used for the scheduling algorithms is not valid anymore. This was compensated in the approximate DP algorithm by introducing the acquisition time as a discretized decision variable (see Section 6.5.2). In this example, the introduction of the off-nadir penalty resulted in a schedule for which 7% less targets are acquired compared to the case without the penalty. On the other hand, the average off-nadir angle at acquisition went from 17.8 to 11.2 degrees.

6.7.6. Improvement of near-optimal slew manoeuvres

One of the main novelties introduced in this research is the use of time-optimal manoeuvres between the target observations. Therefore it is worthwhile to assess the improvement in the scheduling results, compared to the use of non-optimal attitude manoeuvres. A case was set up consisting of 300 targets which are distributed over a single orbit. Now, the DP scheduling is run multiple times, such that for each run the optimal slew time is increased by a certain percentage. In Figure 6.38 it can be seen that a slew manoeuvre for which the duration is sub-optimal with for example 20%, in total 8.3% less targets can be acquired. To get a better perspective, the slews in this sequence have an average duration of around 10 seconds. This means that overestimating the slew time by just 2 seconds already results in a schedule which is almost 10% worse.

As explained in Chapter 5, this research overestimates the slew by 0.5 seconds on average, corresponding to a 2% decrease in scheduling performance. However, the approximate slew manoeuvres in literature are not optimized and based on a conservative estimate. Furthermore, the time-dependency of the manoeuvres is typically neglected. Still, it was shown in Section 6.2.1 that the optimal manoeuvre can change by 20% during the time availability window. In fact, the manoeuvre time typically varies by 15-25% during the time window. Therefore, the approach in this research offers a significant advantage over the estimates in literature. Unfortunately, it is difficult to make a proper comparison between previous research and this example, since the improvement will depend on the schedule case, the satellite design, the exact method of the manoeuvre approximation and the scheduling algorithm. Unfortunately no complete information on either of these was found in literature, such that no good comparison could be made. Still, an approximate number of the possible improvement of the near-optimal manoeuvres can be obtained. For example, in (Lemaître et al., 2002), the slew time is overestimated by 10% to compensate for the time-dependency. In case the slew is not possible, this is corrected for after the scheduling. Furthermore, the slew time is found through an approximation, which is estimated to be at least 10% sub-optimal. Especially for short manoeuvres, the error is expected to be larger. Now, combining both error budgets, the method in this thesis is expected to outperform this approximation by around 10%.

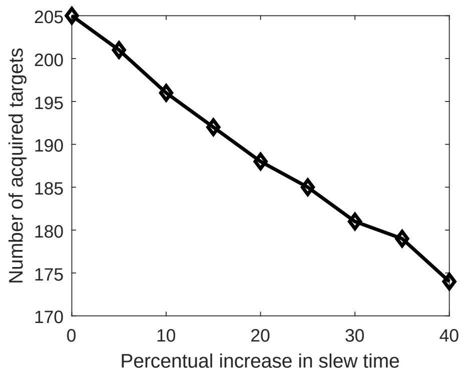


Figure 6.38: Influence of the optimality of the attitude manoeuvres on the scheduling result, given a case of 300 targets over one orbit, solved with the DP(A) algorithm. This figure does not include the 0.5 second margin as introduced in Section 5.3.

7 Simulations

Some practical considerations about the software implementation are given in this chapter. Firstly, some numerical tools are introduced in Section 7.1. Next, the simulation architecture is shown in Section 7.2.

7.1. Numerical tools

This section describes two important numerical tools that will be used to construct the simulation tool. A short introduction about integration and interpolation is given in Sections 7.1.1 and 7.1.2 respectively.

7.1.1. Integrators

Since the equations of motion require to solve an initial value problem (IVP), a numerical integrator is required to estimate the states over time. This is for example required for integration of the kinematics from Section 3.4.3 and for the multiple shooting method from Section 4.3.2. Starting from an initial state vector $\mathbf{x}_0$, a numerical integration can approximate the state at the next time step as (Montenbruck and Gill, 2001)

$$\mathbf{x}(t_0 + h) \approx \mathbf{x}_0 + h\boldsymbol{\Phi} \tag{7.1}$$

where $\boldsymbol{\Phi}$ is an increment function. For example, the Runge-Kutta 4 (RK4) integration scheme is given as

$$\mathbf{x}(t_0 + h) \approx \mathbf{x}_0 + h\boldsymbol{\Phi}_{RK4} = \mathbf{x}_0 + h\frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right) \tag{7.2}$$

Here, h is the step size and the intermediate slopes $\mathbf{k}_1$ to $\mathbf{k}_4$ can be computed as

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}\left(t_0, \mathrm{x}_0\right) \\
\mathbf{k}_2 &= \mathbf{f}\left(t_0 + \frac{h}{2}, \mathbf{x}_0 + h\frac{\mathbf{k}_1}{2}\right) \\
\mathbf{k}_3 &= \mathbf{f}\left(t_0 + \frac{h}{2}, \mathbf{x}_0 + h\frac{\mathbf{k}_2}{2}\right) \\
\mathbf{k}_4 &= \mathbf{f}\left(t_0 + h, \mathbf{x}_0 + h\mathbf{k}_3\right)
\end{aligned} \tag{7.3}$$

Similar expressions exist for higher-order Runge-Kutta integration schemes as shown in (Montenbruck and Gill, 2001, Ch. 4). Furthermore, variable step-size integrators can provide higher efficiency for a desired accuracy. These are however unsuitable for the optimal control integration, but can be used to propagate the position of the satellite.

The Runge-Kutta method is a single-step method. On the other hand, also multi-step integrators exist, which rely on information from previously calculated steps. However, these integrators are typically used when the function derivatives are expensive to evaluate. Furthermore, these methods also require initialisation. Since the dynamics are rather simple and the integration time-span relatively short, the multi-step method offers no advantage for this problem. As a consequence, it was chosen to use the RK4 integrator for the attitude dynamics and the kinematics. An RK8 integrator is used to obtain the satellite orbit.

7.1.2. Interpolation

Throughout this thesis, interpolation has been extensively used. For example, the satellite states are obtained through interpolation during scheduling and also the pseudospectral methods are based on interpolation. Therefore, this section introduces some necessary basic notions about interpolation methods.

Suppose n values $y_i = f(x_i)$ of an unknown function are given over an interval $[x_0, x_n]$. The nodes x_i are ordered such that $x_0 < x_i < x_n$. Moreover, no two nodes x_i are equal. The purpose of interpolation is to construct the interpolant, which is a function passing through the given set of data points. The interpolant $g(x)$ is used to estimate all the function values for the interval, such that $g(x) \approx f(x)$. Two ways of constructing an interpolant are through Lagrange interpolation and spline interpolation. The Lagrange interpolant is a linear combination of the Lagrange basis polynomials ℓ. The specific equations are covered in Section 4.3.4. On the other hand, splines are constructed by connecting a series of polynomials. These piecewise polynomials have to meet certain connecting conditions at the boundary nodes. For example, natural cubic splines are third-order polynomials which ensure that the second derivative of the polynomials is equal at the connecting nodes and equals zero at the end nodes. Next, a linear system of equations can be set-up to solve for the polynomial coefficients (Engeln-Müllges and Uhlig, 1996).

