Optimization of Spacecraft Rendezvous and Docking using Interval Analysis

E. van Kampen, Q.P. Chu and J.A. Mulder

Delft University of Technology, Delft, The Netherlands

This paper applies interval optimization to the fixed-time multiple impulse rendezvous and docking problem. Current methods for solving this type of optimization problem include for example genetic algorithms and gradient based optimization. Unlike these methods, interval methods can guarantee that the globally best solution is found for a given parameterization of the input. The state transition matrix approach for the linearized CW-equations is used to avoid interval integration. Thruster pulse amplitudes are optimized by an interval branch and bound algorithm, which systematically eliminates parts of the control input space that do not satisfy the final time state constraints. Interval analysis is shown to be a useful tool in both sensitivity analysis and nonlinear optimization of the rendezvous and docking problem.

I. Introduction

In this paper Interval Analysis, or more specifically interval optimization, is applied to the rendezvous and docking problem. Interval analysis is the analysis of interval numbers, which form an extension of the regular numbers. As will be shown in this paper, interval numbers are especially well suited for sensitivity analysis and optimization. Other examples of the application of interval analysis to aerospace related problems are for example: finding trim-points for nonlinear aircraft models, pilot model identification, integer ambiguity resolution for aircraft attitude determination, spacecraft re-entry optimization and fuel optimization for constrained spacecraft formation rotations. Although interval analysis has been increasingly applied to nonlinear optimization problems over the last decades, it is still relatively unknown, especially in the field of spacecraft rendezvous and docking. Therefore the goal of this paper is to introduce interval optimization as a viable alternative to the current methods.

There are many ways to classify rendezvous and docking optimization methods. Carter gives a clear overview of classification options for linearized rendezvous, with three main classification groups:

1. Type of coordinate frame.
2. Type of linearization.
3. Type of nominal orbit.

Each of these groups is further subdivided into subgroups and sometimes sub-subgroups, indicating the abundance of available methods. Before looking at some of the linearized methods, it is also worth mentioning the methods that skip the linearization step, but instead look at the fully nonlinear equations of motion. An example of such a nonlinear method is the adaptive output feedback control by Singla, Subbarao, and Junkins, where asymptotic stability can be guaranteed, even under measurement uncertainties, although the transient performance can be sub-optimal.

---

*Ph.D. student, Aerospace Software and Technologies Institute (ASTI), P.O. Box 5058, 2600GB Delft, The Netherlands; e.vankampen@tudelft.nl. Student member AIAA.

†Associate Professor, Control and Simulation Division, Faculty of Aerospace Engineering, P.O. Box 5058, 2600GB Delft, The Netherlands; q.p.chu@tudelft.nl. Member AIAA.

‡Professor, Control and Simulation Division, Faculty of Aerospace Engineering, P.O. Box 5058, 2600GB Delft, The Netherlands; j.a.mulder@tudelft.nl. Member AIAA.
Most rendezvous and docking optimization methods however start from the linearized equations of motion. Once the equations are linearized, state transition matrices can be derived to eliminate direct numerical integration of the equations of motion.

Since rendezvous and docking optimization falls under the much broader category of general trajectory optimization, most trajectory optimization algorithms are suitable to solve the rendezvous and docking problem. Betts provides an extensive survey of numerical methods for trajectory optimization, including: nonlinear programming, optimal control, numerical analysis, (in)direct shooting methods and also, briefly, dynamic programming and genetic algorithms. From these methods, optimal control theory receives the most attention, for example for the rendezvous optimization for bounded thrust levels and for hybrid continuous and impulsive thrust.

Luo, Tang and Lei apply a multi-objective genetic rendezvous optimization using the Clohessy Wiltshire (CW) equations and a state transition matrix. Their multi-objective genetic algorithm, NSGA, optimizes two-, three-, and four-impulse -V-bar and +V-bar homing rendezvous trajectories, resulting in Pareto-optimal solution fronts.

In this paper analytical solutions to the linearized CW-equation are used to compute the spacecraft trajectories for a number of thruster pulses. Different combinations of thruster pulses lead to different trajectories. The optimization problems concerns finding the amplitudes of the control input such that the final time constraints, i.e. relative position and velocities, are satisfied.

In section II the relevant dynamics for the rendezvous and docking problem are presented, together with their analytical solutions. Before explaining the optimization procedure, it is necessary to introduce in some more detail, the theory of Interval Analysis, which is done in section III. Next, the interval optimization method for rendezvous and docking is presented in section IV, together with the simulation results of several types of optimizations relating to rendezvous and docking. A comparison between gradient based optimization and interval optimization for the spacecraft rendezvous and docking problem is presented in section V. Finally a summary of the conclusions is given in section VI.

**II. Relative motion dynamics**

There are a number of alternative representations of the CW-equations in use today, which differ in the definition of the coordinate frame.

Fehse uses the following axes definitions (origin located in target spacecraft, see also figure 1):

- **x**: direction of orbital velocity.
- **y**: opposite direction of angular momentum vector.
- **z**: radial from center of spacecraft to the Earth.

![Coordinate frame definition for CW relative motion equations: x in direction of orbital velocity, y in opposite direction of angular momentum vector, and z along the radial from center of spacecraft to the Earth.](image)

The CW-equations for this frame definition are:

\[
\begin{align*}
\ddot{x} - 2\omega \dot{z} &= 0 \\
\dot{y} + \omega^2 y &= 0 \\
\ddot{z} + 2\omega \dot{x} - 3\omega^2 z &= 0
\end{align*}
\]
These equations are derived under the following assumptions:

1. Spacecraft are in undisturbed Kepler orbits.
2. The distance from the target to the chaser is much smaller than the distance from the target to the center of the orbited mass.
3. The target spacecraft is in a circular orbit.

The CW equations can, under some conditions, be solved analytically. The first part of the solution is the homogeneous part that describes the relative position as a function of the initial relative position and initial velocity only. A particular solution that describes the effect of external forces forms the second part of the solution.

