Carnelli, L. and Dachwald, Bernd and Vasile, Massimiliano (2009)

This version is available at https://strathprints.strath.ac.uk/31138/

Strathprints is designed to allow users to access the research output of the University of Strathclyde. Unless otherwise explicitly stated on the manuscript, Copyright © and Moral Rights for the papers on this site are retained by the individual authors and/or other copyright owners. Please check the manuscript for details of any other licences that may have been applied. You may not engage in further distribution of the material for any profitmaking activities or any commercial gain. You may freely distribute both the url (https://strathprints.strath.ac.uk/) and the content of this paper for research or private study, educational, or not-for-profit purposes without prior permission or charge.

Any correspondence concerning this service should be sent to the Strathprints administrator: strathprints@strath.ac.uk
Evolutionary Neurocontrol: A Novel Method for Low-Thrust Gravity-Assist Trajectory Optimization

Ian Carnelli
ESA, 2001 AZ Noordwijk, The Netherlands
Bernd Dachwald
DLR, German Aerospace Center, 82234 Oberpfaffenhofen, Germany
and
Massimiliano Vasile
University of Glasgow, Glasgow, Scotland G12 8QQ, United Kingdom

The combination of low-thrust propulsion and gravity assists to enhance deep-space missions has proven to be a remarkable task. In this paper, we present a novel method that is based on evolutionary neurocontrollers. The main advantage in the use of a neurocontroller is the generation of a control law with a limited number of decision variables. On the other hand, the evolutionary algorithm allows one to look for globally optimal solutions more efficiently than with a systematic search. In addition, a steepest-ascent algorithm is introduced that acts as a navigator during the planetary encounter, providing the neurocontroller with the optimal insertion parameters. Results are presented for a Mercury rendezvous with a Venus gravity assist and for a Pluto flyby with a Jupiter gravity assist.

Nomenclature

\[ \dot{r} = \text{sun–spacecraft unit vector} \]
\[ g_0 = \text{Earth’s standard gravitational acceleration} \]
\[ I_{sp} = \text{specific impulse} \]
\[ J = \text{fitness function} \]
\[ m_p = \text{propellant mass flow} \]
\[ N = \text{network function} \]
\[ R = \text{rotation matrix} \]
\[ S = \text{spacecraft steering strategy} \]
\[ T = \text{thrust vector} \]
\[ \dot{t} = \text{thrust unit vector} \]
\[ u = \text{spacecraft control vector} \]
\[ v_\infty = \text{hyperbolic excess velocity} \]
\[ x_{1/6} = \text{spacecraft state (r, v)} \]
\[ \alpha = \text{sphere of influence scaling factor} \]
\[ \gamma = \text{thrust clock angle} \]
\[ \Delta = \text{aiming point distance on the B plane} \]
\[ \delta = \text{thrust cone angle} \]
\[ i = \text{orbit inclination} \]
\[ \mu = \text{gravitational constant} \]
\[ n = \text{number of reproductions} \]
\[ \xi = \text{chromosome/individual} \]
\[ \Xi = \text{population of chromosomes/individuals/pilots} \]
\[ \pi = \text{network internal parameters vector} \]
\[ \chi = \text{throttle factor [0, 1]} \]
\[ \Psi = \text{evolutionary neurocontroller} \]

Subscripts

\[ p_l = \text{assisting planet} \]
\[ 1 = \text{before the gravity-assist maneuver} \]
\[ 2 = \text{after the gravity-assist maneuver} \]

Superscript

\[ \ast = \text{optimal} \]

I. Introduction

Gravity assists (GAs) have proven to be the key to interplanetary high-energy missions. They not only make missions feasible that would otherwise be impossible due to large propellant mass fractions, but flybys also make missions more interesting for the scientific community. Additionally, low-thrust (LT) propulsion systems make interplanetary missions more efficient and more flexible, allowing larger launch windows. Hence, the combination of low-thrust propulsion and gravity assists (LTGAs) provides an excellent means to reduce propellant mass-fraction requirements.

However, the design of such trajectories is a nontrivial task. The spacecraft control function on low-thrust arcs is a continuous function of time, and therefore the dimension of the solution space is infinite. The problem is further complicated by considerations of the planet’s phasing, especially when multiple gravity assists are sought. Finally, preliminary analysis tools such as the Tisserand plane or Lambert’s method are not applicable to LT trajectories.

The complexity of the solution of a multiple-LTGA (MLTGA) problem derives from the complexity of the simpler multiple-gravity-assist (MGA) problem. It should be noted that this complexity is not simply due to the hybrid nature of the MLTGA problem. It can be easily shown that even formulating the MGA problem in a homogenous fashion, with continuous decision variables as in the fixed-sequence case, it retains its inherent complexity. This is due to the following reasons: the MGA problem presents a high number of local minima, this number grows with the number of gravity maneuvers, the number of minima further increases if multiple revolutions or deep-space maneuvers are inserted between two subsequent planetary encounters, and the most interesting solutions are generally nested (i.e., their basin of attraction is narrow...
and generally falls within or near a wider basin of attraction of a worse solution.

