SCALABLE TRAJECTORY OPTIMIZATION BASED ON BÉZIER CURVES

T. Jusko, *Student of aerospace engineering, Technische Universität Braunschweig, DE*

E. Stoll, *Institut für Luft- und Raumfahrtsysteme, Technische Universität Braunschweig, DE*

Abstract

As a result of the increasing number of satellites in Earth orbit and the continuous rise of potential for bigger payloads, notably due to the accelerated growth of the private sector, there are many new application areas. An important step for effectively using a multitude of satellites in a large-scale project is the navigational autonomy, meaning that the calculation and optimization of the trajectory takes place on-board the spacecraft. There are already semi-autonomous micro-satellites in operation like the PROBA satellite (ESA) which can complete tasks like attitude control, target identification during fly-by as well as taking pictures. These kind of micro-satellites are specially designed and developed for autonomous service. If the autonomy of trajectory calculation could be realised onboard a spacecraft which where not explicitly designed for this mode of operation, tasks like on-orbit-assembly (OOA) or on-orbit-servicing (OOS) could be solved flexibly and with high efficiency. The limiting factor is the low processing power of spaceborne microprocessors which require scalable algorithms to solve the calculations to a sufficient degree of accuracy. For this reason a new approach to describing position and attitude via Bézier curves is examined in this paper. The motivation comes from Bézier curves having a low computing time due to their mathematically simple structure, which is why they are being used in computer-aided design (CAD) programs and in general computer graphics almost exclusively. Another important part of calculation are the boundary conditions which ensure the protection of the instruments, avoiding collisions as well as aligning tools and cameras at specific targets during fly-by. The objective is to keep the processing power of the optimization process under the given constraints as low as possible by using Bézier curves to describe the trajectory. Furthermore, an analysis of the model regarding the scalability of computing time vs. solution accuracy was conducted in order to examine the flexibility of the approach and if it would be possible to extend the range of application to other hardware combinations. The analysis of three different optimization methods in combination with the model shows that there is a potential for on-board applications.

Glossary

C_j	Weigh values for Gaussian quadrature
FZ	Forbidden zone
$L(f)$	Gaussian quadrature of function f
MZ	Mandatory zone
N	Resolution, number of evaluations for $t \in [0,1]$
$TolCon$	Error margin in boundary conditions
$TolFun$	Difference in object function values per iteration
$TolX$	Difference between variables per iteration
Δt	Step size
$\beta_{i,n}$	i-th Bernstein polynomial of degree n
$\boldsymbol{c}(t)$	Bézier curve of translation
$\boldsymbol{h}_m$	Position of the m-th obstacle
$\boldsymbol{p}_i$	Control points of Bézier curve
$\boldsymbol{r}_{forb.}(t)$	Normal vector from chaser to sun
$\boldsymbol{r}_{tar.}(t)$	Normal vector from chaser to target
δ_i	Angle between chaser axis i and observer
ϵ_i	Angle between chaser axis i and target
γ_i	Angle between chaser axis i and light source
$\hat{x}, \hat{y}, \hat{z}$	Coordinate system of chaser satellite
$\mathbb{H} \otimes \mathbb{H} \rightarrow \mathbb{H}$	Quaternion product
$\mathbf{q}(t)$	Bézier quaternion curve of rotation
$\mathbf{q}_i$	Control points of Bézier quaternion curve
τ_j	Abscissa values for Gaussian quadrature
t	Control variable of Bézier curves

1 INTRODUCTION

The classical approach using a mission control centre to send commands via ground stations is a very robust and safe but also a resource heavy and limited way to operate unmanned spacecraft. With rising numbers of active satellites and space debris in Earth orbit, the complexity of the manoeuvres increases and the notion of autonomous navigation and mission execution becomes increasingly interesting and is currently an active field of research.

There are many studies and concepts regarding various aspects of autonomy of spacecraft. The focus of research has been mainly on the autonomy of detecting targets, readjusting instruments and diagnostic of dysfunctions instead of calculating and performing orbit manoeuvres. One of the most recent projects regarding the autonomy of detecting targets is the SONATE project by the University of Wuerzburg, which will have a autonomous sensor and planning system (ASAP) as well as an automated diagnostic system (ADIA/ADIA++) [1]. It is planned to build a prototype in 2018 which has the possibility to be launched into orbit. The small Proba and Proba-2 satellites (Project for On-Board Autonomy) launched by ESA in 2001 and 2009 are already able to perform tasks like attitude control, target acquisition and image capture on their own [2]. There also have been several missions using full size satellites in orbit to test the feasibility of different aspects of autonomous rendezvous. In 1997 the Japanese national space agency NASDA launched the ETS-VII (Engineering Test Satellite VII) which consists of two main part: the chaser satellite and the target satellite. Using a robot arm attached to the chaser satellite, it was able to conduct the first successful unmanned autonomous rendezvous docking operation. The US Airforce launched the XSS 10 and XSS 11 (Experimental Satellite System) which were able to approach and rendezvous successfully with the second stage of their carrier rockets [3]. In 2005 NASA launched the DART (Demonstration for Autonomous Rendezvous Technology) vehicle which was developed to demonstrate an automated navigation and rendezvous operation. The mission failed prematurely due to a collision with the target spacecraft after completing about half of the planned objectives.

One of the limiting factors for generating a trajectory onboard a spacecraft is the processing power of the microprocessors used for spacecraft [4],[5]. Although the performance increases every generation just like their terrestrial counterparts, new generations take a long time before being thoroughly tested and certified and finally being used in new missions. In order to make an on-board trajectory optimization feasible for a wide range of spacecraft without relying on the improvement of new hardware in the future, processing demands of the computation has to be kept as low as possible. Classical methods like solving the equation of motion for every point on the trajectory deliver exact results but are only achievable with fast ground based computers. Therefor alternative, more efficiency orientated approaches with high scalability may be able to bridge the gap between accuracy and processing power.

Thus, this paper will investigate a method of calculating and optimizing a trajectory under boundary conditions for position and orientation using Bézier curves. The work of Sprunk, Lau, Pfaff and Burgard already showed that Bézier splines can used very effectively to plan 2D trajectories of omnidirectional robots [6]. This class of curves is being used since the 1960's for describing curves in the automotive industry (CAD) and quickly became the most used way of generating free form curves in computer graphics as a whole. Their success is mainly due to their simple mathematical structure. The idea is that by using Bézier curves only a few control points have to be optimized which describe the whole trajectory. The number of points in which the constraints must be satisfied (resolution) can be varied to accomplish high scalability which can be adjusted to the task at hand. The constraints are derived from the objections of a fly-by mission. The spacecraft has to stay above a minimum distant to any other object at any given time. Furthermore one axis, for example the camera, always has to stay in a mandatory zone pointing towards the target while avoiding a forbidden zone toward the sun in order to prevent damaging any optical instruments.