II.A. Homogeneous solution

The homogeneous solution for the CW equation 1 is given by Fehse as:

\[
\begin{align*}
  x(t) &= \left( \frac{4\pi}{\omega} - 6z_0 \right) \sin(\omega t) - \frac{2\pi}{\omega} \cos(\omega t) + (6\omega z_0 - 3\dot{x}_0) t + (x_0 + \frac{2\pi}{\omega}) \\
  y(t) &= y_0 \cos(\omega t) + \frac{2\pi}{\omega} \sin(\omega t) \\
  z(t) &= \left( \frac{2\pi}{\omega} - 3z_0 \right) \cos(\omega t) + \frac{2\pi}{\omega} \sin(\omega t) + (4z_0 - \frac{2\pi}{\omega})
\end{align*}
\]

(2)

Where \( x_0, y_0, z_0, \dot{x}_0, \dot{y}_0, \dot{z}_0 \) are the initial relative positions and velocities and \( \omega \) is the orbital rate of the target. From this solution it can be seen that \( y(t) \) and \( z(t) \) oscillate, while \( x(t) \) contains an additional linear growth over time.

II.B. Particular solutions

The particular solution describes the effect of control inputs, in this case thruster burns, on the relative motion of the spacecraft. For some types of control inputs, the particular solution can be obtained analytically. For a single thruster pulse from \( t_1 \) to \( t_2 \), the particular solution becomes:

\[
x_p = \frac{1}{m_c} Hu
\]

(3)

where the state \( x \) consists of the relative positions \( x, y, z \) appended with the relative velocities \( \dot{x}, \dot{y}, \dot{z} \), while the vector \( u \) contains the amplitudes of the thruster pulses. The matrix \( H \) is divided into an out-of-plane part \( h_2 \) and an in-plane part \( h_1, h_3 \):

\[
H = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix}
\]

(4)

For the in-plane dynamics, the particular solution is formed by:\n
\[
h_1 = \begin{bmatrix}
\frac{1}{\omega} [\cos(\omega(t - t_2)) - \cos(\omega(t - t_1))] + \frac{2}{\omega} \left( (t - t_2)^2 - (t - t_1)^2 \right) \\
\frac{2}{\omega} [\sin(\omega(t - t_2)) - \sin(\omega(t - t_1)) + \omega(t_1 - t_2)] \\
\frac{1}{\omega} [\sin(\omega(t - t_2)) - \sin(\omega(t - t_1))] + 3(t_1 - t_2) \\
\frac{2}{\omega} [\cos(\omega(t - t_1)) - \cos(\omega(t - t_2))]
\end{bmatrix}
\]

(5)

\[
h_3 = \begin{bmatrix}
\frac{2}{\omega} [\sin(\omega(t - t_2)) - \sin(\omega(t - t_1)) + \omega(t_2 - t_1)] \\
\frac{1}{\omega} [\cos(\omega(t - t_2)) - \cos(\omega(t - t_1))] \\
\frac{2}{\omega} [\cos(\omega(t - t_2)) - \cos(\omega(t - t_1))] \\
\frac{1}{\omega} [\sin(\omega(t - t_1)) - \sin(\omega(t - t_2))]
\end{bmatrix}
\]

(6)

The particular solution for the out-of-plane dynamics is constructed with:\n
\[
h_2 = \begin{bmatrix}
\frac{1}{\omega} [\cos(\omega(t - t_2)) - \cos(\omega(t - t_1))] \\
\frac{1}{\omega} [\sin(\omega(t - t_1)) - \sin(\omega(t - t_2))]
\end{bmatrix}
\]

(7)
Particular solutions are linear solutions and can therefore be added to the homogeneous solution or to other particular solutions. In this way a more complex control input structure can be approximated by a summation of single thruster pulses, each with their own particular solution.

Before discussing how interval optimization can be applied to optimize the rendezvous and docking problem using the analytical solutions of the CW-equations, a short introduction to interval arithmetic is presented in the next section.

III. Interval Analysis

Interval analysis is the theory dealing with interval numbers and the arithmetic operations on them. The collection of all arithmetic operations on interval numbers is called interval arithmetic. An interval number is defined as an ordered pair of real numbers \([a, b]\) with \(a \leq b\). An interval parameter is written with brackets within which both the infimum and supremum are given: \([a, b]\), or a single variable: \([x] = [a, b]\).

Interval arithmetic contains the same operators as ordinary arithmetic, for example the basic computational operations of addition, subtraction, multiplication and division:

\[
\begin{align*}
[a, b] + [c, d] &= [a + c, b + d] \\
[a, b] - [c, d] &= [a - d, b - c] \\
[a, b] \cdot [c, d] &= \left[ \min (ac, ad, bc, bd) , \max (ac, ad, bc, bd) \right] \\
\frac{[a, b]}{[c, d]} &= [a, b] \cdot \left[ \frac{1}{d}, \frac{1}{c} \right] \quad \text{if } 0 \notin [c, d]
\end{align*}
\]

The core of interval analysis is to use interval arithmetic to form an inclusion function \([f]([x])\) of any function \(f(x)\). This property of interval arithmetic follows from the inclusion function theorem given by R.E. Moore:

**Theorem III.1.**

If \([f]([x_1], [x_2], ..., [x_n])\) is a rational expression in the interval variables \([x_1], [x_2], ..., [x_n]\), i.e. a finite combination of \([x_1], [x_2], ..., [x_n]\) and a finite set of constant intervals with interval arithmetic operations, then

\[
[x_1]’ \subset [x_1], [x_2]’ \subset [x_2], ..., [x_n]’ \subset [x_n]
\]

implies

\[
[f]([x_1]’, [x_2]’, ..., [x_n]’) \subset [f]([x_1], [x_2], ..., [x_n])
\]

for every set of interval numbers \([x_1], [x_2], ..., [x_n]\) for which the interval arithmetic operations in \([f]\) are defined.

**Proof.** For the proof of this theorem the reader is directed to.

If we take \([x_1], [x_2], ..., [x_n]\) to be the crisp numbers \(x_1, x_2, ..., x_n\) and apply the theorem, then we obtain:

\[
f(x_1, x_2, ..., x_n) \subset [f]([x_1], [x_2], ..., [x_n])
\]

for \(x_1 \subset [x_1], x_2 \subset [x_2], ..., x_n \subset [x_n]\). It states that if the input variables lie within the corresponding intervals we can use interval arithmetic to produce an interval for the output of the function which is guaranteed to contain the crisp function output \(f(x)\).

