AIAA/AAS Astrodynamics Specialist Conference and Exhibit
16 - 19 August 2004, Providence, Rhode Island

AIAA 2004-5405

Evolutionary Neurocontrol: A Smart Method for Global Optimization of Low-Thrust Trajectories

Bernd Dachwald*

German Aerospace Center (DLR), Cologne, Germany

Searching optimal low-thrust trajectories is usually a difficult and time-consuming task that involves much experience and expert knowledge. This is because the convergence behavior of traditional optimizers, which are based on numerical optimal control methods, depends on an adequate initial guess, which is often hard to find. Even if the optimizer converges to an optimal trajectory, this is typically close to the initial guess and rarely close to the (unknown) global optimum. Therefore, those methods are called local trajectory optimization methods. Within this paper, trajectory optimization problems are attacked from the perspective of artificial intelligence and machine learning. Inspired by natural archetypes, a smart global method for low-thrust trajectory optimization is proposed that fuses artificial neural networks and evolutionary algorithms into so-called evolutionary neurocontrollers. This novel method runs without an initial guess and does not require the attendance of an expert in astrodynamics and optimal control theory. This paper details how evolutionary neurocontrol works and how it could be implemented. Furthermore, the performance of the method is assessed for two exemplary interplanetary missions.

I. Introduction

This paper deals with the problem of searching optimal interplanetary trajectories for low-thrust spacecraft. Two propulsion systems are considered: solar sails (large ultra-lightweight reflecting surfaces that utilize solely the freely available solar radiation pressure for propulsion) and solar electric propulsion (SEP) systems. The optimality of a trajectory can be defined according to several objectives, like transfer time or propellant consumption. Because solar sails do not consume any propellant, their trajectories are typically optimized with respect to transfer time alone. Trajectory optimization for spacecraft with a SEP system is less straightforward because transfer time minimization and propellant minimization are mostly competing objectives, so that one objective can only be optimized at the cost of the other. Spacecraft trajectories can also be classified with respect to the terminal constraint. If, at arrival, the position $\mathbf{r}_{\mathrm{SC}}$ and the velocity $\dot{\mathbf{r}}_{\mathrm{SC}}$ of the spacecraft must match that of the target ($\mathbf{r}_{\mathrm{T}}$ and $\dot{\mathbf{r}}_{\mathrm{T}}$, respectively), one has a rendezvous problem. If only the position must match, one has a flyby problem. A spacecraft trajectory is obtained from the (numerical) integration of the spacecraft's equations of motion. Besides the inalterable external forces, the trajectory $\mathbf{x}_{\mathrm{SC}}[t] = (\mathbf{r}_{\mathrm{SC}}[t], \dot{\mathbf{r}}_{\mathrm{SC}}[t])$ is determined entirely by the variation of the thrust vector ('$[t]$' denotes the time history of the preceding variable). The thrust vector $\mathbf{F}(t)$ of low-thrust propulsion systems is a continuous function of time. It is manipulated through the n_u-dimensional spacecraft control function $\mathbf{u}(t)$ that is also a continuous function of time. Therefore, the trajectory optimization problem is to find the optimal spacecraft control function $\mathbf{u}^\star(t)$ that yields the optimal trajectory $\mathbf{x}^\star_{\mathrm{SC}}[t]$. This problem can not be solved except for very simple cases. What *can* be solved, at least numerically, however, is a discrete approximation of the problem. Dividing the allowed transfer time interval $[t_0, t_{f,\max}]$ into τ finite elements, the discrete trajectory optimization problem is to find the optimal spacecraft control history $\mathbf{u}^\star[\bar{t}] \in \mathbb{R}^{n_u \tau}$ that yields the optimal trajectory $\mathbf{x}^\star_{\mathrm{SC}}[t]$ (the symbol $\bar{t}$ denotes a discrete time step; note that only the spacecraft control function is discrete, whereas the trajectory is still continuous). Through discretization, the problem of finding the optimal function $\mathbf{u}^\star(t)$ in infinite-dimensional function space is reduced to the problem of finding the optimal control history $\mathbf{u}^\star[\bar{t}]$ in a $n_u\tau$-dimensional parameter space (a space which is usually still

*Research Engineer, Institute of Space Simulation, Linder Hoehe, 51147 Cologne, Germany, Phone: +49-2203-601 3001, Fax: +49-2203-601 2352, E-Mail: bernd.dachwald@dlr.de, AIAA Member, AAS Member

very high-dimensional). For optimality, some cost function J must be minimized. If the propellant mass m_{P} is to be minimized, $J = m_{\mathrm{P}}(\bar{t}_0) - m_{\mathrm{P}}(\bar{t}_f) = \Delta m_{\mathrm{P}}$ is an appropriate cost function, if the transfer time is to be minimized, $J = \bar{t}_f - \bar{t}_0 = T$ is an appropriate cost function.

II. Traditional Local Low-Thrust Trajectory Optimization Methods

Traditionally, low-thrust trajectories are optimized by the application of numerical optimal control methods that are based on the calculus of variations. These methods can be divided into direct methods, such as nonlinear programming (NLP) methods, and indirect methods, such as neighboring extremal methods and gradient methods. All these methods can generally be classified as *local* trajectory optimization methods (LTOMs), where the term optimization does not mean finding *the best* solution but rather finding *a* solution.[1] Prior to optimization, the NLP methods and the gradient methods require an initial guess for the control history $\mathbf{u}[\bar{t}]$, whereas the neighboring extremal methods require an initial guess for the starting adjoint vector of LAGRANGE multipliers $\boldsymbol{\lambda}(\bar{t}_0)$ (costate vector).[2] Unfortunately, the convergence behavior of LTOMs (especially of indirect methods) is very sensitive to the initial guess, so that an adequate initial guess is often hard to find, even for an expert in astrodynamics and optimal control theory. Similar initial guesses often produce very dissimilar optimization results, so that the initial guess can not be improved iteratively and trajectory optimization becomes more of an art than science.[3] Even if the optimizer finally converges to an optimal trajectory, this trajectory is typically close to the initial guess that is rarely close to the (unknown) global optimum. Because the optimization process requires nearly permanent attendance of the expert, the search for a good trajectory can become very time-consuming and expensive. Another drawback of LTOMs is the fact that the initial conditions (launch date, initial propellant mass, hyperbolic excess velocity vector, etc.) – although they are crucial for mission performance – are generally chosen according to the expert's judgment and are therefore not part of the actual optimization method.

III. Evolutionary Neurocontrol: A Smart Global Low-Thrust Trajectory Optimization Method

To evade the drawbacks of LTOMs, a *smart global* trajectory optimization method (GTOM) was developed by the author.[4] This method was termed InTrance, which stands for **In**telligent **Tra**jectory optimization using **n**euro**c**ontroller **e**volution. To find a near-globally[a] optimal trajectory, InTrance requires only the target body and intervals for the initial conditions as input. Implementing evolutionary neurocontrol (ENC), InTrance runs without an initial guess and does not require the attendance of a trajectory optimization expert. The remainder of this section will sketch the motivation for ENC and explain the underlying concepts, as well as the application of ENC to solve low-thrust trajectory optimization problems.

A. Motivation for Evolutionary Neurocontrol

ENC fuses artificial neural networks (ANNs) and evolutionary algorithms (EAs) into so-called evolutionary neurocontrollers (ENCs). Like the underlying concepts, it is inspired by the *natural* processes of information processing and optimization. Animal nervous systems incorporate *natural* evolutionary neurocontrollers to control their actions, giving them marvelous capabilities. The smart flight control system of the housefly might provide an example. The nervous system of the housefly comprises about 10^5 neurons. This small natural neural network manages the flight control of the fly, as well as many even more difficult tasks. Nature has optimized the performance of the fly's neurocontroller on this tasks through the recombination and mutation of the fly's genetic material and through natural selection: smarter flies produce more offspring and there is a high probability that some of them are even smarter than their parents. So, if a natural evolutionary neurocontroller is able to steer a housefly optimally from A to B, why should an artificial evolutionary neurocontroller not be able to steer a spacecraft optimally from A to B, which seems to be a much simpler problem? The remainder of this section will describe how such an artificial evolutionary neurocontroller could be implemented.

[a] *Near*-globally optimal because global optimality can rarely be proved for real-world problems.

B. Machine Learning

Within the field of artificial intelligence, one important and difficult class of learning problems are reinforcement learning problems, where the optimal behavior of the learning system (called agent) has to be learned solely through interaction with the environment, which gives an immediate or delayed evaluation[b] J (also called reward or reinforcement).[5,6] The agent's behavior – it's strategy – is defined by an associative mapping from situations to actions $\mathsf{S} : \mathcal{X} \mapsto \mathcal{A}$[c]. The optimal strategy $\mathsf{S}^\star$ of the agent is defined as the one that maximizes the sum of positive reinforcements and minimizes the sum of negative reinforcements over time. If, given a situation $X \in \mathcal{X}$, the agent tries an action $A \in \mathcal{A}$ and the environment *immediately* returns a scalar evaluation $J(X, A)$ of the (X, A) pair, one has an immediate reinforcement learning problem. A more difficult class of learning problems are delayed reinforcement learning problems, where the environment gives only a single evaluation $J(X, A)[t]$, collectively for the sequence of (X, A) pairs occurring in time during the agent's operation.