1.1 Simplifications and Assumptions

In order to develop initial algorithms a number of assumptions and simplifications have to be made beforehand. The algorithms only serves as a kinematic basis to see how Bézier curves perform in an optimization environment.

- No influence due to gravity or other forces. The model has no dynamic properties. Different spacecraft do not influence each other.
- Changes in the trajectory can be made at any point in time and are applied instantly. There is no limitation of orbit alteration due to thruster orientation, burning period etc.
- In order to evaluate the computation efficiency of the program, the assumption is made that the solution characteristics don't change across different micro processors or programming languages.

2 TRAJECTORY MODEL BASED ON BÉZIER CURVES

In order to build the Bézier curves for describing the position and the orientation in space, first the form of the polynomial must be defined and also the degree for the respective curve has to be chosen. For $n \in \mathbb{N}_0$ the real polynomials

$$\beta_{i,n} : \mathbb{R} \to \mathbb{R},\ t \to \binom{n}{i} t^i (1-t)^{n-i} \tag{1}$$

with $0 \leq i \leq n$ and the control variable $t \in [0,1]$ are called Bernstein basis polynomials of degree n. Where

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \tag{2}$$

is the binomial coefficient [7]. To model the position, a $n = 4$ quartic polynomial will be used. To describe the rotation a second, $n = 3$ cubic Bézier curves will be needed which also incorporates quaternions instead of Euler angles in order to avoid gimbal lock [8]. The degrees are chosen so that they are able to map a complex scenario without introducing unnecessary computing costs.

2.1 Position

For the description of the position the classical form of the Bézier curve is used. A Bézier curve of degree n with given $n+1$ control points $\boldsymbol{p_0}, \boldsymbol{p_1}, ..., \boldsymbol{p_n}$ which form the so called control polygon, for $t \in [0,1]$ is defined as:

$$\boldsymbol{c}(t) = \sum_{i=1}^{n} \beta_{i,n}(t)\boldsymbol{p}_i \tag{3}$$

in which the Bernstein basis polynomials $\beta_{i,n}(t)$ of degree n are forming the vector space basis of the polynomials [9]. The quartic Bézier curve can be described by its coefficient matrix and exponent vector [10] for ease of use. This yields is the form which is used for implementation in the program. FIGURE 1 shows an example of the curves generated by Eq. (4).

$$\boldsymbol{c}(t) = \begin{pmatrix} 1 \\ t^1 \\ t^2 \\ t^3 \\ t^4 \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 12 & -12 & 4 & 0 \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_0 \\ \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \\ \boldsymbol{p}_4 \end{pmatrix} \tag{4}$$

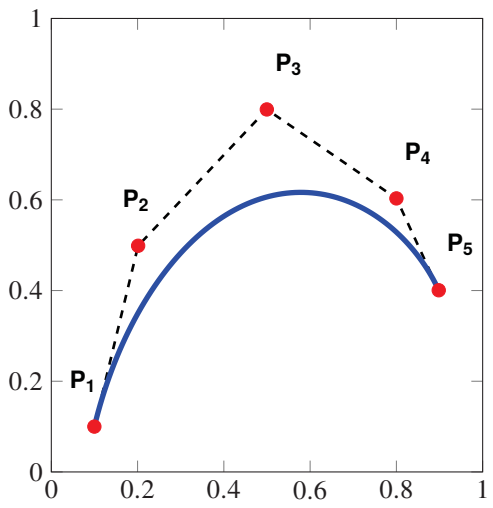

FIGURE 1. Example of a quartic Bézier curve

2.2 Orientation

For the description of the orientation a modified form of the Bézier curve is used which incorporated the use of quaternions. This is done to obtain a more stable form by eliminating the risk of losing a degree of freedom when two of the three axis are driven into a parallel configuration. The derivation of the Bézier quaternion curve follows [11]. In order to have any physical meaning the curves have to be continuous, meaning that the graph is a single unbroken curve with no "holes" or "jumps" ($\lim_{t \to a} p(t) = p(a)$). Given the points $\boldsymbol{p}_0, \boldsymbol{p}_1, .., \boldsymbol{p}_n \in \mathbb{R}^3$, the simplest C^0-continuous curve which interpolates every point $\boldsymbol{p}_i$ at $t = i$ is given by the linear interpolation:

$$\begin{aligned} \boldsymbol{p}(t) &= \boldsymbol{p}_0 + \alpha_1(t)\Delta\boldsymbol{p}_1 + \alpha_2(t)\Delta\boldsymbol{p}_2 + ... + \alpha_n(t)\Delta\boldsymbol{p}_n \\ &= \boldsymbol{p}_0 + \sum_{i=1}^{n} \alpha_i(t)\,(\boldsymbol{p}_i - \boldsymbol{p}_{i-1}) \end{aligned} \tag{5}$$

Given the unit quaternions $\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_n \in \mathbb{S}^3$ from the 3-sphere $\mathbb{S}^3 = \{\mathbf{q} \in \mathbb{H} : \|\mathbf{q}\| = 1\}$, it is possible to formulate an analogous C^0-continuous curve with the control points $\mathbf{q}_i$ and obtain the n-th order Bézier quaternion curve:

$$\mathbf{q}(t) = \mathbf{q}_0 \prod_{i=1}^{n} \exp(\boldsymbol{\Omega}_i \tilde{\beta}_{i,n}(t)) \,, \tag{6}$$

where

$$\boldsymbol{\Omega}_i = \ln(\mathbf{q}_{i-1}^{-1} \otimes \mathbf{q}_i) \quad \text{and} \quad \tilde{\beta}_{i,n}(t) = \sum_{i=0}^{n} \beta_{i,n}(t) \,. \tag{7}$$

The quaternion product, exponential and logarithm can be described in a compact form by partitioning the quaternions into a scalar part $s \in \mathbb{R}$ and a vector part $\boldsymbol{v} \in \mathbb{R}^3$:

$$\mathbf{q}_i = \begin{pmatrix} s_i & \boldsymbol{v}_i \end{pmatrix}^{\mathsf{T}} = \begin{pmatrix} q_{i,0} & q_{i,1} & q_{i,2} & q_{i,3} \end{pmatrix}^{\mathsf{T}} \tag{8}$$

$$\mathbf{q}_1 \otimes \mathbf{q}_2 = \begin{pmatrix} s_1 s_2 - \boldsymbol{v}_1 \cdot \boldsymbol{v}_2 \\ s_1 \boldsymbol{v}_2 + s_2 \boldsymbol{v}_1 + \boldsymbol{v}_1 \times \boldsymbol{v}_2 \end{pmatrix} \tag{9}$$