An important aspect is the following: if \(f(x)\) is a real rational expression in which each variable \(x_i\) occurs only once and only to the first power, then the function evaluations with interval variables bounds the function \(f(x)\) tightly over the set of intervals \([x_i]\), i.e. \(\sup [f]([x]) = \max f(x)\) and \(\inf [f]([\text{mathbf f} x]) = \min_{\text{mathbf f} \in \{\text{mathbf f} x\}} f(\text{mathbf f} x)\). This is a direct consequence of the properties of interval arithmetic. However, when an interval parameter occurs more than once the bounds on the function output can become non-tight, i.e. \(\sup [f]([x]) > \max f(x)\) and \(\inf [f]([x]) < \min f(x) \forall x \in [x]\). This effect is known as the dependency problem.
An important aspect of interval arithmetic is its implementation on digital computers. Since digital computers have limited capacity of representing a number, some form of error bounding is required to obtain correct results. Especially for prolonged integration of differential equations, as is often the case for spacecraft trajectory simulation (formation flying, rendezvous, etc.), machine roundoff errors can have a significant effect on the integrated results. Although some people had been working on bounding rounding errors in the 1950’s, the breakthrough for interval analysis came in the 1960’s with Moore’s book *Interval Analysis*.\(^{16}\) Interval arithmetic can cope with the inherent problems related to the limited precision of numbers in digital computers by replacing a crisp number with an interval where the lower bound is the largest machine-representable number lower than the crisp number and the upper bound is the smallest machine-representable number larger than the crisp number. For example in double precision (64 bit) floating point arithmetic according to IEEE Standard 754,\(^{19}\) the next machine representable number after the number 1 is \(1 + z\), where \(z=2.220446049250313e-16\). This means that the computation \(1 + z/3\) will be equal to 1. Interval arithmetic software, such as the INTLAB\(^{20}\) toolbox for Matlab, will switch rounding modes between up and down and return the interval encompassing the different solutions, such that the computation \(1 + z/3\) will result in the interval \([1, 1 + z]\). All interval computations in this paper have been performed using INTLAB in order to guarantee correct bounding of the roundoff errors.

### III.A. Branch and bound loop

The branch and bound method is an iterative algorithm to efficiently remove parts of the search space that do not contain the global minimum of cost function. It is not a fixed algorithm on its own, but rather a container for several interval methods. For each specific application one can construct a branch and bound algorithm using an appropriate set of interval methods. There has been a gradual change over time from ‘bounding’ to ‘branching’ as available computing power increased. In the time of Moore’s first book\(^{16}\) on interval analysis, mid 1960’s, the focus is on interval contracting methods and splitting of intervals is hardly mentioned. By the time of Jaulin’s book\(^{21}\) on application of interval analysis in 2001, bisection of boxes appears in most of the described global optimization methods. In general, bounding methods are more memory efficient then branching methods, while branching methods can allow for a faster reduction of the search space. With the enormous amount of computer memory and computation speed available today\(^a\), it can be advantageous to aim for more branching first, although it remains an application specific decision what components to put in the branch and bound method.

Figure 2 shows a diagram of a general branch and bound loop. The branch and bound loop will be discussed in more detail in section IV, where this general layout of the branch and bound algorithm is adapted for the rendezvous and docking optimization.

### IV. Interval Optimization for Rendezvous and Docking

In this section it will be shown how interval analysis can be advantageous for rendezvous and docking optimization. In the first place interval analysis is a great tool for sensitivity analysis of the equations of relative motion, i.e. the investigation of how parameters, such as initial position and velocity, affect the trajectory of the spacecraft.

In the second place, several rendezvous and docking interval optimization scenarios are treated:

- **Rendezvous**: bring relative position to zero.
- **Docking**: bring relative position and velocity to zero.
- **Minimize fuel use while docking**.
- **Obstacle avoidance while docking with minimal fuel**.

Attitude dynamics and control are not investigated here, so the spacecraft is assumed to be capable of always firing thrusters in the three directions given by the definition of the coordinate frame in Fig. 1.

---

\(^a\)Moore’s IBM 7094 had 0.35 Million Instructions Per Second (MIPS), while a current INTEL i7 920 system can easily do 6000 MIPS
Fig. 2. Schematic flowchart of the interval branch and bound algorithm. Partitions of the search space that cannot contain the global minimum of the cost function are eliminated. Since these partitions are interval boxes, an infinite number of crisp parameter combinations can be eliminated at once.
IV.A. Sensitivity analysis

Before the rendezvous and docking optimization is performed, this section gives some examples of solutions to the CW equations for both crisp and interval parameters. First the straightforward case without thrusters or other external forces, i.e. only the homogeneous solution is used. Fig. 3 gives the relative position of the chaser with respect to the target for an initial offset in position, no initial offset in velocity, and no additional forces. The motion in $y$ and $z$ direction is oscillatory, while the relative distance in $x$ direction increases with time. Of course this follows also directly from (2), in which the $y$ and $z$ directions only contain oscillatory terms, while the $x$-direction also contains a term that is linearly increasing with time.

\[ x[m] \]
\[ y[m] \]
\[ z[m] \]

\[ t[s] \]

\[ 0 \]
\[ 5000 \]
\[ 10000 \]
\[ 0 \]
\[ 5000 \]
\[ 10000 \]
\[ 0 \]
\[ 5000 \]
\[ 10000 \]

Fig. 3. Homogeneous solution to CW equations, with $x_0 = y_0 = z_0 = 100m$ and zero initial relative velocity.

In the next example, the initial distance between the spacecraft is assumed to be known with limited precision, for example due to limited accuracy of position sensors. The uncertainty on the position is represented by an interval, which is added to the estimated position. As long as the width of the uncertainty interval is chosen wider than the actual measurement error on the position, the true position will be contained in the summation of the estimated position and the uncertainty interval. This is an essential requirement for the interval optimization algorithm. Fig. 4 gives the trajectories for an uncertainty of $[-10, 10]$ in all directions on top of the initial relative position from the previous example. Simply inserting these intervals on top of the crisp values in (2) results in intervals for the relative position. The bounds on the $y$ and $z$ positions are cyclical and limited, while the bounds on the relative $x$ position diverge.

\[ x[m] \]
\[ y[m] \]
\[ z[m] \]

\[ t[s] \]

\[ 0 \]
\[ 5000 \]
\[ 10000 \]
\[ 0 \]
\[ 5000 \]
\[ 10000 \]
\[ 0 \]
\[ 5000 \]
\[ 10000 \]

Fig. 4. Homogeneous solution to CW equations, with $x_0 = y_0 = z_0 = [90, 110]m$ and zero initial relative velocity.

