On-Line Generation of Quasi-Optimal Docking Trajectories

Riccardo Bevilacqua\textsuperscript{a}, Oleg Yakimenko\textsuperscript{b} and Marcello Romano\textsuperscript{b}

\textsuperscript{a}Me.Mo.Mat. Department, University of Rome La Sapienza, Italy
\textsuperscript{b}Department of Mechanical and Astronautical Engineering
Naval Postgraduate School, Monterey, CA, USA

Abstract

Satellites’ rendezvous and docking is a challenging problem for space exploitation. Saving time, fuel, or any other resource, expressed for example as a combination of the two previous ones, is not a straightforward task if onboard calculation is required to make the control autonomous. A direct method for a rapid generation of near-optimal trajectories of proximity maneuvers onboard a flying vehicle (required to dock a target one), with predetermined thrust history along a master direction, is presented. The new direct method, already implemented and tested onboard for the case of a real aircraft by one of the authors, is based on three concepts: high-order polynomials from the virtual arc as reference functions for the spatial coordinates, preset sequence of a master control, reduction of the optimization problem to the determination of a small set of parameters. By doing this the remaining controls act as slaves, guarantying the chaser to move along the desired path. The master thrust has an on-off structure. Seeking of the optimum strategy is transformed to an nonlinear programming problem, then numerically solved through an ad hoc algorithm in accelerated time scale. Examples are reported in order to prove the rapidness of the approach to generate a sub-optimal docking trajectory.

Abbreviations and Acronyms

DMRP = direct method for rapid prototyping
LVLH = local vertical local horizontal
NLP = nonlinear programming
OP = optimization parameter
PMP = Pontryagin maximum principle

Nomenclature

\( a_k \) = coefficients of reference polynomials
\( g \) = acceleration due to gravity
\( J, J_{sc} \) = performance index, scaled performance index
\( k=\{x,y,z\} \) = Cartesian coordinates in LVLH frame
\( m \) = mass
\( N \) = number of nodes
\( n \) = polynomial order
\( O_{sc} \) = slaves control constraint violation scaling factor
$P_k(\tau) = \text{polynomial reference function for the Cartesian coordinates}$

$T_{\text{max}} = \text{maximum thrust}$

t = \text{time}$

$t^*_T (\tau^*_T) = \text{thrust-off instant (arc)}$

$t^{**}_T (\tau^{**}_T) = \text{thrust-on instant (arc)}$

$t_{\text{scaling}} = \text{time scaling factor}$

$u_x, u_y, u_z = \text{control accelerations along the three axis}$

$V = \text{speed}$

$V_{sc} = \text{velocity scaling factor}$

$w = \text{cost index weighting coefficient}$

$p = \text{penalty weighting coefficient}$

$w_{\text{Overflow}} = \text{slaves control constraint violation penalty weighting coefficient}$

$w_{\text{Speed}} = \text{speed penalty weighting coefficient}$

$x_i, i = 1,3 = \text{spacecraft mass center coordinates in the LVLH frame}$

$\delta_T = \text{throttle position}$

$\Delta, \Delta_{ac} = \text{penalty function, scaled penalty function}$

$\lambda = \text{virtual velocity along the virtual arc}$

$\tau = \text{virtual arc}$

$\Delta \tau, \Delta t_j = \text{sampling period}$

$\Xi = \text{vector of optimization parameters}$

Subscripts and Superscripts

$(.)_j = \text{quantity pertaining to the } j^{th} \text{ time node}$

$(.), (.') , (.''), (.''') = \text{arc derivatives}$

$(.), (\cdot), (\cdot') = \text{time derivatives}$

$(\cdot) = \text{relative parameter}$

I. Introduction

Optimization of spacecraft rendezvous and docking has been widely studied, as shown by the most recent literature. Taking the well known Clohessy-Wiltshire equations ([1]) as reference dynamic for the satellites’ relative flight, different approaches can be found regarding the optimization of the docking maneuvers. In [2] the optimization is performed in two stages: the first part of the trajectory is optimized without any restriction on the chaser vehicle position, the last phase of docking is along a fixed direction. The authors of [3] and Error! Reference source not found. optimize the control sequence to pass from an initially stable formation to a new stable configuration. This approach does not give the possibility of considering generic initial conditions for the relative motion of the chaser. In [5] a
quick method for docking optimization on a predetermined path is introduced, based on a strategy similar to that of [6].