(10)

$$\exp(\mathbf{q}) = \exp(s) \begin{pmatrix} \cos(\|\boldsymbol{v}\|_2) \\ \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2} \sin(\|\boldsymbol{v}\|_2) \end{pmatrix}, \quad \ln(\mathbf{q}) = \begin{pmatrix} \ln(\|\mathbf{q}\|_2) \\ \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2} \arccos(\frac{s}{\|\mathbf{q}\|_2}) \end{pmatrix} .$$

The cubic Bézier-quaternion-curve yields the form which will be used for implementation:

(11)

$$\mathbf{q}(t) = \mathbf{q}_0 \otimes \exp(\boldsymbol{\Omega}_1 \tilde{\beta}_{1,3}) \otimes \exp(\boldsymbol{\Omega}_2 \tilde{\beta}_{2,3}) \otimes \exp(\boldsymbol{\Omega}_3 \tilde{\beta}_{3,3}) \,.$$

3 IMPLEMENTATION

3.1 Object function

The function which will be minimized by the solver is defined as the sum of the length [12] of the translation curve (Eq. 4) and the rotation curve (Eq. 9):

$$\min_{\boldsymbol{x}} L = \min_{\boldsymbol{x}} \{L(\boldsymbol{c}) + L(\mathbf{q})\} \,. \tag{12}$$

This means that the movement of the satellites is kept at a minimum. Due to the fact that there are no dynamic influences, in this case the shortest way is equivalent to the most efficient. The length of the curves can be approximated by using the Gaussian quadrature [13]. Applied to the quartic translation curve it yields:

(13)

$$L(\boldsymbol{c}) \approx C_1 \cdot \sqrt{\sum_{j=0}^{2}\left(\frac{dc_j(\tau_1)}{dt}\right)^2} + C_2 \cdot \sqrt{\sum_{j=0}^{2}\left(\frac{dc_j(\tau_2)}{dt}\right)^2} + C_3 \cdot \sqrt{\sum_{j=0}^{2}\left(\frac{dc_j(\tau_3)}{dt}\right)^2} + C_4 \cdot \sqrt{\sum_{j=0}^{2}\left(\frac{dc_j(\tau_4)}{dt}\right)^2} .$$

And in case of the cubic rotation curve it yields a form with only three terms but since the rotation makes use of quaternions it has four contributing elements per sum:

(14)

$$L(\mathbf{q}) \approx C_1 \cdot \sqrt{\sum_{j=0}^{3}\left(\frac{dq_j(\tau_1)}{dt}\right)^2} + C_2 \cdot \sqrt{\sum_{j=0}^{3}\left(\frac{dq_j(\tau_2)}{dt}\right)^2} + C_3 \cdot \sqrt{\sum_{j=0}^{3}\left(\frac{dq_j(\tau_3)}{dt}\right)^2} .$$

The necessary weights C_i and abscissae τ_i are tabulated [14]. If the model would be expanded by implementing the flight dynamics, the object function would be replaced by mission relevant values like the sum of forces and torques, fuel consumption or the travel time. These values can be combined and weighted to create functions which incorporate all desired characteristics of the trajectory.

3.2 Constraints

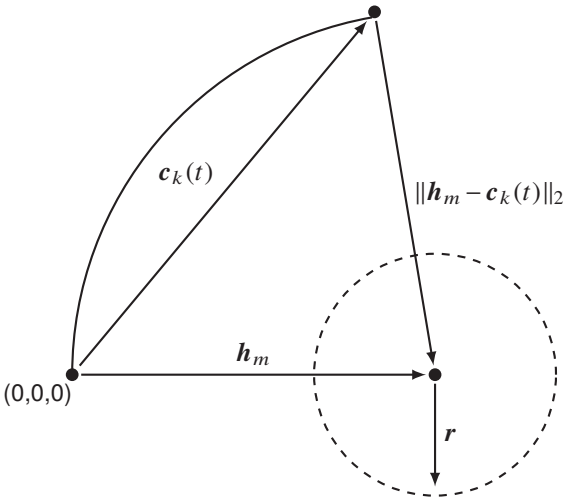

FIGURE 2. Boundary condition for collision avoidance.

The optimization is subject to complex constraints which are formulated as non linear boundary conditions and passed to the solver [15],[16]. The condition which is deployed on the translation curve has the task to avoid collision with all other spacecraft or objects. At any given time there has to be a minimum distance between all involved objects [17]. Mathematically the collision detection is formulated in such a way that the vector originating from the centre of the m-th obstacle pointing towards the k-th chaser has to be bigger than the defined minimum distance over the whole trajectory, therefore creating an forbidden sphere around the obstacle with radius r (FIGURE 2):

(15) $$\|\boldsymbol{h}_m - \boldsymbol{c}_k\|_2 \geq r \ .$$

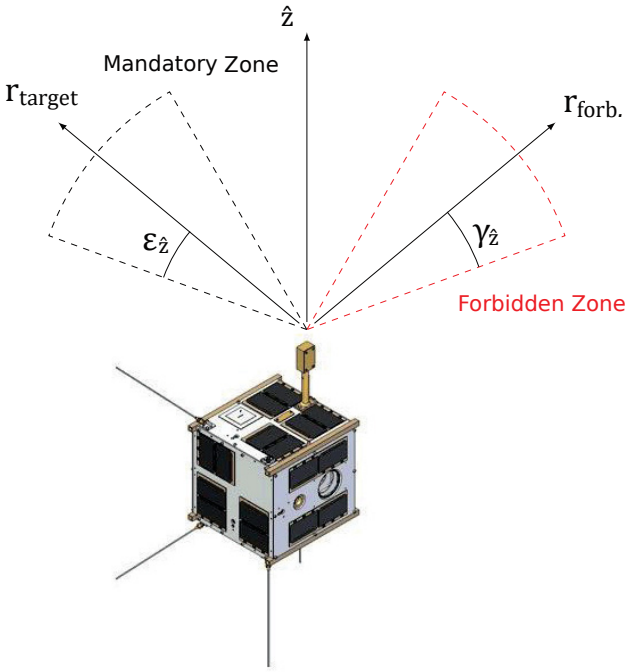

FIGURE 3. Forbidden and mandatory zones [18].