Some of these reasons are related to the mathematical formulation of the problem and can be mitigated with an appropriate approach. For example, if no deep-space maneuvers are considered and the GA maneuvers are modeled through a powered-swingby model, it can be proved that the MGA problem is solvable incrementally in polynomial time, with a small exponent, through a simple branch-and-prune procedure. Others are instead intrinsically related to the physical nature of the problem. In fact, in a MGA trajectory, the outgoing leg from a GA maneuver is highly sensitive to the incoming conditions. This sensitivity narrows down the size of the basin of attraction in comparison with a direct transfer. Furthermore, the ratio between the orbital periods of the planets tends to destroy any periodicity or symmetry in the solution space.

Despite what could seem intuitive, adding low-thrust arcs to a MGA trajectory does not change the global nature of the solution space. The main reason is that low-thrust arcs (in particular, when the solution is optimal) are only locally shaping each trajectory leg, through the entry conditions to a GA maneuver at a lower cost than deep-space maneuvers. This is true unless a multispiral trajectory is inserted between two gravity maneuvers. In this case, low-thrust arcs increase the complexity of the problem. Furthermore, if deep-space maneuvers and an accurate gravity-assist model are considered, the number of possible states for an MGA trajectories grows exponentially with the number of GA maneuvers, even for a fixed sequence. The MLTGA problem therefore inherits the complexity of the MGA problem and adds the local solution of an optimal control problem.

Traditionally, the design and analysis of interplanetary LT trajectories undergoes three main steps:

1) The main objectives are selected and the sequence of flyby bodies is outlined.
2) Possible preliminary trajectories are analyzed.
3) The optimal trajectory is calculated.

The first two steps, typically carried out by an expert in trajectory optimization, generate the initial guess for the third optimization phase. Unfortunately, this approach is typically quite inefficient, as each individual problem has to be solved from scratch and many solutions have to be explored before convergence of the optimization method is achieved. Finally, even if a solution is obtained, in most cases it represents a local optimum far from the global optimal solution. Recent studies have attacked the problem using either deterministic (such as branch and prune) or stochastic global optimization methods (such as evolutionary algorithms).

Though the use of stochastic-based methods allows the treatment of high-dimensional problems that are otherwise intractable with deterministic problems, modeling the control law through either an indirect approach or through direct collocation could not be the most efficient choice in the preliminary phase of the design.

It is proposed here to use evolutionary neurocontrollers (ENCs) instead, which have proven to be able to generate very good solutions to quite complex problems in an effective and efficient way. The ability of ENC's to find, for example, optimal solar photonic assist trajectories to reach the outer solar system with a solar sail was demonstrated by Dachwald. The proposed ENC's are very flexible because they perform a broad search of the solution space without any special requirement on the regularity of the optimization function and of the constraints. They also allow one to accommodate the cooptimization of additional problem parameters (e.g., launch date, hyperbolic excess velocity, etc.).

If we consider the usual classification of the methods for trajectory design, the evolutionary neurocontroller would fall in the class of direct approaches in particular, because the dynamics here are propagated forward in time, in the class of direct shooting methods. This basic classification, however, does not help to understand the essence of the evolutionary neurocontroller. The main difference with respect to the other direct methods is that the neurocontroller generates a time-dependent control law, as for indirect methods, with a low number of decision variables. The advantage can be understood through a simple example: If a multispiral trajectory were to be designed and the number of spirals were not known a priori, a direct shooting methods based on a collocation of the controls would require an increase of the number of decision variables as the number of spirals increases to have a good resolution of the control profile. A neurocontroller would instead require a constant, and small, number of control variables. We can say that as neural networks can be used to map a generic nonlinear function, the neurocontroller can also be used to map a generic controller.

In this paper, we present a novel method to design LT trajectories that is based on ENCs. This new tool is an extension of InTrance (Intelligent Trajectory Optimization Using Neurocontroller Evolution), the global LT trajectory-optimization tool developed by Dachwald.

II. Simulation Models

Some general assumptions are used throughout this paper to simplify the models and to relax the computational effort:

1) Along a low-thrust arc, the spacecraft is subject to the gravity attraction of the sun and to the control acceleration of the engine only. Gravity-assist maneuvers are inserted between two low-thrust arcs as hyperbolic trajectories in the planetocentric reference frame. Finally, along the gravity-assist hyperbola, the spacecraft is subject to the gravity attraction of the GA planet only.
2) The magnitude and direction of the spacecraft’s thrust vector can be achieved instantaneously.
3) The spacecraft systems (e.g., solar arrays, electric thrusters, etc.) do not degrade over time.

InTrance implements several solar sail models, two solar electric propulsion (SEP) systems (NASA’s NSTAR thruster and QinetiQ’s T6 thruster), and a nuclear electric propulsion (NEP) system. The preliminary results presented in this paper make use of the NEP system for the Pluto flyby mission and the (NSTAR) SEP system for the Mercury rendezvous mission.