7.2. Simulation architecture

Since the agile satellite scheduling simulation tool requires several modules, it is important to sketch its architecture. The complete software tool which was developed is referred to as IMPULSE (IMage Planning Utility for agiLe SatEllites). The program is divided into modules as shown in Figure 7.1. All code is written in Matlab, except for the parts that require higher computational efficiency. The code is in these cases optimized by using the C language. The function of each module is described below:

- **Observation request simulator**: as explained in Section 6.1.1, a simulation of the incoming request is used as customer input.
- **Orbit propagator:** the cartesian elements of the orbit in the ECI frame are necessary to calculate the opportunity windows for tracking a target. The propagation of the orbit is performed by the internal OHB software FLASH.
- **Target window determination** the observation requests and orbit are used to determine the target opportunity windows as explained in Section 6.1.2.
- **OCP model:** the optimal attitude manoeuvres should be transcribed such that the OCP problem is formulated as an NLP problem as explained in Section 4.2. The OCP problem was set up using CasADi. This software is a general-purpose tool for numerical optimisation, in particular for optimal control. CasADi provides certain “building blocks” to implement a specific-purpose OCP solver efficiently with modest programming effort (Andersson et al., In Press, 2018).
- **NLP solver:** the solver was chosen to be IPOPT, which uses an IP method to solved the NLP problem.
- **Database generation:** as explained in Chapter 5, a database of solutions for the attitude manoeuvre OCP is constructed, such that the optimal attitude manoeuvre duration case approximated very efficiently within the scheduling algorithm.
- **ANN training:** the database is used to train an ANN model, which can approximate the optimal slew times. The ANN training is performed using the deep learning toolbox from Matlab.
- **Scheduler:** this concerns the main algorithm that solves the scheduling problem. Depending on the problem size, the user can select an appropriate scheduling algorithm.
- **Schedule validation:** the output from the schedule algorithm is validated by calculating the actual slew manoeuvres between the scheduled targets using the OCP model. In case a violation is detected due to the limited accuracy of ANN, the actual slew manoeuvre times is determined and used to build the schedule again.

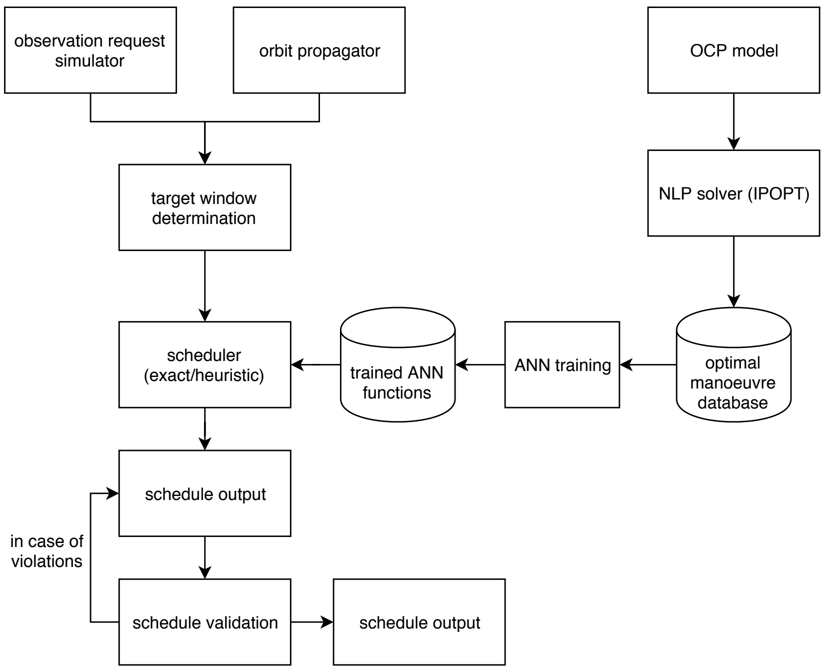


Figure 7.1: Software architecture of IMPULSE.

- **Schedule output:** the output contains the control of the satellite, selection and sequence of the observations and statistics about the scheduling results.

8 Software verification

Many new software tools have been developed during this thesis research. Therefore, it is essential to perform an elaborate verification process on the software. The verification of the optimal guidance software and scheduling algorithms is performed in Sections 8.1 and 8.2 respectively.

8.1. Optimal guidance profile verification

The generation of optimal attitude guidance profiles for a slightly different problem was already verified in Section 4.5.1. There it was shown that the solver produces identical results as in literature for a time-optimal attitude manoeuvre around a single axis. Nevertheless, this case applies different path constraints compared to the attitude manoeuvres in this research and it does not consider a time-dependent final boundary condition. Unfortunately no reference examples were found in literature which consider the full complexity of variable boundary conditions for attitude guidance profiles. Therefore, it was chosen to compare the developed algorithm to an OHB in-house tool which optimizes attitude profiles. Since this verification algorithm solves a fixed-time OCP, the time-optimal manoeuvres are generated using a bisection method such that the final time is decreased until the manoeuvre is not feasible due to constraint violations. Since the developed algorithm introduces the final time as an NLP variable, it can be expected that the profiles generated in this research will slightly outperform the verification algorithm.

The verification is executed for three cases: two short and one long attitude slew for which the input variables are presented in Tables 8.1 and 8.2. The results of the comparison are shown in Table 8.3. As an example, the generated and reference optimal attitude trajectories for case 1 are shown in Figures 8.1 and 8.2 respectively. It can be seen that the developed solver (IMPULSE) slightly outperforms the reference values for each case, as was expected. Furthermore, also the boundary conditions of the quaternions, angular velocities and torques for each case were checked. The difference for all boundaries is in the order of 10^{-8}. This difference has been attributed to the accuracy of the NLP solver, which was set to this tolerance for the reference case.

It is interesting to notice that although the optimal manoeuvre times are comparable, the attitude trajectory and actuator input are significantly different for case 1. The IMPULSE solutions shown more oscillatory behaviour, while the OHB reference case is more smooth. This resembles the conclusion from Section 4.5.1, where it was found that many local optima of equal quality exist for the time-optimal attitude manoeuvres. Another possible explanation is that the reference has no incentive to minimize the final time and has in fact a penalty in the objective function to minimize the energy used for the attitude manoeuvre. This directly leads to less peaks in the torque profile.

Next to the slew between the targets, also the scanning over a target is verified. This was done by importing the scanning attitude profile into the commercial software STK to check whether the target is correctly scanned. By visual inspection, it is examined whether the scanned images match with the required target area. An example of an instantaneous scanning line is shown in Figure 8.3. No discrepancies with the required areas were found. Therefore, also the kinematics of scanning the target are verified.

Table 8.1: Orbital parameters for the attitude manoeuvre validation cases at the center of observing the initial target.

a [m]	e [-]	inc [deg]	RAAN [deg]	AOP [deg]	TA [deg]
$6.8746 \cdot 10^{6}$	6.7910^{-4}	90	0	149.90	235.47

Table 8.2: Input parameters for the attitude manoeuvre validation cases.

Case number	Initial target			Final target		
	latitude [deg]	longitude [deg]	scanning azimuth [deg]	latitude [deg]	longitude [deg]	scanning azimuth [deg]
1	25.1846	-101.2729	0	29.3992	-100.3285	0
2	25.1846	-101.2729	-90	29.3992	-100.3285	90
3	25.1846	-101.2729	45	29.3992	-100.3285	80

Table 8.3: Results of the optimal slew with in-house OHB attitude manoeuvre optimizer.