The boundary condition associated with the rotation curve is the avoidance and alignment of predefined angle zones, as shown in FIGURE 3. For example a scenario could be to align the on-board camera towards a target during fly-by while also avoiding the sun during rotation, in order to lower the risk of damaging the instrument [19],[20]. The conditions can be formulated for each of the spacecraft axis independently (camera towards target, antenna towards earth etc.). The forbidden zones used in this simulation are representing the avoidance of a light source (sun). This can be used to protect optical sensitive equipment like telescopes and cameras.

(16) $$\hat{\boldsymbol{z}} \cdot \boldsymbol{r}_{forb.} - \cos(\gamma_{\hat{z}}) \leq 0$$

The mandatory zones used in this simulation are representing the alignment towards a target object. For example this can be used to inspect a target regarding exterior damage before a docking manoeuvre is initiated.

(17) $$\cos(\epsilon_{\hat{z}}) - \hat{\boldsymbol{z}} \cdot \boldsymbol{r}_{tar.} \leq 0$$

3.3 Initial guess values

Initial guess values for all control points are needed to start the optimization. The smaller the difference between initial guess and actual solution, the shorter the time the computation needs to finish. This can lead to a high efficiency gain when using very sophisticated guesses which are specially adjusted for specific types of problem configurations. These kind of guesses are beyond the scope of this paper. Although a rather primitive form will be used, the results are good enough for the simple configurations that will be evaluated later on.

The initial guesses for the control points are derived from the equilateral triangle formed by the start and end point of the translation curve. This method is based on the experience that most trajectories in which the obstacle is positioned in the same plane as the chaser satellite exhibit a equidistant distribution of control points. The selection of the right initial guess values is very important. If the values are too far off they can cause the solution to diverge.

FIGURE 4 shows how the equilateral triangle is constructed by using the distance between start and end point. Subsequently the initial control points are placed equidistantly on half the hight of the triangle parallel to the base in such a way that the points form a trapezoid. The initial coordinates obtain that way are given by Eq. (16):

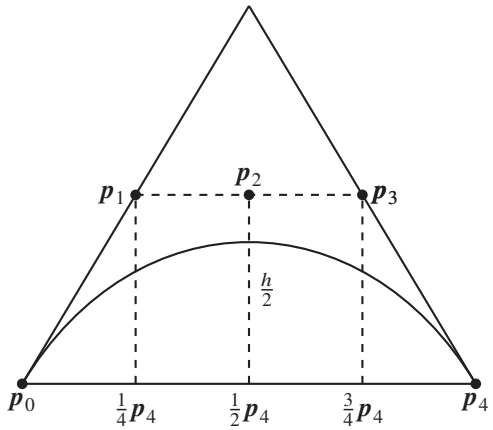

FIGURE 4. Initial guess for translation.

$$
\begin{aligned}
\boldsymbol{p}_{1,init.} &= \left(\tfrac{1}{4}\|\boldsymbol{p}_4\|_2 \quad \tfrac{\sqrt{3}}{4}\|\boldsymbol{p}_4\|_2 \quad 0\right) \\
\boldsymbol{p}_{2,init.} &= \left(\tfrac{1}{2}\|\boldsymbol{p}_4\|_2 \quad \tfrac{\sqrt{3}}{4}\|\boldsymbol{p}_4\|_2 \quad 0\right) \\
\boldsymbol{p}_{3,init.} &= \left(\tfrac{3}{4}\|\boldsymbol{p}_4\|_2 \quad \tfrac{\sqrt{3}}{4}\|\boldsymbol{p}_4\|_2 \quad 0\right) .
\end{aligned} \tag{18}
$$

In order to obtain initial values for the rotation curve the quaternion at the control points $\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_3, \boldsymbol{p}_4$ are calculated which are necessary to align an arbitrary axis of the satellite towards the target FIGURE 5. For all four control points the normal in target direction is plotted as well as the original position of the preceding point. Those vector pairs yield the corresponding quaternion which describes the rotation between two adjacent points. The initial quaternions obtain that way are given by Eq. (17):

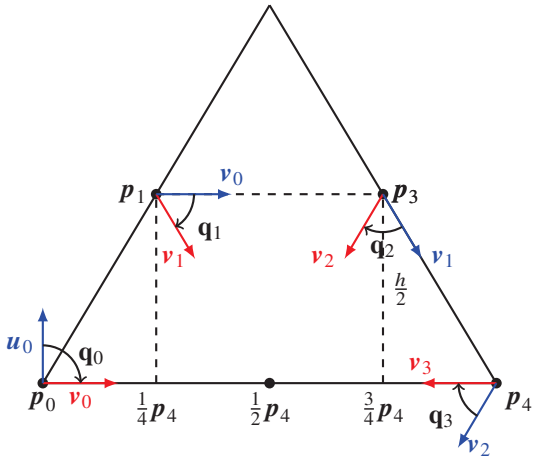

FIGURE 5. Initial guess for rotation.

$$
\begin{aligned}
\mathbf{q}_{0,init.} &= \left((\boldsymbol{u}_0 \cdot \boldsymbol{v}_0) + \sqrt{\|\boldsymbol{u}_0\|_2 + \|\boldsymbol{v}_0\|_2} \quad \boldsymbol{u}_0 \times \boldsymbol{v}_0\right)^{\mathsf{T}} \\
\mathbf{q}_{1,init.} &= \left((\boldsymbol{v}_0 \cdot \boldsymbol{v}_1) + \sqrt{\|\boldsymbol{v}_0\|_2 + \|\boldsymbol{v}_1\|_2} \quad \boldsymbol{v}_0 \times \boldsymbol{v}_1\right)^{\mathsf{T}} \\
\mathbf{q}_{2,init.} &= \left((\boldsymbol{v}_1 \cdot \boldsymbol{v}_2) + \sqrt{\|\boldsymbol{v}_1\|_2 + \|\boldsymbol{v}_2\|_2} \quad \boldsymbol{v}_1 \times \boldsymbol{v}_2\right)^{\mathsf{T}} \\
\mathbf{q}_{3,init.} &= \left((\boldsymbol{v}_2 \cdot \boldsymbol{v}_3) + \sqrt{\|\boldsymbol{v}_2\|_2 + \|\boldsymbol{v}_3\|_2} \quad \boldsymbol{v}_2 \times \boldsymbol{v}_3\right)^{\mathsf{T}}
\end{aligned} \tag{19}
$$

3.4 Optimization methods

The simulation was run with three different optimization methods in order to compare their accuracy, computational demands and overall scalability for the given model and scenarios. Since the program is written in MATLAB the choice of algorithms is rather limited without buying third party software which is beyond the scope of this paper. Therefor the three main algorithms that MATLAB provides (INT,SQP,GA) are being tested which in most cases are the base platforms on which more sophisticated algorithms are build upon [21].

