Quasi-Optimal Control For Path Constrained Relative Spacecraft Maneuvers based on Dynamic Programming

R. Bevilacqua[1] and M. Romano[2*]

[1] *Ph. D. Candidate, University "La Sapienza", Mathematical Methods and Models for Applied Sciences, Via A. Scarpa 16, I-00161, Rome, Italy.*
[2] *Assistant Professor, US Naval Postgraduate School, Department of Mechanical and Astronautical Engineering, Code MAE/MR, 700 Dyer Rd., Monterey, California, 93943.*

Received: ???? ?, ????; Revised: ???? ?, ????

Abstract: Autonomous close flight and docking of a chaser spacecraft to a target are still challenging problems. In this paper the Hill-Clohessy-Wiltshire equations are taken as dynamic model and inverted, after a variable change, in order to be used by a control algorithm to drive the chaser spacecraft along a specified path. The path parameterization is performed by using cubic B-splines and by having the curvilinear abscissa as parameter. The proposed optimization algorithm uses dynamic programming to find quasi-optimal controls. The number of optimization parameters is drastically reduced by working only on the acceleration component along the vehicle trajectory. The shape of the path can be chosen according to the specific maneuver requirements. In particular, the optimization algorithm is split into a trajectory planner which generates the best tangential acceleration sequence through backward exploration of a tree of possible policies, and a control generator which inverts the parameterized dynamics in order to get the thrusters commands sequence. The optimization algorithm has been coded in Simulink as a library of Embedded Functions and has been experimentally proved to run in Real Time.

Keywords: *Satellites Rendezvous, Dynamic Programming, Dynamic Inversion, Cubic B-Splines.*

Mathematics Subject Classification (2000): Sub-Optimal Control

* Corresponding author: mromano@nps.edu.

Nomenclature

$\vec{a},\ a$ = Generic Vector and its Magnitude
$\mathbf{B}$ = Universal Transformation Matrix for Cubic B-Splines
ds = Path Section Length
$\frac{d^2(.)}{dt^2}$ = Time Derivatives with respect to the Inertial Frame
$\frac{\delta(.)}{\delta s}$ = Partial Derivative with respect to s
$\vec{\gamma}$ = Cubic B-splines parameter
J = Cost Functional
$\mathbf{K}$ = Matrix containing the Vector Positions of the Cubic B-Splines Control Points
$LVLH$ = Local Vertical Local Horizontal Coordinate System
μ = Earth Gravitational Constant
N_L = Number of Levels in Dynamic Programming Tree
N_C = Number of Values for Control Discretization
$\hat{n}$ = Path Normal Unit Vector
$\vec{\omega}_{LVLH}$ = Angular Velocity in LVLH
$\vec{r}$ = Position Vector
$\underline{\underline{r}}$ = Dyadic of Position Vector $\vec{r}$
$\vec{r}_c$ = Chaser Position Vector in the Inertial Frame
$\vec{r}_t$ = Target Position Vector in the Inertial Frame
$\vec{r}_{rel}$ = Chaser-Target Relative Position Vector
$\vec{r}^* = [r_x,\, r_y,\, 0]^T$
ρ = Local Curvature of Path
s = Curvilinear Abscissa
t = Time
$\hat{\tau}$ = Path Tangent Unit Vector
$\vec{u}$ = Acceleration Control Vector
$x,\, y,\, z$ = Components of $\vec{r}_{rel}$ in LVLH
$\underline{\underline{1}}$ = Unitarian Dyadic
$(.)_0\,,\ (.)_f$ = Value at Initial and Final Time
$\dot{(.)},\ \ddot{(.)}$ = Time Derivatives in LVLH
$(.)_x\,(.)_y\,(.)_z$ = Components along $x,\ y,\ z$ axis of LVLH

1 Introduction

Many researches have been performed on modeling ([1, 2, 3, 4, 5, 6, 7, 8]) and optimizing ([9]) the maneuvers for Spacecraft Rendezvous and Docking, but real-time implementation of the optimal control is still a difficult task. In [10] and [11] the maneuvers for passing from an initially stable relative motion to a final stable state are optimized. In these works there is no possibility of considering generic initial conditions for the relative position and velocity of the chaser vehicle with respect to the target. In [9] 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.