The importance of this research field can be found in several applications, one above all can be the Automated Transfer Vehicle developed by the European Space Agency. On-the-ground experimentation of algorithms and hardware for satellites’ relative control is also under a strong development in the last years ([7]).

The proposed strategy is a direct optimization method already used in for the control of aircraft (see [8] and references therein). The basic idea is to parameterize the trajectory in a way that allows for independent choice of path and velocity profile. The approach permits to reduce the functional problem into an NLP, with a rather small number of optimization parameters. Although this method obviously maintains the main disadvantage of all direct methods: it gives near optimal instead of optimal solution, its proven robustness makes it a good candidate for onboard real-time implementation.

The paper contributing to real-time autonomous control of the satellites’ docking procedure is organized as follows: Section II introduces the reference dynamics, the boundary conditions for the maneuver and the control limits. Section III deals with the optimal control problem statement while Section IV treats the synthesis of the control. Section V reports the reference trajectory through high order polynomials. Section VI introduces the master control arc history and the algorithm. Section VII refers to the states computation, performance index and penalty function. Obtained results can be found in Section VIII. Finally, Section IX concludes the paper.

II. Physical System and Relative Dynamics

The physical system is shown on Fig.1. The LHLV coordinate system attached to the target body frame is used ([1]). This LHLV coordinate system is assumed to be stabilized on a circular orbit around the Earth. It is also assumed that the reaction wheels on the chaser are used to maintain its angular velocity to be equal to that of the target, $\omega_{LHLV}=\omega$, i.e. the chaser does not change its attitude in the LHLV frame. Therefore, the chaser has only three translational degrees of freedom.

Figure 1: Chaser in the local horizontal local vertical coordinate system.
The system of nonlinear equations driving the chaser’s dynamics written in LHLV coordinate frame is shown below:

\[
\begin{align*}
\dot{x} &= V_x \\
\dot{V}_x &= 2\omega V_y + 3\omega^2 x + u_x = f_1(x, V_y) + u_x \\
\dot{y} &= V_y \\
\dot{V}_y &= -2\omega V_x + u_y = f_2(V_x) + u_y \\
\dot{z} &= V_z \\
\dot{V}_z &= -\omega^2 z + u_z = f_3(z) + u_z
\end{align*}
\] (1)

In the vector form system (2) looks as follows:

\[
\dot{\xi} = f(\xi, u).
\] (2)

The six states \(\xi = [x, y, z, V_x, V_y, V_z]^T\) are three coordinates and three components of the velocity vector. The chaser’s relative position can be controlled by multiple thrusters that can produce independent limited control inputs in each direction: \(u_x\), \(u_y\) and \(u_z\) (\(u = [u_x, u_y, u_z]^T\)). The magnitudes of these control inputs are limited by \(U_{\text{max}}\) (by maximum thrust \(T_{\text{max}}\)).

In general it might be required to satisfy the following sets of boundary conditions at the initial and final points:

\[
\begin{align*}
x(t_0) &= x_0 & \dot{x}(t_0) &= \dot{x}_0 &= V_{x0} & \ddot{x}(t_0) &= \ddot{x}_0 \\
y(t_0) &= y_0 & \dot{y}(t_0) &= \dot{y}_0 &= V_{y0} & \ddot{y}(t_0) &= \ddot{y}_0 \\
z(t_0) &= z_0 & \dot{z}(t_0) &= \dot{z}_0 &= V_{z0} & \ddot{z}(t_0) &= \ddot{z}_0 \\
x(t_f) &= x_f & \dot{x}(t_f) &= \dot{x}_f = V_{xf} & \ddot{x}(t_f) &= \ddot{x}_f \approx 0 \\
y(t_f) &= y_f & \dot{y}(t_f) &= \dot{y}_f = V_{yf} & \ddot{y}(t_f) &= \ddot{y}_f \approx 0 \\
z(t_f) &= z_f & \dot{z}(t_f) &= \dot{z}_f = V_{zf} & \ddot{z}(t_f) &= \ddot{z}_f \approx 0
\end{align*}
\] (3)

The chaser starts from whatever current condition it has (including current accelerations) and should maneuver itself precisely into the docking position with near-zero speed and near-zero acceleration.