Case number	OHB [s]	IMPULSE (10th order) [s]	IMPULSE (15th order) [s]
1	17.34	16.81	16.59
2	78.36	77.33	71.30
3	24.61	24.18	23.12

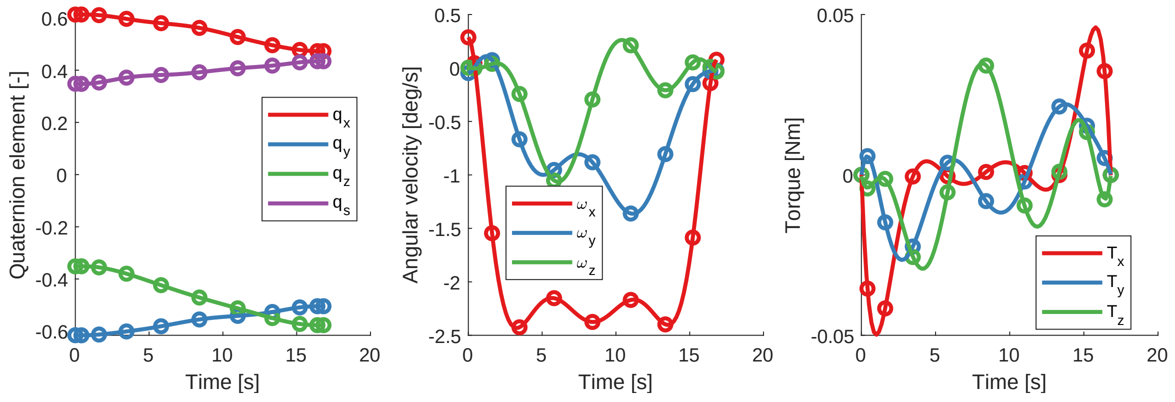


Figure 8.1: Attitude and control states for optimal attitude manoeuvre for case 1 using IMPULSE.

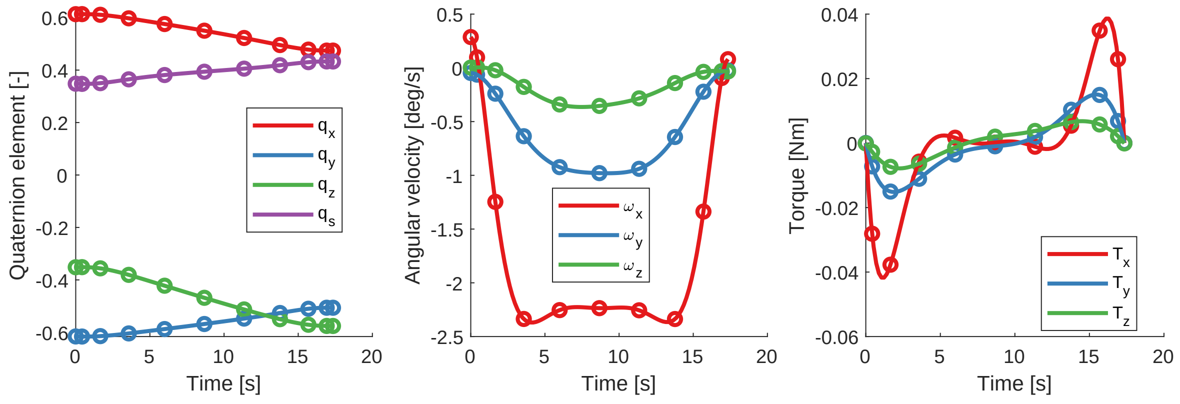


Figure 8.2: Attitude and control states for optimal attitude manoeuvre for case 1 using OHB software.

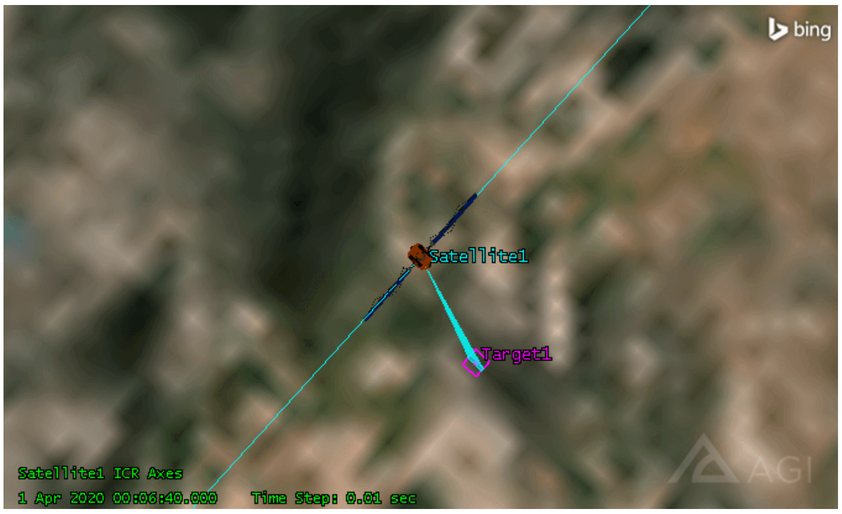


Figure 8.3: Scanning of the initial target of case 1, visualized in STK (Courtesy of AGI).

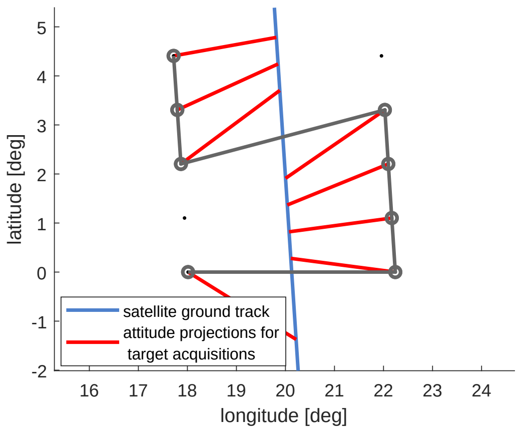


Figure 8.4: Verification case for 10 targets, adopted by removing the last two targets in the sequence of case 3 in Section 6.7.1

8.2. Schedule algorithm verification

The verification of the scheduling algorithm is carried out in two different parts. The first part is the verification of the scheduling with a brute-force solution. Secondly, it is verified whether the target sequences as produced by the scheduling algorithms always satisfy the minimum manoeuvre constraint. That is, whether the approximation by the ANN does not underestimate the required slew time.

The optimality of the scheduling result is validated by a brute-force solution which goes over all possible sequences. This solution is compared to the exact DP solution, which should give the same result. The other algorithms are not considered since these do not guarantee the optimal solution. Now, the brute-force solution is generated for 10 targets, since the computer runs out of memory for larger cases. The same target cases as presented in Section 6.7.1 are used, with the difference that the last two visible targets are removed. The schedule for the remaining 10 targets was found to be always identical. An example case of such a schedule is shown in Figure 8.4. This validation case again demonstrates the efficiency of the exact DP algorithm. Here, the brute-force solution is obtained in 321 seconds, while the exact DP algorithm only required 0.5 seconds. In case enough RAM memory is available, the calculation of a schedule for 12 targets using the brute-force technique would already require half a day of computation time.

As a next step, it is important to validate if the produced schedule is actually flyable by the satellite.

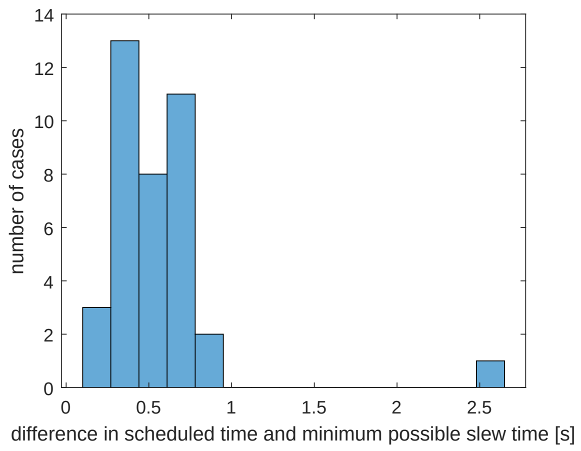


Figure 8.5: Verification of the exact solution of case 1 in Section 6.7.2.

This is done by taking the result from the optimizer and computing the slews in between the scheduled target using the OCP solver as presented in Chapter 4. This procedure is demonstrated here for case 1 from Section 6.7.2. As can been in Figure 8.5, no violations occurred for this case, since the difference in scheduled time is always larger than zero. It is also interesting to note that the outlier at 2.5 seconds is not an error because of the ANN. In this case, the target could not be imaged after a short slew since it was not yet in view. Therefore, the slew takes 2.5 seconds longer than the fastest slew possible.