The main field of application of the interior-point-method (INT) are linear and quadratic problems, although they can also be used to solve some non-linear problems. In comparison to the more common simplex-methods, the interior-point-method features fast convergent solutions for sparsely populated problems. On the other hand, these methods are not suitable for solving a great number optimization problems in series, which is an important requirement for many whole-number optimizations [22].

The sequential quadratic programming method (SQP) is very effective for solving bounded non-linear optimization problems. In every iteration the algorithm approximates the non-linear problem with a quadratic one which is easy to

solve. The constraints however are approximated linearly [23].

Evolutionary algorithms (GA) are a class of stochastic, metaheuristic optimization methods inspired by nature. Similar how in nature a problem is solved by adapting to the environment through random mutations, the algorithm creates a generation of solutions in every iteration in which only the best ones will be chosen to create the next generation until the best possible solution is found [24]. This behaviour gives the algorithm a versatile area of applications with the drawback of lower accuracy and higher computation power requirement [25].

MATLAB provides several abortion criteria in order to vary the accuracy of the solution and therefor the needed computational power [26].

$TolX$ is defined as the smallest allowed step size of a variable. If the algorithm tries to make a step smaller than $\|\boldsymbol{x}_i - \boldsymbol{x}_{i+1}\|_2$ the process will be aborted (FIGURE 6). The smaller the step size the more accurate the solution will be.

$TolFun$ is defined as the smallest allowed difference in the object funtion (the function which is subject to the minimization) at any given iteration step. If $|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_{i+1})| \leq TolFun$ the process will be aborted (FIGURE 6).

$TolCon$ is defined as the biggest allowed exceedance of the boundary conditions. If the algorithm detects a violation of the constraint $c(\boldsymbol{x}) > TolCon$ at point $\boldsymbol{x}$ MATLAB will prompt the user about the current status of the optimization but the calculation will not be stopped. Although it is not an abortion criterion in the literal sense it is very useful for evaluating the scalability of the program.

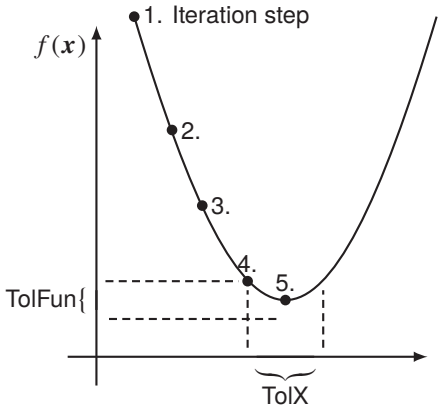

FIGURE 6. Abortion criteria TolX and TolFun.

4 SIMULATION RESULTS

The simulation was carried out with three different scenarios: a fly-by with one target, a fly-by with a target and an observer [27] and an assembly mission consisting of a resource storage and the assembly-site (FIGURE 7). Every scenario was run with all three optimization methods under different abortion criteria $TolX = [1e-2, 1e-4, ..., 1e-10]$ and resolutions of the Bézier curves, which means the number of evaluations in the region $t \in [0,1]$ with $\Delta t = \frac{1}{N}$ is ranging from $N = [20, 40, ..., 100]$. These parameters result in 225 reading points. In this paper only scenario 1 will be shown in detail but for the scalability analysis all of the points will be considered.

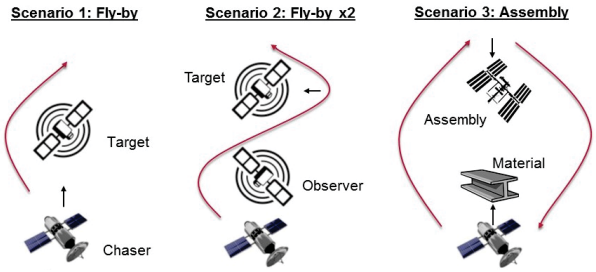

FIGURE 7. Applied scenarios.

The configuration of the fly-by scenario consists of the chasing satellite, the target and one light source (sun) TABLE 1. The chaser starts with distance to the target and proceeds to fly around it without falling below the minimum distance given by the constraints. The rotation along the fly-by is done by keeping the yaw axis of the chaser in a cone of $\pm 15°$ around the normal vector towards the target. Furthermore the yaw and pitch axis orientation are not allowed to enter a $\pm 22.5°$ cone around the normal vector towards the light source at any point in trajectory. Only the roll axis is free from any constraints. Start and end point of the trajectory are predefined, all other points along the way are being calculated by the optimization process. The object function, which is subject to the minimization, is set to the sum of the length of the translation and rotation curve. Altogether the scenario has one translational and three rotational boundary conditions. The results are shown in FIGURE 8-12 and TABLE 2. FIGURE 8 displays the trajectory in 3D space. It shows that the distance between start and end point has been minimized while also respecting all three constraints. It can also be observed that the final control points lie in the same plane as the initial guess values, and by comparing the numerical values we see that the guess was fairly good. FIGURE 9 shows a more detailed picture of the rotational constraints. The 2D plot in geographic coordinates as well as the 3D projection unto the unit sphere show that the forbidden zone is being avoided by the affected axis while keeping the camera axis in the mandatory zone.

TABLE 1. Scenario 1 initial parameter

Solver	SQP	TolX=1e-6	TolCon=1e-6	$\Delta t = 0.02$
Start	[0 0 0]			
End	[10 0 0]			
Light	[1 5 2.5]	R=0.5	$\gamma_y = 45°$	Forbidden y
Light	[1 5 2.5]	R=0.5	$\gamma_z = 45°$	Forbidden z
Target	[6 0 0]	R=3	$\epsilon_z = 30°$	Mandatory z

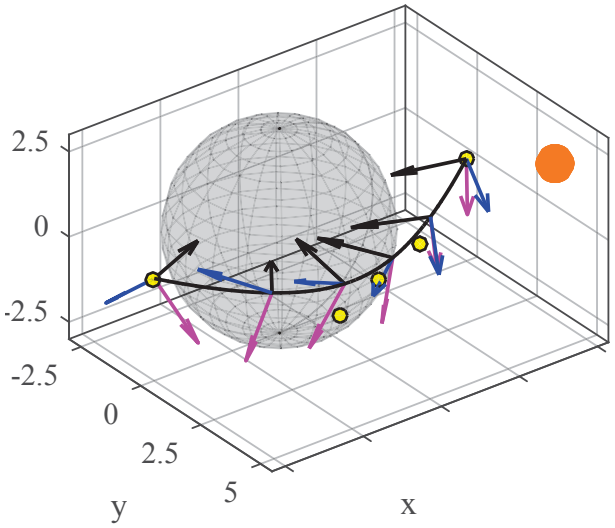

FIGURE 8. 3D plot of the trajectory.
Magenta line: x-axis of chaser satellite.
Blue line: y-axis of chaser satellite.
Black line: z-axis of chaser satellite.
Black sphere: Avoidance radius of target.
Orange sphere: Light source (sun).
Yellow point: Control points of bézier curve.

