AIAA Guidance, Navigation, and Control Conference
08 - 11 August 2011, Portland, Oregon

AIAA 2011-6657

Trajectory Optimization Based on Interval Analysis

E. de Weerdt * , E. van Kampen † , Q.P. Chu ‡ , J.A. Mulder §

Delft University of Technology, Control and Simulation Division

P.O. Box 5058, 3600 GB Delft, The Netherlands

Trajectory optimization has been a large field of research for many years. The drawback is that for non-convex, constrained problems practically all available solvers cannot guarantee that the globally optimal trajectory is found. Interval analysis based solvers however can provide this guarantee. Interval analysis has been applied to trajectory optimization before, but the previously presented methods suffered from major drawbacks which limited their application to small scale problems. In this paper a new interval based method is introduced which incorporates state parameterization to prevent explicit integration. The performance of the proposed method is demonstrated by applying it to a spacecraft formation flying optimization problem. The results are compared with a gradient based solver and it is shown that the interval method is guaranteed to find the global optimal solution. Finally the first steps for another new trajectory optimization method based on interval analysis and direct collocation are presented.

I. Introduction

The problem of trajectory optimization is encountered in many applications such as formation flying and rendezvous and docking in the aerospace industry, path planning in robotics, sailing/shipping route optimization in the maritime industry, and ideal trajectory construction in the racing industry. In most applications the optimization is non-linear with possibly multiple local-minimums and the desire for methods which guarantee that the global optimal solution is found grows. A general problem description within which trajectory optimization fits is given in Pinter:[1]

$$\begin{aligned} &\min f(x) \\ &x \in D := \{x : l \leq x \leq u,\ \ g(x) \leq 0\} \end{aligned} \tag{1}$$

where the optimal trajectory of the state x has to be found which minimizes the (cost) function f while satisfying all constraints g. The state is constrained by a lower bound l and a upper bound u which together define the domain D within which the solution must be sought. Throughout the years, many researchers have developed method for solving the trajectory optimization problems. Betts[2] classified two methods translating the initial problem of trajectory optimization into a finite dimensional problem: non-linear programming and optimal control. He showed that the two methods can be considered equal so in this paper the non-linear programming approach is considered. Betts also made a review of (many of) the available numeric analysis based solvers. The key aspect for a solver to be able to find the optimal solution within finite time is the definition of the parameter adaptation. For instance, when applying direct shooting, one can apply a gradient based adaptation scheme for the parameters which determine the trajectory. As is well known, gradient based methods, e.g. the Levenberg-Marquardt method, can end up in a local minimum. Genetic algorithms, and Monte-Carlo optimization schemes can also not provide a guarantee of finding the optimal solution within finite time. Methods which do have this guarantee are Lipschitz based algorithms and interval analysis.[3–5]

*PhD student, student AIAA member
†Assistant professor, AIAA member
‡Associate professor, AIAA member
§Professor, AIAA member

Lipschitz optimization algorithms are applied to cases where the cost function $J : D \to \mathbb{R}, D \subseteq \mathbb{R}^n$ is a Lipschitz function such that:

$$|J(\mathbf{x}) - J(\mathbf{y})| \leq L \|\mathbf{x} - \mathbf{y}\|, \ \forall \mathbf{x}, \mathbf{y} \in D \tag{2}$$

for a specific Lipschitz constant $L \in \mathbb{R}$. The previous relation can be used to discard regions of the domain D. When (in)equality constraints are present, a method must be available to guarantee that they are not violated over the entire trajectory. It may be difficult to derive Lipschitz functions for the entire trajectory, e.g. consider the process of guaranteed integration. Interval analysis uses interval arithmetic[3,6] instead of standard, crisp valued, arithmetic to perform the optimization. Interval arithmetic can be used to derive Lipschitz constants. The principles of optimization are similar as both methods converge to the global optimum by discarding infeasible regions of the search space. However, interval analysis can handle a much larger class of optimization problems containing discontinuities, numeric integrations and sub-routines. Finding bounds on the cost function and propagation of (in)equality constraints comes naturally from the use of interval arithmetic which makes interval analysis preferable. Examples of applications of interval analysis to aerospace related problems are: finding trim-points for nonlinear aircraft models,[7] pilot model identification,[8] integer ambiguity resolution for aircraft attitude determination,[9,10] spacecraft re-entry optimization[11] and fuel optimization for constrained spacecraft formation rotations.[12]

Interval analysis has been applied to trajectory optimization by several researchers: Chu et al.,[13] Filipe et al.[11] The approach taken in the previous applications was to parameterize the control and perform subsequent guaranteed integration in a direct shooting context. Although successful for small scale problems, the inherent problems of interval analysis limit the applicability to larger scale problems. Specifically the dependency effect encountered in the integration process, called the temporal de-correlation effect, causes a blowup of the trajectories.[11] The key to successfully applying interval analysis is to use a problem framework which does not include explicit integration. State parameterization and direct collocation[14,15] are two methods of transforming the original problem into a framework having implicit integration which is classified by Betts[2] as a direct transcription method.

State parameterization methods represent the trajectory by a parameterized function approximation, such as a polynomial or spline. Next, the corresponding control action that is required to follow the trajectory is derived from the equations of motion. By varying the coefficients of the function approximator, the cost function, that usually depends on both state and control, can be minimized.

Direct collocation transforms the trajectory optimization problem into a non-linear programming (NLP) problem, which does not include any explicit integration steps, by replacing the ODEs with a set of defect constraints. The constructed NLP problem is typically solved by available solver based on gradient methods[16] and genetic algorithms. Direct transcription methods has been successfully applied in many fields such as spacecraft trajectory optimization,[17,18] aircraft trajectory optimization[19,20] etc. As stated, in most cases the applied methods cannot provide guarantees on finding the global optimal solution (region of convergence may be very small) and moreover can have severe problems of finding any valid trajectories given the (in)equality constraints imposed on the problem. Interval analysis is a covering method thus having the guarantee of finding the optimal solution. Moreover, any constraint can be easily applied and typically decreases the computational load since they can be used to reject parts of the solution space. When combining interval analysis with direct collocation the aspect of polynomial representation becomes very important. In this paper Bezier polynomials are selected to reduce the dependency effects to a minimum. The final solution is found using a branch and bound algorithm such as the Moore-Skelboe algorithm or Hansens' algorithm.[5]

This paper is organized as follows. Section II gives a brief introduction and description of interval arithmetic. Thereafter, previous attempts of using interval analysis for trajectory optimization are analyzed (section III). Specifically the aspects which limit the application to small scale problems are discussed. In section IV the state parameterization method is discussed, followed by an application of this method to a formation flying optimization problem in section V. The method of direct collocation is presented briefly in section VI and the optimal form of direct collocation is given such that it is most suited to be applied in combination with interval analysis. Finally, in Section VII conclusions are drawn and future research directions are indicated.

II. Interval analysis

Interval analysis is the theory dealing with interval numbers and the arithmetic operations on them[21,22]. 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 either the infimum and supremum are given $[a,b]$ or a single variable $[x]=[a,b]$. Interval arithmetic contains the same operands as ordinary arithmetic such as the basic computational operations of addition, subtraction, multiplication and division:

$$[a,b]+[c,d]=[a+c,b+d] \tag{3}$$
$$[a,b]-[c,d]=[a-d,b-c] \tag{4}$$
$$[a,b]\cdot[c,d]=[\min(ac,ad,bc,bd),\max(ac,ad,bc,bd)] \tag{5}$$
$$\frac{[a,b]}{[c,d]}=[a,b]\cdot[1/d,1/c] \qquad if 0 \notin [c,d] \tag{6}$$

The core of interval analysis is to use interval arithmetic to form an inclusion function $[f([\mathbf{x}])]$ of any function $f(\mathbf{x})$. This property of interval arithmetic follows from the inclusion function theorem given by R.E. Moore[21,23]:

Theorem II.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

$$[x_1]' \subset [x_1],[x_2]' \subset [x_2],...,[x_n]' \subset [x_n] \tag{7}$$

implies

$$\left[f\left([x]_1',[x]_2',...,[x]_n'\right)\right] \subset [f([x_1],[x_2],...,[x_n])] \tag{8}$$

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 [23]. □

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:

$$f(x_1,x_2,...,x_n) \subset [f([x_1],[x_2],...,[x_n])] \tag{9}$$

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, interval arithmetic can be used to produce guaranteed bounds on the crisp function output $f(\mathbf{x})\,\forall \mathbf{x} \in [\mathbf{x}]$.
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. $\sup[f([x])] = \max_{x\in[x]} f(x)$ and $\inf[f([x])] = \min_{x\in[x]} f(x)$. 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. $\sup[f([\mathbf{x}])] > \max_{\mathbf{x}} f(\mathbf{x})$ and $\inf[f([\mathbf{x}])] < \min_{\mathbf{x}} f(\mathbf{x})\ \forall \mathbf{x} \in [\mathbf{x}]$.