Similar to adding uncertainty in the initial relative position, uncertainty can also be added on the initial relative velocities. Fig. 5 gives the homogeneous solution computed with interval numbers for the initial velocities. In this case the initial velocity is set to $[0, 0.1]$ for each direction, while the initial relative distance of the spacecraft is set to zero. From the interval inclusion (Theorem III.1) it follows that since the interval for the initial velocity contains zero, the resulting motion should also contain the trajectory for the case where there is no initial velocity and no initial offset in position. Clearly this is true in Fig. 5, where the
lines $x = 0, y = 0, z = 0$ are contained in the area between the lower and upper bounds of the trajectory. Similar to the case of an interval for the relative initial position, the bounds on the $y$ and $z$ positions are cyclical and limited, while the bounds on the relative $x$ position diverge.

\[ x_0 = [0, 0], y_0 = [0, 0], z_0 = [0, 1] \]

The CW equations are an excellent example of the sensitivity analysis capabilities of interval analysis. Any of the parameters or variables can be replaced by an interval and the effects on the relative motion will become apparent. For example the somewhat unrealistic but theoretically interesting case where the orbital rate of the satellites is uncertain with a margin of plus and minus 10 percent on the nominal value. From Fig. 6 it can be concluded that the interval bands on all three position variables increase over time, in contrast to the previous case with uncertain initial relative position.

\[ x_0 = y_0 = z_0 = 100 \text{ m} \]

Effects of thruster pulses on the relative motion are computed with the particular solution in (3). First the regular crisp pulse magnitude is considered. The effect of a single pulse in the $x$ direction, starting at $t = 5000$ is given in Fig. 7, where the same initial conditions are used as in Fig. 3. In blue is the original homogeneous solution for no thruster input, while the red line shows the combined effect of the homogeneous solution and the particular solution from the thruster pulse. Since the in-plane and out-of-plane equations of motion are decoupled, there is no influence of a pulse in $x$-direction on the $y$-component of the relative position.

Analogous to the homogeneous solution, any uncertainty on the magnitude of the thruster pulse can be represented by an interval. If an uncertainty bound is added to the magnitude of the thruster pulse, the range of possible trajectories can be determined by substituting interval numbers in the particular solution. Fig. 8 shows the effect of a 50 percent uncertainty on the thruster pulse from Fig. 7.

These examples show the possibilities of interval analysis applied to the analytic solution of the CW equations. The next step will be to use interval optimization techniques to optimize some practical trajectories, such as docking while avoiding certain regions, or docking with minimal fuel use.
Fig. 7. Particular solution to CW equations, with $x_0 = y_0 = z_0 = 100\,\text{m}$ and zero initial relative velocity. A thruster pulse in the $x$ direction is given at $t = 5000\,\text{s}$. Blue is the original homogeneous solution, red is the combination of homogeneous and particular solution.

Fig. 8. Particular solution to CW equations, with $x_0 = y_0 = z_0 = 100\,\text{m}$ and zero initial relative velocity. A thruster pulse with an interval amplitude in the $x$ direction is given at $t = 5000\,\text{s}$. Blue is the original homogeneous solution, red is the combination of homogeneous and particular solution.
IV.B. Rendezvous

For the rendezvous case the goal is just to get the relative position close to zero, i.e. to get two spacecraft colliding with each other. Relative velocities will be dealt with in the docking section. The control input has the form of pulsed thruster burns. Specification of the input structure is required to reduce the complexity of the optimization problem. In this case the input is structured as a set of three fixed-length thruster pulses of varying magnitude, one in each direction. The timing of the pulses is also fixed in the input structure. For the first example the three pulses are given at time zero (see Fig. 9). The optimization problem is now to determine the magnitude of the pulses in such a way that the relative position of the spacecraft is as close to zero as possible at a given final time.

The initial search space for the pulse magnitudes is set as:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} [-50, 50] \\ [-50, 50] \\ [-50, 50] \end{bmatrix}$$  \hspace{1cm} (12)$$

Fig. 9. Input structure for the rendezvous optimization: 3 pulses at $t=0$, one in each direction.

The initial search space for the pulse magnitudes is set as:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} [-50, 50] \\ [-50, 50] \\ [-50, 50] \end{bmatrix}$$  \hspace{1cm} (12)$$

Fig. 10. Complete trajectory space for initial search space consisting of three interval pulses at $t=0$, one each in $x, y,$ and $z$, with magnitude $[-50, 50]$. The initial relative position is $100 \ [m]$ in all directions, while the initial relative velocity is zero.

If the complete search space as set by these intervals is entered into the equations governing the relative motion, then the result is as depicted in Fig. 10. A set of conclusions can be obtained from this figure. Firstly, all the regions outside of the resulting intervals for the relative position cannot be reached, whatever the crisp magnitudes of the pulses are (as long as they are within the search space). Secondly, the shape of the intervals for the $x, y,$ and $z$ trajectories gives an insight into the underlying dynamics, as was discussed earlier. Finally it can be concluded that the solution to the rendezvous optimization must lie within the
interervals given here. Since the point of zero relative position is included in all three intervals at the final time, there might be a set of pulses that will satisfy the goal of rendezvous, but this cannot yet be guaranteed. If the point of zero relative position had not been included at the final time, then it would be certain there is no solution within this search space.

Since the in-plane and out-of-plane motions are uncoupled, they are optimized separately. For the in-plane motion there are two interval input parameters: pulse magnitude in x-direction and pulse magnitude in z-direction. These inputs are used to build two particular solutions which are then added to the homogeneous solution to obtain the complete trajectory. The cost function is based on the distance between the spacecrafts at \( t = t_f \). In interval notation, the cost functions for in-plane \([J_i]\) and for out-of-plane \([J_o]\) become:

\[
[J_i] = |[x(t_f)]| + |[z(t_f)]| \tag{13}
\]

\[
[J_o] = |[y(t_f)]| \tag{14}
\]

Fig. 11 contains the schematic block diagram of the interval branch and bound algorithm adapted to the method of rendezvous optimization using the homogeneous and particular solutions of the CW-equations. The input for the branch and bound loop consists of three elements. First the search space has to be defined. In the current example of rendezvous optimization, these are the intervals for the magnitudes of the thruster pulses. The user has to set a lower and upper bound on this magnitude, which can be derived for example from thruster specifications or mission requirements. Once the search space is defined, the next input to the branch and bound loop is the input structure. In the current example this input structure specifies at which times the thrusters are fired. Finally, the initial conditions, i.e.: initial relative position and velocity, are required to compute the homogeneous solution.