From the perspective of machine learning, a spacecraft steering strategy may be defined as an associative mapping S that gives – at any time along the trajectory – the current spacecraft control $\mathbf{u}(\bar{t})$ from some input $\mathbf{X}(\bar{t}) \in \mathcal{X}$ that comprises the variables that are relevant for the optimal steering of the spacecraft (the current state of the relevant environment). Because the trajectory is the result of the spacecraft steering strategy, the trajectory optimization problem is actually a problem of finding the optimal spacecraft steering strategy $\mathsf{S}^\star$. This is a delayed reinforcement problem because a spacecraft steering strategy can not be evaluated before its trajectory is known. Only then a reward can be given according to the fulfillment of the optimization objective(s) and constraints. One obvious way to implement spacecraft steering strategies is to use artificial neural networks because they have been successfully applied to learn associative mappings for a wide range of problems.

C. Artificial Neural Networks and Neurocontrol

Being inspired by the processing of information in animal nervous systems, ANNs are a computability paradigm that is alternative to conventional serial digital computers. ANNs are massively parallel, analog, fault tolerant, and adaptive.[7] They are composed of processing elements (called neurons) that model the most elementary functions of biological neurons. Linked together, those elements show some characteristics of the brain, for example, learning from experience, generalizing from previous examples to new ones and extracting essential characteristics from inputs containing noisy and/or irrelevant data, so that they are relatively insensitive to minor variations in its input to produce consistent output.[8]

Because the neurons can be connected in many ways, ANNs exist in a wide variety. Here, however, only feedforward ANNs are considered. Feedforward ANNs have typically a layered topology, where the neurons are organized in a number of neuron layers. The first neuron layer is called the input layer and has n_i input neurons that receive the network's input. The last neuron layer is called the output layer and has n_o output neurons that provide the network's output. All intermediate layers/neurons are called hidden layers/neurons. A feedforward ANN, as it is used here, can be regarded as a continuous parameterized function (called network function)

$$\mathsf{N}_{\boldsymbol{\pi}} : \mathcal{X} \subseteq \mathbb{R}^{n_i} \to \mathcal{Y} \subseteq (0, 1)^{n_o}$$

that maps from a set of inputs $\mathcal{X}$ onto a set of outputs $\mathcal{Y}$. The parameter set $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_m\}$ of the network function comprises the m internal parameters of the ANN (the weights of the neuron connections and the biases of the neurons).

ANNs have been successfully applied as neurocontrollers (NCs) to reinforcement learning problems.[8] An ANN controls a dynamical system by providing a control $\mathbf{Y}(t) \in \mathcal{Y}$ from some input $\mathbf{X}(t) \in \mathcal{X}$ that contains the relevant information for the control task. Note that the NC's behavior is completely characterized by its network function $\mathsf{N}_{\boldsymbol{\pi}}$ (that is again completely characterized by its parameter set $\boldsymbol{\pi}$). If the correct output is known for a set of given inputs (the training set), the difference between the given output and the correct output can be utilized to learn the optimal network function $\mathsf{N}^\star := \mathsf{N}_{\boldsymbol{\pi}^\star}$ by adapting $\boldsymbol{\pi}$ in a way that minimizes this difference for all input/output pairs in the training set. A variety of learning algorithms has been developed for this kind of learning, the backpropagation algorithm – a gradient-based method – being the most widely known. Unfortunately, learning algorithms that rely on a training set fail when the correct

[b] This evaluation is analogous to the cost function in optimal control theory. To emphasize this fact, it will be denoted by the same symbol, J.

[c] $\mathcal{X}$ is called state space and $\mathcal{A}$ is called action space.

output for a given input is not known, as it is the case for delayed reinforcement learning problems. The next section will address a learning method that may be used for determining $\mathsf{N}^\star$ in this case.

D. Evolutionary Algorithms and Evolutionary Neurocontrol

EAs are robust methods for finding global optima in very high dimensional search spaces. They have been successfully applied as a learning method for ANNs,[9–11] as well as for a wide range of other optimization problems. EAs use a vocabulary borrowed from biology. The key element of an EA is a population that comprises numerous individuals $\boldsymbol{\xi}_{j\in\{1,\ldots,q\}}$, which are potential solutions for the given optimization problem. All individuals of the (initially randomly created) population are evaluated according to a fitness function[d] J for their suitability to solve the problem. The fitness of each individual $J(\boldsymbol{\xi}_j)$ is crucial for its probability to reproduce and to create offspring into a newly created population because fitter individuals are selected with a greater probability for reproduction than less fit ones. The selected parents undergo a series of genetic transformations (mutation, recombination) to produce offspring that consists of a mixture of the parents genetic material. Under this selection pressure, the individuals – also called chromosomes or strings – strive for survival. After some reproduction cycles, the population converges against a single solution $\boldsymbol{\xi}^\star$, which is in the best case the globally optimal solution for the given problem. EAs can be employed for searching the NC's optimal network function because a NC parameter set can be mapped onto a real-valued string that provides an equivalent description of the network function. By searching for the fittest individual, the EA searches for the optimal spacecraft trajectory. Figure 1 sketches the transformation of a chromosome into a trajectory.

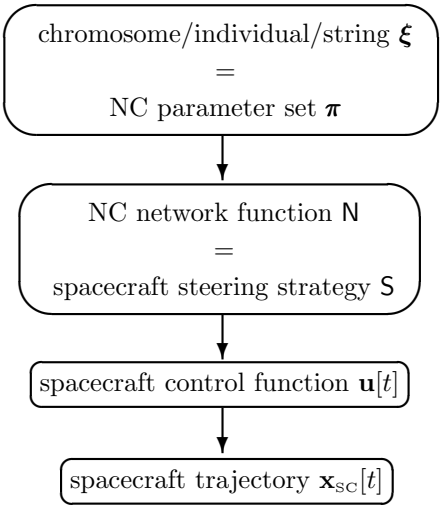


Figure 1. Transformation of a chromosome into a trajectory

E. Neurocontroller Input and Output

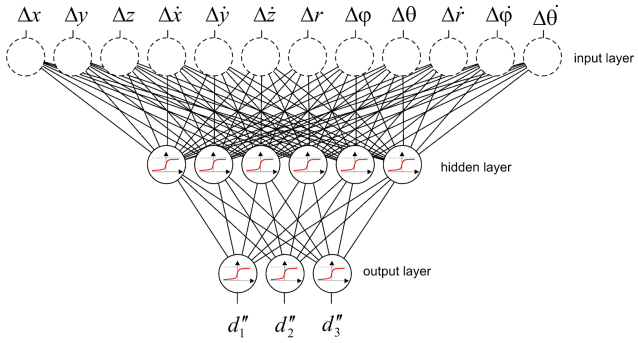
Two fundamental questions concerning the utilization of a NC for spacecraft steering are: (1) "What **input** should the NC get?" (or "What should the NC **know** to steer the spacecraft?") and (2) "What **output** should the NC give?" (or "What should the NC **do** to steer the spacecraft?"). To be robust, a spacecraft steering strategy should be time-independent: to determine the currently optimal spacecraft control $\mathbf{u}(\bar{t}_i)$, the spacecraft steering strategy should have to know – at *any* time step $\bar{t}_i$ – only the current spacecraft state $\mathbf{x}_{\mathrm{SC}}(\bar{t}_i)$ and the current target state $\mathbf{x}_{\mathrm{T}}(\bar{t}_i)$, hence $\mathsf{S} : \mathcal{X} = \{(\mathbf{x}_{\mathrm{SC}}, \mathbf{x}_{\mathrm{T}})\} \mapsto \{\mathbf{u}\}$. If a propulsion system other than a solar sail is employed, the current propellant mass $m_{\mathrm{P}}(\bar{t}_i)$ might be considered as an additional input, $\mathsf{S} : \mathcal{X} = \{(\mathbf{x}_{\mathrm{SC}}, \mathbf{x}_{\mathrm{T}}, m_{\mathrm{P}})\} \mapsto \{\mathbf{u}\}$. The number of potential input sets, however, is still large because $\mathbf{x}_{\mathrm{SC}}$ and $\mathbf{x}_{\mathrm{T}}$ may be given in coordinates of any reference frame and in combinations of them. The difference $\mathbf{x}_{\mathrm{T}} - \mathbf{x}_{\mathrm{SC}}$ may be used as well, also in coordinates of any reference frame and in combinations of them. Two potential input sets are depicted in figure 2.

Each output neuron gives a value $Y_i \in (0, 1)$. The number of potential output sets is also large because there are many alternatives to define $\mathbf{u}$, and to calculate $\mathbf{u}$ from $\mathbf{Y}$. The following approach gave good results for the majority of problems: the NC provides a three-dimensional output vector $\mathbf{d}'' \in (0, 1)^3$ from which a unit vector $\mathbf{d}$ in the desired thrust direction (called direction unit vector) is calculated via

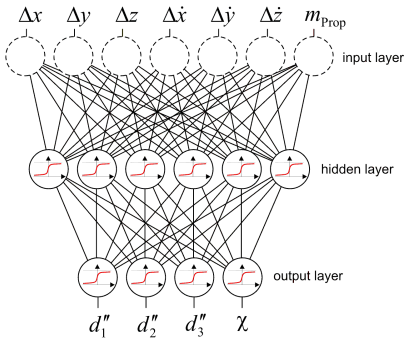
$$\mathbf{d}' = 2\mathbf{d}'' - \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in (-1, 1)^3 \qquad \text{and} \qquad \mathbf{d} = \mathbf{d}'/|\mathbf{d}'| \tag{1}$$

For solar sailcraft, $\mathbf{u} = \mathbf{d}$, hence $\mathsf{S} : \{(\mathbf{x}_{\mathrm{SC}}, \mathbf{x}_{\mathrm{T}})\} \mapsto \{\mathbf{d}\}$ (see figure 2(a)). For SEP spacecraft, the output must include the engine throttle χ, so that $\mathbf{u} = (\mathbf{d}, \chi)$, hence $\mathsf{S} : \{(\mathbf{x}_{\mathrm{SC}}, \mathbf{x}_{\mathrm{T}}, m_{\mathrm{P}})\} \mapsto \{\mathbf{d}, \chi\}$ (see figure 2(b)).