A. NEP System Model

In this model, the maximum thrust $T_{\text{max}}$ and the specific impulse $I_{sp}$ are assumed to be fixed. The maximum propellant mass flow rate $\dot{m}_{p,\text{max}}$ (required to generate $T_{\text{max}}$) is

$$\dot{m}_{p,\text{max}} = \frac{|T_{\text{max}}|}{|v_\infty|} I_{sp} g_0$$

where $v_\infty$ is the exhaust velocity and $g_0$ is Earth’s standard gravitational acceleration. A throttle factor $0 \leq \chi \leq 1$ is used to control the propellant mass flow rate, so that

$$\dot{m}(\chi) = \chi \dot{m}_{p,\text{max}}$$

and

$$T(\chi) = \chi T_{\text{max}}$$

B. SEP System Model

The key parameter for a SEP system is the input power $P_{\text{PPU}}$ that is available to the power processing unit (PPU). This power is proportional to that delivered by the solar arrays $P_{SA}$ (which is $\sim 1/r^2$, where $r$ represents the solar distance). The available power $P_{\text{PPU}}$ must also take into account the power required to operate the spacecraft’s systems $P_{\text{SYS}}$.
where $P_{\text{min}}$ and $P_{\text{max}}$ represent the minimum and maximum power at which the propulsion system can operate (respectively, 0.5 and 2.0 kW). According to Williams and Coverstone-Carroll [12], the following polynomial approximation for the propellant mass flow rate (in milligrams/second) and thrust (in millinewtons) can be used in the power range $P_{\text{min}} \leq P_{\text{PPU}} \leq P_{\text{max}}$:

$$
\dot{m}_p(\chi, r) = 0.74343 + 0.20951 P_{\text{PPU}} (1 \text{ kW}) + 0.25205 P_{\text{PPU}}^2 (1 \text{ kW}^2)
$$

The minimum and maximum mass flow rate and thrust are then

$$
\dot{m}_{p,\text{min}} = 0.9112 \text{ mg/s} \quad \dot{m}_{p,\text{max}} = 2.1707 \text{ mg/s}
$$

$$
T_{\text{min}} = 15.251 \text{ mN} \quad T_{\text{max}} = 71.298 \text{ mN}
$$

Finally, the expression for the specific impulse is

$$
I_{sp}(\chi, r) = \frac{|T(\chi, r)|}{\dot{m}_p(\chi, r) g_0}
$$

### III. Equations of Motion for EP Spacecraft

When a spacecraft employs chemical thrusters to generate the required $\Delta V$, the maneuvers can be considered to be impulsive, due to the high level of thrust and consequently short burning times. However, if a spacecraft uses electric propulsion, the burning times are comparable with the spacecraft’s orbital period, and hence the thrust has to be included within the equations of motion. Therefore, the differential equations system is

$$
\dot{\mathbf{r}} = \mathbf{v} \quad \dot{\mathbf{v}} = -\frac{\mu}{r^2} \mathbf{e}_r + \frac{\mathbf{T}}{m_{\text{SC}}}
$$

where $\mathbf{e}_r$ is the unit vector pointing from the attracting body $B$ to the spacecraft, and $\mu = GM_B$ is the gravitational constant. Using Eq. (3), one gets for NEP

$$
\dot{\mathbf{r}} = \frac{T_{\text{max}}}{m_{\text{SC}}} \mathbf{e}_r - \frac{\mu}{r^2} \mathbf{e}_r
$$

### IV. Gravity-Assist Model

Labusky et al. [13] proposed an analytical method that allows an approximate computation of gravity assists. We have implemented this method into InTrance to avoid direct numerical integration of the equations of motion within the assisting body’s sphere of influence (SOI). This way, the required computation time is reduced considerably. Nevertheless, this method provides a quite accurate description of the gravity-assist maneuver.

Given the spacecraft state $x_{sc} = (r_{sc}, v_{sc})$ on the SOI before the gravity-assist maneuver, the purpose of this method is to provide the spacecraft state on the planet’s SOI after the gravity assist. The spacecraft’s position and velocity in the heliocentric reference system $\{R_1, V_1\}$ when entering the SOI, as well as those of the planet $\{R_{pl,1}, V_{pl,1}\}$, are supposed to be known (see Fig. 3). According to the SOI approximation, the coordinate system is changed into the planetocentric reference frame. The spacecraft’s state thus becomes

$$
\mathbf{r}_1 = \mathbf{R}_1 - \mathbf{R}_{pl,1}
$$

$$
\mathbf{v}_1 = \mathbf{V}_1 - \mathbf{V}_{pl,1}
$$

where $|\mathbf{R}_1| = R_{SOI,pl}$, the subscript 1 indicates a variable before the gravity-assist maneuver, and the subscript pl is used to indicate a variable of the assisting planet. From the planetocentric point of view, the spacecraft approaches from infinity with a nonzero velocity. Therefore, the trajectory is represented as hyperbolic, and the outbound velocity and position vectors result from a rotation around the angular momentum vector $\mathbf{h}_{sc}$ with respect to the planet (see Fig. 3). The outbound state of the spacecraft is then obtained using

$$
\begin{bmatrix}
  x_2 \\
  y_2 \\
  z_2
\end{bmatrix} = \mathcal{R}(\phi) \begin{bmatrix}
  x_1 \\
  y_1 \\
  z_1
\end{bmatrix}
$$

$$
\begin{bmatrix}
  v_{x2} \\
  v_{y2} \\
  v_{z2}
\end{bmatrix} = \mathcal{R}(\psi) \begin{bmatrix}
  v_{x1} \\
  v_{y1} \\
  v_{z1}
\end{bmatrix}
$$