There are no other constraints on states, but once again thrusters’ thrust is limited by \(T_{\text{max}}\).

**III. Formulation of Optimization Problem**

The Bolza ([9]) formulation of optimization problem for system (1) looks like follows. The problem is to choose the control input \(u(t)\) to minimize:
\[ J = \int_{a}^{b} f(t) \, dt \]  

subject to (2) and:

\[ \xi(t_0) = \xi_0, \quad \xi(t_f) = \xi_f. \]  

The performance index \( J \) we would like to minimize can be either the time required to perform the docking maneuver or overall amount of propellant spent to produce thrust or their combination. Therefore in Bolza formulation (4) \( f_0 = 1 \) for the time minimum problem, \( f_0 = u_x^2 + u_y^2 + u_z^2 \) for the propellant minimum problem and \( f_0 = 1 + w(u_x^2 + u_y^2 + u_z^2 - 1) \) in the most general case where \( w \) is the weighting coefficient.

Let us write the Hamiltonian of the system (1):

\[ H = p_x V_x + p_y V_y + p_z V_z + p_{v_x} f_1(x, V_y) + p_{v_y} f_2(V_x) + p_{v_z} f_3(z) + p_{v_x} u_x + p_{v_y} u_y + p_{v_z} u_z - f_0 \]  

The set of adjoint differential equations for the costate vector \( p_{\xi} \) then looks like follows ([10]):

\[ \dot{p}_x = -3 \omega^2 p_{v_x}, \quad \dot{p}_{v_x} = -p_x V_x + 2 \omega p_{v_x} \]
\[ \dot{p}_y = 0, \quad \dot{p}_{v_y} = -p_y V_y + 2 \omega p_{v_y} \]
\[ \dot{p}_z = \omega^2 p_{v_z}, \quad \dot{p}_{v_z} = -p_z V_z \]  

Before we proceed with the synthesis of the optimal control for the \( \pi \)-system (1)+(7) let’s make an important observation. Even if we solve the problem (4)+(2)+(5) using any numerical method we will not be able to satisfy all of the boundary conditions (3). Specifically, we couldn’t meet the requirements imposed onto initial and final accelerations. While it maybe not so important for the theoretical solution we would definitely want to meet these boundary conditions in the practical application – onboard spacecraft control system. Even if we need to sacrifice a fraction of the performance index. (Of course we can introduce new states and then we could meet all conditions (3) but it would complicate the optimization problem even further involving introducing dynamic constraints on the new states which used to be controls.)
IV. Synthesis of the Optimal Control

The optimal control \( u_{opt} = [\tilde{u}_x, \tilde{u}_y, \tilde{u}_z]^T \) for the problem (4)+(2)+(5) by definition can be found from \( u_{opt} = \arg \max_u H(u) \). Therefore, analyzing (6) we may conclude that the optimal control input \( \tilde{u}_k \) in each of three channels \( k=\{x,y,z\} \) is defined by:

\[
\tilde{u}_k = \text{sign}p_{vi} \min \left( \frac{|p_{vi}|}{2w}, U_{max} \right)
\]  

(8)

From equation (8) it is seen that the singular control arcs (when \( p_{vi} \equiv \dot{p}_{vi} = 0 \)) correspond to \( \tilde{u}_k \equiv 0 \). It is also seen that if the weighting coefficient \( w \) is small enough then we have a bang-singular-bang control. It is worth mentioning that from the standpoint of physical realization (construction of thrusters) bang-bang (on-off) control is also preferable (that actually casts the problem as a discrete optimization problem).