[d]This fitness function is also analogous to the cost function in optimal control theory. To emphasize this fact, it will be denoted by the same symbol, J.



(a) Example for a NC that implements a solar sailcraft steering strategy



(b) Example for a NC that implements a SEP spacecraft steering strategy

Figure 2.

F. Evolutionary Neurocontroller Design

Figure 3 shows how an ENC may be applied for low-thrust trajectory optimization. To find the optimal spacecraft trajectory, the ENC method runs in two loops. Within the (inner) trajectory integration loop, a NC steers the spacecraft according to its network function $\mathsf{N}_{\boldsymbol{\pi}_j}$ that is completely defined by the NC's parameter set $\boldsymbol{\pi}_j$. The EA in the (outer) NC optimization loop holds a population of NC parameter sets, $\Xi = \{\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_q\}$, and examines all of them for their suitability to generate an optimal trajectory. Within the trajectory optimization loop, the NC takes the current spacecraft state $\mathbf{x}_{\mathrm{SC}}(\bar{t}_{i\in\{0,\ldots,\tau-1\}})$ and that of the target $\mathbf{x}_{\mathrm{T}}(\bar{t}_i)$ as input, and maps them onto some output. For SEP spacecraft, the input includes the current propellant mass $m_{\mathrm{P}}(\bar{t}_i)$ and the output includes the current throttle $\chi(\bar{t}_i)$. The first three output values are interpreted as the components of $\mathbf{d}''(\bar{t}_i)$, from which the direction unit vector $\mathbf{d}(\bar{t}_i)$ is calculated via Eq. (1). Now, the spacecraft control $\mathbf{u}(\bar{t}_i)$ is calculated from the NC output. Then, $\mathbf{x}_{\mathrm{SC}}(\bar{t}_i)$ and $\mathbf{u}(\bar{t}_i)$ are inserted into the equations of motion and (numerically) integrated over one time step $\Delta\bar{t} = \bar{t}_{i+1} - \bar{t}_i$ to yield $\mathbf{x}_{\mathrm{SC}}(\bar{t}_{i+1})$. The new state is fed back into the NC. The trajectory integration loop stops when the accuracy of the trajectory is sufficient or when a given time limit is reached. Then, back in the NC optimization loop, the NC's trajectory is rated by the fitness function $J(\boldsymbol{\pi}_j)$. The fitness of $\boldsymbol{\pi}_j$ is crucial for its probability to reproduce and to create offspring. Under this selection pressure, the EA breeds more and more suitable steering strategies that generate better and better trajectories. Finally, the EA converges against a single steering strategy, which gives in the best case a near-globally optimal trajectory $\mathbf{x}^{\star}_{\mathrm{SC}}[t]$.

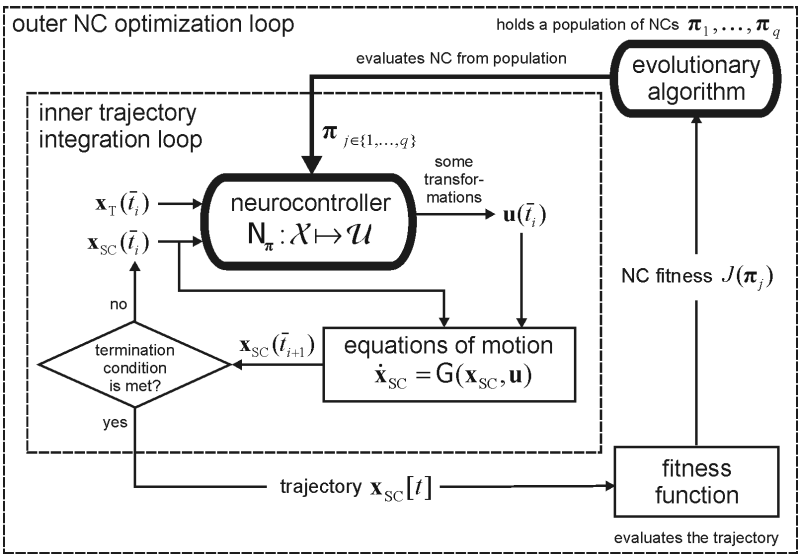


Figure 3. Low-thrust trajectory optimization using ENC

G. Additionally Encoded Problem Parameters

If an EA is already employed for the optimization of the NC, it is manifest to use it also for the co-optimization of additional problem parameters. This can be done without major additional effort. InTrance encodes the following parameters additionally on the chromosome, making them an explicit part of the optimization

problem: (1) the launch date, (2) the launch velocity vector (hyperbolic excess velocity vector), and (3) the initial propellant mass (except for solar sailcraft).

H. Neurocontroller Fitness Assignment

The optimality of a trajectory might be defined with respect to various (primary) objectives (e.g., transfer time or propellant consumption). When an ENC is used for trajectory optimization, the accuracy of the trajectory with respect to the terminal constraints must also be considered as a secondary optimization objective because it is not explicitly stated elsewhere and needs therefore not to be satisfied throughout the search process. If, for example, the transfer time for a rendezvous is to be minimized, the fitness function must include the transfer time $T = \bar{t}_f - \bar{t}_0$, the final distance to the target $\Delta r_f = |\mathbf{r}_{\mathrm{T}}(\bar{t}_f) - \mathbf{r}_{\mathrm{SC}}(\bar{t}_f)|$, and the final relative velocity to the target $\Delta v_f = |\dot{\mathbf{r}}_{\mathrm{T}}(\bar{t}_f) - \dot{\mathbf{r}}_{\mathrm{SC}}(\bar{t}_f)|$, hence $J(T, \Delta r_f, \Delta v_f)$. If, for example, the propellant mass for a flyby problem is to be minimized, T and Δv_f are not relevant but the consumed propellant $\Delta m_{\mathrm{P}} = m_{\mathrm{P}}(\bar{t}_0) - m_{\mathrm{P}}(\bar{t}_f)$ must be included in the fitness function, hence $J(\Delta m_{\mathrm{P}}, \Delta r_f)$. Because the ENC unlikely generates a trajectory that satisfies the terminal constraints $\Delta r_f = 0\,\mathrm{m}$ and $\Delta v_f = 0\,\mathrm{m/s}$ exactly, a maximal allowed distance $\Delta r_{f,\max}$ and a maximal allowed relative velocity $\Delta v_{f,\max}$ have to be defined. Using $\Delta r_{f,\max}$ and $\Delta v_{f,\max}$, the distance and relative velocity at the target can be normalized:

$$\Delta R = \frac{\Delta r}{\Delta r_{f,\max}} \qquad \Delta R_f = \frac{\Delta r_f}{\Delta r_{f,\max}} \tag{2}$$

$$\Delta V = \frac{\Delta v}{\Delta v_{f,\max}} \qquad \Delta V_f = \frac{\Delta v_f}{\Delta v_{f,\max}} \tag{3}$$

Furthermore, it is necessary to define a measure for the accuracy of the trajectory with respect to the terminal constraints, e.g.:

$$\Delta X = \sqrt{\frac{1}{2}\left(\Delta R^2 + \Delta V^2\right)} \qquad \Delta X_f = \sqrt{\frac{1}{2}\left(\Delta R_f^2 + \Delta V_f^2\right)} \tag{4}$$

Because in the beginning of the search process most individuals do not achieve the required accuracy, a maximal transfer time $T_{\max}$ must be defined for the numerical integration of the trajectory. Now, sub-fitness functions may be defined with respect to the primary and the secondary optimization objectives. It was found that the performance of ENC depends strongly on an adequate choice of the sub-fitness functions and on their composition to an (overall) fitness function. This is reasonable because the fitness function has not only to decide autonomously which trajectories are good and which are not, but also which trajectories are promising for future cultivation. The primary sub-fitness functions

$$J_T = 1000 \cdot \left(1 - \frac{T}{T_{\max}}\right) \qquad J_{m_{\mathrm{P}}} = \frac{m_{\mathrm{P}}(\bar{t}_0)}{2m_{\mathrm{P}}(\bar{t}_0) - m_{\mathrm{P}}(\bar{t}_f)} - \frac{1}{3} \tag{5}$$

and the secondary sub-fitness functions

$$J_r = \log\left(\frac{1}{\Delta R_f}\right) \qquad J_v = \log\left(\frac{1}{\Delta V_f}\right) \tag{6}$$

were empirically found to produce good results. They have been used for all trajectory calculations within this paper. J_r and J_v are positive, if the respective accuracy requirement is fulfilled and negative, if it is not. Another empirical finding is that the search process should first concentrate on the accuracy of the trajectory and then on the primary optimization objective. Therefore, the sub-fitness functions for the primary optimization objectives are modified to

$$J'_T = \begin{cases} 0 & \text{if } J_r < 0 \vee J_v < 0 \\ J_T & \text{if } J_r \geq 0 \wedge J_v \geq 0 \end{cases} \qquad J'_{m_{\mathrm{P}}} = \begin{cases} 0 & \text{if } J_r < 0 \vee J_v < 0 \\ J_{m_{\mathrm{P}}} & \text{if } J_r \geq 0 \wedge J_v \geq 0 \end{cases} \tag{7}$$