When the inputs for the branch and bound loop have been specified, the loop starts by adding the particular solutions for the current box (interval vector of thruster impulse amplitudes) to the homogeneous solution and computing the interval cost function. The best estimate of the minimum value of the cost function is given by the upper bound of the cost function evaluation. When this upper bound is lower than the current estimate of the minimum value, it is set as the new minimum value estimate. Initially, there is no information on the best estimate of the minimum value of the cost function \( J_{\text{min}}^* \), so it is set to infinity:

\[
J_{\text{min}}^* = \infty \tag{15}
\]

After each new cost function evaluation for a certain interval parameter vector \([P]\), \( J_{\text{min}}^* \) is updated as follows:

\[
J_{\text{min}}^* = \min (J_{\text{min}}^*, \sup ([J([P])])) \tag{16}
\]

If the lower bound of the cost function evaluation is higher than the current estimate of the minimum value, the box \([P]\) can be deleted from the search space:

\[
\inf ([J([P])]) > J_{\text{min}}^* \rightarrow \text{delete} \ [P] \tag{17}
\]

At this point any additional constraints, for example originating from obstacle avoidance requirements, are imposed on the solution. If a box does not satisfy the constraints, it will be removed. The next step in the loop is to check if the box satisfies the stopping criteria, which can be formulated either as a minimum box width or as a minimum value of the cost function width:

\[
diam ([P]) < \varepsilon_P \quad , \quad diam ([J]) < \varepsilon_J \tag{18}
\]

where \( diam ([x]) = \sup [x] - \inf [x] \).

Boxes satisfying the stopping criteria are stored in a list of solutions, while boxes not satisfying the criteria are split into sub-boxes. The order in which boxes are split influences the total required computation time to solve the problem. Unfortunately, it is in general not possible to determine the optimal order in advance. In this paper a volume reduction approach is taken, meaning that a multidimensional interval vector is always split in the dimension that is widest, leading to the fastest reduction in the total volume of the search space. On the implementation side all interval boxes are split at the same time, i.e. parallel splitting, doubling the number of boxes (and the memory footprint) each time. With an interpreted language such as MATLAB this implementation is much faster than sequential splitting.

11 of 23

American Institute of Aeronautics and Astronautics
Fig. 11. Schematic structure of the branch and bound algorithm for rendezvous and docking optimization using the analytical solution to the CW equations.
If dimension $p$ of interval vector $[\mathbf{x}]$ has the largest width, then the bisection yields two new interval vectors:

$$
[x]_1 = ([x^1], [x^2], ..., [\inf [x^p]], \text{mid } [x^p], ..., [x^n])^t
$$

$$
[x]_2 = ([x^1], [x^2], ..., [\text{mid } [x^p]], \text{sup } [x^p], ..., [x^n])^t
$$

Fig. 12 shows the results of the interval optimization algorithm applied to the rendezvous problem. The three optimal thruster pulse amplitudes are determined to be:

$$
[u^*] = \begin{bmatrix}
[u_1] \\
[u_2] \\
[u_3]
\end{bmatrix} = \begin{bmatrix}
[2.118117, 2.118120] \\
[-0.144100, -0.144088] \\
[-0.236034, -0.235930]
\end{bmatrix}
$$

With these pulses the relative position at the specified final time (here 10000) is as close to zero as possible with this input structure. The value of the cost function for this optimal interval input is:

$$\[J((u^*)) = [0, 0.0145]\]$$

The next step is to get the relative velocity to zero as the spacecraft meet.

### IV.C. Docking optimization

In this scenario it is required that the relative position between the target and chaser becomes zero and that the velocity at the moment of contact is below a certain threshold. The cost functions for in-plane and out-of-plane motion can therefore be chosen as:

$$
[J_1] = ||x(t_f)|| + ||z(t_f)|| + ||\dot{x}(t_f)|| + ||\dot{z}(t_f)||
$$

$$
[J_0] = ||y(t_f)|| + ||\dot{y}(t_f)||
$$

For the rendezvous problem and the docking problem, one can choose to put the requirements for zero relative distance and velocity into either the cost function or to impose them as constraints. For the results in this section, a combination is made by defining the cost function based on the relative position together with a constraint on the velocity at $t_{final}$:

$$
[J_1] = ||x(t_f)|| + ||z(t_f)||, \quad ||\dot{x}(t_f)|| < \varepsilon_x, \quad ||\dot{z}(t_f)|| < \varepsilon_z
$$

This constraint will remove any boxes that do not have the zero relative velocity at $t_{final}$ in them. That is: no combination of thruster pulse amplitudes in the input box results in an interval trajectory that contains the zero relative velocity.
As in Sec. IV.B, a predefined input structure is required to reduce the complexity of the problem. With only the three thruster burns at \( t_0 \) as in the rendezvous case, the velocity cannot become zero for all three directions at \( t_{final} \). So a set of three additional pulses is added at 75% of \( t_{final} \), resulting in the input structure that is given in Fig. 13.

![Input structure for the docking optimization: 3 pulses at \( t = 0 \), one in each direction, and 3 pulses at \( t = \frac{3}{4}(t_f - t_0) \), one in each direction.](image)

Fig. 13. Input structure for the docking optimization: 3 pulses at \( t = 0 \), one in each direction, and 3 pulses at \( t = \frac{3}{4}(t_f - t_0) \), one in each direction.

Fig. 14 shows the results of the interval optimization algorithm applied to this docking problem. The six optimal thruster pulses are determined to be:

\[
[u^*] = \begin{bmatrix}
[u_1] \\
[u_2] \\
[u_3] \\
[u_4] \\
[u_5] \\
[u_6]
\end{bmatrix} = \begin{bmatrix}
[2.39238, 2.39247] \\
[0.13409, 0.13418] \\
[0.66661, 0.66709] \\
[-0.23650, -0.23633] \\
[1.08624, 1.08633] \\
[1.62119, 1.62137]
\end{bmatrix}
\]  \hspace{1cm} (25)

With these pulses the relative position and velocity at the specified final time are as close to zero as possible with this input structure. Because there are no more inputs after 75% of \( t_{final} \), all relative velocities must be brought to zero by the last pulse. Obviously the relative position is fixed as soon as the relative velocities go to zero. Therefore the first set of pulses at \( t_0 \) has to be optimized such that the relative position at the time of the second set pulses is zero. The second set of pulses then reduces the remaining velocities, such that all docking requirements are met.

![Optimal docking trajectory for input consisting of three interval pulses at \( t = 0 \), and three pulses at 75% of \( t_{final} \).](image)

Fig. 14. Optimal docking trajectory for input consisting of three interval pulses at \( t = 0 \), and three pulses at 75% of \( t_{final} \).