Now that the optimal control has been synthesized (although additional analysis needs to be performed to establish the rules for switching to/from the singular control arcs) the optimal control problem can be reduced to the problem of parameter optimization. In our particular case it means that we would guess on the final time \( t_f \) and initial values of costates \( \xi(t_0) \), then knowing the structure of the optimal control \( u_{opt}(t) = u_{opt}(\xi(t),p_{\xi}(t)) \) integrate the \( \pi \)-system (1)+(7). At the end we compare the final values of states with the given ones (5) and final costates with zero, and since in general they will not coincide we repeat integration of the \( \pi \)-system trying to tune \( t_f \) and \( p_{\xi}(t_0) \) to match the required final conditions (5) and \( p_{\xi}(t_f) = 0 \).

However, knowing how difficult it is to solve the problem (4)+(2)+(5) numerically in real time it maybe possible to define (parameterize) the controls time histories directly and then to only integrate the original system (1). Figure 2 represents an example of such a profile (suggested by discretization of equation (8)) and defined by several \((N)\) switching points \( t_n^k, n=1,N \) (this profile has to be established in each channel \( k=\{x,y,z\} \)).

![Figure 2: Parameterized control inputs time history.](image-url)
Varying the final time $t_f$ and location of the switching points $t^k_n$, $n=1,N$, $k=x,y,z$ of the control-time history profiles it maybe possible to satisfy most of boundary conditions (3) (not guaranteed though).

Although there is no a priory hints on the number of switching points and the sequence of control inputs, we could start from some small number of switching points alternating all possible values of controls and then increase $N$ if necessary to achieve more feasible solution. For instance, even with as little as four switching points shown on Fig.2 we may explore a wide variety of control profiles including pure bang-bang control as demonstrated on Fig.3.

![Figure 3: Pure bang-bang control profiles available with four switching points as defined on Fig.2.](image)

Although this latter approach may lead to the real-time algorithm, its robustness will not be guaranteed. The algorithm may diverge. No solution may exist. Not all boundary conditions can be satisfied. No predictions on the shape of the trajectory (its feasibility) can be made upfront even if solution exists. That is why the following introduces the direct method of calculus of variations as a mean to find a near-optimal solution to problem (4)+(2)+(3) in real-time.

**V. Introducing the Reference Trajectory**

We start from the “end” defining the near-optimal trajectory we want the chaser to follow upfront. Moreover, to be able to separate the trajectory from the speed profile we use some artificial argument $\tau$ rather than time $t$ ([8]). It should be understood from the very beginning though, that in our particular case by doing this
we are losing independency in controls (spacecraft should fly along the trajectory, hence its speed vector should always be tangent to this trajectory and as it will be shown later that, in turn, implies certain relationships between $u_x$, $u_y$, and $u_z$). However, introducing the reference trajectory allows satisfying the majority of the boundary conditions (3) upfront. It also excludes the possibility of “wild” unpredicted trajectories during the following parameter optimization.

As mentioned above, each of three chaser’s coordinates should be represented by some parameterized reference function versus virtual arc $\tau$, $P_x(\tau)$, $P_y(\tau)$ and $P_z(\tau)$, respectively. Without loss of generality we further consider a single class of reference functions, namely polynomials to show how its unknown coefficients can be determined and what can be done to assure an additional flexibility of the reference trajectory (the other class of reference functions might be trigonometric functions).