All of the above references prove the effectiveness of their respective approaches by using numerical simulations. None of them perform hardware experimentation on real-time computing. Furthermore, the current docking missions, such as the Space

Shuttle-International Space Station (ISS) mating, and the planned Automated Transfer Vehicle-ISS supply service, do not include any closed-loop optimization for the rendezvous trajectories.

The objective of our research is threefold: first, we want to be able to impose a certain path in order to guarantee a priori the safety of the maneuver, second we want to track the set path with minimum propellant consumption, third, we want our solution to be suitable for real time implementation.

The proposed strategy is a direct optimization method similar to the one used in [12] and [13] for the control of robotic manipulators and in [14] for the control of aircraft. The basic idea is to parameterize the trajectory in a way that allows for independent choice of path and velocity profile. In order to obtain such feature, we use here the curvilinear abscissa as in [12] and [13].

Dynamic Programming ([15]) is used as the optimization method. This approach has been proved to be suitable for real-time implementation, especially when a sub-optimal solution is searched through a step-by-step optimization of the trajectory ([16, 17]).

Our contribution is focused on real-time autonomous control during the very last phases of satellites docking procedure.

The paper is organized as follows: Section 2 formally states the problem. Section 2.1 describes the dynamic model and the inversion of the dynamics. Section 3 presents the path parameterization via cubic B-splines. Section 4 is dedicated to the developed optimization algorithm. Section 5 reports some significant results obtained via numerical simulations. Section 6 illustrates the Real-Time test which has been performed to prove the capability of running in real-time. Finally, Section 7 concludes the paper.

2 Problem Statement

The aim of the present work is to quickly design an optimal control sequence which drives the chaser vehicle towards the target on a specified path $\vec{r}_{rel} = \vec{r}(s)$. Without loss of generality, the boundary conditions are assumed to be $\vec{r}_{rel}(t_0) = \vec{r}_0$, $\vec{r}_{rel}(t_f) = \vec{0}$.

The following minimum-propellant cost function is considered for the optimal control problem:

$$J = \int_{t_0}^{t_f} |u|^2 dt \tag{1}$$

2.1 Dynamic Model and Dynamics Inversion

We use the well known Hill-Clohessy-Wiltshire equations in order to model the system dynamics. We consider the x axis of the LVLH coordinate system centered on the target directed along the radial line, the y axis directed along the orbital velocity, and the z axis consequently directed to complete the right frame.

The HCW equations can be written in vector form as:

$$\ddot{\vec{r}}_{rel} + 2\vec{\omega}_{LVLH} \times \dot{\vec{r}}_{rel} - \omega^2_{LVLH}\vec{r}^{\,*}_{rel} = \frac{\mu}{r_t^5}\left(3\underline{\underline{r_t}} - r_t^2\underline{\underline{1}}\right) \cdot \vec{r}_{rel} + \vec{u} \tag{2}$$

As stated above, the proposed optimization approach is based on the search for a suboptimal policy to drive the chaser vehicle along a specified path. The specified path is parameterized in terms of the curvilinear abscissa as follows ([13]):

$$\vec{r}_{rel} = \vec{r}\,(s) = [x\,(s)\,,\, y\,(s)\,,\, z\,(s)] \tag{3}$$

Differentiation with respect to time and substitution of this relation into Eq. (2) gives:

$$\frac{\delta^2\vec{r}}{\delta s^2}\dot{s}^2 + \frac{\delta\vec{r}}{\delta s}\ddot{s} + 2\vec{\omega}_{LVLH}\times\frac{\delta\vec{r}}{\delta s}\dot{s} - \omega^2_{LVLH}\vec{r}^{*} = \frac{\mu}{r_t^5}\left(3\underline{\underline{r_t}} - r_t^2\underline{\underline{1}}\right)\cdot\vec{r} + \vec{u} \tag{4}$$