9 Conclusions and recommendations

This chapter gives the conclusions of this research in Section 9.1 and the recommendations in Section 9.2.

9.1. Conclusions

This section summarizes the results of this research and provides the answers to the research questions, as introduced in Section 1.2.1. The AEOS scheduling optimization problem as treated in this thesis is shown in Section 2.1 to consist of two levels. The high-level problem is the scheduling for observation of the targets and the low-level problem is to find the optimal manoeuvre times between those targets. First, the conclusions about the introduced novelties for the low-level optimization problem are stated below:

- **Kinematics:** agile satellites often use line scanners for high image quality. These scanners require that the velocity vector of the scanner ground track is perpendicular to the scanning line. The derivation of a new ODE for the kinematics to scan the target accordingly is given in Section 3.4. The proper functioning of this ODE has been verified.
- **Optimal guidance profiles:** the time-optimal manoeuvres between the targets are complicated because of the time-dependent final boundary condition. Since the final boundary is the solution to an ODE, the final boundary condition is obtained by constructing an interpolant which depends on the final time. The optimal guidance profiles could be obtained robustly using the IPOPT solver and CGL pseudospectral collocation as transcription method. It was found that the guidance profile of the time-optimal manoeuvres is typically not a bang-bang solution, except for short attitude manoeuvres. Instead, the introduction of the ellipsoidal constraints on the angular velocity and torque of the satellite results in nonlinear behaviour for the torque input profile.
- **Database generation:** a database of optimal attitude manoeuvres is generated, which required a fast and robust solution to the OCP. Therefore, a new shape-based method for optimal attitude manoeuvres was proposed as initial guess for the solver. Together with other robustness enhancements, a database of 600,000 solutions of the OCP could be generated in five days of computation time on a desktop.
- **Optimal manoeuvre time approximation:** since the time-optimal manoeuvres must be solved many times within the scheduling algorithm and the solution to the optimal manoeuvres is too computationally expensive, an approximation of the optimal manoeuvres that can be solved much faster is necessary. During this research, a large effort went into finding an approximation which can cover all the complexities of the attitude manoeuvres. It was concluded to construct a database of solutions, depending on nine input parameters, and obtain the approximated manoeuvre times by training a neural network. The best performance was obtained using a network of two hidden layers with 50 neurons, which is trained using the Levenberg-Marquardt algorithm with Bayesian regularization. Furthermore, several neural networks are specifically trained for overlapping subsets of the complete dataset to increase the accuracy. The optimal attitude manoeuvre times could be approximated within a maximum error of 1 second. Furthermore, the function to obtain the approximated slew can be evaluated 200,000 times per second.

The conclusions about the high-level scheduling problem are as follows:

- **Assumptions:** it was decided to discard the observation time as decision variable. Furthermore, it was found that including the scanning azimuth angle as a decision variable offers a limited improvement of 1-2% to scheduling result. On the other hand, when leaving out these variables, the computation time can be reduced by a significant amount. The remaining decision variables, as used in this research, are the selection of targets, the sequence of targets and their scanning direction.
- **Exact dynamic programming:** while previous work has shown difficulties solving instances of more than 12 targets, this research proposes a novel exact algorithm using the dynamic programming framework which can handle larger problem sizes. The algorithm was shown to be able to solve up to 100 targets up to optimality for a restricted number of visible targets per epoch. Also, the cases of 12 targets could always be solved under two minutes for the most difficult cases. Therefore, this should not be considered as a difficult problem anymore.
- **Scheduling algorithms comparison:** four different scheduling algorithms are implemented: a simulated annealing algorithm, an exact dynamic programming algorithm, an approximate dynamic programming algorithm and a greedy algorithm. For cases up to 12 targets, it is recommended to use the SA or DP(E) algorithm. For a large problem size of 50 targets, the greedy algorithm has a remarkable good performance while being very fast. On the other hand, the simulated annealing algorithm and approximate dynamic programming algorithm can improve the objective value up to 10% at a higher computational cost. Looking at even larger problem instances of up to 500 targets, the approximate dynamic programming outperforms the other algorithms by up to 10%, while the algorithm is still able obtain a solution within 30 seconds for up to 500 targets. Therefore, this algorithm should be preferred for larger problem sizes.
- **Scheduling improvements:** the implementation of the approximate guidance profiles provides an estimated improved scheduling performance of approximately 10%. This improvement is possible since the time-dependency of the manoeuvres is considered, and the optimal manoeuvres can be successfully approximated using the artificial neural network.

9.2. Recommendations

As for any research, answering a research question raises many new questions. An overview of recommendations for further research is listed below:

- **Uncertainty:** any kind of uncertainty is left out of consideration. However, several uncertainties, most notably from weather forecasts, have a considerable impact on the best schedule in practice. Uncertainties from weather forecasts could be used to obtain an optimal schedule with maximum likelihood. In this case, the areas where clouds prevent the image from being observed should also be considered as an additional constraint.
- **On-board scheduling:** for this research, the scheduling happens on ground and the activity plan is sent to the satellite on a daily basis. When clouds could obstruct the target visibility, this problem can be averted by using a scanner on-board to detect the clouds and implement an autonomous algorithm which can reschedule the targets based on the sensor data. One of the challenges of such an autonomous algorithm would be that the schedule needs to be updated very fast with only little computational resources. Also, it is difficult to implement optimal guidance profiles for on-board scheduling. Still, for some business models, it might not be acceptable to reschedule targets on-board after it was scheduled for the customer.
- **Optimal control:** so far, only the guidance profiles for the satellite attitude are used. A more realistic simulation of the actual attitude profile in orbit could be obtained using closed loop control. Several techniques such as model predictive control or a linear quadratic regulators could be explored. Even more, this may impact the vibration analysis made in Section 4.5.3, since control may induce other vibrations. An important advantage of this method is that the stabilization time can be estimated based on the settling time of the optimal control profile. So instead of assuming a constant stabilization, only the minimum necessary stabilization time is considerd.

- **Satellite operational modes:** currently, the target sequence is optimized for the case when the satellite is in 'target acquisition mode'. However, when no targets are being observed, the satellite is active in other modes such as the Sun pointing mode or the ground contact mode. Depending on the selection of the scanning azimuth angle $\alpha_{s,0}$ and the scheduling of the sequence, the slew that is required to switch to another operational mode will be influenced. In order to further optimize the operational use of the satellite, the slews to these other models should also be considered at the start and the end of the target sequence. As an example, a slew from the Sun to a first target in the sequence could possibly take 30 seconds longer depending on which angle $\alpha_{s,0}$ is selected. Therefore, it is important to consider this decision variable as well for the scheduling optimization in this case.

- **Multi-orbit/constellation analysis:** all targets in the research have a single visibility window. However if the scheduling is considered over multiple orbits or if a constellation of satellites is used, the target can be visible for multiple time windows. Exploring this option would unfortunately greatly increase the search space.

- **Complex image products:** the analysis should be extended for certain other image products such as custom areas or stereoscopic requests. These kind of image products were already considered by Lemaître et al. (2002), but the techniques in this paper can not be directly applied for the newly developed algorithms in this thesis.