As stated by equalities (3) we need to satisfy up to the second derivative of Cartesian coordinates at both ends of the trajectory. It’s natural to require that the second derivate (proportional to accelerations) to be as smooth as at least third-order polynomial. Therefore, for each coordinate $k = \{x,y,z\}$ we may write the following:

$$P'_k(\tau) = a_{k2} + a_{k3}\tau + a_{k4}\tau^2 + a_{k5}\tau^3 = \sum_{l=2}^{5} a_{kl}\tau^{l-2}. \quad (9)$$

Integrating equation (9) twice yields

$$P'_k(\tau) = \sum_{l=1}^{5} \frac{a_{kl}\tau^{l-1}}{\max(1, l-1)} \quad \text{and} \quad P_k(\tau) = \sum_{l=0}^{5} \frac{a_{kl}\tau^l}{\max(1, l(l-1))} \quad (10)$$

so that six coefficients $a_{kl}$, $l = 0, 5$ for each $k = \{x,y,z\}$ can now be defined from the following linear matrix equation:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{12}\tau_f^4 & \frac{1}{20}\tau_f^5 \\ 0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{12}\tau_f^4 \\ 0 & 0 & 1 & \tau_f & \tau_f^2 & \tau_f^3 \end{bmatrix} \begin{bmatrix} a_{k0} \\ a_{k1} \\ a_{k2} \\ a_{k3} \\ a_{k4} \\ a_{k5} \end{bmatrix} = \begin{bmatrix} k_0 \\ k_1' \\ k_2' \\ k_3' \\ k_4' \\ k_5' \end{bmatrix}. \quad (11)$$

To write this matrix equation the derivatives of coordinates in (3) were converted to the new argument using the so-called speed factor.
\[ \lambda = \frac{d\tau}{dt} \]  

so that

\[ k' = \lambda^{-1} \dot{k} \quad \text{and} \quad k'' = \lambda^{-2}(\ddot{k} - \dot{k}\lambda') \]  

Once determined the coefficients of the reference functions and the reference trajectory itself will depend on the only varied parameter, \( \tau_f \). But what if additional flexibility is needed? Then we can increase the order of the reference polynomials and use the higher-order derivatives at both ends as additional varied parameters. For instance in our particular case it makes sense using 6th or even 7th order polynomials instead of 5th order polynomials (10) and then use the third derivatives of coordinates with respect to the virtual arc at the both ends of the trajectory as varied parameters.

**VI. Introducing Master Control Arc History**

Let us first convert the system (1) to the new argument. Using the speed factor (12) we can write

\[
\begin{align*}
    x' & = \lambda^{-1} V_x \\
    y' & = \lambda^{-1} V_y \\
    z' & = \lambda^{-1} V_z \\
    V'_x & = \lambda^{-1} f_1(x, V_x) + \lambda^{-1} u_x \\
    V'_y & = \lambda^{-1} f_2(V_y) + \lambda^{-1} u_y \\
    V'_z & = \lambda^{-1} f_3(z) + \lambda^{-1} u_z
\end{align*}
\]  

(14)

Combining the first three equations of (14) as

\[
\sqrt{V_x^2 + V_y^2 + V_z^2} = V = \lambda \sqrt{x'^2 + y'^2 + z'^2}
\]  

(15)

specifically addresses the issue of independency of the trajectory and the velocity along it. Having the trajectory defined with respect to the virtual arc \( \tau \), i.e. having \( x', y' \) and \( z' \) defined, still leaves a possibility of varying the magnitude of the speed via varying the speed factor \( \lambda \). However, the orientation of the speed vector with respect to the trajectory is completely determined by this trajectory (regardless its argument) and can be defined by two Euler angles as

\[
tg\varphi = \frac{x'}{y'} = \frac{V_x}{V_y}, \quad tg\theta = \frac{x'}{\sqrt{y'^2 + z'^2}} = \frac{V_x}{\sqrt{V_y^2 + V_z^2}}
\]  

(16)

That means that we cannot longer vary all three controls, \( u_x \), \( u_y \) and \( u_z \), independently. We should define one master control (in the predominant direction), say \( u_x \), and then define two others so that the direction of the velocity vector is tangent to the trajectory (meaning that equalities (16) hold).
Now, as suggested by the optimal control theory we assume the master control arc profile to be bang-singular-bang as shown on Fig.4 (which represents exactly the same control profile as on Fig.2 but with respect to the virtual arc $\tau$ rather than time $t$).

![Figure 4: Suggested control profile for the master control.](image)

Obviously two other controls, $u_y$ and $u_z$, will not be bang-singular-bang anymore so that the pulse-width-modulation should be used to produce continuous accelerations in these two channels.