The tangential acceleration profile $\ddot{s}$ to be tracked by the chaser spacecraft is considered as the free parameter for the optimization. Once the tangential acceleration profile is obtained, the actual controls (accelerations along the three axis) are then determined by inverting the dynamics equation. In particular, by starting from Eq.(4), it results:

$$\vec{u} = -\frac{\mu}{r_t^5}\left(3\underline{\underline{r_t}} - r_t^2\underline{\underline{1}}\right)\cdot\vec{r} + \frac{\delta^2\vec{r}}{\delta s^2}\dot{s}^2 + \frac{\delta\vec{r}}{\delta s}\ddot{s} + 2\vec{\omega}_{LVLH}\times\frac{\delta\vec{r}}{\delta s}\dot{s} - \omega^2_{LVLH}\vec{r}^{*} \tag{5}$$

Since we work in terms of acceleration along the path, it is not practical to impose directly constraints on the controls by using Eq. (5).

3 Geometric Representation of the Path

We use cubic B-spline curves to represent the trajectory $\vec{r}_{rel} = \vec{r}\,(s)$. B-splines have the advantage of narrowly propagating the local changes ([18]). Given n control points, a first and second order continuous curve which fits them is univocally determined by a composition of $n-1$ B-splines. As the positions of the control points change, the curve shape changes consequently. Each spline is defined by four control points and has the parametric representation:

$$\vec{r}\,(\vec{\gamma}) = \vec{\gamma}BK \tag{6}$$

Where $\vec{r}\,(\vec{\gamma})$ is the position vector of a generic point of the spline, $\vec{\gamma}$ is the parameter vector, defined as:

$$\vec{\gamma} = \left[\begin{array}{cccc} \gamma^3 & \gamma^2 & \gamma & 1 \end{array}\right],\ 0 \le \gamma \le 1 \tag{7}$$

K contains the vector positions of the control points:

$$K = \left[\begin{array}{cccc} \vec{r}_0 & \vec{r}_1 & \vec{r}_2 & \vec{r}_3 \end{array}\right]^T \tag{8}$$

and B is the universal transformation matrix, obtained by imposing continuity, which contains the same numerical values for every B-spline ([18]):

$$B = \frac{1}{6}\left[\begin{array}{cccc} -1 & 3 & -3 & 1 \\ 3 & 6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{array}\right] \tag{9}$$

Accordingly, the partial derivatives of the path with respect to the arc length, needed in Eq. (4) and Eq. (5), are given by:

$$\frac{\delta\vec{r}}{\delta s} = \frac{\frac{\delta\vec{r}}{\delta\gamma}}{\left|\frac{\delta\vec{r}}{\delta\gamma}\right|},\quad \frac{\delta^2\vec{r}}{\delta s^2} = \frac{\frac{\delta^2\vec{r}}{\delta\gamma^2}}{\left|\frac{\delta\vec{r}}{\delta\gamma}\right|^2} - \frac{\delta\vec{r}}{\delta\gamma}\frac{\frac{\delta\vec{r}}{\delta\gamma}\cdot\frac{\delta^2\vec{r}}{\delta\gamma^2}}{\left|\frac{\delta\vec{r}}{\delta\gamma}\right|^4} \tag{10}$$

where:

$$\frac{\delta\vec{r}}{\delta\gamma} = \begin{bmatrix} 3\gamma^2 & 2\gamma & 1 & 0 \end{bmatrix} BK, \ \frac{\delta^2\vec{r}}{\delta\gamma^2} = \begin{bmatrix} 6\gamma & 2 & 0 & 0 \end{bmatrix} BK \tag{11}$$

Since the B splines, by definition, do not pass through the end points, two additional artificial control points are added at the extremes of the curve.