- **DP bounds:** the newly developed exact algorithm in this research considers all possible paths to find the optimal solution, since it is based on the Held-Karp algorithm. However, also an approximate technique exists, which is based on the DP principles: the Held-Karp lower bound estimation (Held and Karp, 1971). As a maximization problem is considered here, this means that an estimated bound of the maximum possible profit of the scheduling can be obtained with this technique. On the other hand, the heuristics presented in this thesis can give an estimated lower bound of the optimal results. By reducing the gap between those two bounds until they match, optimality can be proven without using an exact algorithm. A first demonstration of this approach for the AEOS scheduling problem, by using a relaxation of the constraints, is given by Habet et al. (2010). However, the implementation of a high-quality upper bound estimation for the AEOS scheduling problem is expected to be very challenging. Still, this is a promising approach since it could be directly implemented in the exact dynamic programming algorithm of this research. The latter can not efficiently deal with the case when many targets are seen at the same time. For these time windows, the performance could be increased by using the DP bounds. Considering that the TSP has been solved for thousands of cities using this approach, a successful implementation is expected to be able to generate optimal schedules for much larger instances.

- **Approximate algorithm with time windows:** similar as for the exact dynamic programming, a new algorithm could be constructed which solves the travelling salesman problem locally for different time windows with meta-heuristics. Since the SA algorithm was shown to be powerful for small problem instances, this approach could be taken as first try to solve the local problems. The developed techniques for the exact dynamic programming can not be applied, such that new strategies are required.

- **Additional constraints:** the memory (ground station contacts) and energy constraints are neglected in this research. Although the memory constraint is typically not critical, the ground stations contacts are essential to deliver the images in time to the customer. Therefore, a model to optimize the system response time can be of interest for example in case of disaster management. Also the influence of introducing energy constraints should be investigated.

- **Sequence scanning angle:** the scanning angles was taken in this research based on a fast acquisition or energy efficient mode. Especially for short target sequences, optimizing the scheduling while including decision variable $\alpha_{s,0}$ can lead to potential improvements.

A Dimensional cuts in database

This appendix chapter contains a series of 2-dimensional cuts in the complete database of optimal control solutions. By reducing the dimensionality, some insight about the relation between the parameters can be gained. The following conclusions can be made about the presented cases:

- **Case 1:** clearly, if the off-nadir direction has a large influence on the optimal slew time. Even more, also the azimuth of the target has a non-symmeterical influence due to the relative motion of the satellite w.r.t the target.
- **Case 2:** compared to case 1, changing the scanning direction considerably changes the optimal slew time.
- **Case 3:** compared to case 1, the influence of the AOP is rather small, but should still be considered. Differences in optimal slew time of around 0.5 seconds are observed w.r.t. case 1.
- **Case 4:** in this case the next target is laying 'ahead' of the previous target. The optimal slew times never exceed 20 seconds.
- **Case 5:** in this case the next target is laying 'behind' the previous target. The optimal slew times can exceed 35 seconds.
- **Case 6:** if the difference in azimuth scanning direction angle is large, this value will largely influence the optimal slew times.

Table A.1: Parameters settings for dimensional cut cases. The parameters which are taken as variables are indicated by 'v'.

Case number	$\boldsymbol{\eta}_0$ **[deg]**	$\boldsymbol{\alpha}_{t,0}$ **[deg]**	$\boldsymbol{\alpha}_{s,0}$ **[deg]**	**Initial scanning direction (F/B)**	$\boldsymbol{\eta}_f$ **[deg]**	$\boldsymbol{\alpha}_{t,f}$ **[deg]**	$\Delta\boldsymbol{\alpha}_s$ **[deg]**	**Final scanning direction (F/B)**	$\boldsymbol{\nu}_0$ **[deg]**
1	v	v	0	F	0	0	0	F	0
2	v	v	0	F	0	0	0	B	0
3	v	v	0	F	0	0	0	F	90
4	v	v	0	F	30	0	0	F	0
5	v	v	0	F	30	180	0	F	0
6	v	v	0	F	0	0	90	F	0

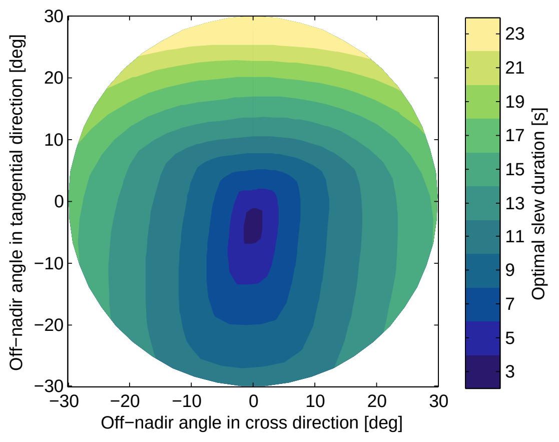


Figure A.1: Case 1 of the dimensional cuts in the database.

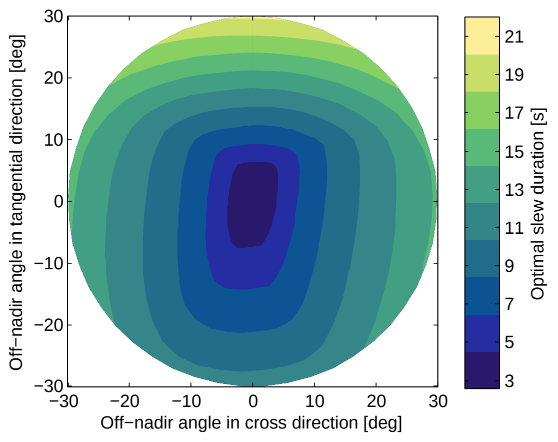


Figure A.2: Case 2 of the dimensional cuts in the database.

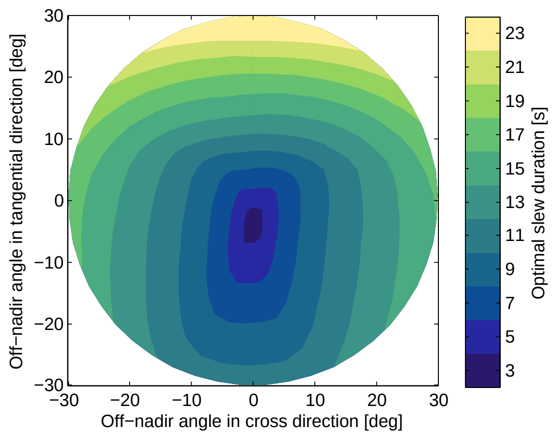


Figure A.3: Case 3 of the dimensional cuts in the database.

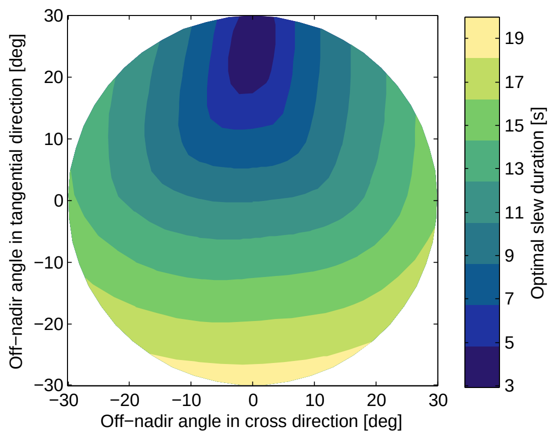


Figure A.4: Case 4 of the dimensional cuts in the database.

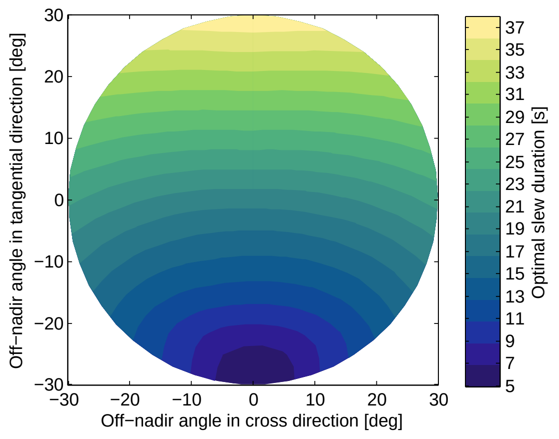


Figure A.5: Case 5 of the dimensional cuts in the database.