The parameter optimization routine may be established as shown on Fig.5.

![Figure 5: Parameter optimization flow chart.](image)

Given the boundary conditions (3) we first define reference functions $P_x(\tau), P_y(\tau)$ and $P_z(\tau)$, and compute their coefficients using these boundary conditions (and initial guesses on the third derivatives in case higher than $5^{th}$ order polynomials were employed for more flexibility). For the master control we also establish a bang-singular-bang arc profile defined by several switching points. These switching points, $\tau_i, i=1,4$, along with the length of the virtual arc $\tau_f$ (and possibly values of the higher-order derivatives of the coordinates at initial and/or final points) form the vector of variable parameters $\Xi$. 
Next, we numerically solve the problem (integrating just one instead of all state equations and applying inverse dynamics for the rest of them). The transition between the virtual arc $\tau$ and time $t$ is made using the speed factor

$$
\lambda = \frac{\sqrt{x'^2 + y'^2 + z'^2}}{\sqrt{V_x'^2 + V_y'^2 + V_z'^2}}
$$

Then, we estimate the performance index $J$ and compound the aggregated penalty $\Delta$. The existence of this penalty is caused by the fact that the constraints on two slave controls are necessarily met and that the boundary conditions for the velocity vector components are not satisfied (since one equation was integrated and two others are related to it via dynamic constraints (16)).

Now, we apply any standard nonlinear constrained minimization routine to minimize the performance index keeping the penalty within a certain tolerance $\varepsilon$

$$
\min_{\Xi} J \bigg|_{\Delta \leq \varepsilon}.
$$

VII. Computation of States, Performance Index and Penalty

We start from dividing the virtual arc $\tau_f$ onto $N$-1 equal pieces $\Delta \tau = \frac{\tau_f}{N-1}$ so that we have $N$ equidistant nodes $j = 1, N$. All states and the master control at the first point $j = 1$ (corresponding to $\tau_1 = \tau_0 = 0$) are defined. Additionally we define $\lambda_1 = 1$.

Then, for each of the subsequent $N$-1 nodes $j = 2, N$ we do the following. We compute the current values of coordinates $x, y$ and $z$ using polynomials $x_j = P_x(\tau_j)$, $y_j = P_y(\tau_j)$ and $z_j = P_z(\tau_j)$ respectively. Next, knowing the master control from the predetermined arc history $u_{x,j-1} = u_x(\tau_{j-1})$ we integrate the forth equation of system (14)

$$
V_{x,j} = V_{x,j-1} + \lambda_{j-1}^{-1} \left( f_x(x_{j-1}, V_{y,j-1}) + u_{x,j-1} \right) \Delta \tau.
$$

To assure the correct direction of the velocity vector we apply relations (16) to obtain two other velocity components

$$
V_{y,j} = V_{y,j} \frac{y'_j}{x'_j} \quad \text{and} \quad V_{z,j} = \frac{z'_j \sqrt{V_{x,j}'^2 + V_{y,j}'^2}}{\sqrt{x'_j^2 + y'_j^2}}
$$
and therefore calculate the magnitude of speed

$$|V_j| = \sqrt{V_{x,j}^2 + V_{y,j}^2 + V_{z,j}^2}. \quad (21)$$

Now that we know the change in the chaser’s position and the magnitude of speed we may compute the time interval between \((j-1)\)th and \(j\)th nodes

$$\Delta t_{j-1} = \frac{2}{|V_j| + |V_{j-1}|} \sqrt{\sum_{i=1}^{3} (\xi_{i,j} - \xi_{i,j-1})^2} \quad (22)$$

and the current value of the speed factor

$$\lambda_j = \frac{\Delta \tau}{\Delta t_{j-1}}. \quad (23)$$

Current time then equals to

$$t_j = t_{j-1} + \Delta t_{j-1} \quad (t_i = 0). \quad (24)$$

Finally, using the last two equations of the system (14) we find the values of two slave controls that yield speed components (20)

$$u_{x,j-1} = \frac{V_{y,j} - V_{y,j-1}}{\Delta \tau} \lambda_j + 2 \omega V_{x,j-1} \quad \text{and} \quad u_{z,j-1} = \frac{V_{z,j} - V_{z,j-1}}{\Delta \tau} \lambda_j + \omega^2 z_{j-1} \quad (25)$$