where $\phi$ is the rotation angle of the position vector, $\psi$ is the rotation angle of the velocity vector, and $\mathcal{R}$ is a rotation matrix. The complementary angles are given by

$$
\phi = \pi + \phi - 2\gamma \quad \psi = 2\delta - 2\arctan \frac{\mu}{\Delta v_{\infty}}
$$

where

$$
\gamma = \pi - \arccos \frac{\mathbf{R}_1 \cdot \mathbf{V}_1}{|\mathbf{R}_1||\mathbf{V}_1|}
$$

and $v_{\infty} = \sqrt{v_1^2 - 2\mu/r_1}$ represents the hyperbolic excess velocity. Finally, $\Delta$ is the aiming point distance defined as the distance between the center of the planet and the inbound asymptote.
The ENC architecture used in this work is sketched in Fig. 3. This particular ENC design reflects its application to solve LT trajectories optimization problems. To do so, the ENC runs in two loops. Within the inner trajectory-integration loop, a NC steers the spacecraft according to its network function \( N \) provided by the EA (that runs in the outer loop). The NC’s parameter set \( \pi \), which represents an individual, is therefore constant during integration. It can be thought of as a pilot who steers the spacecraft until the termination condition is met. That is, the pilot has either reached the final target or a maximum travel time (defined by the user). In the outer optimization loop, the EA holds a population \( \Xi = \{ \pi_1, \pi_2, \ldots, \pi_N \} \) of pilots (or chromosomes/individuals). The EA evaluates all pilots (i.e., NC parameter sets \( \pi_{i(\ell-1:\ell)} \)), one at a time, for their suitability to generate an optimal trajectory. Within the trajectory-optimization loop, the NC takes the actual spacecraft state \( x_{sc}(\ell) \) and that of the target body \( x_{t}(\ell) \) as input values and maps them onto the output values from which the spacecraft control vector \( u(\ell) \) can be calculated. At this point, \( x_{sc}(\ell) \) and \( u(\ell) \) are inserted into the equations of motion, which are numerically integrated over one time step \( \Delta t = t_{\ell+1} - t_{\ell} \), to yield \( x_{sc}(\ell+1) \). This state is fed back into the NC. Again, the trajectory-integration loop stops when the termination condition is met. At this time, the NC’s parameter set (i.e., its trajectory) is rated by the EA’s fitness function \( J(\pi) \). This fitness value is crucial to the reproduction probability of \( \pi \). Under the selection pressure of the environment, the EA breeds NCs that, in turn, generate increasingly better trajectories. The EA that is used within InTrance finally converges to a single steering strategy that gives, in the best case, a globally optimal trajectory \( x_{opt}^t[\ell] \).

**B. Objective Function**

The optimality of a trajectory can be defined with respect to different (primary) objectives such as transfer time or propellant consumption. When an ENC is used for trajectory optimization, the trajectory accuracy to the terminal constraints must also be considered as a secondary optimization objective, as they are not explicitly stated elsewhere. By defining a maximal allowed distance \( \Delta r_{\text{f}, \text{max}} \) and relative velocity \( \Delta v_{\text{r}, \text{max}} \) at the target, the trajectory accuracy can be defined as follows with respect to the terminal constraints:

\[
\Delta X_{\ell} = \sqrt{\frac{1}{2}(\Delta r_{\text{f}, \text{max}}^2 + \Delta v_{\text{r}, \text{max}}^2)}
\]  

Fig. 3 From chromosome to optimal trajectory.
Therefore, to minimize the transfer time \( T \) for a rendezvous, the following function is used:

\[
J(T, \Delta r_f, \Delta v_f) = J_f + \frac{1}{c_4 + (1 - c_4)\Delta X_f} \tag{25}
\]

A tournament-selection scheme has been used for the selection of the individuals from the population \([11]\). This way, the selection probability does not depend on the scaling of the fitness function. The value chosen for \( c_4 \) guarantees that once the final constraints are fulfilled, improvements in the primary optimization objective are rated much higher than improvements in the fulfillment of the final constraints (but if two individuals have the same flight time or propellant consumption, the more accurate one is always preferred).

In the case of a planetary flyby, only the constraint on the position must be met, whereas the final velocity is set free. If the transfer time is to be minimized in this case, then

\[
J(T, \Delta r_f) = J_m + \frac{1}{c_4 + (1 - c_4)\Delta R_f} \tag{26}
\]

To minimize the propellant mass for a flyby, \( J_f \) is replaced with \( J_{m_f} \):

\[
J(\Delta m_p, \Delta r_f) = J_{m_f} + \frac{1}{c_4 + (1 - c_4)\Delta R_f} \tag{27}
\]

VI. ENC Architecture Options

Two different approaches to solve the LTGA problem are straightforward. In one case, the whole trajectory (i.e., the inbound leg from the departure point to the flyby body and the outbound leg from the flyby body to the target body) is optimized by a single NC. This choice is quite straightforward, as it requires only the input to the ANN and the chromosome length to be modified, but it preserves the structure of the EA operators. However, in such an approach, there is no objective evidence that the ENC will actually seek for gravity-assist maneuvers and exploit them to gain orbital energy (only some expectations based on the ENC ability to perform solar photonic assist for solar sail trajectories) [12].