To guide the search process, sub-fitness functions for other trajectory parameters (e.g., eccentricity or orientation of the orbital plane) might be introduced in the same way. They might be used as long as $J_r < 0 \vee J_v < 0$ and be discarded when $J_r \geq 0 \wedge J_v \geq 0$. This guidance of the search process, however, is at the expense of

the simplicity of the fitness function. To minimize the transfer time for a rendezvous, two different fitness functions might be conceived:

$$J_1(T, \Delta r_f, \Delta v_f) = J'_T + \frac{1}{\sqrt{\Delta R_f^2 + \Delta V_f^2}} \tag{8}$$

$$J_2(T, \Delta r_f, \Delta v_f) = J'_T + \frac{1}{\sqrt{2 \cdot \max(\Delta R_f, \Delta V_f)^2}} \tag{9}$$

Using J_1, a poor final distance can be compensated with a good relative velocity and vice versa. This is not possible for J_2. It was found empirically that J_1 is superior to J_2 for most problems. For some problems, however, J_1 might run into a local optimum, where it yields very good final distances but fails to match the final velocity. To minimize the propellant mass for a rendezvous, the fitness functions are similar, but J'_T is replaced with J'_{m_P}:

$$J_1(\Delta m_\mathrm{P}, \Delta r_f, \Delta v_f) = J'_{m_\mathrm{P}} + \frac{1}{\sqrt{\Delta R_f^2 + \Delta V_f^2}} \tag{10}$$

$$J_2(\Delta m_\mathrm{P}, \Delta r_f, \Delta v_f) = J'_{m_\mathrm{P}} + \frac{1}{\sqrt{2 \cdot \max(\Delta R_f, \Delta V_f)^2}} \tag{11}$$

To minimize the transfer time for a flyby at the target, only the positions must match:

$$J(T, \Delta r_f) = J'_T + \frac{1}{\Delta R_f} \tag{12}$$

In the same way, to minimize the propellant mass for a flyby, J'_T is replaced with J'_{m_P}:

$$J(\Delta m_\mathrm{P}, \Delta r_f) = J'_{m_\mathrm{P}} + \frac{1}{\Delta R_f} \tag{13}$$

IV. InTrance Implementation Specific Issues

The implementation of InTrance is largely based on approaches that have been proposed to avoid – or at least to mitigate – the problems that are associated with standard genetic algorithms (GAs). InTrance implements: (1) real-valued parameter encoding, (2) multi-objective tournament selection with steady-state reproduction, (3) real delta coding, and (4) non-standard evolutionary operators.

A. Real-valued Parameter Encoding

Using a binary representation of the optimization parameters, GAs are often not able to provide high-precision solutions. Experiments like those performed by Michalewicz indicate that the real-valued parameter representation is faster, more consistent from run to run, and provides a higher precision, especially in large search spaces.[12] It is also closer to the problem space, which facilitates the development of problem-specific evolutionary operators.

B. Multi-Objective Tournament Selection and Steady-State Reproduction

The two most important issues in evolutionary search are selective pressure and population diversity, both being strongly related: an increase in selective pressure decreases population diversity and vice versa. A selective pressure that is too strong supports premature convergence to a local optimum, whereas a selective pressure that is too weak makes the search ineffective. A computationally efficient selection method is **tournament selection**, where a parent individual is selected by choosing randomly two individuals from the population and allowing only the better one to reproduce. Using tournament selection, the selective pressure remains constant throughout the search process because the reproduction probability of each individual is independent of its absolute fitness.[13] Another advantage of tournament selection is that each tournament can be performed (randomly) with respect to a different optimization objective. Such a selection mechanism prefers individuals that perform reasonably well with respect to *all* objectives, allowing multi-objective optimization. (It is like a duel between two cowboy gunslingers. To survive a duel, one must draw fast *and* aim accurately.)

Another approach that has been selected due to its computational efficiency is **steady-state reproduction** (also called one-at-a-time reproduction), where only one reproduction takes place at each time step. It is computationally less expensive than the generational reproduction method used by standard GAs.[4] Steady-state reproduction in combination with tournament selection is conceptually very simple (figure 4): two tournaments are performed to determine the two parent chromosomes, which are the winners of the tournaments. They stay in the population, while the two tournament losers are replaced by the two offspring chromosomes.

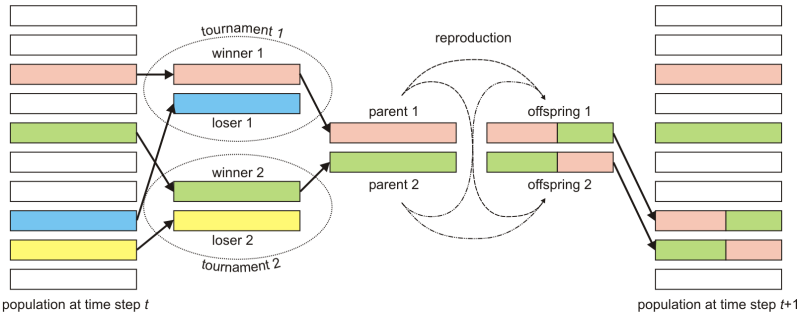


Figure 4. Steady-state reproduction with tournament selection

C. Real Delta Coding

Delta coding (DC) was proposed by Whitley et al. to enhance the precision and convergence behavior of genetic search (for binary strings).[14] It is based on the idea that a string can also express a distance $\boldsymbol{\delta}$ to some previous solution $\mathbf{h}$, called interim solution or partial solution, which is the best known solution so far. Using DC, each individual $\boldsymbol{\xi}_j$ consists of two parts, the partial solution and the $\boldsymbol{\delta}$-chromosome: $\boldsymbol{\xi}_j = \mathbf{h} + \boldsymbol{\delta}_j$. At any time, only a dynamically selected subspace of the total search space is explored. This subspace is constructed around the most recent partial solution. DC starts with the initial run of a GA. After the population has converged, the best $\boldsymbol{\delta}$-chromosome is added to the old partial solution to form the new partial solution. After that, a new population is created within an new (reduced or enlarged) search subspace that is centered around the new partial solution. By periodically re-initializing the population, DC avoids premature convergence. DC provides a mechanism that reduces or enlarges the size of the current search subspace.[14] The reduction mechanism allows the algorithm to focus the search on search subspaces that appear promising, whereas the expansion mechanism allows the algorithm to explore previously overlooked portions of the search space in later search. The author extended in Refs. 10 and 15 the idea of DC to real-valued strings (floating point delta coding, FPDC). The algorithm that is used for InTrance is a revised version of FPDC and should be termed **real delta coding** (RDC).

RDC runs in cycles, called epochs. Within each epoch e_i, a dynamically selected subspace $\mathcal{H}_i$ of the total (ℓ-dimensional) search space $\mathcal{H}$ around the most recent partial solution $\mathbf{h}(e_i)$ is explored. For the first epoch e_0, the search subspace

$$\mathcal{H}_0 = [-\delta_{\max}(e_0), \delta_{\max}(e_0)]^\ell \subseteq \mathcal{H} \subset \mathbb{R}^\ell$$

is constructed around the partial solution $\mathbf{h}(e_0) = \mathbf{0}$ and the population for the first time step t_0 of epoch e_0 is initialized at random, $\Xi^{t_0}(e_0) = \{\boldsymbol{\delta}_1^{t_0}(e_0), \ldots, \boldsymbol{\delta}_q^{t_0}(e_0)\}$ (without loss of generality, it is assumed that the individuals are always sorted according to their fitness). Then, the EA – as described above – runs until the epochal convergence criterion is met. This convergence criterion depends on the relative improvement within the last ν time steps: if at some time step t, the relative improvement within the last ν time steps, $J(\boldsymbol{\xi}_1^t) - J(\boldsymbol{\xi}_1^{t-\nu})$, is less than some small value ε, the population is converged and $t =: t_c$ ('c' for convergence). After convergence, $\boldsymbol{\xi}_1^{t_c}(e_0)$, the best found solution in epoch e_0, is taken as the partial solution $\mathbf{h}(e_1)$ for the next epoch. To guarantee the convergence of the algorithm, RDC uses no search subspace expansion mechanism. The search subspace reduction mechanism is very simple: $\delta_{\max}(e_1) = \kappa \cdot \delta_{\max}(e_0)$, where $0 < \kappa < 1$ is a user-defined parameter. Within the new search subspace

$$\mathcal{H}_1 = [h_1(e_1) - \delta_{\max}(e_1), h_1(e_1) + \delta_{\max}(e_1)] \times \ldots \times [h_\ell(e_1) - \delta_{\max}(e_1), h_\ell(e_1) + \delta_{\max}(e_1)] \subset \mathbb{R}^\ell$$

$\Xi^{t_0}(e_1)$ is again initialized at random, and so on. This is done until the RDC convergence criterion is met. This convergence criterion is similar to the epochal convergence criterion: if, at some epoch e_i, the relative improvement to the last epoch, $J(\mathbf{h}(e_i)) - J(\mathbf{h}(e_{i-1}))$, is less than some small value ε, RDC is said to be converged. For an ideal RDC performance, the search within each search subspace must be very extensive (ν and κ large, ε small). In practice, however, a trade-off must be made between search effort and search duration. To put the decision which values to take not on the user, a robust setting for ν and ε is hard-wired in InTrance.