For this optimization problem, there is only one optimal solution, i.e.: the solution in Fig. 14. There is no other combination of pulses that results in docking, so with this input structure it makes no sense to look at fuel optimization.
IV.D. Minimize fuel use while docking

In this section the goal is to minimize fuel use while docking. Fuel use is represented here by the summed amplitude of all the thruster pulses. As was shown in the previous section, there is only a single solution for docking when 6 pulses are used, so in this section the input structure is appended with 3 additional pulses. Fig. 15 gives an overview of the input structure. With these additional pulses there are infinitely many solutions to the docking problem. One of the thruster pulses can be changed slightly and the other pulses can be corrected for this change, such that the relative position and velocity are zero at t\text{final}. The interval optimization algorithm shows that there are infinitely solutions which lie on a line in the search space.

The cost function for this problem consists of a penalty term proportional to the magnitude of the thruster pulses, with constraints on the final position(ε\text{x},ε\text{y},ε\text{z}) and velocity(ε\dot{x},ε\dot{y},ε\dot{z}):

\[ J = \sum_{i=1}^{N\text{ pulses}} |[u_i]| \]

\[ |x(t_f)| < \varepsilon_x, \quad |y(t_f)| < \varepsilon_y, \quad |z(t_f)| < \varepsilon_z \]

\[ |\dot{x}(t_f)| < \varepsilon_{\dot{x}}, \quad |\dot{y}(t_f)| < \varepsilon_{\dot{y}}, \quad |\dot{z}(t_f)| < \varepsilon_{\dot{z}} \]

An important aspect of the fuel optimization problem is that there is a conflict between the cost function minimization and satisfying the constraints at t\text{final}. For the rendezvous and docking cases, minimization of the cost function automatically means that the final state constraints are satisfied, since the lowest cost function occurs where final position and velocity are zero.

In the case of fuel optimization, the best strategy for minimization of fuel use is to do nothing, but this solution will obviously not satisfy the final constraints. The danger in the branch and bound algorithm is that an infeasible solution (i.e. not satisfying the final constraints) is used to lower the estimate of the cost function minimum. When this has happened, then the feasible solutions might not be able to obtain this low value of the cost function, which means they are deleted by the branch and bound algorithm. The result is that in the end all boxes are deleted and no solutions are found.

To prevent this from happening, the current best estimate of the minimum value of the cost function may only be lowered if the box under investigation actually leads to a solution that satisfies all constraints.

![Diagram of input structure for fuel optimization](attachment:input_structure.png)

Fig. 15. Input structure for the fuel optimization: 3 pulses at t = 0, one in each direction, 3 pulses at t = \frac{1}{2}(t_f - t_0), one in each direction, and 3 pulses at t = \frac{3}{4}(t_f - t_0), one in each direction.

Fig. 16 shows the most fuel efficient docking trajectory for this input structure. This trajectory is generated by evaluating the hull of all remaining boxes from the branch and bound algorithm. Clearly the final state constraints have been satisfied. The amplitudes of the thruster pulses corresponding to this
solution are:

\[
[u^*] = \begin{bmatrix}
  [u_1] \\
  [u_2] \\
  [u_3] \\
  [u_4] \\
  [u_5] \\
  [u_6] \\
  [u_7] \\
  [u_8] \\
  [u_9]
\end{bmatrix} = \begin{bmatrix}
  [2.167283, 2.167286] \\
  [0.134140, 0.134158] \\
  [-0.000003, 0.000004] \\
  [-0.161050, -0.161048] \\
  [-0.000030, 0.000012] \\
  [-1.056956, -1.056940] \\
  [0.149778, 0.149781] \\
  [1.086223, 1.086259] \\
  [-0.000003, 0.000018]
\end{bmatrix}
\] (27)

See also Fig. 15 for the timing and direction of these pulses. Three of the pulses have been optimized to zero amplitude: the first and third pulse in \(z\)-direction as well as the second pulse in the \(y\)-direction. For the \(y\)-direction this means that the first pulse brings the relative position to zero at the time of the third pulse, which then brings the relative velocity to zero. For the coupled \(x\) and \(z\) directions, things are more complicated.

The sum of amplitudes of this solution is: \([4.7554, 4.7556]\), while the sum of amplitudes of the thruster pulses for the docking solution in Sec. IV.C is: \([6.1368, 6.1380]\). It can be concluded that even with the addition of three pulses, the total fuel use has decreased. From all trajectories that lead to docking for this input structure, the solution found by the interval optimization algorithm is the one that requires the least amount of fuel.

**IV.E. Obstacle avoidance while docking**

In this section the goal is to perform a fuel-optimized docking maneuver, while avoiding a prescribed area. This procedure can simulate docking along specific direction, e.g. R-bar or V-bar, as is done in ref.,\(^{11}\) or docking while avoiding external structures such as solar panels. Of course attitude information is required when the attitude of the target spacecraft is not constant with respect to the CW-frame, but the trajectory optimization itself is still valid without this information.

The additional requirements in this optimization problem can either be imposed through the cost function or through constraints. In this case it is more natural to describe the obstacles in the constraints and leave...
the cost function the same as the fuel-optimization cost function:

$$J = \sum_{i=1}^{N_{\text{pulses}}} |u_i|$$  \hspace{1cm} (28)

$$\begin{align*}
    &|[x(t_f)]| < \varepsilon_x, &|[y(t_f)]| < \varepsilon_y, &|[z(t_f)]| < \varepsilon_z \\
    &|[\dot{x}(t_f)]| < \varepsilon_x, &|[\dot{y}(t_f)]| < \varepsilon_y, &|[\dot{z}(t_f)]| < \varepsilon_z
\end{align*}$$

If the obstacles are given as intervals \([x_{\text{obs}}, y_{\text{obs}}, z_{\text{obs}}]\), then the constraint that has to be satisfied for all time steps is:

$$[[x(t)] \cap [x_{\text{obs}}]] \cup [[y(t)] \cap [y_{\text{obs}}]] \cup [[z(t)] \cap [z_{\text{obs}}]] =$$  \hspace{1cm} (29)

During the branch and bound loop any remaining boxes resulting in a trajectory that enters the obstructed area, at any time, are deleted. As was the case in Sec. IV.D, the minimum estimate of the cost function should only be lowered by a solution that satisfies all the constraints, otherwise the optimization terminates with all thruster pulse amplitudes optimized to zero.