The inclusion theorem is the key to using interval arithmetic as a global non-linear solver. For any given non-linear optimization problem one can use the inclusion function of the cost function to search for the global optimum. The process of converging to the global optimum is a process of elimination: via application of a branch and bound algorithm. The process is graphically represented in Figure 1. Given a search domain $[x]$ one can split it into n parts ($[x_i], i=1,2,...,n$ - green surfaces). For each part, the inclusion function of the cost function $[J]$ is computed using interval arithmetic. A guaranteed estimate of the lowest cost function value ρ can now be made: $\rho = \min_i \sup([J([x_i])])$. Since the value of ρ is guaranteed one can automatically

discard all domains $[x_i]$ for which the infimum (the lower interval bound) of the cost function is higher than ρ. These are the hatched areas in figure 1 and it is guaranteed that the global minimum does not lie in those domains. The solution space is formed by the remaining domains and the process starts again by splitting the remaining boxes. To limit the computational load one must limit the number of boxes or reduce the search domain as fast as possible. This can be done by lowering the value of ρ as fast as possible such that more and more boxes can be discarded. Many box selection methods and advanced inclusion function theorem are available in literature[3, 5, 24–26] and will not be discussed here.

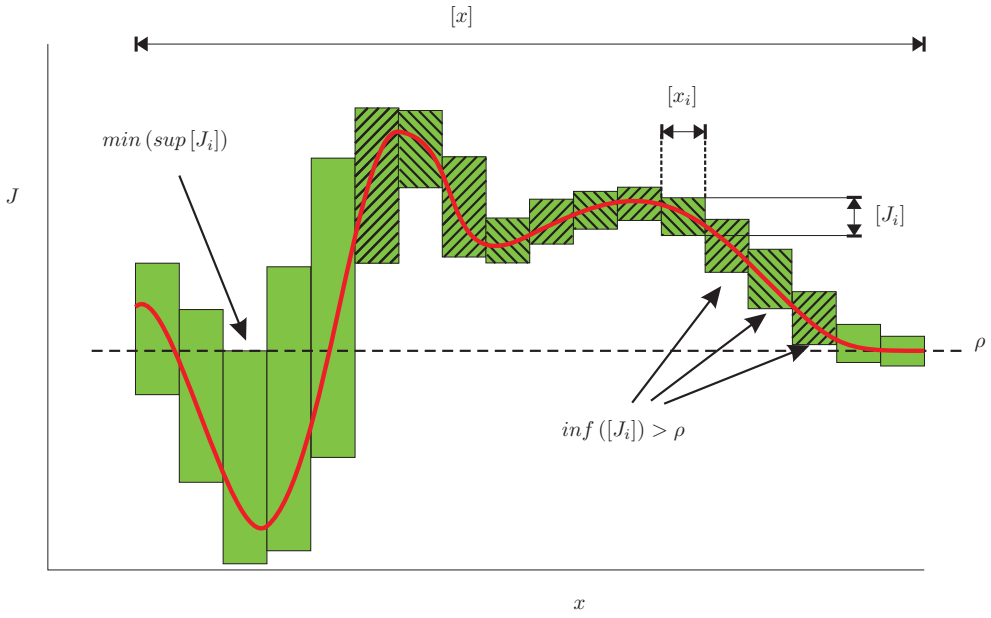


Figure 1. Principles of the interval branch and bound algorithm (hatched boxes can be eliminated based on the minimum cost function value estimate ρ)

The very basic description of the branch and bound principle given here is used to highlight one very important aspect: satisfying the constraints. If constraints are imposed on the problem, then the estimate of the minimum cost function value, ρ, may only be lowered if all constraints are valid for at least one (crisp) point within the box:

$$\rho = \min\left(\rho, \sup\left[J_i\left([x_i]\right)\right]\right) \quad \Longleftrightarrow \quad g\left(x\right) \leq 0,\ x \in [x_i] \tag{10}$$

Next to trying to find a single point within $[x_i]$ which satisfies the constraints one can also prove the existence of such a point by applying theorem 10.12.1 in Hansen.[24] Despite the availability of an existence theorem it can be very difficult to proof that there is a point which satisfies the constraints. Not being able to lower ρ means that less boxes can be discarded and that the computational load increases, often dramatically. Constraint handling in case of direct collocation will be further discussed in section VI.

III. Control parameterization

To redefine the infinite dimensional problem into a finite dimensional, both Chu[13] and Filipe[11] applied control parameterization. Besides forming a finite dimensional problem control parameterization ensures that the found trajectory is in accordance with the Equations Of Motion (EOM), i.e. the found trajectory can actually be followed. Moreover, any constraints on the control variables are easy to apply and can often be eliminated by choosing the correct parameterization. Another advantages of applying control parameterization is that *if* a set of trajectories is found to satisfy the constraints, then the entire set of trajectories can be marked as feasible. This allows one to solve problem in which one does not require one optimal trajectory but one want to determine the entire set of feasible trajectories, such as the problem given in Filipe,[11] where the goal is to find the set of initial states from which a final point can be reached, given

an interval of control actions. In later sections, direct collocation is applied which introduces approximate solutions to the EOM for a given control trajectory. This makes the problem solvable but will limit the application to problems for which a crisp set of trajectories needs to be found, i.e. trajectories leading to the minimal cost function value. One should therefore keep in mind that the control parameterization method based on interval analysis does have unique and useful properties.

However, the major drawback of control parameterization is the integration of the equations of motion that is required to compute the trajectory. It is this problem that makes the large dimensional problems unsolvable. Besides the wrapping effect, the integration inherently causes another effect called the temporal decorrelation effect. The temporal decorrelation effect causes the integration blow-up as is observed in previous works[11] and will be demonstrated using the following simple example.

EXAMPLE TEMPORAL DECORRELATION: INTEGRATION BLOW-UP

The task at hand is to find all feasible trajectories for the system

$$\dot{x} = u, \quad u(a,t) = at - a \tag{11}$$

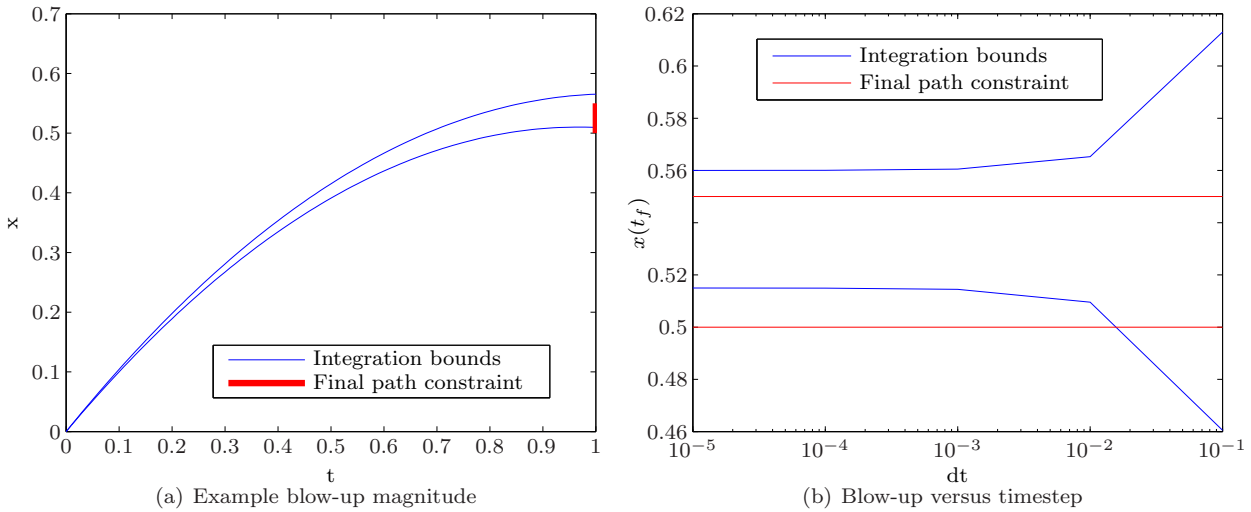
with the constraints given by:

$$\begin{aligned} x(t_0) &= 0 \\ x(t_f) &= [0.5, 0.55] \\ |u(t)| &\leq 1 \\ t_0 &= 0 \\ t_f &= 1 \end{aligned} \tag{12}$$