D. Non-Standard Evolutionary Operators

Standard genetic operators that work on binary chromosomes can not be applied directly to real-valued chromosomes. New genetic operators, which are tailored to work on real-valued strings, have to be designed. Besides the analogues to the one-point crossover operator and the bit mutation operator that is used in standard GAs, many evolutionary operators have been proposed so far. InTrance implements three crossover operators (figure 5): (1) The implemented **one-point crossover** operator[16] works analogous to its binary pendant. (2) If the **uniform crossover** operator[17] is applied, it is decided (randomly) for each locus in the first offspring which parent contributes its parameter in that position. The second offspring receives the parameter from the other parent. Because uniform crossover exchanges single parameters and not string segments, it can combine features regardless of their relative location on the string. (3) If the **crossover nodes** operator[16] is applied, it is decided (randomly) for each neuron in the first offspring which parent contributes its parameters for that neuron. The second offspring receives the neuron parameters from the other parent. This prevents that the logical subgroups of the string – the parameters of a single neuron – are torn apart.

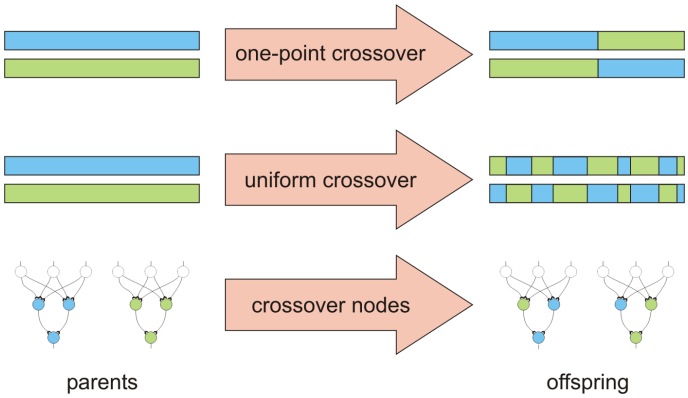


Figure 5. InTrance evolutionary operators

All three crossover operators only exchange real numbers between the chromosomes but do not change the numbers themselves. This can only be done by the mutation operator. WHITLEY pointed out that no mutation operator is necessary for DC because the population is re-initialized at regular intervals.[14] Preliminary InTrance tests, however, have revealed that this might be different for real-valued strings because the absence of a mutation operator leads for small population sizes to premature convergence within the epochs. Therefore, InTrance implements a computationally efficient mutation operator, which should be termed **fast uniform mutation**: it is decided for the entire chromosome (with the mutation probability $0 \leq p_m \leq 1$), whether or not a single parameter on the chromosome is to be mutated. If the chromosome is to be mutated, one locus is randomly selected. If, for example, the i^{th} locus of chromosome $\boldsymbol{\delta}_j$ is to be mutated, its parameter δ_{ji} is replaced with a new one $\delta'_{ji} \in [-\delta_{\max}, +\delta_{\max}]$.

V. Results

To assess the performance of ENC for low-thrust trajectory optimization, InTrance was used to recalculate trajectories for problems for which trajectories have been found in the literature (henceforth called reference problems/trajectories). Two of them are used within this paper to investigate the influence of various factors (accuracy requirements, EA population size, NC topology) on the convergence behavior of the method and the quality of the obtained solutions: a Mercury rendezvous using solar sail propulsion and a near-Earth asteroid (NEA) rendezvous using solar sail and solar electric propulsion. The results for further reference missions can be found in Ref. 4.

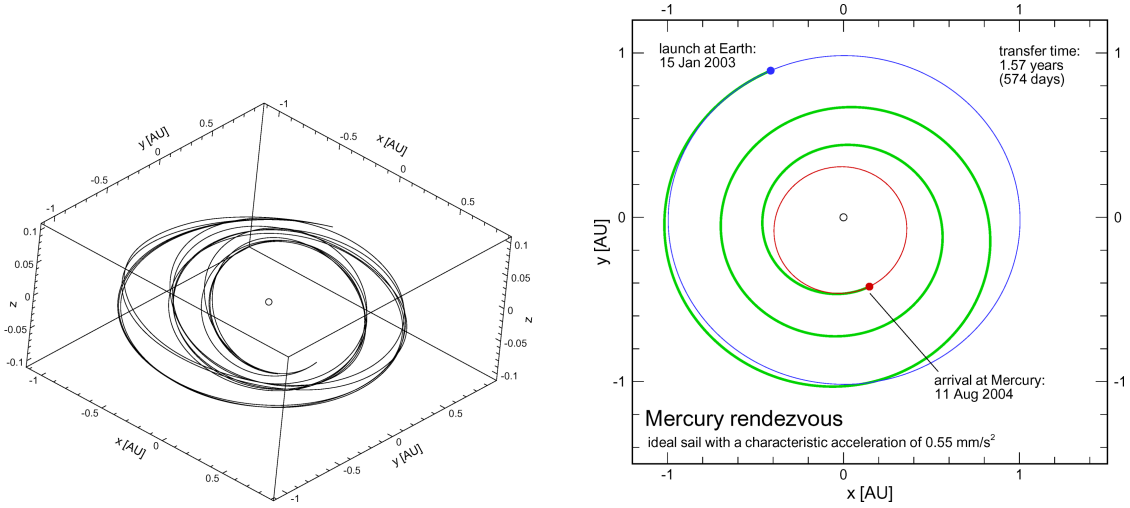
A. Mercury Rendezvous

Within this section, the convergence behavior of ENC and the quality of the obtained solutions is assessed for an exemplary rendezvous mission to Mercury. For an ideal solar sail with a characteristic acceleration[e] of $0.55\,\text{mm/s}^2$ a (locally) optimal trajectory was calculated in Refs. 18-19 using a LTOM. This reference trajectory launches from Earth on 15 Jan 03 and takes 665 days to rendezvous Mercury, if the solar sailcraft is inserted directly into an interplanetary trajectory with zero hyperbolic excess energy ($C_3 = 0\,\text{km}^2/\text{s}^2$).

[e] maximum acceleration of a solar sailcraft at Earth distance from the sun

1. *Convergence Behavior*

To evaluate its convergence behavior, InTrance was run five times – using five different initial NC populations – for the launch date of the reference trajectory (reference launch date). A 12–30–3 neurocontroller (12 input neurons, 1 hidden layer with 30 hidden neurons, 3 output neurons) was used, where the input neurons receive the current solar sailcraft state $\mathbf{x}_{\mathrm{SC}}$ and the current target state $\mathbf{x}_{\mathrm{T}}$ in cartesian coordinates, and the output neurons define the direction unit vector $\mathbf{d}$. On the basis of preliminary InTrance-runs, which indicated that the optimal transfer takes less than 600 days, the maximum transfer time was set to $T_{\max} = 600\,\mathrm{days}$. For discretization, this time interval was divided into $\tau = 600$ finite elements of equal length, so that the NC was allowed to change the sail attitude once every day. The final accuracy limit was set to $\Delta r_{f,\max} = 100\,000\,\mathrm{km}$ and $\Delta v_{f,\max} = 100\,\mathrm{m/s}$.[f] The population size was set to $q = 50$. Figure 6(a) shows the resulting trajectories for the five InTrance-runs.



(a) Trajectories for five different initial populations

(b) Best InTrance-trajectory

Figure 6. Mercury rendezvous trajectories (reference launch date)

The best trajectory (figure 6(b)) is 91 days ($\Delta T\% = 16\%$)[g] faster than the reference trajectory, revealing that the latter is far from the global optimum. The final distance is $\Delta r_f \approx 57\,000\,\mathrm{km}$ and the final relative velocity to Mercury is $\Delta v_f \approx 57\,\mathrm{m/s}$, both being well better than the required accuracy limits. The small variance of the five solutions gives evidence for a good convergence behavior of ENC.

2. *Different Population Sizes and Accuracy Requirements*

To assess the influence of the population size and the required accuracy ($\Delta r_{f,\max}$ and $\Delta v_{f,\max}$) on the results, InTrance was run for three different population sizes ($q = 25; 50; 100$) and two final accuracy limits (FAL1: $\Delta r_{f,\max} = 1\,000\,000\,\mathrm{km}$, $\Delta v_{f,\max} = 500\,\mathrm{m/s}$; FAL2: $\Delta r_{f,\max} = 100\,000\,\mathrm{km}$, $\Delta v_{f,\max} = 100\,\mathrm{m/s}$), using the same 12–30–3 NC. The results are shown in table 1 and 2. Table 3 shows the average runtime for the calculations.

Table 1. Transfer times to Mercury for different population sizes (FAL1, five different initial populations)

Population size	Transfer time T [days]					Average	Std. dev.
	run 1	run 2	run 3	run 4	run 5		
25	566	573	577	570	572	571.6	4.04
50	569	568	573	571	568	569.8	2.17
100	570	**564**	**564**	567	569	**566.8**	2.77

[f] $\Delta v_{f,\max} = 100\,\mathrm{m/s}$ was also used in Refs. 18-19, whereas $\Delta r_{f,\max}$ is not given there.
[g] $\Delta T\% = (T_{\mathrm{InTrance}} - T_{\mathrm{R}})/\min(T_{\mathrm{R}}, T_{\mathrm{InTrance}}) \cdot 100\%$, where T_{R} is the transfer time of the reference trajectory.