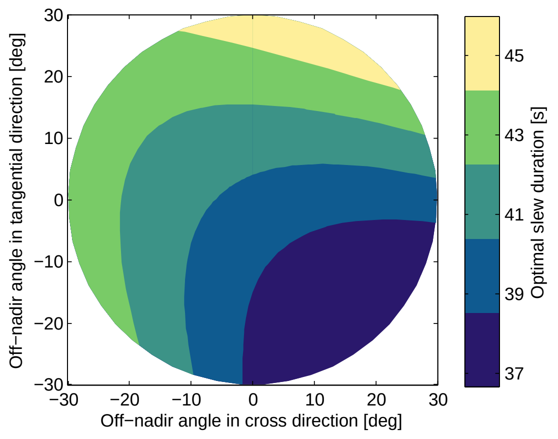


Figure A.6: Case 6 of the dimensional cuts in the database.

B
Sparse grid interpolation

The following introduction about sparse grids is based on (Garcke and Griebel, 2012, p. 11-12). The sparse grid modelling is established by piecewise multilinear basis functions. The basis function $\phi(x)$ which is typically used for the sparse grid method is the standard hat function, which is defined as follows

$$\phi(x) = \begin{cases} 1 - |x| & \text{if } x \in [-1, 1] \\ 0 & \text{otherwise} \end{cases} \tag{B.1}$$

The above equation holds for a basis function of level 1. This basis function is adopted to generate the general one-dimensional basis functions $\phi_{l,i}(x)$ using translation and dilation:

$$\phi_{l,i} = \phi(2^l x - i),\ l, i \in \mathbb{N} \tag{B.2}$$

Here, the index l refers to the level and i indicates the location of the basis function on the grid. This can again be extended, such that the basis functions in dimension d are defined as

$$\phi_{\mathbf{l},\mathbf{i}} = \prod_{j=1}^{d} \phi_{l_j,i_j}(x_j) \tag{B.3}$$

For convenient indexing, the vectors **l** = $[l_1, \cdots, l_d]$ and **i** = $[l_i, \cdots, i_d]$ are used here. Using these basis functions, the function space V_l can now be defined when combining functions $\phi_{l,i}(x)$ as:

$$V_\mathbf{l} = \text{span}\left\{\phi_{\mathbf{l},\mathbf{i}} :\ 1 \leq \mathbf{i} \leq 2^\mathbf{l} - 1\right\} \tag{B.4}$$

Next, also a hierarchical increment function space W_l can be defined as follows:

$$W_l = \text{span}\left\{\phi_{l,i} :\ 1 \leq i \leq 2^l - 1,\ i \text{ odd}\right\} \tag{B.5}$$

Finally, using these definitions, the sparse grid method can be defined. The sparse grid space $\hat{V}_n^S$ of level n in dimension d is defined as the direct sum of the subspaces:

$$V_n^S = \bigoplus_{\sum_{i=1}^{d}(l_i) \leq n+d-1} W_\mathbf{l} \tag{B.6}$$

This method is mathematically constructed by an optimization of the number of degrees of freedom with respect to the resulting accuracy. A proof can be found in (Griebel, 2005).This can be compared to the full grid space V_n:

$$V_n = \bigoplus_{\max_i(l_i) \leq n} W_\mathbf{l} \tag{B.7}$$

To demonstrate how the basis functions are used for interpolation, a one-dimensional example in Figure B.3 shows the appoximation of a parabola by the hierarchical basis W_3.

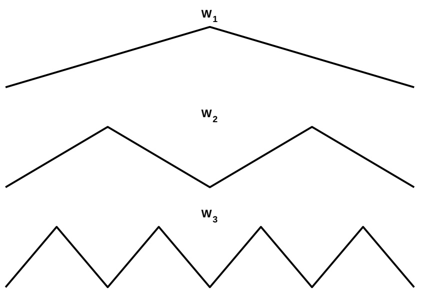


Figure B.1: Piecewise hierarchical basis for level 1, 2 and 3.

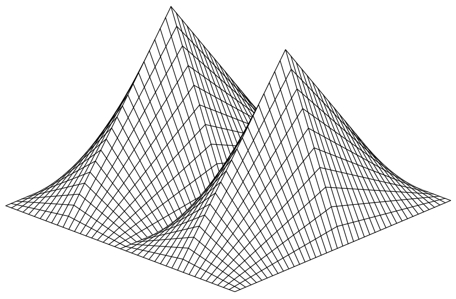


Figure B.2: Piecewise bilinear basis function $W_{2,1}$

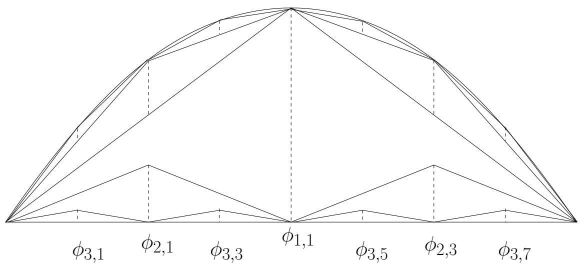


Figure B.3: Interpolation of a parabola using W_3 (Garcke and Griebel, 2012, p. 62).

Now that the sparse grid space is defined, it is still possible to select how the indices i, as introduced in Equation B.2, will be structured. Klimke and Wohlmuth (2005) demonstrate that a Clenshaw-Curtis grid is generally a good choice. Only if very strict tolerances are set, more advance grid types are recommend. As shown in Figures B.4 and B.5, the Clenshaw-Curtis grid has equidistant sampling, including nodes at the boundaries.

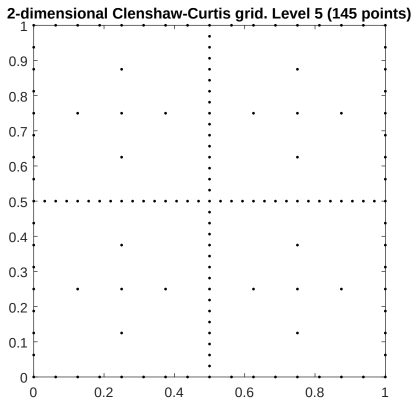


Figure B.4: 2-dimensional Clenshaw-Curtis sparse grid.

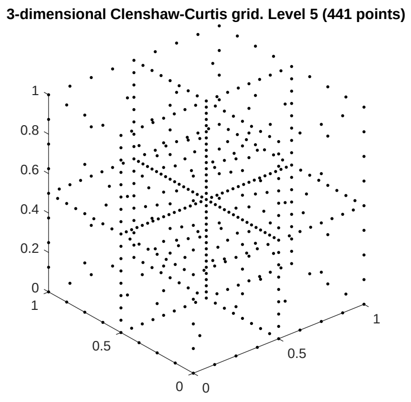


Figure B.5: 3-dimensional Clenshaw-Curtis sparse grid.