The solution to this problem is given by

$$\begin{aligned} & \dot{x} && = -a + at \\ \rightarrow \quad & x(t) && = x(0) - a(t - t_0) + \tfrac{1}{2} a (t - t_0)^2 \\ \rightarrow \quad & a && = \tfrac{x(t) - x(0)}{-(t-t_0) + 0.5(t-t_0)^2} \\ & && = [-0.55, -0.5] * 2 \end{aligned} \tag{13}$$

Suppose at a given moment of time in the optimization process the box $a = [-0.545, -0.530] * 2$ is processed. Since this box lies within the true solution ($a = [-0.55, -0.5] * 2$), this box is a valid solution to the problem. However, nothing can be concluded when looking at the computed interval of the final state: $x(t_f) = [0.5095, 0.5654] \notin [0.5, 0.55]$ (see Figure 2(a)). The integration is performed using $\Delta t = 0.01s$ for which the wrapping effect is present although limited (see Figure 2(b)). □



(a) Example blow-up magnitude

(b) Blow-up versus timestep

Figure 2. Integration blow-up example - demonstration of blow-up and wrapping effect

The reason for the integration blow-up is the temporal decorrelation effect: for a given control parameterization there are infinite realizations of the control trajectory within the given bounds. From interval analysis (using standard integration) point of view, all control trajectories given in Figure 3 for parameterization $u = a \; \forall \; t, a = [-1, 1]$ are valid. Stated mathematically, the control parameterization with interval coefficients defines the set U within which infinite realizations of trajectories are possible:

$$U = [\underline{\mathbf{u}}(a,t), \bar{\mathbf{u}}(a,t)] \, \forall t \tag{14}$$

Given this class which defines bounds for each time instant t a valid trajectory realization is any trajectory which fulfills:

$$\tilde{\mathbf{u}}(t) \subseteq U \; \forall t \tag{15}$$

The interval analysis inclusion theorem states that the computed output of the integration must include the output of all possible control trajectory realizations within U:

$$\int_{t_0}^{t_f} [f(\mathbf{x}(t), \tilde{\mathbf{u}}(t))] \, dt \quad \subseteq \int_{t_0}^{t_f} [f(\mathbf{x}(t), U)] \, dt = [\mathbf{x}(t_f)] \tag{16}$$

Therefore, when performing numerical integration, there is no enforcement of the parameterization (no correlation between time steps, hence the name temporal decorrelation) which leads to integration blow-up if the final state is a non-monotonic function of the control parameterization coefficients. In other words, at each time step in the integration the worst case control action can be selected (within U) without any constraints between the control actions of different time steps. It is the temporal decorrelation effect combined with the dependency and wrapping effect which makes larger problems unsolvable as is observed in the work of Chu and Filipe.[11,13]

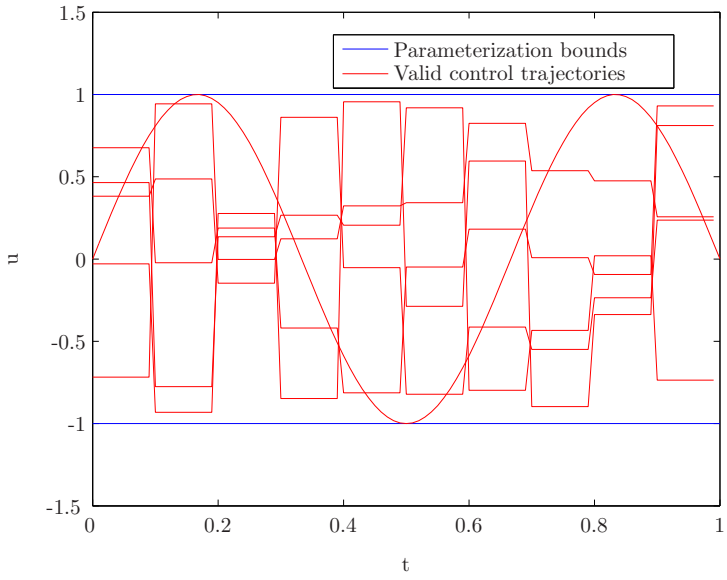


Figure 3. Integration blow-up example: valid control trajectories within a given parameterization ($u = [-1, 1] \forall t$)

Avoiding the temporal decorrelation effect can be realized using techniques which prevent the numerical integration with interval parameters. The most obvious method is that of algebraic integration which however is not possible for most realistic trajectory optimization problems. In Riehl et al.[19] a overview of possible implicit integration methods are given (including Pseudo-Spectral integration). The methods transform the problem of numerical integration by replacing the ODEs with a set of defect constraints. The distinction between the methods is the manner in which the states are represented (polynomial type, polynomial order, number of nodes) and the manner in which the defects are computed (collocation defect, quadrature defect, defect through explicit integration). The next section will discuss a state parameterization method that can also be used to avoid the temporal decorrelation effect.

IV. State parameterization

This section discusses the approach of state parameterization in the field of trajectory optimization. State parameterization is one method to transform the infinite dimensional, dynamic optimization problem into a finite dimensional optimization problem. The transformed problem is solved using an interval based solver, which is guaranteed to find the globally optimal solution within the initial search space.

In its general form, the trajectory optimization problem can be written as:

$$\begin{array}{ll} J\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right),t\right) & = \Phi\left(\mathbf{x}\left(t_0\right),t_0,\mathbf{x}\left(t_f\right),t_f\right) + \\ & \int_{t_0}^{t_f} L\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right),t\right)dt \\ \mathbf{p}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right)\right) & \leq 0 \\ \mathbf{q}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right)\right) & = 0 \\ \frac{\partial \mathbf{x}(t)}{\partial t} & = \mathbf{f}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right),t\right) \end{array} \tag{17}$$

where $\mathbf{x}$ is the state vector and $\mathbf{u}$ is the control vector. Equality constraints ($\mathbf{q}$) and inequality constraints ($\mathbf{p}$) can be imposed on the problem. This includes (control and states) path constraints and endpoint constraints ($\Phi = 0$). Dynamic constraints in the form of $\mathbf{f}$ are assumed to always be present. Finally, J is the cost functional (or cost function), which is defined by the endpoint cost Φ and the integral of the Lagrangian $\mathcal{L}$ over time domain $t \in [t_0, t_f]$. Both time values t_0, t_f may be unknown, but in most problems the begin time is known or only the length of the time interval $\tau = t_f - t_0$ is of importance.

When implementing state parameterization, one substitutes all (independent) states with a parameterized model describing the path of each state with respect to time ($\mathbf{x}(t) \rightarrow \mathbf{x}(\mathbf{c}, t)$, where $\mathbf{c}$ is the vector containing the adaptable parameters):

$$\begin{array}{ll} J\left(\mathbf{x}\left(\mathbf{c},t\right),\mathbf{u}\left(t\right),t\right) & = \Phi\left(\mathbf{x}\left(\mathbf{c},t_0\right),t_0,\mathbf{x}\left(\mathbf{c},t_f\right),t_f\right) + \\ & \int_{t_0}^{t_f} L\left(\mathbf{x}\left(\mathbf{c},t\right),\mathbf{u}\left(t\right),t\right)dt \\ \mathbf{p}\left(\mathbf{x}\left(\mathbf{c},t\right),\mathbf{u}\left(t\right)\right) & \leq 0 \\ \mathbf{q}\left(\mathbf{x}\left(\mathbf{c},t\right),\mathbf{u}\left(t\right)\right) & = 0 \\ \frac{\partial \mathbf{x}(\mathbf{c},t)}{\partial t} & = \mathbf{f}\left(\mathbf{x}\left(\mathbf{c},t\right),\mathbf{u}\left(t\right),t\right) \end{array} \tag{18}$$

The control paths are deduced from the dynamic constraints:

$$\mathbf{u}\left(t\right) = \mathbf{h}\left(\frac{\partial \mathbf{x}\left(\mathbf{c},t\right)}{\partial t}, \mathbf{f}\left(\mathbf{x}\left(\mathbf{c},t\right),\mathbf{u}\left(t\right),t\right)\right) \tag{19}$$

For all EOM that can be rewritten into an expression for the control as a function of state and state derivatives:

$$\mathbf{u}\left(t\right) = \mathbf{h}\left(\frac{\partial \mathbf{x}\left(\mathbf{c},t\right)}{\partial t}, \mathbf{x}\left(\mathbf{c},t\right),t\right) \tag{20}$$

one can derive an analytic expression in terms of the optimization parameters. When the separation is not possible and the relation must be written as

$$\mathbf{u}\left(t\right) = \mathbf{h}_1\left(\frac{\partial \mathbf{x}\left(\mathbf{c},t\right)}{\partial t}, \mathbf{x}\left(\mathbf{c},t\right),t\right) + \mathbf{h}_2\left(\mathbf{x}\left(\mathbf{c},t\right),\mathbf{u}\left(t\right),t\right) \tag{21}$$

one must apply a numeric solver to determine the control values. Interval based solvers specifically designed for inverse problems can be found in the book of Jaulin et al.,[27] e.g. the SIVIA optimization algorithm. When the control cannot be written as an analytic expression, the complexity of the problem is greatly enhanced. In this paper only problems for which the inverse problem can be analytically solved are treated.