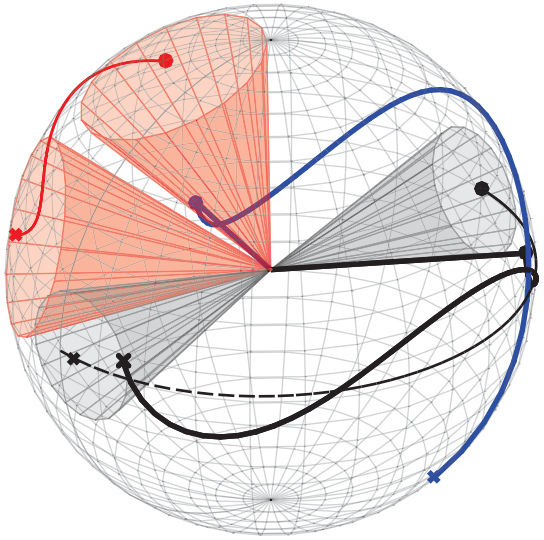

FIGURE 10. Rotation projected unto the unit sphere.
Blue line: y-axis of chaser satellite.
Black line: z-axis of chaser satellite.
Black dashed line: Normal vector chaser to target.
Black cone: Mandatory zone.
Red line: Normal vector from chaser to sun.
Red cone: Avoidance zone.

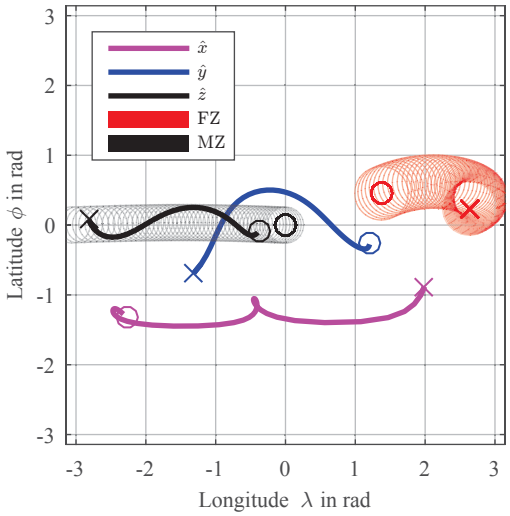

FIGURE 9. Orientation displayed in geographic coordinates.

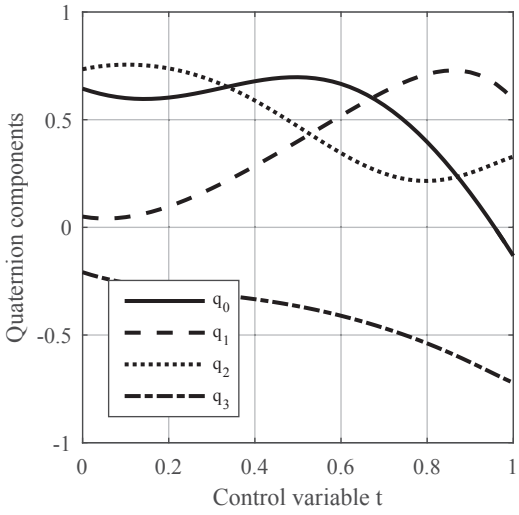

FIGURE 11. Development of the quaternion values.

For comparing the processing demands to other systems the calculation time has to be converted to Floating Point Operations (FLOPs). Since MATLAB doesn't support a FLOP count any more the performance of the test computer was measured with Intel Linpack 10.3.4. and clocked in at 76.91 GFLOP/s. Every simulation was run in parallel mode to ensure that all physical cores are being used.

FIGURE 12 shows how the solution develops as the iteration steps increase. In this case the function value decreases almost quadratically before approaching the solution in a asymptotic manner. The constraint violation has a much steeper decline and already reaches its minimum at the 10th iteration.

TABLE 2. Scenario 1 initial guesses and output

Translation	Initial guess	End value
p_1	[2.5, 4.33, 0]	[3.45, 2.60, -0.10]
p_2	[5, 4.33, 0]	[5.42, 3.38, -0.13]
p_3	[7.5, 4.33, 0]	[7.25, 4.17, -0.13]
Rotation		
$\mathbf{q}_0$	[0.71, 0, 0.71, 0]	[0.64, 0.05, 0.73, -0.20]
$\mathbf{q}_1$	[0.75, 0, 0.46, -0.46]	[0.38, -0.07, 0.82, -0.41]
$\mathbf{q}_2$	[0.98, 0, 0.12, -0.12]	[0.73, 0.67, -0.09, 0.03]
$\mathbf{q}_3$	[0.99, 0, 0.02, -0.02]	[-0.13, 0.59, 0.33, -0.72]
Computation		
Arch length		12.10255
Iterations		20
Duration		19.917770 s

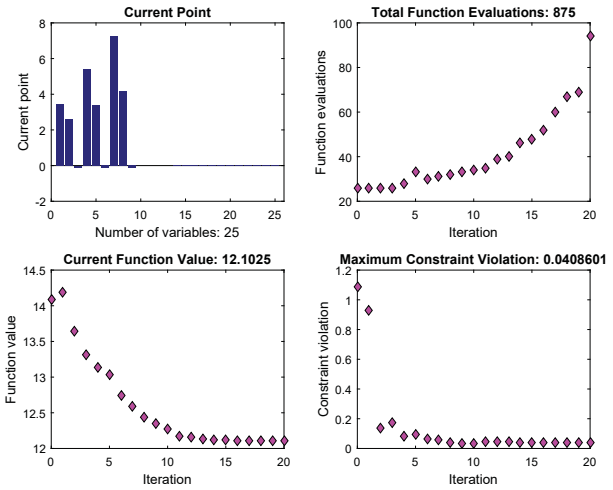

FIGURE 12. Development of the solution along the iteration steps .