C Example solution of TSP using DP

This chapter covers an example of the DP implementation of the TSP, as support of Section 6.5.1. The theory is applied to the case of AEOS scheduling where the time window constraints are removed. The objective is to find the sequence of the four given targets, which gives the minimal time to observe all of them. First, consider the matrix $d_{\min}$, which contains the optimal slew times between each targets, such that $d_{\min}(T_i, T_j)$ gives the slew from target T_i to target T_j:

$$d_{\min} = \begin{bmatrix} 0 & 1.5 & 8.6 & 15.3 \\ 1.2 & 0 & 6.0 & 4.1 \\ 10.3 & 7.1 & 0 & 9.9 \\ 8.3 & 3.0 & 11.8 & 0 \end{bmatrix} \tag{C.1}$$

As opposed to the classical TSP, this is an asymmetric matrix, which is considered in the general formulation of Algorithm 2. The formulation also slightly deviates from the TSP as the last target is not connected to the first target. Now, it can be shown that $2^n - 1 = 15$ different subsets can be found using the four targets. First, the smallest subsets are considered, which consists of a single target. it is assumed that all targets have the same starting time, which is zero. Therefore, using the notation as was used in Section 6.5.1, the resulting arrays for the smallest subsets is:

$$\begin{aligned} p(\{1\}, 1) &= 0 \\ p(\{2\}, 2) &= 0 \\ p(\{3\}, 3) &= 0 \\ p(\{4\}, 4) &= 0 \end{aligned} \tag{C.2}$$

Now these first nodes are used to construct the subsets consisting of two elements. Taking the first subset $\{1, 2\}$ and the associated values in the profit matrix p:

$$\begin{aligned} p(\{1,2\}, 1) &= p(\{2\}, 2) + d_{\min}(T_2, T_1) = 1.2 \\ p(\{1,2\}, 2) &= p(\{1\}, 1) + d_{\min}(T_1, T_2) = 1.5 \end{aligned} \tag{C.3}$$

If a single element is excluded from the set $\{1, 2\}$, only a single permutation of the resulting subset exists. Therefore, there is only a single previous target possible, such that $m(\{1,2\}, 1) = 2$ and $m(\{1,2\}, 2) = 1$. For the remaining subsets with two elements:

$$\begin{aligned}
p(\{1,3\},1) &= p(\{3\},3) + d_{\min}(T_3,T_1) = 10.3 \\
p(\{1,3\},3) &= p(\{1\},1) + d_{\min}(T_1,T_3) = 8.6 \\
p(\{1,4\},1) &= p(\{4\},4) + d_{\min}(T_4,T_1) = 8.3 \\
p(\{1,4\},4) &= p(\{1\},1) + d_{\min}(T_1,T_4) = 15.3 \\
p(\{2,3\},2) &= p(\{3\},3) + d_{\min}(T_3,T_2) = 7.1 \\
p(\{2,3\},3) &= p(\{2\},2) + d_{\min}(T_2,T_3) = 6.0 \\
p(\{2,4\},2) &= p(\{4\},4) + d_{\min}(T_4,T_2) = 3.0 \\
p(\{2,4\},4) &= p(\{2\},2) + d_{\min}(T_2,T_4) = 4.1 \\
p(\{3,4\},3) &= p(\{4\},4) + d_{\min}(T_4,T_3) = 11.8 \\
p(\{3,4\},4) &= p(\{3\},3) + d_{\min}(T_3,T_4) = 9.9
\end{aligned} \tag{C.4}$$

Next, consider the subsets with three elements:

$$\begin{aligned}
&p(\{1,2,3\},1) = \min\{p(\{2,3\},2) + d_{\min}(2,1), p(\{2,3\},3) + d_{\min}(3,1)\} = \min\{7.1 + 1.2, 6.0 + 10.3\} = 8.3 \\
&p(\{1,2,3\},2) = \min\{p(\{1,3\},1) + d_{\min}(1,2), p(\{1,3\},3) + d_{\min}(3,2)\} = \min\{10.3 + 1.5, 8.6 + 7.1\} = 11.8 \\
&p(\{1,2,3\},3) = \min\{p(\{1,2\},1) + d_{\min}(1,3), p(\{1,2\},2) + d_{\min}(2,3)\} = \min\{1.2 + 8.6, 1.5 + 6.0\} = 7.5 \\
&p(\{1,2,4\},1) = \min\{p(\{2,4\},2) + d_{\min}(2,1), p(\{2,4\},4) + d_{\min}(4,1)\} = \min\{3.0 + 1.2, 4.1 + 8.3\} = 4.2 \\
&p(\{1,2,4\},2) = \min\{p(\{1,4\},1) + d_{\min}(1,2), p(\{1,4\},4) + d_{\min}(4,2)\} = \min\{8.3 + 1.5, 15.3 + 3.0\} = 9.8 \\
&p(\{1,2,4\},4) = \min\{p(\{1,2\},1) + d_{\min}(1,4), p(\{1,2\},2) + d_{\min}(2,4)\} = \min\{1.2 + 15.3, 1.5 + 4.1\} = 5.6 \\
&p(\{1,3,4\},1) = \min\{p(\{3,4\},2) + d_{\min}(2,1), p(\{3,4\},4) + d_{\min}(4,1)\} = \min\{11.8 + 1.2, 9.9 + 8.3\} = 13.0 \\
&p(\{1,3,4\},3) = \min\{p(\{1,4\},1) + d_{\min}(1,3), p(\{1,4\},4) + d_{\min}(4,3)\} = \min\{8.3 + 8.6, 15.3 + 11.8\} = 16.9 \\
&p(\{1,3,4\},4) = \min\{p(\{1,3\},1) + d_{\min}(1,4), p(\{1,3\},3) + d_{\min}(3,4)\} = \min\{10.3 + 15.3, 8.6 + 9.9\} = 18.5 \\
&p(\{2,3,4\},2) = \min\{p(\{3,4\},3) + d_{\min}(3,2), p(\{3,4\},4) + d_{\min}(4,2)\} = \min\{11.8 + 7.1, 9.9 + 3.0\} = 12.9 \\
&p(\{2,3,4\},3) = \min\{p(\{2,4\},2) + d_{\min}(2,3), p(\{2,4\},4) + d_{\min}(4,3)\} = \min\{3.0 + 6.0, 4.0 + 11.8\} = 9.0 \\
&p(\{2,3,4\},4) = \min\{p(\{2,3\},2) + d_{\min}(2,4), p(\{2,3\},3) + d_{\min}(3,4)\} = \min\{7.1 + 4.1, 6.0 + 9.9\} = 11.2
\end{aligned} \tag{C.5}$$

Finally, the following numbers can be obtained for the largest subsets:

$$\begin{aligned}
p(\{1,2,3,4\},1) &= \min\{12.9 + 1.2, 9.0 + 10.3, 11.2 + 8.3\} = 14.1 \\
p(\{1,2,3,4\},2) &= \min\{13.0 + 1.5, 16.9 + 7.1, 18.5 + 3.0\} = 14.5 \\
p(\{1,2,3,4\},3) &= \min\{4.2 + 8.6, 9.8 + 6.0, 5.6 + 11.8\} = 12.8 \\
p(\{1,2,3,4\},4) &= \min\{8.3 + 15.3, 11.8 + 4.1, 7.5 + 9.9\} = 15.9
\end{aligned} \tag{C.6}$$

Since this problem does not go back to the first node, one obtains four possible optimal tours, depending on which node is taken as the first or last one. Since the objective is smallest when finishing with T_3, the tour with final cost 12.8 is selected. Now that all 15 subsets are enumerated and the best tour is selected, the solution is found by backtracking. The optimal sequence is $4 \rightarrow 2 \rightarrow 1 \rightarrow 3$.

Bibliography

J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.

S. Augenstein, A. Estanislao, E. Guere, and S. Blaes. Optimal scheduling of a constellation of earth-imaging satellites, for maximal data throughput and efficient human management. In *Proceedings of the Twenty-Sixth International Conference on International Conference on Automated Planning and Scheduling*, pages 345–352, London, UK, 2016. AAAI Press.

G. Beaumet, G. Verfaillie, and M. Charmeau. Estimation of the minimal duration of an attitude change for an autonomous agile earth-observing satellite. In *Principles and Practice of Constraint Programming – CP 2007*, pages 3–17, Berlin, Heidelberg, 2007. Springer.

J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21:193 – 207, 1998.

J. T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Cambridge University Press, New York, USA, 2nd edition, 2009.

K. D. Bilimoria and B. Wie. Time-optimal three-axis reorientation of rigid spacecraft. *Journal of Guidance, Control, and Dynamics*, 16:446 – 452, 1993.