Once all states and controls are computed, we may estimate the performance index

$$J = (1-w)t_N + w \sum_{j=0}^{N-1} \left( u_{x,j}^2 + u_{y,j}^2 + u_{z,j}^2 \right) \Delta t_j \quad (26)$$

and form the penalty as

$$\Delta = w_p \sum_{k} (V_{k,N} - \dot{k}_j)^2 + (1-w_p) \sum_{k} \max \left( 0, |u_{k,j}| - U_{\max} \right)^2 \quad (27)$$

where \(k = \{x,y,z\}\) and \(w_p\) is the penalty weighting coefficient.

For numerical convergence of the algorithm, time, the slaves control constraint violation w.r.t. the limit on maximum trust and the discrepancy on final velocity, have been re-scaled obtaining the following modified performance index and penalty:
\[
J_{sc} = t_{sc} (1-w)t_N + w \sum_{j=0}^{N-1} \left( u_{x,j}^2 + u_{y,j}^2 + u_{z,j}^2 \right) \Delta t_j
\]

\[
\Delta_{sc} = w_{\text{speed}} V_{sc} \sum_k (V_{k,N} - \dot{k}_j)^2 + w_{\text{overflow}} O_{sc} \sum_k \max \left( 0; |u_{k,j}| - U_{\text{max}} \right)^2
\]

(28)

Coefficients have been adjusted to bring the values to the same order.

VIII. Simulation Results

Two examples are reported in order to show how the algorithm is capable to generate in a short time the command sequence to drive the chaser towards the target in a sub-optimal way.

The boundary conditions are the same for the two test cases, only the weighting ratio between propellant and time is changed showing how the resulting trajectories differ from each other. In the first case propellant is more important to be saved than the time required for docking: \( w = 0.9 \). While in the second test the weighting coefficient is 0.1, giving more importance to execute the maneuver in a short time.

The numerical values used in both simulations are shown in Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Units</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height above the Earth surface</td>
<td>km</td>
<td>981.46</td>
</tr>
<tr>
<td>Initial relative position</td>
<td>m</td>
<td>(-60, -40, 0)</td>
</tr>
<tr>
<td>Initial relative velocity</td>
<td>m/s</td>
<td>(0.005, 0, 0)</td>
</tr>
<tr>
<td>Required relative final position</td>
<td>m</td>
<td>(0, 0, 0)</td>
</tr>
<tr>
<td>Required relative final velocity</td>
<td>m/s</td>
<td>(0.0005, 0, 0)</td>
</tr>
<tr>
<td>Initial and final relative accelerations</td>
<td>m/s²</td>
<td>(0, 0, 0)</td>
</tr>
<tr>
<td>Maximum relative acceleration (thrust)</td>
<td>m/s²</td>
<td>0.0138</td>
</tr>
<tr>
<td>Number of points for computation along arc ( \tau )</td>
<td>-</td>
<td>200</td>
</tr>
<tr>
<td>Time scaling factor</td>
<td>1/s</td>
<td>1/1000</td>
</tr>
<tr>
<td>Control constraint violation scaling factor</td>
<td>(m/s²)⁻¹</td>
<td>40</td>
</tr>
<tr>
<td>Final discrepancy on velocity scaling factor</td>
<td>(m/s)⁻¹</td>
<td>200</td>
</tr>
</tbody>
</table>

Table 1: Numerical values for the test cases.