When considering systems that are fully controllable, the use of state parameterization prevents explicit integration of the dynamic constraints (the EOM), significantly reducing the computational load. Moreover, the end-point constraints can be implicitly satisfied when the right parameterization type is selected. To demonstrate this, a simple example is given.

Consider the following optimization problem:

$$\begin{array}{lll} \min\limits_{u} J & = \int\limits_{t_0}^{t_f} u\left(t\right)^2 dt & \\ \ddot{x}\left(t\right) & = u\left(t\right) - 0.1\dot{x}\left(t\right) & \\ x_{t_0} & = 0 & \dot{x}_{t_0} = 0 \\ x_{t_f} & = 10 & \dot{x}_{t_f} = 0 \end{array} \tag{22}$$

Suppose that $x(t)$ is parameterized by a polynomial. To satisfy all end-point constraints, one must at least have a third order polynomial. If a third order polynomial is used all coefficients are fixed and the optimization problem is finished (no free variables left to optimize). As an example, for a single 3rd order simplex spline the coefficients are:

$$\left.\begin{array}{ll} x\left(\mathbf{c}, t_0\right) & = c_0 \\ \dot{x}\left(\mathbf{c}, t_0\right) & = 3\left(c_1 - c_0\right) \\ x\left(\mathbf{c}, t_f\right) & = c_3 \\ \dot{x}\left(\mathbf{c}, t_f\right) & = 3\left(c_3 - c_2\right) \end{array}\right\} \quad \begin{array}{ll} c_0 & = x_{t_0} \\ c_1 & = x_{t_0} + \frac{\dot{x}_{t_0}}{3} \\ c_2 & = x_{t_f} - \frac{\dot{x}_{t_f}}{3} \\ c_3 & = x_{t_f} \end{array} \tag{23}$$

The simple derivative expressions of the simplex spline and the property that the polynomial value at t_0 and t_f is fully defined by the first and last coefficient respectively, make the simplex spline the ideal representation for state parameterization. With the simplex spline definition one can directly set the end-point constraints independently of the polynomial order (for d > 3 the first two and the last two coefficients are directly set and fixed, leaving the other (d-3) coefficients free).

In general, considering a single domain $[t_0, t_f]$ and a d^{th} order polynomial, the simplex spline can be written as:[28]

$$p_d\left(\mathbf{c}, t\right) = \sum_{i=0}^{d} c_i \frac{d!}{(d-i)!i!}\left(1-\tilde{t}\right)^{d-i}\tilde{t}^i \tag{24}$$

where $\tilde{t}$ is the normalized time:

$$\tilde{t} = \frac{t - t_0}{t_f - t_0} \tag{25}$$

and c_i denote the polynomial coefficients. The advantages of using simplex splines for the state parameterization are:

- Direct relation between end-point constraints and coefficient values.
- Simple operator definitions (multiplication, addition, subtraction, derivative).
- Coefficient domains are equal in magnitude, unlike ordinary polynomials.

An important property is that the end-point conditions are implicitly satisfied and that each end-point condition is related to a single coefficient. This means that the optimization problem dimension is reduced by the number of end-point constraints.

In order to derive an analytic expression for the control trajectories in terms of the optimization parameters, one needs to be able to take the n^{th} order derivative of the state with respect to time. In case of the simplex splines the n^{th} order derivative can be computed by applying the following expression n times:

$$\frac{\partial p_d\left(\mathbf{c}, t\right)}{\partial t} = \sum_{i=0}^{\hat{d}} \hat{c}_i \frac{\hat{d}!}{(\hat{d}-i)!i!}\left(1-\tilde{t}\right)^{\hat{d}-i}\tilde{t}^i = p_{\hat{d}}\left(\hat{\mathbf{c}}, t\right) \tag{26}$$

where $\hat{d} = d - 1$ and $\hat{c}_i$ is given by $\hat{c}_i = \frac{d}{t_f - t_0}\left(c_{i+1} - c_i\right)$. One can see that the derivative of a spline is again a spline (of one order lower), making the evaluation of derivatives very easy.

V. Simulation results

The main aspects of state parameterization have been discussed in the previous section. When applying state parameterization using simplex splines to the problem of satellite formation flying, the following can be concluded:

- The control paths can be analytically derived using the EOM.
- The end-point constraints are implicitly satisfied thus do not need to be considered during the optimization.
- The application of the simplex splines makes the cost function convex with respect to the free coefficients in the simplex splines. However, the addition of nonlinear constraints will necessitate the use of a global solver.

A. Formation flying optimization problem

The motion of a satellite (follower i) relative to another satellite (leader) which is in a circular orbit about the body can, under the assumption of close proximity, be expressed by the following equations of motion:[29]

$$\begin{aligned} \ddot{x} - 2\omega\dot{z} &= \tfrac{Q_x}{m} \\ \ddot{y} + \omega^2 y &= \tfrac{Q_y}{m} \\ \ddot{z} + 2\omega\dot{x} - 3\omega^2 z &= \tfrac{Q_z}{m} \end{aligned} \tag{27}$$

commonly called the Hill-Clohessy-Wiltshire (HCW) equations, where x, y, z are the position coordinates of the follower relative to the leader, ω is the orbital angular rate of the leader, m is the mass of the follower, and Q is the specific force acting on the follower. The specific force is the summation of all acting forces on the spacecraft, such as thruster forces, third body forces, gravity field gradient forces, and other disturbance forces. For this example it is assumed that there are only thruster forces and that the follower has three thrusters along its axes (assumed to be parallel to the coordinate frame in which the equation of motion are expressed) $\mathbf{T} = [T_x, T_y, T_z]$.

The goal is to transport the follower from a begin state $\mathbf{x}(0)$, consisting of 3 positions and 3 velocities, to a final state $\mathbf{x}(t_f)$ where the final time t_f is given (see Table 1).

Table 1. Formation flying optimization problem settings.

parameter	value	unit
m	1	kg
t_0	0	s
t_f	10,000	s
N_t	101	-
ω_0	0.0011	rad/s
$\mathbf{x}_0$	$[10, 400, 10, 0, 0, 0]$	[m,m/s]
$\mathbf{x}_{t_f}$	$[0, -100, 0, 0, 0, 0]$	[m,m/s]

The cost function is designed to minimize the fuel consumption:

$$J = \int_{t_0}^{t_f} \mathbf{T}^T \mathbf{T} dt \tag{28}$$

Under the assumption that the control is smooth and its derivative with respect to time is small, the cost function can be written as:

$$J = \sum_{j=1}^{N_t} \mathbf{T}(t_j)^T \mathbf{T}(t_j) \Delta t \tag{29}$$

where N_t is the number of time instances and t_j is the j^{th} time instance. The time instances are uniformly distributed over the time domain $[t_0, t_f]$. The approximation of the integral simplifies the computation of the cost function considerably, while having a negligible influence on the solutions set.

The end state $\mathbf{x}(t_f)$ represents a position in a configuration of satellites (and zero velocity) of which an example is given in figure 4. The constraints imposed on the problem are the mentioned end-point constraints, the dynamic constraints imposed by the EOM, and possibly dynamic constraints representing the minimum inter-satellite distance (to avoid collisions):

$$p_{ij}\left(\mathbf{x}_i, \mathbf{x}_j\right) = r^2 - \left(\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 + \left(z_i - z_j\right)^2\right) \leq 0 \tag{30}$$

where i and j denote the two satellites under consideration and r is the minimum inter-satellite distance. As for the cost function, the constraints (if needed) are evaluated at specified instances in the time domain. The consequence of this approach is that the constraints, e.g. the inter-satellite distance, may be momentarily violated between two time instances. The minimal inter-satellite distance during that time will not be significantly lower than r, provided that the dynamics are slow varying and/or the time instances are close to each other.