4.1 Scalability

The obtained data points for all optimization methods per scenario are shown in FIGURE 13-15. For each scenario the object function values and the constraints error are plotted over the computing effort. Considering all the results with respect to how small the object function was able to get and the corresponding error margin in the constraints, different characteristic tendencies of the individual algorithm can be observed. The solution behaviour is greatly influenced by the number and configuration of the boundary conditions.

For the fly-by scenario the SQP has good scalability in the constraint margin but the object function stays almost constant. In the double fly-by, both values are scalable and for the assembly scenario both stay constant. The INT algorithm scales both parameters for all scenarios but also has the highest amount of computing demands. The genetic algorithm shows consistency across the tests because the error in the boundary condition is always zero.

Globally speaking the SQP features fast calculation with scalable constraint deviations. The GA offers precise solutions with greater computing demands and scalable object function values. The INT algorithm has the broadest and most flexible scaling range for all considered parameters which also results in the lowered efficiency in terms of computation time. A rough estimation of the scalability per invested TFLOPS is given by $\Delta L/TFLOPS \approx -3.8\%$ for the minimization of the object function and $\Delta Tolcon/TFLOPS \approx -0.57°$ for the adherence of the boundary conditions.

5 CONCLUSION AND OUTLOOK

The examined optimization methods were able to produce sufficiently precise and scalable solutions for the trajectory based on Bézier curves. The advantages of the Bézier curves are reflected in the compact and simple structure of the program thus forming a good base for further implementation of functionality with relatively low effort. Due to the fact that translation and rotation depend on the same control variable, a logical consistency is introduced besides the numerical advantage which creates a clearly arranged and easy to handle environment for the usage of the program.

A processing speed in the range of the sampling rate of spacecraft sensors in order to permit the application on autonomous satellites could not be achieved with the solver methods provided by MATLAB. The results show that despite the low processing cost of Bézier curves in comparison to traditional trajectory models, the choice of a appropriate optimization algorithm has a great impact on quality, efficiency and scalability of the solution. It has to be mentioned that the efficiency of the calculation has a great potential of improvement. Because MATLAB is an interpreter language, the first and most important step in order to enhance the processing speed is the change to a compiler language like C++ or FORTRAN, in which the source code is translated into machine code. Depending on the code a gain of processing speed by a factor of 10x-100x is possible.

The results furthermore show that out of the examined algorithms, the sequential quadratic programming algorithm has the highest potential for the usage of on board applications. It has to be assumed that algorithms which are specially adjusted to the problem could utilise the potential of the Bézier trajectory considerably more.