A second approach instead requires the use of multiple ENCs. Here, each ENC optimizes only a single leg of the trajectory. In other words, the mission is divided into sequential phases: the first phase starts from the departure body and ends at the SOI of the first flyby body. The gravity-assist maneuver at this moment is computed analytically to give the outbound conditions (subscript 2), then the second leg is optimized by a second ENC until the next SOI or the final target body is reached. This reasoning can be extended to \( n + 1 \) ENCs \((\Psi_{j=1:n+1})\) performing \( n \) gravity-assist maneuvers. At first sight, this second approach seems very promising because InTrance has proven to be capable of finding optimal \( \Psi_{j=1:n+1} \). By creating new evolutionary operators and providing a flyby sequence, one can be quite confident of finding the optimal overall trajectory for this sequence \( \Psi_{j=1:n+1} \). Although the latter approach is quite attractive, it fails in one particular point that is the goal of this work: designing a fully automatic tool that does not require the attendance of a trajectory-optimization expert. If the flyby sequence were to be given as an input to InTrance, either an expert or a second ad hoc algorithm would have to provide it. For this reason, the single-ENC approach has been chosen.

VII. Gravity-Assist Optimization

Within the single-ENC approach, the first step is to provide the ENC with the appropriate input information. This corresponds to the knowledge the ENC should have to optimally steer the spacecraft and eventually perform gravity assists. In addition to the state of the spacecraft and the final target, the ENC should also know where the possible gravity-assist bodies are and how they are moving. In other words, the NC’s inputs must include additional information such as \( x_{nc} = x_{pl} \).

The second step is to introduce the analytical gravity-assist calculator into the inner trajectory-integration loop of the ENC. Here, a patched-conics approximation is used: at every integration step, a...
check is performed to determine whether the spacecraft has entered
the SOI of a gravity-assist body. If this is the case, the maneuver is
calculated analytically and the outbound conditions (subscript 2) are
used as initial conditions for the successive numerical integration,
which is carried out until the termination condition is met.

Finally, the third step corresponds to the gravity-assist maneuver
optimization. To do so, the $B$ plane, which can be thought of as a
target attached to the assisting body, is taken as reference system.

The topology of the solution space for the optimal planetary
maneuver was assessed by plotting the fitness values corresponding
to a large-enough number of targeting points $(\tilde{x}, \tilde{y})$ on the $B$ plane
for one exemplary gravity assist. Once a grid is defined on the plane,
some virtual spacecraft were used to probe the solution space by
forcing them to aim at each grid point. This way, a sufficient subset
of all possible maneuvers was tested and rated with respect to the fitness
function. Following this procedure, a plot can be made in which the
solution space for this exemplary gravity assist is particularly smooth
and regular (here, the Venus gravity assist of the test case in Sec. VII
is used).

For this exemplary gravity assist, the two extremes (i.e., the
absolute maximum and the absolute minimum) are very well defined,
and only a local minimum and maximum are present on the boundary
of the SOI, because of the slight inclination of the plane
along the positive $x$ axis. A few characteristics of the solution space
are noteworthy:

1) As the gravity-assist geometry is incorporated in the
controller’s steering strategy (that is fixed once a controller
is selected because its parameters are constant), each ENC
can only fly one of the many possible GA geometries. Thus, the $B$
plane corresponds to a fixed point in space with a frozen incoming
velocity vector (only the thrust profile may vary before and after the
GA, but not the geometry of the flyby) characterized by a solution
space with a single optimum.

2) The two global extremes are symmetrically located with respect
to the origin and separated by about the diameter of the planet. There
is in fact only one semi-$B$ plane that bends the spacecraft trajectory
toward the target body, hence boosting the spacecraft in the right
direction. Also, the two extremes are close to the flyby body in which
its gravitational field is strongest and the trajectory’s curvature is
most affected.

3) The solution space in the outer regions is almost flat. This is also
an expected result because the maneuvers performed close to the SOI
boundaries (outer region of the equatorial circular plane in Fig. 5)
have small effect on the trajectory.

4) No other hills or valleys are found in the domain. This suggests
adopting a local optimization method rather than a global optimizer
for the maneuver, for the sake of saving computing time. This simple
structure of the solution space, however, might not generally be the
case. Especially if multiple gravity assists are involved, a global
method might offer advantages.

5) In this particular case, the fitness values are in the range
$0 < J < 0.9$ (note that $J \geq 1$ corresponds to fulfillment of the final
constraints), but the values close to the optimum are located in a very
small region of the domain space.

Following the preceding analysis, we have chosen a steepest-
ascend method as the optimization algorithm. The goal is to find the
highest fitness-function value: that is, the highest reward $J$ that can be
attributed to the navigator. Because local optima are present only in
the outer regions of the SOI, there is no risk of converging to a local
optimal solution if the initial point $(x_i, y_i)$ is chosen to be close
enough to the planet. The strategy, which consists of following the
path along the steepest gradient, is illustrated in Fig. 3. Using this
simple algorithm, an efficient navigator for the ENC is designed. The
best gravity-assist maneuver $(x, y)^*$ is located within a few iterations
(see Fig. 5) and is so efficient that the fitness of individuals that
perform a gravity assist far outweighs the fitness of the individuals
who do not.