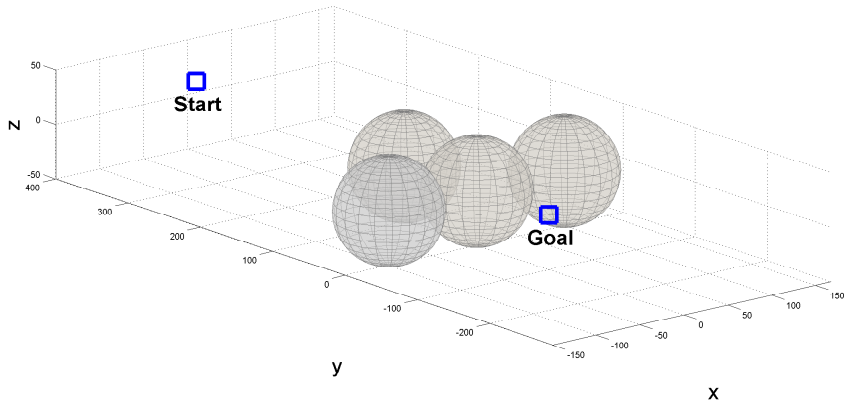


Figure 4. Example of formation flying satellites. The goal is to transport the fifth satellite to the open spot in the formation (Goal) without crossing any space defined by the spheres.

B. Optimization results

The formation flying optimization problem above is solved by using the interval trajectory optimization algorithm with state paramterization. The states x, y, z are parameterized by 5th order interval splines (see eq.24), resulting in 6 interval coefficients per state. The parameterized states are differentiated twice and entered into equation 27 to compute the control forces $\mathbf{T} = [T_x, T_y, T_z]$. Next the cost function is computed and it is checked of the constraints have been met. In an interval branch-and-bound loop(see Section II), the cost function is minimized, resulting in the optimal set of coefficients.

Table 2 gives the minimum cost function value that is found by the interval optimization algorithm, together with the corresponding spline coefficients. Also in the table are the three best results J_1, J_2, J_3 from 100 gradient based optimization attempts. The corresponding trajectories are visualized in figure 5. It can be observed that the interval solution has a lower cost function value and is therefore a better solution.

Another important observation is that the gradient based optimization leads to different solutions, depending on the initial value of the spline coefficients. This does not happen with the interval method, since the initial values for the spline coefficients cover the whole search area and thus do not have to be changed between trials.

Table 2. Resulting cost function value and spline coefficients for the formation flying optimization problem with states parameterized by 5th order splines. The interval result J_{IA} is compared with the three best results J_1, J_2, J_3 from 100 gradient based optimization attempts. The corresponding trajectories are visualized in figure 5.

Method	J	**c**
Interval analysis (J_{IA})	1.358E-6	$\{535.156, 492.032, -1622.656,$ $1142.253, 293.573, -329.365\}$
Gradient based (J_1)	1.731E-6	$\{-600.669, -355.159, -1389.200,$ $933.505, -289.394, 305.299\}$
Gradient based (J_2)	2.148E-6	$\{547.592, 361.230, -983.228,$ $434.092, 214.610, -239.320\}$
Gradient based (J_3)	2.197E-6	$\{556.146, 508.422, -945.581,$ $489.875, 279.622, -301.878\}$

VI. Direct collocation and interval analysis

Apart from state parameterization, direct collocation can also be used to prevent the explicit interval integration that is the main drawback of control parameterization methods. Some initial findings into the combination of interval analysis and direct collocation will presented in this section.

Direct collocation is a method for transforming the original dynamic optimization problem (which includes explicit integration) into a NLP problem. The basic principle is that the time domain is split into several phases $[t_i, t_{i+1}]$ and that the state x and control values u are set for the begin and end of each phase. Using the state and control values and the EOM (f) one can determine the state derivative values $\dot{x}(t_i), \dot{x}(t_{i+1})$. The state value and the state derivative values at two time instances provide enough conditions to define a cubic polynomial $x^i(c,t)$ per time phase i. Once the polynomial is determined it can be used to compute the 'defect' Δ_i:

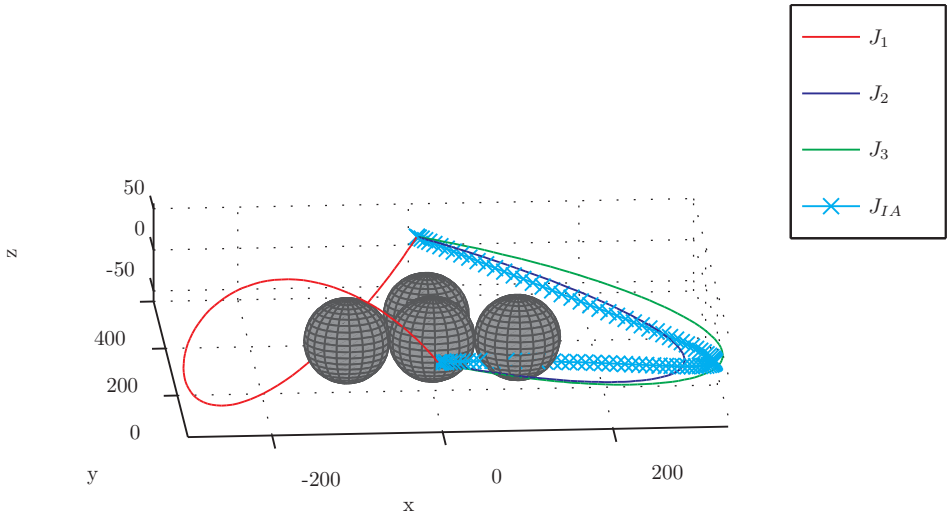
$$\Delta_i = \left.\frac{dx\,(c,t)}{dt}\right|_{t=t_{\Delta,i}} - f\left(x\left(c, t_{\Delta,i}\right), u\left(t_{\Delta,i}\right)\right) \tag{31}$$

where $t_{\Delta,i}$ denotes the time where the defect is computed. The key aspect of direct collocation is the assumption that the polynomial represents the solution to the EOM (for the given control) accurately if the defects are (close to) zero. It must be stressed that direct collocation yields an *approximate* trajectory which *may* not fully comply with the solution found via explicit integration using the same control profile. This is the main difference between using the control parameterization method given in section III (explicit integration) and direct collocation. It also means that the found trajectory is not guaranteed to be the optimal solution of the original problem. However, as explained in section III, implicit integration is required to make the problem solvable. The original problem, equation 1, is transformed into the new problem definition using direct collocation:

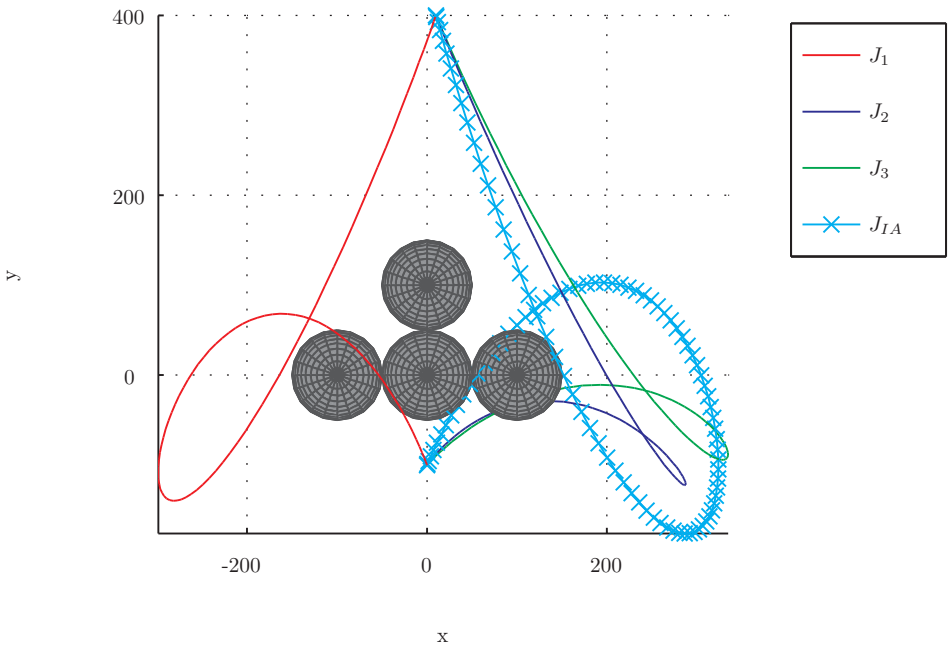
$$\begin{aligned} &\min_c J\left(x\left(c,t\right)\right) \\ &x \in D := \{x : l \le x \le u,\ \ g\left(x\right) \le 0\} \end{aligned} \tag{32}$$

where the state trajectories are determine by the adaptable coefficients c.