Unfortunately the Matlab® \textit{fmincon} function for constrained non-linear optimization to solve (18) failed to work (it is known to be quite unstable.). Therefore the Matlab® \textit{fminsearch} function has been employed in both simulations as optimization means to work on five varied parameters, i.e., the virtual arc length and the master control switches. In this case the performance index \( J_{sc} \) and penalty function \( \Delta_{sc} \) (28) were blended together using an appropriate weighting coefficient.
It worth mentioning that in some sense the \textit{fminsearch} function was more preferable because of another reason too. This reason is that it employs zero-order (non-gradient) rather than gradient algorithms (like Nelder-Mead downhill simplex algorithm) assuring unconditioned finding of at least local minimum, i.e. certain reliability for the future on-line implementation.

The software used in the modeling stage is based on a call to a Simulink\textsuperscript{®} model every time an index and penalty evaluation is required. This obviously slows down the overall process w.r.t. a simple Matlab\textsuperscript{®} script and even more w.r.t. a C code, but still demonstrates a very satisfactory relative CPU time (percentage of time required by the machine w.r.t. the complete maneuver required time). In what follows the relative CPU time is reported for two different machines: an AMD Athlon 2600 MHz processor, and a Pentium III 1200 MHz processor.

**Simulation test case 1**

An initial guess on virtual arc length and position of the four switching points was: 
\[\tau_f = 5.5, \quad \tau_1 = 0.007\tau_f, \quad \tau_2 = 0.1\tau_f, \quad \tau_3 = 0.33\tau_f, \quad \text{and} \quad \tau_4 = 0.4\tau_f.\]

The resulting trajectory, with the corresponding optimized values of the OPs controls’ behavior, velocity history, fuel consumption and other significant parameters is shown on Fig.6.

![Figure 6: Results for simulation test case 1.](image-url)
Number of iterations is rather small (<100). Note how the final velocity is of the same order of the required one (0.0005m/s) and the fact that there is no control constraint violation, i.e. the slaves are respecting the imposed bounds.

Relative CPU time came out to be 2.9% with the faster machine (AMD Athlon 2600 MHz) and 4.3% with the Pentium III, 1200 MHz.

**Simulation test case 2**

For this test case, where we still optimize a combination of fuel and time, but giving more importance to the rapidity of the maneuver execution, the initial guess was $\tau_f = 9$, $\tau_1 = 0.007\tau_f$, $\tau_2 = 0.2\tau_f$, $\tau_3 = 0.5\tau_f$, and $\tau_4 = 0.6\tau_f$.

The results are shown on Fig.7.

![Figure 7: Results for simulation test case 2.](image)

Again, the required iterations are less than 100. For this maneuver, more demanding than the Simulation test case 1, we obtain a small control constraint violation of 5.9%, and the final velocity discrepancy is slightly higher than in the previous case.

Having required minimizing time, with a small consideration of propellant expenditure in this case, it results in the possibility of limit violations in the slaves’ behavior. Note how the final time is ~35% lower than in test case 1, as expected (1982 s vs. 3025s). At the same time the propellant expenditure raised from 0.5 units to 0.898 units (~80% increase).
Relative CPU time is 4.2% on the AMD Athlon 2600 MHz and 6% on the Pentium III, 1200 MHz.

IX. Conclusion

The proposed direct method for trajectory optimization shows many advantages. First of all it guarantees the boundary conditions to be satisfied, no “wild” trajectories arise during optimization, an analytical (parametrical) representation of the reference trajectory is possible, and, the last but not the least, a small number of OPs have to be considered, requiring only a few iterations (<100) to generate a solution. These two last features, together with a low relative CPU time for convergence, make possible to employ DMRP on board of a spacecraft for real-time prototyping of rendezvous and docking maneuvers. Resulting trajectory generation algorithms can be easily integrated with existing navigation/control algorithms.

Next development of the present study will be the implementation of the algorithm with the C language and the hardware-in-the-loop test. At this time the constrained optimization routine based on the zero-order Hooke-Jeeves method currently being developed by the authors is expected to be fully tested and ready. Also, the authors intend to rewrite the original system of differential equations driving system’s dynamics (1) in the wind (trajectory) coordinate frame to see whether it simplifies computations and benefits in further reducing the relative CPU time.

References