P. Bouman, N. Agatz, and M. Schmidt. Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, 72(4):528–542, 2018.

A. E. Bryson and Y. C. Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Taylor & Francis Group, New York, USA, 1975.

C. Büskens and D. Wassel. The ESA NLP solver WORHP. In *Modeling and Optimization in Space Engineering*, volume 47, pages 99 – 131. Springer, New York, 2012.

A. Caubet and J. Biggs. Optimal attitude motion planner for large slew maneuvers using a shape-based method. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, Boston, USA, 2013.

X. Chu, C. Yuning, and X. Lining. A branch and bound algorithm for agile earth observation satellite scheduling. *Discrete Dynamics in Nature and Society*, 137:415 – 433, 2017.

B. Conway. *Spacecraft Trajectory Optimization*. Cambridge Aerospace Series. Cambridge University Press, Cambridge, 2010.

J.F. Cordeau and G. Laporte. Maximizing the value of an earth observation satellite orbit. *Journal of the Operational Research Society*, 56(8):962–968, 2005.

G. Denis, A. Claverie, X. Pasco, J.P. Darnis, B. de Maupeou, M. Lafaye, and E. Morel. Towards disruptions in earth observation? New earth observation systems and markets evolution: Possible scenarios and impacts. *Acta Astronautica*, 137:415 – 433, 2017.

G. El-baghdady and M.S. El–Azab. Chebyshev–Gauss–Lobatto pseudo–spectral method for one–dimensional advection–diffusion equation with variable coefficients. *Sohag Journal of Mathematics*, 3:7–14, 01 2016.

G. Engeln-Müllges and F. Uhlig. *Numerical Algorithms with C*. Springer, Berlin, Heidelberg, 1996.

F. D. Foresee and M.T. Hagan. Gauss-Newton approximation to Bayesian learning. In *IEEE International Conference Neural Networks*, volume 3, pages 1930 – 1935, Houston, USA, 1997.

J. Garcke and M. Griebel. *Sparse Grids and Applications*. Springer, Bonn, Germany, 2012.

D. Garg. *Advances in global pseudospectral methods for optimal control*. PhD thesis, University of Florida, 2011.

P. E. Gill, M. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *Society for Industrial and Applied Mathematics*, 47:99 – 131, 2005.

A. Globus, J. Crawford, J. Lohn, and A. Pryor. A comparison of techniques for scheduling earth observing satellites. In *Proceedings of the 16th Conference on Innovative Applications of Artifical Intelligence*, pages 836–843, San Jose, California, 2004. AAAI Press.

R. Grasset-Bourdel, G. Verfaillie, and A. Flipo. Building a really executable plan for a constellation of agile earth observation satellites, 2011.

M. Griebel. *Sparse grids and related approximation schemes for higher dimensional problems*. Foundations of Computational Mathematics - FoCM, Bonn, Germany, 2005.

D. Habet, M. Vasquez, and Y. Vimont. Bounding the optimum for the problem of scheduling the photographs of an agile earth observing satellite. *Computational Optimization and Applications*, 47: 307–333, 10 2010.

M. T. Hagan, H. B. Demuth, M. H. Beale, O. De Jesús, and M. T. Hagan. *Neural Network Design*. Martin Hagan, USA, 2nd edition, 2014.

L. He, X. Liu, G. Laporte, Y.-W. Chen, and Y. Chen. An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. *Computers & Operations Research*, 100:45 – 52, 2018.

M. Held and R. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.

M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees: Part ii. *Mathematical Programming*, 1(1):6–25, 1971.

A. Israr. Vibration and modal analysis of low earth orbit satellite. *Shock and Vibration*, 2014(1):1–8, 2014.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 (4598):671–680, 1983.

A. Klimke and B. Wohlmuth. Algorithm 847: Spinterp: Piecewise multilinear hierarchical sparse grid interpolation in Matlab. *ACM Trans. Math. Softw.*, 31:561–579, 12 2005.

A.R. Klumpp. Singularity-free extraction of a quaternion from a direction-cosine matrix. *Journal of Spacecraft and Rockets*, 13(12):754 – 755, 2002.

T.N. Krishnamurti, L. Stefanova, and V. Misra. *Tropical Meteorology: An Introduction*. Springer, New York, 2013.

V. Kurkova. Kolmogorov's theorem and multilayer neural networks. *Jneural networks -Oxford-*, 5(2): 501, 1992.

M. Lemaître, G. Verfaillie, F. Jouhaud, J. Lachiver, and N. Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6(5):367 – 381, 2002.

X. Liu, G. Laporte, Y.-W. Chen, and R. He. An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time. *Computers & Operations Research*, 86, 04 2017.

J. Lu, Y. Chen, R. He, and Y.-W. Chen. A simple and fast heuristic algorithm for time-dependent AEOS scheduling problem. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 07 2018.

D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1991.

O. Montenbruck and E. Gill. *Satellite Orbits: Models, Methods and Applications*. Springer, Berlin, Germany, 2nd edition, 2001.

S. Nag, A. S. Li, and J. H. Merrick. Scheduling algorithms for rapid imaging using agile cubesat constellations. *Advances in Space Research*, pages 891 – 913, 2017.

C. Oguz, F. S. Salman, and Z. B. Yalçın. Order acceptance and scheduling decisions in make-to-order systems. *International Journal of Production Economics*, 125(1):200 – 211, 2010.

K. D. Palagachev and M. Gerdts. *Numerical Approaches Towards Bilevel Optimal Control Problems with Scheduling Tasks*, pages 205–228. Springer International Publishing, Cham, 2017.

G. Peng, P. Vansteenwegen, X. Liu, L. Xing, and X. Kong. An iterated local search algorithm for agile earth observation satellite scheduling problem. In *2018 SpaceOps Conference*, Marseille, France, 2018. American Institute of Aeronautics and Astronautics.

D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. PhD thesis, Institut für Informatik, Technische Universität München, 2010.

J. J. Qu, W. Gao, M. Kafatos, R. E. Murphy, and V. V. Salomonson. *Earth science satellite remote sensing: Vol. 1: Science and instruments*. Springer, 2006.

A. V. Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135, 01 2010.

C. E. Rasmussen. *Gaussian Processes in Machine Learning*, pages 63–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

Y. She, S. Li, and Y. Zhao. Onboard mission planning for agile satellite using modified mixed-integer linear programming. *Aerospace Science and Technology*, 72:204–216, 2018.

A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14 (3):199–222, 2004.

E.-G. Talbi. *Metaheuristics: From Design to Implementation*. John Wiley & Sons, New Jersey, 2009.

O. Tekinalp, T. Elmas, and I. Yavrucuk. Gimbal angle restricted control moment gyroscope clusters. In *Recent Advances in Space Technologies*, pages 585 – 590, Istanbul, Turkey, 2009.

L. N. Trefethen. *Spectral Methods in MatLab*. Society for Industrial and Applied Mathematics, Oxford, 2000.

G. Verfaillie and M. Lemaître. Selecting and scheduling observations for agile satellites: Some lessons from the constraint reasoning community point of view. In *Principles and Practice of Constraint Programming — CP 2001*. Springer, Berlin, 2001.

K. F. Wakker. *Fundamentals of Astrodynamics*. Institutional Repository, Delft University of Technology, Delft, The Netherlands, 2015.

B. Wie. *Space Vehicle Dynamics and Control*. AIAA Education Series. American Institute of Aeronautics and Astronautics, Ames, Iowa, 2nd edition, 2008.

A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25 – 57, 2006.

R. Xu, H. Chen, X. Liang, and H. Wang. Priority-based constructive algorithms for scheduling agile earth observation satellites with total priority maximization. *Expert Systems with Applications*, 51, 01 2016.