In this paper only the basic form of direct collocation is considered: state and control values are set at the nodes (also known as control points), third order polynomials represent the trajectories in between, and there is a linear relation for the control in each phase. The control is assumed to be continuous up to order zero, meaning that the state is continuous up to order 1 (i.e. state derivatives for neighboring phases match at the nodes). Examples of possible trajectory representations as a function of the number of time phases,



(a) 3D view



(b) x-y plane

Figure 5. State parameterization, constrained problem, spline order 5 - Three most optimal trajectories (J_1, J_2, J_3) found out of a 100 runs when applying the gradient based solver to the constrained problem and the solution found using interval analysis J_{IA}. For each run of the gradient based solver the free coefficients have been randomly initialized in the domain $[-1500, 1500]$

using the setting given above, are given in Figures 6 to 8. As one can see, the absolute magnitude of the defects (generally speaking) decreases with increasing number of phases. The goal of this section is to show that the proposed interval based solver always leads to the optimal solution *within the given framework* (problem, type of direct collocation, number of phases etc). The solver given in this paper is easily extended to other forms of direct collocation, such as those in the work of Hu et al.[30] In order to apply the interval analysis based solver to the NLP problem generated via direct collocation, a brief discussion on the parameterization type used in the scheme is required. From the interval analysis point of view, specific forms are more suited than others, and this is explained in the next section. Thereafter a discussion on handling the defects and defining the cost function is presented.

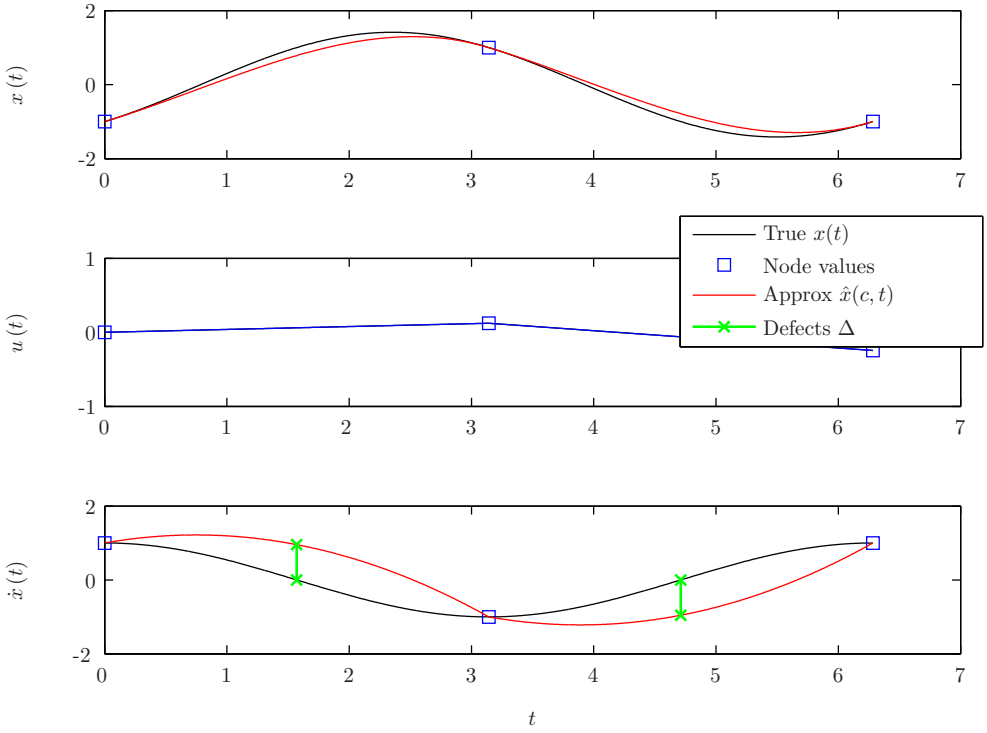


Figure 6. Priciples of direct collocation - 2 time phases

A. Parameterization type

When dealing with crisp numbers, the manner in which the polynomial per phase is represented is not crucial (they all represent the same line). However, when dealing with interval arithmetic, the representation does become very important as is illustrated with the following example.

Example polynomial representation

Without loss of generality, consider a single phase of the trajectory (time domain $t = [0, 1]$) for which the state and state derivative values (computed using the EOM and the state/control values) at the nodes are given as intervals (blue lines Figure 9):

$$\begin{pmatrix} [x(0)] \\ [x(1)] \\ [\dot{x}(0)] \\ [\dot{x}(1)] \end{pmatrix} = \begin{pmatrix} [-0.1, 0.1] \\ [0.5, 0.65] \\ [0.1, 0.15] \\ [0, 0.05] \end{pmatrix} \tag{33}$$

By applying interval analysis an inclusion function of the state derivative is determined and the result is again an interval: $[\dot{x}] = [f([x], [u])]$. Following the procedure of direct collocation for a 3^{rd} order polynomial