Table 2. Transfer times to Mercury for different population sizes (FAL2, five different initial populations)

Population size	Transfer time T [days]					Average	Std. dev.
	run 1	run 2	run 3	run 4	run 5		
25	585	592	580	579	588	584.8	5.45
50	**574**	578	589	579	589	**581.8**	6.83
100	584	590	576	583	585	583.6	5.03

For the less demanding FAL1, the trajectories are faster and the standard deviation of the transfer time is lower. Unlike it may be expected, the quality of the solutions does not depend considerably on the population size. Table 3 shows, however, that the search duration depends substantially on the population size and on the required accuracy. The dependency on the population size is straightforward. A larger population takes longer to converge against a point in the search space where no further improvement is probable. Therefore, the population size had been introduced into the RDC epochal convergence criterion, so that the EA is allowed to search longer when the population size is large. The dependency of the search duration on the final accuracy limit might be attributed to the employed fitness function, which allots little value to further improvements in Δr_f and Δv_f if the required accuracy is already achieved ($\Delta X_f \leq 1$). Consequently, a more demanding accuracy requirement leaves more room for improvements, hence delaying the convergence of RDC. For the following calculations, $q = 50$ and FAL2 has been used, if it is not stated otherwise.

Table 3. Average runtime on a personal computer with a 1.3 GHz processor

Accuracy limit	Average runtime [hours]		
	$q = 25$	$q = 50$	$q = 100$
FAL1	3.0	5.8	7.9
FAL2	5.0	6.8	11.6

3. Different Neurocontrollers

For the calculations above, a 12–30–3 neurocontroller had been used, where the input neurons receive the current solar sailcraft state and the current target state in cartesian coordinates. Because this NC is only one of many possible NCs that may be used for this trajectory optimization problem, different input sets and different numbers of hidden neurons/layers have been tested. (Investigations and results for different NC output sets have been reported before.[20]) Table 4 shows the different NC input sets that have been considered.

Table 4. Tested NC input sets

Notation	NC input set $\mathcal{X}$
(c) 12–□–3	$\mathbf{x}_{\mathrm{SC}}$ and $\mathbf{x}_{\mathrm{T}}$ in cartesian coordinates
(c) 6–□–3	$\mathbf{x}_{\mathrm{T}} - \mathbf{x}_{\mathrm{SC}}$ in cartesian coordinates
(p) 12–□–3	$\mathbf{x}_{\mathrm{SC}}$ and $\mathbf{x}_{\mathrm{T}}$ in polar coordinates
(e) 6–□–3	$\mathbf{x}_{\mathrm{T}} - \mathbf{x}_{\mathrm{SC}}$ as orbital element differences ($a_{\mathrm{T}} - a_{\mathrm{SC}}$, etc.)
(cp) 24–□–3	$\mathbf{x}_{\mathrm{SC}}$ and $\mathbf{x}_{\mathrm{T}}$ in cartesian and polar coordinates
(cp) 12–□–3	$\mathbf{x}_{\mathrm{T}} - \mathbf{x}_{\mathrm{SC}}$ in cartesian and polar coordinates
(ce) 12–□–3	$\mathbf{x}_{\mathrm{T}} - \mathbf{x}_{\mathrm{SC}}$ in cartesian coordinates and as orbital element differences
(pe) 12–□–3	$\mathbf{x}_{\mathrm{T}} - \mathbf{x}_{\mathrm{SC}}$ in polar coordinates and as orbital element differences
(cpe) 36–□–3	$\mathbf{x}_{\mathrm{SC}}$ and $\mathbf{x}_{\mathrm{T}}$ in cartesian and polar coordinates and as orbital elements
(cpe) 18–□–3	$\mathbf{x}_{\mathrm{T}} - \mathbf{x}_{\mathrm{SC}}$ in cartesian and polar coordinates and as orbital element differences

(□ = wildcard for the neurons in the hidden layer(s))

Table 5 shows the results for the NCs that achieved the required final accuracy limit (FAL2) in all of five

Table 5. Transfer times to Mercury for different steering strategy sets and different NC topologies (FAL2, reference launch date, five different initial populations)

NC topology	Transfer time T [days]					Average	Std. dev.
	run 1	run 2	run 3	run 4	run 5		
(c) 12–10–3	592	582	587	593	577	586.2	6.76
(c) 12–20–3	579	583	579	577	580	579.6	2.19
(c) 12–30–3	574	578	589	579	589	581.8	6.83
(c) 12–40–3	583	579	575	578	578	578.6	2.88
(c) 12–15–15–3	577	578	577	581	592	581.0	6.36
(cp) 24–20–3	580	584	575	**573**	591	580.6	7.23
(cp) 24–30–3	579	576	**573**	578	575	**576.2**	2.39
(cp) 24–40–3	591	575	580	581	586	582.6	6.11
(cp) 24–15–15–3	575	579	578	580	583	579.0	2.92
(c) 6–30–3	591	592	583	579	581	585.2	5.93
(cpe) 36–30–3	583	589	579	584	579	582.8	4.15
(cpe) 18–30–3	581	591	585	577	587	584.2	5.40

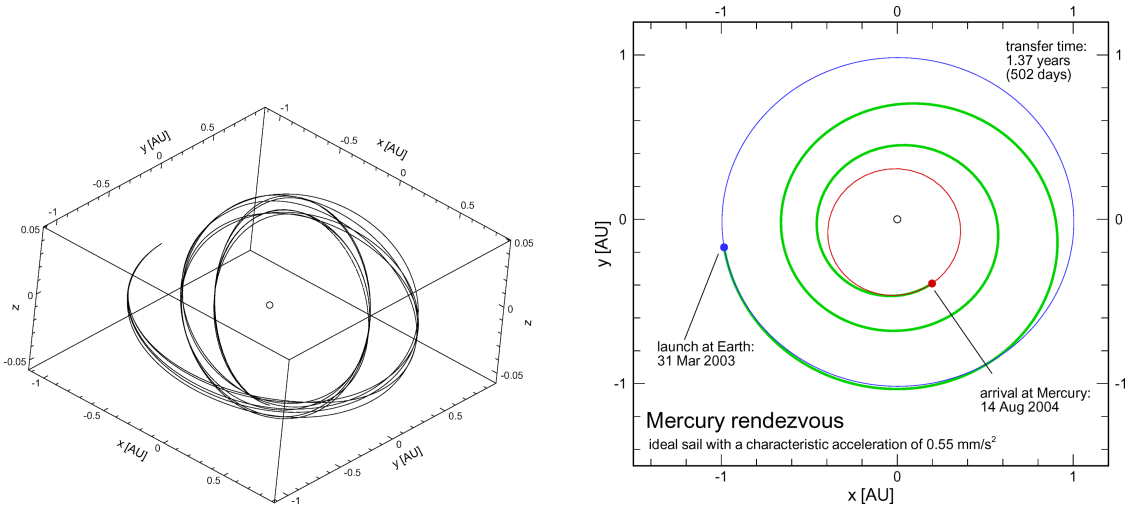
InTrance-runs. On average, the best results (best solution, best average, low standard deviation) have been obtained using the (cp) 24–30–3 neurocontroller. Note that only the NCs with cartesian inputs achieved the required accuracy in all cases, although the motion of spacecraft in interplanetary space is better described in polar coordinates or by orbital elements. This might be due to the fact that the variation of the states is larger in cartesian coordinates than in polar coordinates or in orbital elements, thus providing a more substantial input to the NC. The number of hidden neurons has – at least for this trajectory optimization problem – little effect on the results. Even NCs with a very small number of hidden neurons provide acceptable results for this problem.

4. Optimization of the Launch Date

To find the optimal launch date for the Mercury rendezvous, InTrance was used to determine the shortest orbit transfer. Because no rendezvous with the target body but only with the target orbit is required in this case, the launch interval was set to 1 year, the orbital period of Earth[h]. After five InTrance runs, the shortest found orbit transfer takes $T = 510$ days to reach the orbit of Mercury. For this orbit transfer, the solar sailcraft's angular position at launch is $\varphi_{\mathrm{SC}}(\bar{t}_0) \doteq -2.90$ and at arrival $\varphi_{\mathrm{SC}}(\bar{t}_0 + T) \doteq -0.83$. By scanning the planetary positions, it can be found that within a 1 year-interval around the reference launch date, the constellation of Earth and Mercury is most similar to that of the optimal orbit transfer solution for a launch on 27 Mar 03 (where $\varphi_{\mathrm{Earth}}(27\,\mathrm{Mar}\,03) \doteq -3.02$ and $\varphi_{\mathrm{Mercury}}(27\,\mathrm{Mar}\,03 + 510\,\mathrm{days}) \doteq -0.83$). InTrance was run five times for the launch date that was expected to be optimal (27 Mar 03). However, to allow the steering strategy to compensate for the small difference in Earth's angular position at launch, the launch date was not prescribed exactly, but InTrance was allowed to choose the launch date from the interval [26 Mar 03 00:00, 31 Mar 03 00:00]. The maximum transfer time was set to $T_{\max} = 600$ days (with $\tau = 600$).