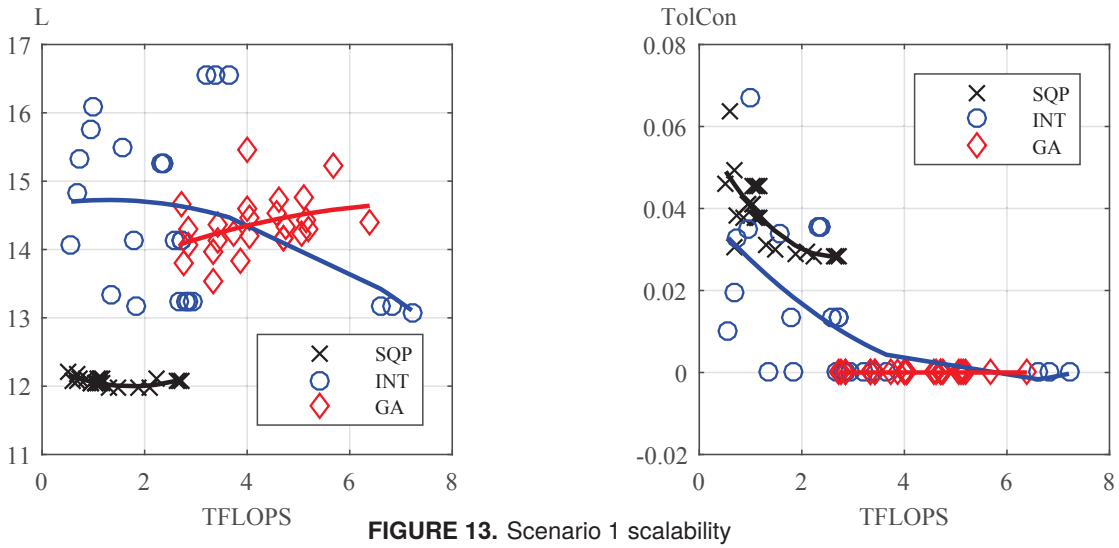

FIGURE 13. Scenario 1 scalability

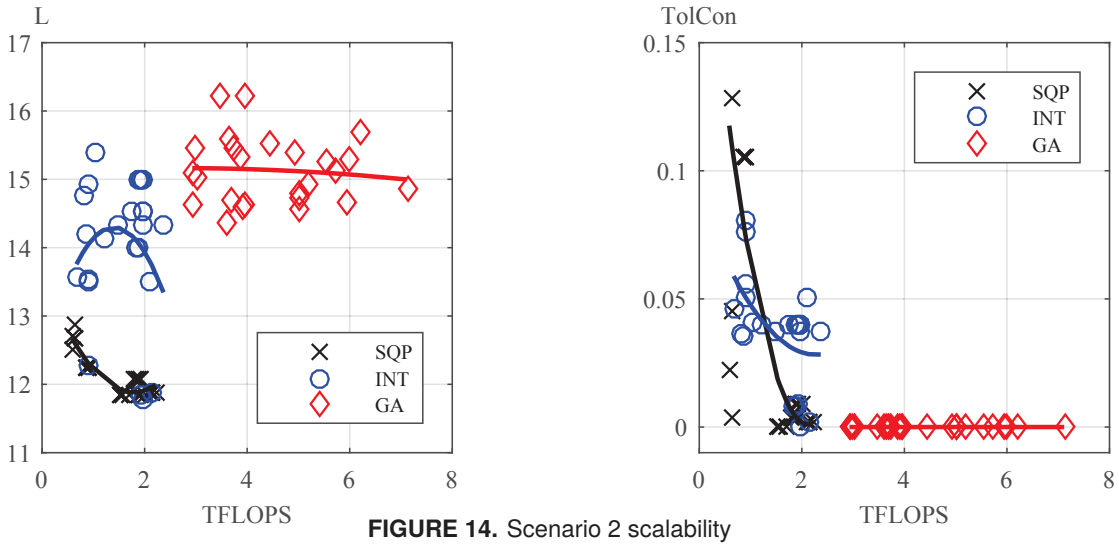

FIGURE 14. Scenario 2 scalability

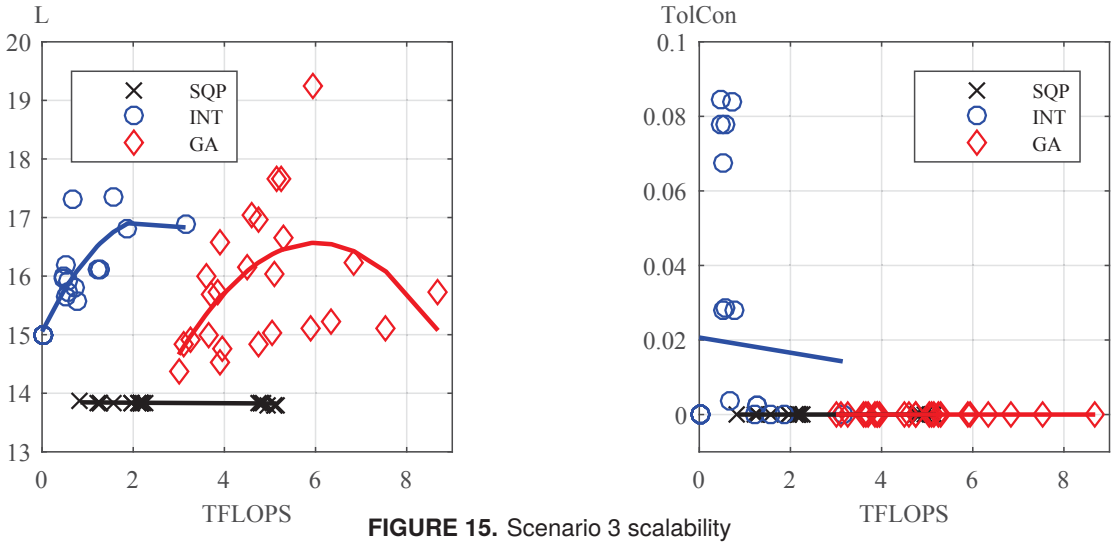

FIGURE 15. Scenario 3 scalability

References

[1] Prof. Dr. Hakan Kayal. *Autonome Nanosatelliten in Arbeit*. https://www.uni-wuerzburg.de/sonstiges/meldungen/single/artikel/autonome-nanosatelliten-in-arbeit/, 2016. Accessed: 2016-08-08.

[2] ESA. *Proba: Facts and Figures*. http://m.esa.int/Our_Activities/Observing_the_Earth/Proba-1/Facts_and_figures, 2016. Accessed: 2016-08-08.

[3] NASA. *On-Orbit Satellite Servicing Study - Project Report*. http://ssco.gsfc.nasa.gov/images/NASA_Satellite%20Servicing_Project_Report_0511.pdf, 2010. Accessed: 2016-01-17.

[4] Simone D'Amico Andreas Grillenberger Sergio De Florio, Eberhard Gill. *Performance comparison of microprocessors for space-based navigation applications*. DLR Space Flight Technology Department, 2009.

[5] Ran Ginosar. *Survey of processors for space*. DASIA, 2012.

[6] Patrick Pfaff Wolfram Burgard Christoph Sprunk, Boris Lau. *Online Generation of Kinodynamic Traajectories for Non-Circular Omnidirectional Robots*. IEEE International Conference on Robotics and Automation, 2011.

[7] P.P. Korovkin. *Encyclopedia of Mathematics*. Springer, 2003.

[8] Simon Särkka. *Notes on Quaternions*. Simon Särkka, 2007.

[9] Mike "Pomax" Kamermans. *A Primer on Bézier Curves*. http://pomax.github.io/bezierinfo/, 2011. Accessed: 2015-12-11.

[10] L. Maisonobe. *Drawing an elliptical arc using polylines, quadratic or cubic Bezier curves*. L. Maisonobe, 2003.

[11] Sung Yong Shin Myoung-Jun Kim, Myung-Soo Kim. *A General Construction Scheme for Unit Quaternion Curves with Simple High Order Derivatives*. Korea Advanced Institute of Science and Technology KAIST, 2001.

[12] Harro Heuser. *Lehrbuch der Analysis*. Teubner Verlag, 1990.

[13] Mike "Pomax" Kamermans. *Arc length*. http://pomax.github.io/bezierinfo/#arclength, 2011. Accessed: 2015-12-11.

[14] Mike "Pomax" Kamermans. *Gaussian Quadrature Weights and Abscissae*. http://pomax.github.io/bezierinfo/legendre-gauss.html, 2011. Accessed: 2015-12-11.

[15] Matthias Gerdts. *Einführung in die linearen und nichtlinearen Optimierung*. Matthias Gerdts, 2011.

[16] F. Javier Heredia. *Tutorial Nonlinear Optimization*. Dept. of Statistics and Operational Research, 2011.

[17] Jonathan P. How Eric Feron Arthur Richards, Tom Schouwenaars. *Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming*, volume 25. Journal of Guidance,Control,Dynamics, 2002.

[18] Sophie Jusko (Illustration). *Animiertes Gif von künstlichen Satelliten*. http://www.gifmania.com.de/Animierte-Gifs-Weltraum/Kunstliche-Satelliten-88599.gif, 2016. Accessed: 2016-01-05.

[19] Ran Dai Chuanhchuang Sun. *Spacecraft Attitude Control under Constrained Zones via Quadratically Constrained Quadratic Programming*. Iowa State University, 2011.

[20] Mehran Mesbahi Unsik Lee. *Quaternion based optimal spacecraft reorientation under complex attitude constrained zones*. University of Washington, Seattle, 2001.

[21] *MathWorks ®Documentation: Find minimum of constrained nonlinear multivariable function*. http://de.mathworks.com/help/optim/ug/fmincon.html, 2015. Accessed: 2016-01-11.

[22] Yinyu Ye. *Interior-Point Algorithms*. Wiley, 1997.

[23] Stephen J. Wright Jorge Nocedal. *Numerical Optimization*. Springer, 2006.

[24] *MathWorks ®Documentation: Find minimum of function using genetic algorithm*. http://de.mathworks.com/help/gads/ga.html, 2015. Accessed: 2016-01-05.

[25] D. Ashlock. *Evolutionary Computation for Modeling and Optimization*. Springer, 2006.

[26] *MathWorks ®Documentation: Tolerances and Stopping Criteria*. http://de.mathworks.com/help/optim/ug/tolerances-and-stopping-criteria.html, 2015. Accessed: 2016-01-17.

[27] Fumio Miyazaki Yasuhiro Masutani, Motoshi Matsushita. *Flyaround Maneuvers on a Satellite Orbit by Impulsive Thrust Control*. Graduate School of Engineering Science Osaka, 2001.