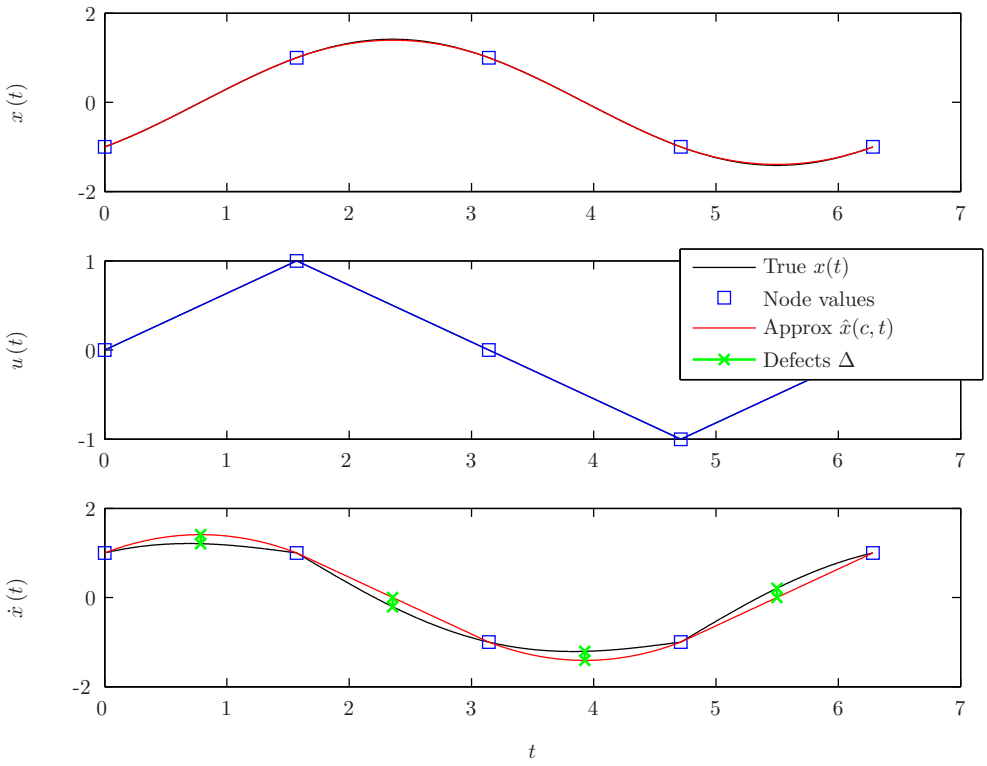


Figure 7. Priciples of direct collocation - 4 time phases

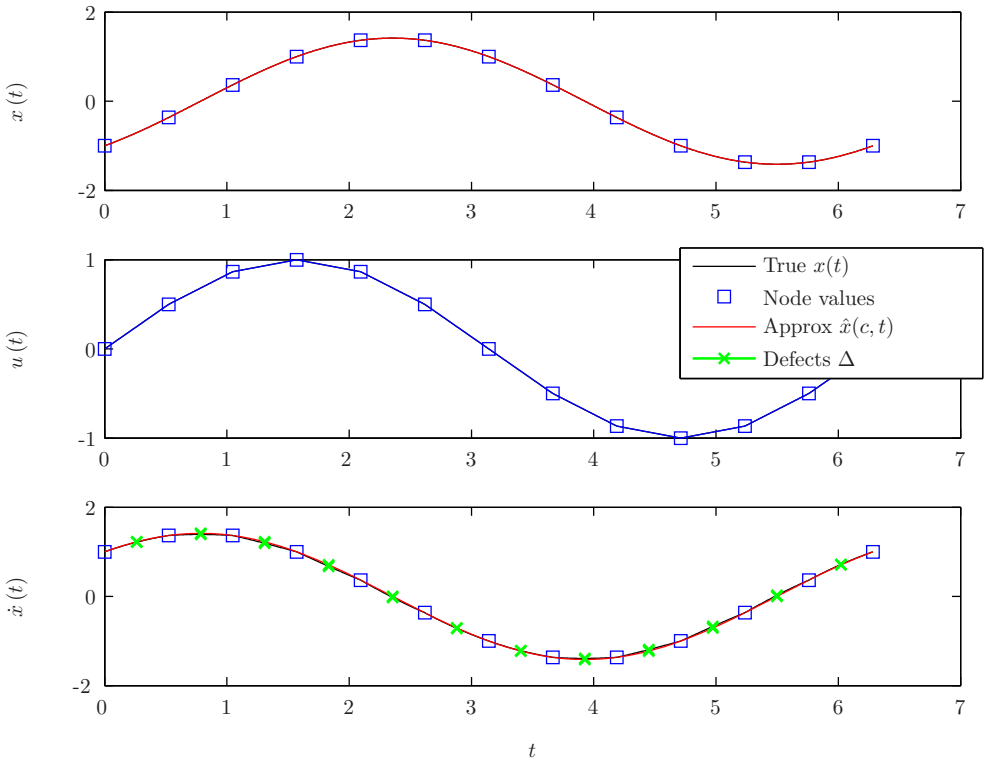


Figure 8. Priciples of direct collocation - 12 time phases

in power form one would obtain the following coefficient values:

$$y^p = \sum_{i=0}^{d} c_i t^i$$
$$\begin{pmatrix} [c_0] \\ [c_1] \\ [c_2] \\ [c_3] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{pmatrix}^{-1} \begin{pmatrix} [x\,(0)] \\ [x\,(1)] \\ [\dot{x}\,(0)] \\ [\dot{x}\,(1)] \end{pmatrix} = \begin{pmatrix} [-0.1, 0.1] \\ [0.1, 0.15] \\ [0.85, 2.05] \\ [-1.4, -0.6] \end{pmatrix} \tag{34}$$

Evaluation of the polynomial with interval coefficients will result in the set of trajectories represented in Figure 9. One can see that the polynomial output for $t = 1$ is not equal to the given state value. Moreover, when performing taking the time derivative of the polynomial (Figure 9, bottom plot) one can see that the $\dot{x}$ for $t = 0$ is exactly equal to the given value ($[c_1] = [\dot{x}(0)]$) but for $t = 1$ a huge difference is present. Should one have used the Bezier representation of the same polynomial:

$$y^b = \sum_{i=0}^{d} c_i \begin{pmatrix} d \\ i \end{pmatrix} t^i \,(1-t)^{d-i},\ t \in [0, 1] \tag{35}$$

then the coefficient values are given by:

$$\begin{pmatrix} [c_0] \\ [c_1] \\ [c_2] \\ [c_3] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{pmatrix}^{-1} \begin{pmatrix} [x\,(0)] \\ [x\,(1)] \\ [\dot{x}\,(0)] \\ [\dot{x}\,(1)] \end{pmatrix} = \begin{pmatrix} [-0.1, 0.1] \\ [-0.067, 0.15] \\ [0.483, 0.65] \\ [0.5, 0.65] \end{pmatrix} \tag{36}$$

In Figure 9, the set of trajectories is given for the second representation. It can be observed that the obtained set of polynomials lies within the set corresponding to the power form polynomial. Moreover, the begin and end conditions $[x(0)], [x(1)]$ are exactly matched. Evaluating the differentiated polynomial however shows that both end points for $\dot{x}$ are over-estimated. The latter can be improved by noting that the derivative of a Bezier polynomial is again a Bezier polynomial for which the begin and end conditions are equal to the first and last coefficients. By setting the first and last coefficients for x to the intervals $[\dot{x}(0)], [\dot{x}(1)]$ *after* algebraic differentiation of the Bezier polynomial, one can obtain a sharp inclusion. Note that the second coefficient is determined using the Bezier polynomial for x, thus by all begin and end conditions. The result is given in Figure 10 together with the trajectories formed by all combinations of $\sup, \inf$ of the begin and end conditions (denoted as 'possible trajectory'). The trajectories are correctly bounded which follows from application of the inclusion function theorem II.1.

When applying interval analysis to direct collocation the polynomial representation which has the sharpest bound for both x and $\dot{x}$ is desired. As shown in the previous example the (adapted) Bezier polynomial representation generates very sharp bounds. Moreover, the domain of the coefficients within which the optimal values lie needs to be known to solve the NLP problem. For Bezier polynomials the coefficient are in the same order of magnitude while those of the power form are (generally) not. Therefore, the search domain for the Bezier polynomial coefficients is easier to define. In this paper the Bezier polynomial is selected for the reasons given above. Note that the polynomial order can easily be increased if more constraints are available.
In the following section the location and type of defects used in this paper is explained. Moreover, the definition of the cost function is discussed which in turn defines the output.

B. Handling defects and defining the cost function

In the example of the principle of direct collocation (Figure 6 to 8), there is one defect per phase (between nodes) and the defect was located in the middle of the phase ($t_\Delta, i = (t_i + t_{i+1})/2$). This definition of the defect is but one of many (see Riehl[19]). Since it is the most commonly used defect definition in literature, it will also be used in this paper. Besides the definition of the defect is it important for which time stamps

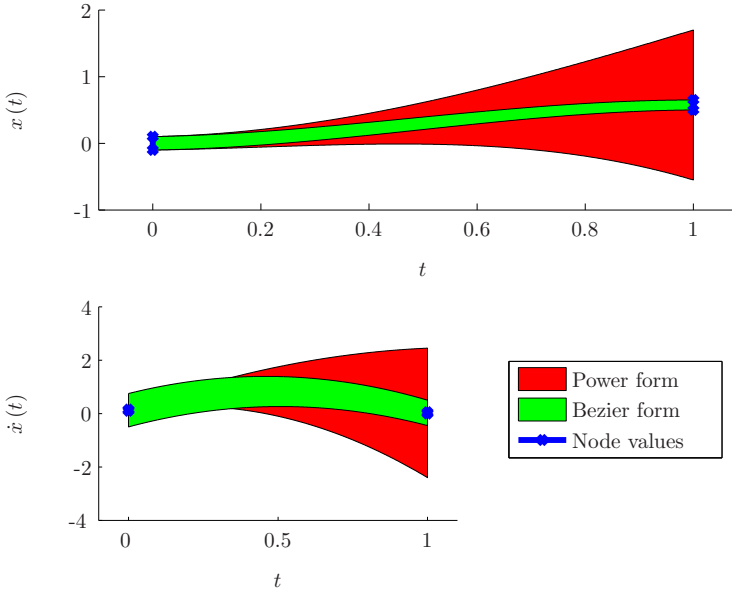


Figure 9. Example polynomial representation: power form versus Bezier form

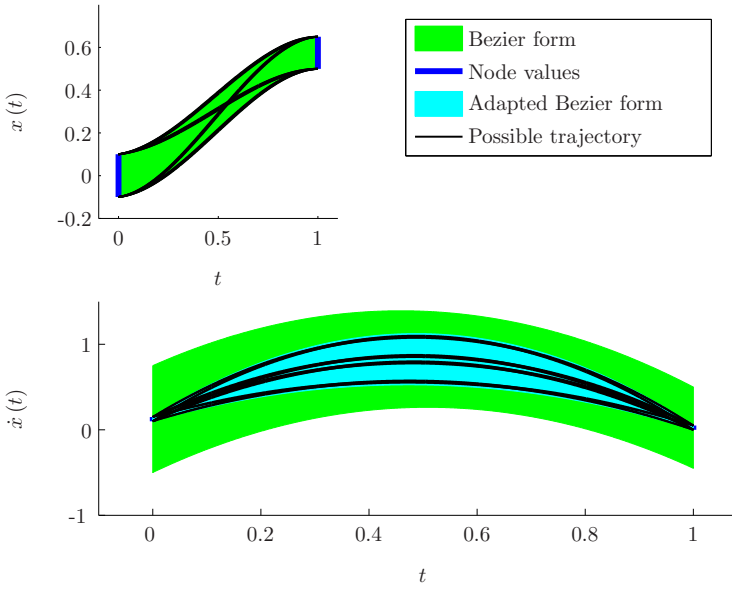


Figure 10. Example polynomial representation: adapted Bezier form

the defect is computed and how it is incorporated in the NLP problem.