Figure 7(a) shows that all five InTrance-trajectories differ only little. This small variance of the solutions gives evidence for a good convergence behavior of ENC. Taking 502 days to rendezvous Mercury, the best InTrance-trajectory (figure 7(b)) is now 163 days (32%) faster than the reference trajectory. The final distance to Mercury is $\Delta r_f \approx 20\,000$ km and the final relative velocity to Mercury is $\Delta v_f \approx 20$ m/s, both being well better than the required values (FAL2). The optimal launch date was found to be 31 Mar 03, 75 days later than the reference launch date. Note that the optimal transfer time to rendezvous Mercury is even better than the previously found optimal orbit transfer time of 510 days, which is obviously not optimal ex post. This might be attributed to the larger launch interval (1 year instead of 5 days), which makes the search more difficult for the EA because different launch dates within a 1 year interval require significantly different steering strategies, whereas the good steering strategies within a 5 day interval are more similar.

[h] The orbital period of the target body is not relevant for the orbit transfer problem.



(a) Trajectories for five different initial populations

(b) Best InTrance-trajectory

Figure 7. Mercury rendezvous trajectories (optimized launch date)

B. Near-Earth Asteroid Rendezvous

Within this section, the convergence behavior of ENC and the quality of the obtained solutions is further assessed for an exemplary rendezvous mission to a near-Earth asteroid (1996FG$_3$). For solar sailcraft with a characteristic acceleration of 0.14 mm/s^2 (ideally reflecting 50 m × 50 m solar sail, launch mass 148 kg, useful mass[i] 75 kg) a (locally) optimal trajectory was calculated in Refs. 21-22 using a LTOM. This reference trajectory launches from Earth on 13 Aug 06 and takes 1640 days to rendezvous 1996FG$_3$, if the solar sailcraft is inserted directly into an interplanetary trajectory with an hyperbolic excess energy of $C_3 = 4\,\mathrm{km}^2/\mathrm{s}^2$.

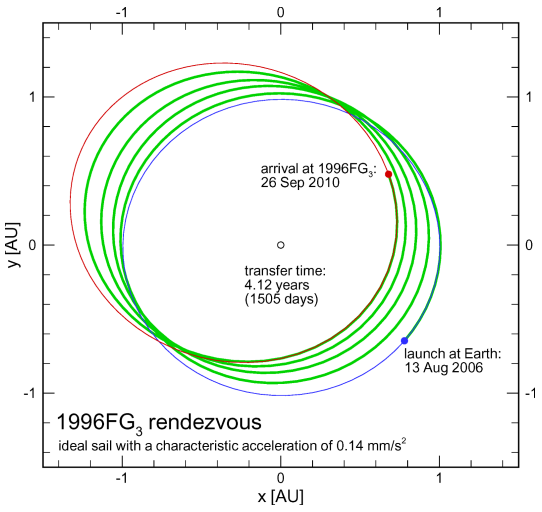
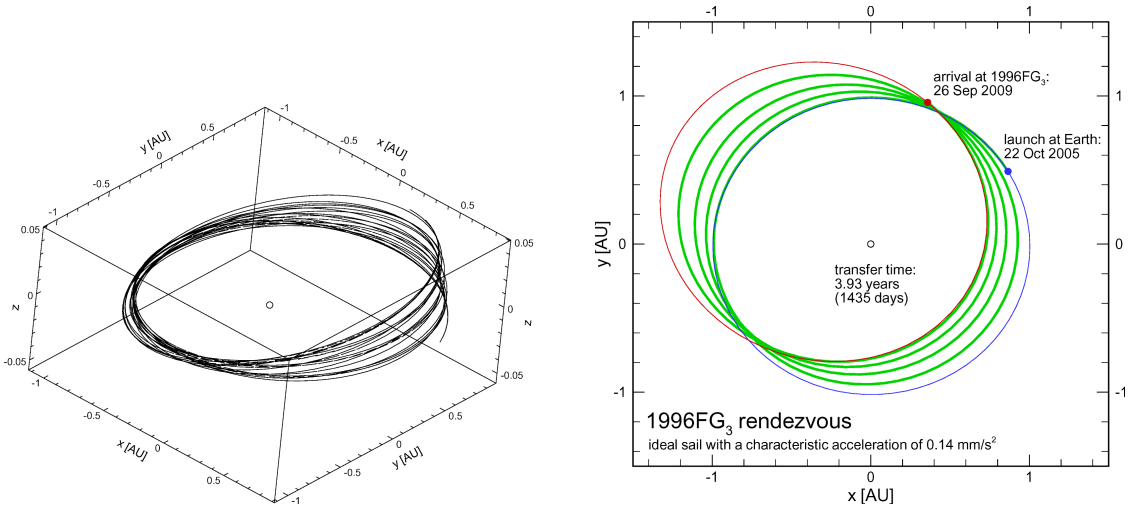


Figure 8. Best InTrance-trajectory for the 1996FG$_3$ rendezvous (reference launch date)

In the first experiment, InTrance was run five times for the reference launch date but with zero hyperbolic excess energy. The accuracy limit for the distance and the relative velocity at the target was set to $\Delta r_{f,\max} = 300\,000\,\mathrm{km}$ and $\Delta v_{f,\max} = 100\,\mathrm{m/s}$, respectively, which is compatible with the reference trajectory. The best found InTrance-trajectory (figure 8) is 135 days faster (9.0%) than the reference trajectory, while reducing at the same time the C_3-requirement from $4\,\mathrm{km}^2/\mathrm{s}^2$ to $0\,\mathrm{km}^2/\mathrm{s}^2$, thus permitting a reduction of the launcher requirements and eventually of launch costs. The final distance to 1996FG$_3$ is $\Delta r_f \approx 200\,000\,\mathrm{km}$ and the final relative velocity is $\Delta v_f \approx 65\,\mathrm{m/s}$, both being better than the required values.

In the second experiment, InTrance was used to find the optimal launch date for the 1996FG$_3$ rendezvous problem (with $C_3 = 0\,\mathrm{km}^2/\mathrm{s}^2$). Figure 9(a) shows the solutions for five runs with different initial NC populations. The small variance of the five solutions gives evidence for a good convergence behavior of ENC. Taking 1435 days to rendezvous 1996FG$_3$, the trajectory is now 205 days (14%) faster than the reference trajectory. The final distance to 1996FG$_3$ is $\Delta r_f \approx 267\,000\,\mathrm{km}$ and the final relative velocity is $\Delta v_f \approx 89\,\mathrm{m/s}$, both being better than the required values. The optimal launch date was found to be 22 Oct 05, 295 days earlier than the reference launch date.

[i] spacecraft bus plus scientific payload



(a) Trajectories for five different initial populations

(b) Best InTrance-trajectory

Figure 9. 1996FG$_3$ **rendezvous (optimized launch date)**

The third experiment was to find out whether a given C_3 of 4 km^2/s^2 could be spent more efficiently than done by the reference trajectory. The optimal launch date for this problem was found to be 12 Feb 06, a half year earlier than the reference launch date. Figure 10 shows the best found trajectory. It takes only 944 days to rendezvous 1996FG$_3$, being 696 days (74%) faster than the reference trajectory.

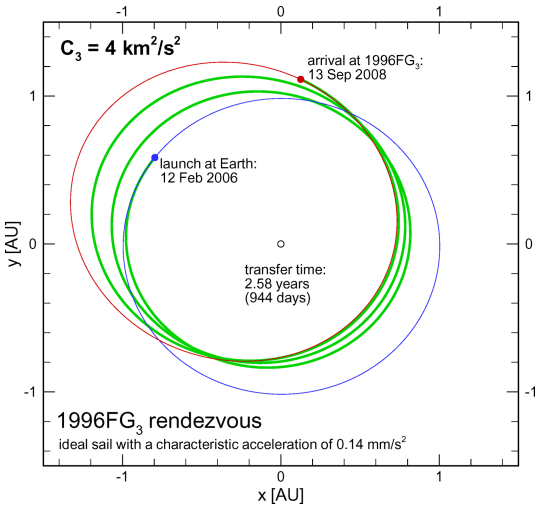
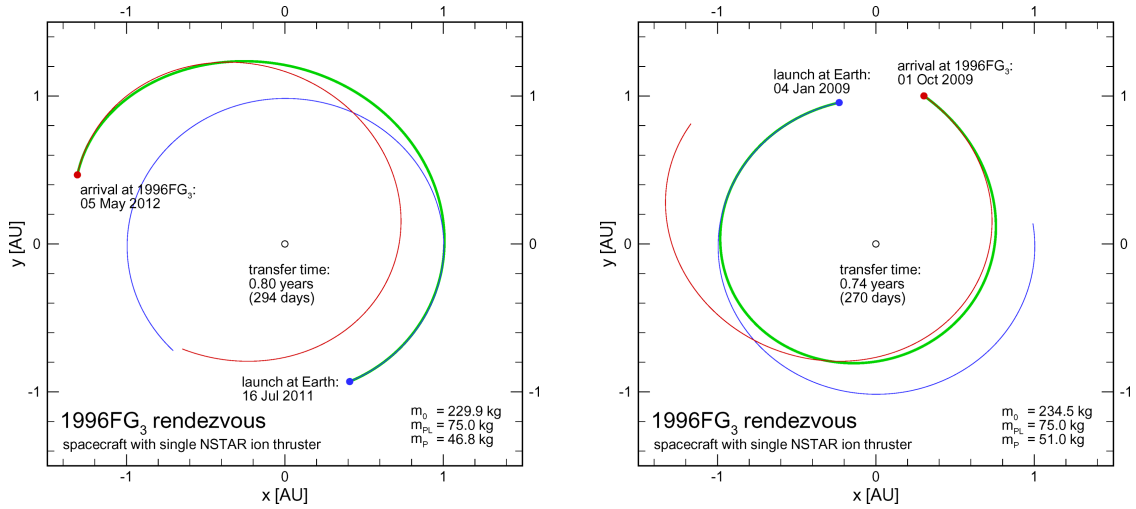