The input structure is chosen equal to that in Fig. 15. Also the initial conditions are equal to those in Sec. IV.D, such that the resulting trajectory is exactly the same if the obstacle is removed. In the right of Fig. 18 the \(x\) and \(z\) components of the unobstructed trajectory are plotted against each other. An obstacle is then selected to deliberately obstruct this trajectory, in this case the obstacle is selected as a rectangle in the \(xz\)-plane with vertices:

- \((-50, 5)\)
- \((-50, 50)\)
- \((100, 50)\)
- \((100, 5)\)

In this example the obstacle has no thickness, so no component in the \(y\)-direction, but since the out-of-plane dynamics are uncoupled from the in-plane dynamics, the same optimization procedure can be performed for obstacles in the \(y\)-direction. The left part of Fig. 18 as well as Fig. 17 give the results of the optimization. The chaser spacecraft is now forced to approach the target from the opposite direction, because the obstacle blocks the original docking path. An additional benefit of interval optimization is that it can tell when there is no possible way for the chaser to reach the target using any of the thruster amplitudes in the search space. When the obstacle is placed over the target, all boxes are deleted during the branch and bound loop and the algorithm will report that no solution is possible.

Because the trajectory in Sec. IV.D is the minimal fuel trajectory, the obstacle avoidance trajectory should have a higher fuel cost. The amplitudes of the pulses corresponding to the solution in Fig. 17 are:

$$\begin{bmatrix}
    u_1 \\
    u_2 \\
    u_3 \\
    u_4 \\
    u_5 \\
    u_6 \\
    u_7 \\
    u_8 \\
    u_9
\end{bmatrix}^* =
\begin{bmatrix}
    2.152571, 2.152579 \\
    0.134140, 0.134158 \\
    0.627943, 0.627955 \\
    -0.010904, -0.010901 \\
    -0.000030, 0.000012 \\
    -1.253981, -1.253940 \\
    0.149778, 0.149781 \\
    1.086223, 1.086259 \\
    -0.000002, 0.000047
\end{bmatrix}$$  \hspace{1cm} (30)

The sum of these pulses is [5.2800, 5.2803], which is indeed larger than the value for minimal fuel use.
Fig. 17. Trajectory for fuel minimization while docking and avoiding obstacles with input consisting of three interval pulses at $t = 0$, three pulses at 50% of $t_{final}$, and three pulses at 75% of $t_{final}$.

Fig. 18. x-z trajectory for fuel minimization while docking. On the right the fuel optimization from Sec. IV.D. On the left the additional constraints due to the obstacle are satisfied. The black star represents the initial position; the red circle is the final position.
V. Comparison between interval optimization and gradient based optimization

In this section a comparison between the results of the interval optimization and a gradient-based optimization are presented. As was discussed previously, the cost function for all cases is convex, which leads to the hypothesis that a gradient based algorithm should also be able to find the global minimum. However, in the last optimization case, the obstacles introduce nonlinear constraints into the problem, effectively creating multiple local minima in the cost function, which leads to the hypothesis that gradient based algorithms will have problems obtaining the global minimum. As gradient-based algorithm MATLAB’s *fmincon* function is chosen, which allows equality and inequality constraints in the optimization.

V.A. Rendezvous optimization

The rendezvous optimization has three parameters, one pulse in each direction, and three equality constraints: final position equal to zero in all directions. The cost function is simply taken as the sum of absolute magnitudes of the pulses. Table 1 gives the result for the gradient based optimization together with the previously computed interval optimization results. Clearly, the same solution is found by both methods. A random starting point for the gradient based method is used that lies within the initial interval that defines the search space for the interval optimization. For any random starting point, the gradient-based method always ends up in the same minimum, which confirms that this optimization problem is convex.

<table>
<thead>
<tr>
<th>Input</th>
<th>IA</th>
<th>GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[u_1]$</td>
<td>2.118117, -2.118120</td>
<td>2.1181</td>
</tr>
<tr>
<td>$[u_2]$</td>
<td>-0.144100, -0.144088</td>
<td>-0.1441</td>
</tr>
<tr>
<td>$[u_3]$</td>
<td>-0.236034, -0.235930</td>
<td>-0.2360</td>
</tr>
<tr>
<td>$\sum_{i=1}^{3}</td>
<td>[u_i]</td>
<td>$</td>
</tr>
</tbody>
</table>

V.B. Docking optimization

The docking optimization has six parameters, two pulses in each direction, and six equality constraints: final position and final velocity equal to zero in all directions. The cost function is again taken as the sum of absolute magnitudes of the pulses. Table 2 gives the result for the gradient based optimization together with the previously computed interval optimization results. Clearly, the same solution is found by both methods. Similar to the rendezvous case, for any random starting point, the gradient-based method always ends up in the same minimum, which confirms that this optimization problem is also convex.

<table>
<thead>
<tr>
<th>Input</th>
<th>IA</th>
<th>GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[u_1]$</td>
<td>2.39238, 2.39247</td>
<td>2.3924</td>
</tr>
<tr>
<td>$[u_2]$</td>
<td>0.13409, 0.13418</td>
<td>0.1341</td>
</tr>
<tr>
<td>$[u_3]$</td>
<td>0.66661, 0.66709</td>
<td>0.6668</td>
</tr>
<tr>
<td>$[u_4]$</td>
<td>-0.23650, -0.23633</td>
<td>-0.2364</td>
</tr>
<tr>
<td>$[u_5]$</td>
<td>1.08624, 1.08633</td>
<td>1.0863</td>
</tr>
<tr>
<td>$[u_6]$</td>
<td>1.62119, 1.62137</td>
<td>1.6213</td>
</tr>
<tr>
<td>$\sum_{i=1}^{6}</td>
<td>[u_i]</td>
<td>$</td>
</tr>
</tbody>
</table>

V.C. Fuel optimization

The fuel optimization has nine parameters, three pulses in each direction, and six equality constraints: final position and final velocity equal to zero in all directions. The cost function is again taken as the sum of
absolute magnitudes of the pulses. Table 3 gives the result for the gradient based optimization together with the previously computed interval optimization results. Clearly, the same solution is found by both methods. Similar to the rendezvous and docking cases, for any random starting point, the gradient-based method always ends up in the same minimum, which confirms that this optimization problem is also convex.

Table 3. Comparison between gradient-based (GB) optimization and interval optimization (IA) for the fuel optimization case.