The solution to the direct collocation problem should be a trajectory for which the defects are close to zero while the objective, e.g. minimal fuel expenditure, is met. To determine the solution one can use the defects either as inequality constraints, i.e. $\|\Delta_i\| < \epsilon$, or one can add a weighted term based on the defects in the cost function:

$$J = J_0 + \gamma \|\Delta_i\|^2 \tag{37}$$

where J_0 denotes the part of cost function representing the objective of the (original) trajectory optimization problem. Using the defect as inequality constraint will guarantee that the defect will be small and therefore that the found trajectory is a valid one (from direct collocation point of view). Moreover, if no valid trajectory is found, one automatically knows that the number of phases is too low. The latter is a desirable effect since the solution which is sought should be a feasible trajectory. However, looking from the interval analysis perspective, specifically considering the branch and bound algorithm, using the defects as constraints poses a problem. As discussed in section II, the estimate of the cost function value ρ can only be lowered if all constraints are satisfied for at least one point which, if ϵ is small, may be very difficult to proof.

To prevent the use of the defect as constraints, one can add a penalty term to the cost function as described earlier. When doing so any combination of crisp values in the parameter intervals will yield a valid trajectory in the sense that no (defect based) constraints are violated. This eases the implementation of the interval based solver considerably since the computational load is reduced.

VII. Conclusions and recommendations

A new interval based trajectory optimization method has been introduced that is based on state parameterization. Like previous interval trajectory optimization algorithms, this method provides the guarantee that the globally optimal solution is found, but it avoids the problems related to explicit interval integration that occur when a control parameterization approach is taken. The states are parameterized by splines, because with splines the end point constraints are easily translated into coefficient values.

Application of the new algorithm to a formation flying trajectory optimization problem with nonlinear constraints shows that the interval method can find better trajectories than conventional gradient based solvers, which get stuck in local minimums.

Finally, some initial steps into another type of interval trajectory optimization algorithm, that is based on the principle of direct collocation, have been presented. By using direct collocation the problem is transformed into a static optimization problem which is solved using interval analysis. Although not yet implemented, it is expected that numerical results will show that the new approach will also find the global optimal solution. Compared to existing interval based solvers applying the concept of control parameterization, it is believed that the new solver has lower computational load and is able to solve larger scale trajectory optimization problems.

References

[1]Pinter, J., *Global Optimization in Practice: State of the Arts and Perspectives*, chap. 11, Springer Science - Business Media, LCC 2009, 2009, pp. 377 – 404.

[2]Betts, J., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, 1998, pp. 193 – 207.

[3]R.E.Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, NJ., 1966.

[4]Hansen, E. and Walster, G., *Global Optimization Using Interval Analysis*, Marcel Dekker, Inc. and Sun Microsystems, Inc., 2004, ISBN 0-8247-4059-9.

[5]Jaulin, L., Kieffer, M., Didrit, O., and Walter, E., *Applied Interval Analysis*, Springer-Verlag London Berlin Heidelberg, 2001, ISBN 1-85233-219-0.

[6]Hansen, E., *A Generalized Interval Arithmetic*, Vol. 29, Springer Berlin/Heidelberg, 1975.

[7]van Kampen, E., Chu, Q., Mulder, J., and van Emden, M., "Nonlinear Aircraft Trim Using Interval Analysis," *Proceeding of the AIAA Guidance, Navigation, and Control Conference andExhibit*, August 2007.

[8]van Kampen, E., Zaal, P., de Weerdt, E., Chu, Q., and Mulder, J., "Optimization of Human Perception Modeling Using Interval Analysis," *Jounal of Guidance, Control, and Dynamics*, Vol. 33, 2010, pp. 42–52.

[9]van Kampen, E., de Weerdt, E., Chu, Q., and Mulder, J., "Aircraft Attitude Determination Using GPS and an Interval Integer Ambiguity Resolution Algorithm," *AIAA Guidance, Navigation, and Control Conference, Chicago, Illinois, Aug. 10-13, 2009*, 2009.

[10]de Weerdt, E., van Kampen, E., Chu, Q., and Mulder, J., "Integer Ambiguity Resoluition Using Interval Analysis," accepted for publication in the winter 2008 edition, ION Journal of Navigation.

[11]Filipe, N., de Weerdt, E., van Kampen, E., Chu, Q., and Mulder, J., "Terminal Area Energy Management Trajectory Optimization Using Interval Analysis," *Proceedings of the AIAA Guidance, Navigation, and Control Conference 2009*, , No. AIAA-2008-6212, Aug. 2009.

[12]de Weerdt, E., Chu, Q., and Mulder, J., "Neural Network Output Optimization Using Interval Analysis," *IEEE Transactions on Neural Networks*, Vol. 20, No. 4, 2009, pp. 638–653.

[13]Chu, W., Mooij, E., van Kampen, E., and Chu, Q., "A Feasibility Study to the Application of Interval Analysis to Re-entry Trajectory Optimization," *Proceedings of the AIAA Atmospheric Flight Mechanics Conference and Exhibit*, 2008, pp. 40–48.

[14]Dickmanns, E. and Wells, K., "Approximate Solutions of Optimal Control Problems Using Third Order Hermite Polynomial Functions," *Springer Lecture Notes, Com. Sci.*, Vol. 27, 1975, pp. 158 – 166.

[15]Hargraves, C. and Paris, S., "Direct Trajectory Optimization Using Nonlinear Programming and Collocation," *Journal of Guidance, Control, and Dynamics*, Vol. 10, 1987, pp. 338 – 342.

[16]staff, S. R., "SIGEST," *SIAM Rev.*, Vol. 47, No. 1, 2005, pp. 97–97.

[17]Huntington, G. and Rao, A., "Optimal Reconfiguration of Spacecraft Formations Using the Gauss Pseudospectral Method," *Jounal of Guidance, Control, and Dynamics*, Vol. 31, 2008, pp. 689 – 698.

[18]Enright, P. and Conway, B., "Optimal Finite-Thrust Spacecraft Trajectories Using Collocation and Nonlinear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 14, 1991, pp. 981 – 985.

[19]Riehl, J., Paris, S., and W.K.Sjauw, "Comparison of Implicit Integration Methods for Solving Aerospace Trajectory Optimization Problems," *AIAA/AAS Astrodynamics Specialist conference and Exhibit*, August 2006, pp. AIAAA 2006–6033.

[20]B.R.Geiger, Horn, J., Sinsleu, G., Ross, J., Long, L., and Niessner, A., "Flight Testing a Real-Time Direct Collocation Path Planner," *Journal of Guidance, Control, and Dynamics*, Vol. 6, 2008, pp. 1575 – 1586.

[21]Moore, R., *Interval Analysis*, Prentice-Hall, Inc., 1966.

[22]Hickey, T., Ju, Q., and van Emden, M., "Interval Arithmetic: from Principles to Implementation," *Journal of the ACM*, Vol. 48, No. 5, 2001, pp. 1038–1068.

[23]Moore, R., *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, PA, 1979.

[24]Hansen, E. and Walster, G., *Global Optimization Using Interval Analysis*, Marcel Dekker, Inc. and Sun Microsystems, Inc., 2nd ed., 2004.

[25]Makino, K. and Berz, M., "Taylor Models and Other Validated Functional Inclusion Functions," *International Journal of Pure and Applied Mathematics*, Vol. 4, No. 4, 2003, pp. 379–456.

[26]Corliss, G., editor, *Taylor Series Models in Deterministic Global Optimization*. Springer-Verlag, 2000.

[27]Jaulin, L., Kieffer, M., Didrit, O., and Walter, E., *Applied Interval Analysis*, Springer, 2001.

[28]Lai, M. J. and Schumaker, L. L., *Spline Functions on Triangulations*, Cambridge University Press, 2007.

[29]sković, J. B. and Mehra, R., "A multiple model-based reconfigurable flight control system design," *Proceedings of the 37th IEEE Conference on Decision and Control*, Vol. 4, December 1998, pp. 4503 – 4508.

[30]Hu, G., Ong, C., and Teo, C., "Direct Collocation and Nonlinear Programming for Optimal Control Problem Using an Enhanced Transcribing Scheme," *Proceedings of the 1999 IEEE International Symposium on Computer Aided Control System Design*, 1999.