The gravity-assist maneuver is performed when the spacecraft
reaches the SOI of an assisting body. Here, the trajectory integration
is stopped and the spacecraft’s velocity vector defines the orientation
of the $B$ plane. At this stage, the actual aiming point $\mathbf{r}_1$ is disregarded
and the coordinate set $(x, y)^*$ optimized by the navigator is used
instead. This point $\mathbf{r}_1$ is projected onto the SOI to provide the initial

---

4 It is a planar coordinate system that contains the focus of the hyperbola
and is perpendicular to the incoming asymptote. The origin of the reference
system is located at the center of the assisting body, the $x$ axis is defined by
the intersection of the $B$ plane with the trajectory plane, and the $y$ axis
is orthogonal to the $x$ axis.

---

**Fig. 5** Two-dimensional plot of $B$-plane fitness topology.

**Fig. 6** Three-dimensional plot of $B$-plane topology.

**Fig. 7** Search algorithm.
position for the analytical gravity-assist calculator. The discontinuity that is introduced is minimized during the optimization process. This is done by introducing the distance $|\mathbf{r}_1 - \mathbf{r}_f|$ in a subfitness function

$$J_{\text{distance}} = \frac{1}{|\mathbf{r}_1 - \mathbf{r}_f|}$$

that is to be maximized once the primary goal is achieved [$J_r \geq 0$ and $J_t \geq 0$, as defined in Eq. (2)]. Finally, the gravity-assist maneuver is computed and the initial conditions for the following trajectory leg are obtained.

Because of the modest size of the SOIs, the chances for an ENC to encounter a sphere of influence are very small. To solve this problem, artificial spheres of influence are introduced to enlarge the real SOI sizes (with the exception of Jupiter and Saturn) by a factor $\alpha > 1$. The idea is to let the ENC learn in an easier environment and gradually raise the difficulty by letting $\alpha \rightarrow 1$ while the number of reproductions $v$ in the population increases. Empirically, the following definition of $\alpha$ was found to provide a good learning environment to the ENC:

$$\alpha = \begin{cases} 
\frac{1000 + v}{1000 + 5000} & \text{if } v < 600 \\
1 & \text{if } v \geq 600
\end{cases}$$

The 600th-generation threshold is chosen to provide the ENC with the real dynamic environment, typically after the primary goals are achieved and before the secondary goals are optimized. Note that the number of generations required is problem-dependent. However, no LTGA optimization problem was found to require less than ~3000 generations.

VIII. Test Cases

The algorithm defined in the previous section proved to be an efficient navigator for the ENC. The idea is to let the ENC (pilot) concentrate only on the trajectory-optimization task and support each pilot with a navigator whom he can consult whenever a SOI is crossed. Two missions are used as a reference to assess the validity of this method: a flyby mission to Pluto with a Jupiter gravity assist and a rendezvous mission to Mercury with a Venus gravity assist. The first mission is somewhat easier due to the very large size of Jupiter’s SOI. The second mission is much more challenging due to the demanding $\Delta V$ requirement to reach Mercury.

A. New Horizon Mission to Pluto

The proposed approach was implemented in an extended version of InTrance called InTrance-GA. To assess its ability to optimize LTGA trajectories, a flyby mission to Pluto calculated by Vasile [3] was chosen as a reference. In that paper, the design of the NEP trajectories was performed with a direct transcription method based on finite elements in time (implemented in a software tool called DITAN [3]). However, because the problem presents a number of possible solutions dependent on the launch window and transfer time, a global optimization strategy was used [8] to generate sets of promising initial guesses. By setting the launch date and escape velocity to those used in the reference and the terminal conditions so that the arrival at Pluto is before 2020, InTrance-GA could generate results that are comparable with the so-called fast transiti option in the reference. This was most probably achieved due to a very short integration step (i.e., one day). Thus, the objective of this simulation was not to optimize the launch conditions and find a global optimum solution, rather to explore the local level of accuracy that can be reached once a solution is found. The results summarized in Table I were obtained by evolving an initial population size of 50 individuals for 1367 generations, requiring a total computation time of 23,795 s and reaching an fitness-function value of $J = 33.77$. As can be seen, there is very good agreement between the two solutions. In particular, the propellant mass consumptions are nearly identical (the NEP model described in Sec. II.A. was used). Note that InTrance-GA was able achieve a very good local convergence so that virtually no refinement of the solution was needed. However, a minimum final flyby distance at Pluto as high as $\Delta r_f = 1.69$ km was reached and could not be further minimized, whereas $J_{\text{distance}} = (1755 \text{ km})^{-1}$ was considered sufficiently low to consider the overall solution to be accurate.

B. Mercury Rendezvous

As a second test case to assess our method’s performance, a rendezvous trajectory to Mercury, as calculated by Debban et al. [17] and De Pascale [18], was chosen. This optimization goal is particularly challenging for several reasons, including Mercury’s proximity to the sun and the eccentricity and inclination of its orbit. Even assuming circular coplanar orbits of both Earth and Mercury, a Hohmann transfer requires a launch $V_{\infty}$ of at least 7.5 km/s, with the resulting arrival $V_{\infty}$ as high as 9.6 km/s [19]. The idea of using chemical propulsion to remove this hyperbolic excess velocity is prohibitive, and hence LTGA techniques offer lower propellant requirements.