Figure 10. 1996FG$_3$ **rendezvous (optimized launch date): Best InTrance-trajectory for** $C_3 = 4\,\mathrm{km}^2/\mathrm{s}^2$

To assess the trajectory optimization capability of ENC also for low-thrust propulsion systems other than solar sails, and to assess the capability of near-term solar sail propulsion for this NEA rendezvous mission, InTrance was used to calculate trajectories for spacecraft with an already existing SEP system, NASA's NSTAR ion thruster, which was flown on the Deep Space 1 mission. The mission objective was as for the solar sail: deliver a useful mass of 75 kg to 1996FG$_3$. In contrast to solar sailcraft trajectory optimization, SEP spacecraft trajectories may not only be optimized with respect to transfer time but also with respect to the required propellant mass, and usually a trade-off between both optimization objectives has to be made, so that an optimal solution is only one of many (Pareto-)optimal solutions. Figure 11 exemplifies two InTrance-solutions for this problem. Using an NSTAR thruster, the same useful mass of 75 kg could be delivered to 1996FG$_3$ within 294 days (with a propellant mass of 46.8 kg) or even within 270 days, if slightly more propellant (51.0 kg) is consumed. The results demonstrate that, at least for this mission, a near-term solar sail is outperformed by the SEP option, if only the transfer time is considered. This is not surprising because the required ΔV for the transfer is moderate. The launch mass of the SEP option, however, is larger (229.9 kg and 234.5 kg, respectively) than for the solar sail option (148.0 kg), requiring eventually a heavier and more expensive launch vehicle. If ground operation costs can be kept low (e.g., due to a high on-board autonomy during transfer), and if the transfer time plays a subordinate role with respect to cost, the solar sail might be the favorable option for such a mission. In any case, on the way to more advanced solar sails, as they are required for high-ΔV missions, the development of near-term solar sails is an indispensable first stepping stone, even if their current performance is not superior to that of state-of-the-art electric propulsion systems.



(a) $m_{\mathrm{P}} = 46.8\,\mathrm{kg}$ (b) $m_{\mathrm{P}} = 51.0\,\mathrm{kg}$

Figure 11. Trajectory options for a $1996\mathrm{FG}_3$ rendezvous with SEP spacecraft

VI. Summary and Conclusions

Within this paper, low-thrust trajectory optimization was attacked from the perspective of machine learning. Inspired by natural archetypes, a novel method for spacecraft trajectory optimization was proposed that fuses artificial neural networks and evolutionary algorithms into evolutionary neurocontrollers. This method was termed InTrance, which stands for **In**telligent **Tra**jectory optimization using **n**euro**c**ontroller **e**volution. From the perspective of machine learning, a trajectory is regarded as the result of a spacecraft steering strategy that manipulates the spacecraft's thrust vector according to the current state of the spacecraft and the target. An artificial neural network is used as a so-called neurocontroller to implement such a spacecraft steering strategy. This way, the trajectory is defined by the internal parameters of the neurocontroller. An evolutionary algorithm is used to find the optimal network parameters. The trajectory optimization problem is solved if the parameter set that generates the optimal trajectory is found. Using an evolutionary algorithm for the optimization of the neurocontroller parameters, this algorithm may be also used to find the optimal initial conditions for the mission.

Within this paper, InTrance was applied to two interplanetary low-thrust trajectory optimization problems, a Mercury rendezvous and a near-Earth asteroid rendezvous. For both missions, reference trajectories had been found in the literature. The re-calculation of the reference problems revealed that those trajectories, which have been generated using a local trajectory optimization method, are quite far from the global optimum. Using InTrance, the transfer times could be reduced considerably. Because evolutionary neurocontrollers explore the trajectory search space more exhaustively than a human expert can do by using traditional optimal control methods, they are able to find spacecraft steering strategies that generate better trajectories, which are closer to the global optimum. For mission feasibility analysis, the obtained InTrance-trajectories are sufficiently accurate with respect to the terminal constraints. If a more accurate solution is required, the InTrance-solution might be used as an initial guess for some local trajectory optimization method. Unlike the traditional methods, InTrance runs without an initial guess and does not require the attendance of an expert in astrodynamics and optimal control theory. Being problem-independent, the application field of evolutionary neurocontrol may be extended to a variety of other optimal control problems.

References

[1]Vasile, M., "A Global Approach to Optimal Space Trajectory Design," AAS/AIAA Space Flight Mechanics Meeting, Ponce, Puerto Rico, February 2002, AAS 03-141.

[2]Stengel, R., *Optimal Control and Estimation*, Dover Books on Mathematics, Dover Publications, Inc., New York, 1994.

[3]Coverstone-Carroll, V., Hartmann, J., and Mason, J., "Optimal multi-objective low-thrust spacecraft trajectories," *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, 2000, pp. 387–402.

[4]Dachwald, B., *Low-Thrust Trajectory Optimization and Interplanetary Mission Analysis Using Evolutionary Neurocontrol*, Doctoral thesis, Universität der Bundeswehr München; Fakultät für Luft- und Raumfahrttechnik, 2004.

[5]Keerthi, S. and Ravindran, B., "A Tutorial Survey of Reinforcement Learning," Tech. rep., Department of Computer Science and Automation, Indian Institute of Science, Bangalore, 1995.

[6]Sutton, R. and Barto, A., *Reinforcement Learning*, MIT Press, Cambridge, London, 1998.

[7]Rojas, R., "Was können neuronale Netze?" *Mathematische Aspekte der angewandten Informatik*, edited by R.-H. Schulz, Wissenschaftsverlag, Mannheim, 1994, pp. 55–88, (in German).

[8]Dracopoulos, D., *Evolutionary Learning Algorithms for Neural Adaptive Control*, Perspectives in Neural Computing, Springer, Berlin, Heidelberg, New York, 1997.

[9]Whitley, D., Dominic, S., Das, R., and Anderson, C., "Genetic Reinforcement Learning for Neurocontrol Problems," *Machine Learning*, Vol. 13, 1993, pp. 259–284.

[10]Tsinas, L. and Dachwald, B., "A combined Neural and Genetic Learning Algorithm," *Proceedings of the* 1st *IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, 27–29 June 1994, Orlando, USA*, Vol. 2, IEEE, Piscataway (NJ), USA, 1994, pp. 770–774.

[11]Yao, X., "Evolutionary Artificial Neural Networks," *Encyclopedia of Computer Science and Technology*, edited by A. Kent et al., Vol. 33, Marcel Dekker Inc., New York, 1995, pp. 137–170.

[12]Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, Heidelberg, New York, third, revised and extended ed., 1999.

[13]Bäck, T., "Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms," First IEEE Conference on Evolutionary Computation, Orlando, USA, 1994.

[14]Whitley, D., Mathias, K., and Fitzhorn, P., "Delta coding: An iterative search strategy for genetic algorithms," Fourth International Conference on Genetic Algorithms, San Mateo, USA, 1991.

[15]Dachwald, B., *Optimierung des Lernverhaltens neuronaler Netze mit Hilfe genetischer Algorithmen*, Diploma thesis, Universität der Bundeswehr München, Fakultät für Luft- und Raumfahrttechnik, Institut für Meßtechnik, July 1993, (in German).

[16]Montana, D. and Davies, L., "Training Feedforward Neural Networks Using Genetic Algorithms," 1989 Joint Conference on Artificial Intelligence, Los Altos, USA, 1989.

[17]Syswerda, G., "Uniform Crossover in Genetic Algorithms," Third International Conference on Genetic Algorithms, San Mateo, USA, 1989.

[18]Leipold, M., *Solar Sail Mission Design*, Doctoral thesis, Lehrstuhl für Flugmechanik und Flugregelung; Technische Universität München, 1999, DLR-FB-2000-22.

[19]Leipold, M., Seboldt, W., Lingner, S., Borg, E., Herrmann, A., Pabsch, A., Wagner, O., and Brückner, J., "Mercury Sun-Synchronous Polar Orbiter with a Solar Sail," *Acta Astronautica*, Vol. 39, No. 1-4, 1996, pp. 143–151.

[20]Dachwald, B., "Optimization of Interplanetary Solar Sailcraft Trajectories Using Evolutionary Neurocontrol," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 1, pp. 66–72.

[21]Jessberger, E., Seboldt, W., Glassmeier, K.-H., Neukum, G., Pätzold, M., Arnold, G., Auster, H.-U., deNiem, D., Guckenbiehl, F., Häusler, B., Hahn, G., Hanowski, N., Harris, A., Hirsch, H., Kührt, E., Leipold, M., Lorenz, E., Michaelis, H., Möhlmann, D., Mottola, S., Neuhaus, D., Palme, H., Rauer, H., Rezazad, M., Richter, L., Stöffler, D., Willnecker, R., Brückner, J., Klingelhöfer, G., and Spohn, T., "ENEAS – Exploration of Near-Earth Asteroids with a Sailcraft," Tech. rep., August 2000, Proposal for a Small Satellite Mission within the Space Sciences Program of DLR.

[22]Seboldt, W., Leipold, M., Rezazad, M., Herbeck, L., Unkenbold, W., Kassing, D., and Eiden, M., "Ground-based Demonstration of Solar Sail Technology," 51st International Astronautical Congress, Rio de Janeiro, Brazil, 2000, IAF-00-S.6.11.