<table>
<thead>
<tr>
<th>Input $u_i$</th>
<th>IA $[2.167283, 2.167286]$</th>
<th>GB $2.1673$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>$[0.134140, 0.134158]$</td>
<td>$0.1341$</td>
</tr>
<tr>
<td>$u_2$</td>
<td>$[-0.000003, 0.000004]$</td>
<td>$0.0000$</td>
</tr>
<tr>
<td>$u_3$</td>
<td>$[-0.161050, -0.161048]$</td>
<td>$-0.1610$</td>
</tr>
<tr>
<td>$u_4$</td>
<td>$[-0.000030, 0.000012]$</td>
<td>$0.0000$</td>
</tr>
<tr>
<td>$u_5$</td>
<td>$[-1.056956, -1.056940]$</td>
<td>$-1.0569$</td>
</tr>
<tr>
<td>$u_6$</td>
<td>$[0.149778, 0.149781]$</td>
<td>$0.1498$</td>
</tr>
<tr>
<td>$u_7$</td>
<td>$[1.086223, 1.086259]$</td>
<td>$1.0863$</td>
</tr>
<tr>
<td>$u_8$</td>
<td>$[-0.000003, 0.000018]$</td>
<td>$0.0000$</td>
</tr>
<tr>
<td>$\sum_{i=1}^{9}</td>
<td>u_i</td>
<td>$</td>
</tr>
</tbody>
</table>

V.D. Obstacle optimization

The obstacle avoidance optimization has nine parameters, three pulses in each direction and six equality constraints: final position and final velocity equal to zero in all directions. An additional constraint for this case is that the chaser satellite does not enter the forbidden area at any time step. The cost function is again taken as the sum of absolute magnitudes of the pulses. Table 4 gives the result for the gradient based optimization, for a random initial condition, together with the previously computed interval optimization results. Clearly, the solutions are different and the cost function found by the interval optimization algorithm is lower.

Table 4. Comparison between gradient-based (GB) optimization and interval optimization (IA) for the obstacle avoidance case.

<table>
<thead>
<tr>
<th>Input $u_i$</th>
<th>IA $[2.152571, 2.152579]$</th>
<th>GB $2.0333$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>$[0.134140, 0.134158]$</td>
<td>$-0.3576$</td>
</tr>
<tr>
<td>$u_2$</td>
<td>$[0.627943, 0.627955]$</td>
<td>$-0.0103$</td>
</tr>
<tr>
<td>$u_3$</td>
<td>$[-0.010904, -0.010901]$</td>
<td>$-0.1644$</td>
</tr>
<tr>
<td>$u_4$</td>
<td>$[-0.000030, 0.000012]$</td>
<td>$1.1299$</td>
</tr>
<tr>
<td>$u_5$</td>
<td>$[-1.253981, -1.253940]$</td>
<td>$-1.7597$</td>
</tr>
<tr>
<td>$u_6$</td>
<td>$[0.149778, 0.149781]$</td>
<td>$0.2871$</td>
</tr>
<tr>
<td>$u_7$</td>
<td>$[1.086223, 1.086259]$</td>
<td>$2.0449$</td>
</tr>
<tr>
<td>$u_8$</td>
<td>$[0.000002, 0.000047]$</td>
<td>$-0.9040$</td>
</tr>
<tr>
<td>$\sum_{i=1}^{9}</td>
<td>u_i</td>
<td>$</td>
</tr>
</tbody>
</table>

VI. Conclusions

In this paper an interval optimization algorithm is applied to the problem of fixed-time spacecraft rendezvous optimization for multiple thruster impulses. Since spacecraft rendezvous and docking is a trajectory optimization problem, most trajectory optimization algorithms have previously been applied to the rendezvous and docking problem, for example: nonlinear programming, optimal control, numerical analysis,
Fig. 19. Minimum cost function value found by a gradient based optimization algorithm for the case of rendezvous and docking while avoiding obstacles. Each trial starts at a random initial point located within the initial interval box that defines the search space for the interval optimization algorithm.

Fig. 20. Trajectory found by gradient based fuel minimization while docking and avoiding obstacles with input consisting of three interval pulses at $t = 0$, three pulses at 50% of $t_{final}$, and three pulses at 75% of $t_{final}$. 
The relative dynamics of two satellites are described by the Clohessy-Wiltshire equations, a set of second order differential equations. Because direct interval integration quickly leads to divergence in the solution, the CW-equations are linearized and solved analytically. The analytical solution, consisting of a homogeneous solution describing the effect of the initial conditions, and a set of particular solutions describing the effect of thruster pulses, is then used to generate interval trajectories. Although the optimization problem is now convex, nonlinear constraints will be added in the form of obstacles that need to be avoided during docking. These additional constraints create multiple local minima in the optimization problem.

Interval analysis is a great tool for sensitivity analysis of the equations of relative motion, i.e. investigation of how parameters such as the initial conditions affect the trajectory of the spacecraft. Four optimization problems are solved in this paper: spacecraft rendezvous, docking, docking with minimum fuel use, and finally docking with minimum fuel use while avoiding arbitrarily placed obstacles.

For rendezvous it is only required that the distance between the spacecraft is zero at a prescribed time. There is only one solution in terms of thruster amplitudes when the input structure consists of three pulses at initial time in each of the three orthogonal directions of the Hill frame.

For docking optimization it is required that the distance and relative velocity is zero at a prescribed time. There is no solution when the only inputs are at initial time, but with an additional set of three pulses a solution can be found. The first pulse ensures that that relative distance is zero at the time of the second pulse, which then removes any remaining relative velocity.

To minimize fuel use, an additional set of three impulses is added, giving a total of nine thruster pulses. With this input structure there are infinitely many ways to achieve docking. The interval optimization algorithm finds the combination of pulse amplitudes that results in docking with minimal fuel use. Care has to be taken that the minimum cost function estimate is only lowered by boxes that satisfy all final state constraints.

In the last optimization, an obstruction is placed in the trajectory corresponding to minimum fuel use. It is straightforward to impose these constraints in the branch and bound loop and any boxes leading to a trajectory crossing the obstruction are eliminated, resulting in an alternative trajectory. The results show that the chaser now approached the target from the opposite direction. Smart placement of imaginary obstructions can therefore be used to optimize the trajectory to any docking port on the target.

A comparison of the obtained interval results with a gradient based optimization algorithm shows that the interval method finds solutions with a lower value of the cost function. This indicates that multiple local minima are present in the optimization problem and the gradient based algorithms can get stuck in one of these local minima.
Overall it can be concluded that the interval optimization algorithm provides a better insight in the problem through the sensitivity analysis and provides optimal rendezvous and docking trajectories with minimum fuel use while avoiding obstacles, making it a good alternative to the existing optimization algorithms.

References