In the paper by Debban et al. [17], the whole computational effort is focused on the search for a proper initial guess [using the Satellite Tour Design Program (STOUR)-LTGA, a shape-based method that implements exponential sinusoid functions], which is then optimized using a direct method (implemented in a software tool called GALLOP). The chosen trajectory shape is a two-dimensional curve in polar coordinates. Here, the target’s out-of-plane position at the time of the in-plane encounter is matched by using additional thrust acceleration acting along or against the spacecraft’s angular momentum vector for some final portion of the leg’s thrust arc. By assuming constant $I_{\text{sp}}$, the propellant consumption is then estimated as a fraction of the initial spacecraft mass based on the time integral of the thrust acceleration and the rocket equation. Finally, the effect on the time of flight (TOF) is neglected. Thus, the trajectory along the $z$ coordinate is well approximated only for small inclination changes. An evolution of this method (implemented in a software tool called IMAGO) by De Pascale [18] describes the trajectory by assigning a

<table>
<thead>
<tr>
<th>Table 1 Earth–Jupiter–Pluto mission with NEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DITAN</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Earth launch date</td>
</tr>
<tr>
<td>Launch $V_{\infty}$</td>
</tr>
<tr>
<td>Mass at departure</td>
</tr>
<tr>
<td>Mass at arrival</td>
</tr>
<tr>
<td>$I_{\text{sp}}$</td>
</tr>
<tr>
<td>Thrust</td>
</tr>
<tr>
<td>Jupiter encounter</td>
</tr>
<tr>
<td>Pluto arrival</td>
</tr>
</tbody>
</table>
set of shapes to a set of so-called pseudoequinoctial elements, allowing a fully three-dimensional description. These methods provide interesting results to assess the performance of InTrance-GA. The obtained results, together with the reference values, are shown in Table 2. Here, the SEP model as described in Sec. II.B was used, and the launch conditions were optimized over the following ranges: the launch date was between 15 June 2002 and 15 September 2002 and the launch $V_1$ range was set between 2 and 3 km/s. There is good agreement between InTrance-GA and IMAGO over the launch date, time of flight, and arrival date. The different SEP, trajectory model, and wider range of variability of the initial $V_1$ might instead justify the difference with GALLOP. InTrance-GA was able to rendezvous with Mercury after 2842 generations, requiring a total computation time of 41,484 s, which once again demonstrates the complexity of this rendezvous problem. A final fitness value of $J_{\text{distance}}$ was reached, though the final rendezvous values were somewhat relaxed to $\Delta r_{f,\text{max}} = 0.005 \text{ km}$ and $\Delta v_{f,\text{max}} = 300 \text{ m/s}$. Once again, $J_{\text{distance}}$ proved to be an easier parameter to minimize, and a fairly low value of $(1230 \text{ km})^{-1}$ was reached.

Figure 9 shows the trajectory found by InTrance-GA. The ENC manages to use the gravity-assist maneuver to obtain an initial orbit-inclination change $\Delta i$ (the scale on the z axis is exaggerated for the sake of visibility). The remaining inclination change $\Delta i = i_f - i_i$ is carried out by the spacecraft’s electric propulsion system. In addition to two correction maneuvers, the first immediately after departure and the second slightly before the Venus flyby, all thrust phases work to achieve the required orbit-inclination change.

The thrust and coast arcs can be better seen from Fig. 10, which provides a view from above the ecliptic plane, and from the spacecraft’s throttle level shown in Fig. A1. Note that there are no clear switching points and the thrust is continuously throttled from 0 to its maximum. The solution is therefore expected to be locally suboptimal. Nevertheless, it represents a good first guess for this transfer trajectory.

### IX. Conclusions

We have presented a novel method (termed InTrance-GA) for low-thrust gravity-assist trajectory optimization, which is based on the use of evolutionary neurocontrollers. Based on an enumerative analysis of the solution space in the $B$ plane, a separate algorithm was implemented to optimize the gravity-assist maneuvers. This steepest-ascent algorithm acts as a navigator that the neurocontroller consults whenever it enters the sphere of influence of an assisting planet. Also, we have introduced artificially enlarged spheres of influence, due to their small size as compared with typical trajectory radii. This allows the evolutionary neurocontroller to learn in an easier environment, while gradually raising the difficulty by reducing the size to the real dimensions over time.

Preliminary results for a Pluto flyby with a Jupiter gravity assist and for a Mercury rendezvous with a Venus gravity assist have been presented. They show excellent agreement between the calculated solutions and the reference trajectories. In particular, in one of the
cases, InTrance-GA was able to perform a global search of the solution space in a way that required virtually no further refinement of the solution.

**Appendix A: Gravity-Assist Model**

The rotation matrix $\mathcal{R}$, necessary to calculate the gravity-assist maneuver, is obtained by successive rotations: a first set of rotations transforms the Cartesian coordinate system into the polar orbital reference frame $\mathcal{O}$ defined by the three orthogonal unit vectors $\hat{e}_x$, $\hat{e}_y$, and $\hat{e}_z$, where $\hat{e}_x$ is along the radial sun–spacecraft direction, $\hat{e}_y$ is along the spacecraft’s orbital angular momentum vector, and the orbit transversal vector is $\hat{e}_z = \hat{e}_x \times \hat{e}_y$. The peculiarity of this reference frame is that the $\hat{e}_x$ axis changes constantly along the orbit. In other words, the angle $\xi$ between $\hat{e}_x$ and $\hat{e}_y$ equals the inclination of the orbit only at the nodes, whereas it is null at right angles. The value of this angle can be obtained using the spacecraft velocity vector expressed in polar coordinates $(v_r, v_\theta, v_\phi)$:

$$\xi = \arctan \frac{v_\theta}{v_r}$$  \hspace{1cm} (A1)

Thus, the first set of rotation matrices that transforms from the Cartesian into the polar orbital reference frame is obtained by three subsequent rotations about $\hat{e}_\phi$, $\hat{e}_r$, and $\hat{e}_\theta$. It follows that

$$\mathcal{R}_{\text{pol,orb}} = \mathcal{R}_\phi(\xi_0) \cdot \mathcal{R}_r(-\theta_0) \cdot \mathcal{R}_\theta(\phi_0)$$  \hspace{1cm} (A2)

where $\phi_0$ and $\theta_0$ represent the azimuth and elevation of the spacecraft at the moment it enters the SOI:

$$\phi_0 = \arctan \frac{v_\phi}{v_r} \quad \theta_0 = \arctan \frac{z}{\sqrt{x^2 + y^2}}$$  \hspace{1cm} (A3)

Once the reference frame is set to the polar orbital, the second rotation is the one that actually carries out the maneuver. Hence, the velocity and position vectors are rotated about the orbital angular momentum vector $\hat{e}_z$ by, respectively, angles $\varphi$ and $\psi$ defined by Eq. (A9) through the rotation matrix $\mathcal{R}_\psi$. Finally, a third set of rotations is used to bring the reference system back to Cartesian. These rotation matrices can be easily obtained by transposing Eq. (A2):

$$\mathcal{R}_{\text{can}} = \mathcal{R}_{\text{pol,orb}}^T = (\mathcal{R}_\phi(\xi_0) \cdot \mathcal{R}_r(-\theta_0) \cdot \mathcal{R}_\theta(\phi_0))^T$$  \hspace{1cm} (A4)

All the subsequent rotation matrices have now been defined and the final expression for the overall rotation matrix $\mathcal{R}$ in Eqs. (17) and (18) can be provided:

$$\mathcal{R}(\psi) = \mathcal{R}_{\text{can}} \cdot \mathcal{R}_\psi(\varphi) \cdot \mathcal{R}_{\text{pol,orb}} = \mathcal{R}_\phi(\xi_0) \cdot \mathcal{R}_r(-\theta_0) \cdot \mathcal{R}_\theta(\phi_0) \cdot \mathcal{R}_\psi(\varphi) \cdot \mathcal{R}_\phi(\xi_0) \cdot \mathcal{R}_r(-\theta_0) \cdot \mathcal{R}_\theta(\phi_0)$$  \hspace{1cm} (A5)

$$\mathcal{R}(\psi) = \mathcal{R}_{\text{can}} \cdot \mathcal{R}_\psi(\varphi) \cdot \mathcal{R}_{\text{pol,orb}} = \mathcal{R}_\phi(\xi_0) \cdot \mathcal{R}_r(-\theta_0) \cdot \mathcal{R}_\theta(\phi_0) \cdot \mathcal{R}_\psi(\varphi)$$  \hspace{1cm} (A6)

Using Eqs. (17) and (18) together with Eqs. (A5) and (A6), the spacecraft’s planetocentric Cartesian state after the maneuver is defined. Its components can be added to those of the assisting planet to define the spacecraft’s location and velocity at the moment it leaves the SOI in the heliocentric ecliptic coordinate system:

$$\mathbf{r}_2 = \mathbf{r}_1 + \mathbf{R}_{\text{pl,2}}$$  \hspace{1cm} (A7)

$$\mathbf{v}_2 = \mathbf{v}_1 + \mathbf{V}_{\text{pl,2}}$$  \hspace{1cm} (A8)

where subcript 2 is used to indicate a variable after the gravity assist has occurred (i.e., at time $t + \Delta t_{\text{ga}}$), and $\mathbf{R}_{\text{pl,2}}$ and $\mathbf{V}_{\text{pl,2}}$ can be computed using the osculating elements of the body and the duration of the maneuver. The time of flight for a hyperbolic orbit within the SOI is defined as

$$\Delta t_{\text{fl}} = 2 \sqrt{\frac{a}{\mu}} \left( \sin \delta - \frac{H}{\sin \delta} \right)$$  \hspace{1cm} (A9)

where $H$ is the hyperbolic eccentric anomaly. Its value is determined by the semimajor axis $a$ of the hyperbola, the semirotation angle $\delta$, and the radius $R_{\text{SOI}}$ of the SOI:

$$H = \cosh^{-1} \left( 1 + \frac{R_{\text{SOI}}}{a} \sin \delta \right) \quad a = \left( \frac{|V_j|^2}{\mu} - \frac{2}{R_{\text{SOI}}} \right)^{-1}$$  \hspace{1cm} (A10)

**Acknowledgment**

The authors would like to express their gratitude to Wolfgang Seboldt at DLR, German Aerospace Center, for always finding the time for suggestions and in-depth discussions.

**References**