In order to convert the value of the global arc length along the path s to the corresponding value of the parameter along the local spline γ it is sufficient to numerically find the zero of the following function:

$$h(\gamma) = s - \left(s_{0_{current\,spline}} + \int_0^\gamma \frac{ds}{d\gamma} d\gamma \right) \tag{12}$$

where

$$\frac{ds}{d\gamma} = \sqrt{\left(\frac{dx(\gamma)}{d\gamma}\right)^2 + \left(\frac{dy(\gamma)}{d\gamma}\right)^2 + \left(\frac{dz(\gamma)}{d\gamma}\right)^2} \tag{13}$$

4 Optimization Approach by Dynamic Programming

Dynamic Programming ([15]) is very useful in problems where one needs to take subsequent decisions. The word “Dynamic” states that the decisions are sequentially taken. Sequenced problems can be solved according to the Bellman’s Principle of Optimality:

Definition 4.1 An optimal strategy has the property that, no matter the initial state and decision, the future decision’s set has to constitute an optimal strategy with respect to the state reached according to the decisions taken until that moment

Dynamic Programming takes the decisions one by one. At every step the optimal policy for the future is found, independently of the past decisions. The cost function is divided in the sum of elementary costs, one for each segment of the trajectory. In the present work we adopt a similar approach to the one of [16].

The curvilinear acceleration on the specified curve is taken as the parameter for the optimization. By limiting to one the number of variables the algorithm has to work with, the issue of “curse of dimensionality” ([16]) is greatly mitigated. Once the optimal $\ddot{s}$ profile is determined, the controls are calculated according to Eq. (5). On a certain segment of the entire path, where the acceleration $\ddot{s}$ is kept constant, we know that the section length ds needs a time interval t to be run:

$$\begin{gathered} t = -\frac{\dot{s}_0}{\ddot{s}} \pm \sqrt{\left(\frac{\dot{s}_0}{\ddot{s}}\right)^2 + \frac{2ds}{\ddot{s}}} \\ \left(s(t) = s_0 + \dot{s}_0 t + \ddot{s}\frac{t^2}{2} ds = \dot{s}_0 t + \ddot{s}\frac{t^2}{2} \rightarrow t^2 + \frac{2\dot{s}_0}{\ddot{s}} t - \frac{2ds}{\ddot{s}} = 0 \right) \end{gathered} \tag{14}$$

The complete path is divided into segments of length ds, the controls and time requested to run each segment are calculated and finally the integral 1 is evaluated. In particular, the algorithm tests different possibilities, in terms of possible sequence of levels of $\ddot{s}$, by using the tree approach described below. Feasibility of the sequences is tested by imposing the satisfaction of the following constraints:

$$\begin{cases} s(t_f) = s_f = trajectory\ length \\ \dot{s}(t_f) = 0 = final\ velocity \end{cases} \tag{15}$$

The tree of possible policies is a graph structure, with no cycles, in which every node generates different branches, every one connected to a subsequent node, called son-node. A son can generate other sons. Every node, but the root, has only one entering branch which belongs to the parental node. The root does not have any father. Starting from the initial condition, the tree of possible trajectories is built in an incremental way, by exploring the reachable states in order to find the optimal path. Every step adds new branches (so new nodes) to the tree, only if they are acceptable: in particular, only positive velocities are admitted. Therefore, nodes with zero velocity and zero acceleration are not taken into account. Moreover, the cases when Eq. (14) does not have real solutions are discarded, since this implies that the current section cannot be run with the current value of $\ddot{s}$. A cost value is associated to every node. The cost is calculated through a trapezoidal integration formula of the control magnitude. The algorithm goes on till the stopping condition is reached. At this point it is necessary to run the tree backwards in order to find the optimal policy, i.e. the one which brings to the final node at minimum cost.

The algorithm outline follows (see also Figure 4.1):

1. start from the initial condition s_0, $\dot{s}_0$ (ROOT);
2. apply one of the N_C possible value of $\ddot{s}$ until the trajectory segment ds is completed
3. repeat step 2 for each possible value of acceleration: one generation is then obtained;
4. calculate the cost function for each node of the present generation;
5. starting from every son, repeat step 2 and 3 and obtain the second generation;
6. iterate step 2, 3 and 4 until the final desired condition s_f, $\dot{s}_f$ is reached;
7. recognize the minimum cost node of the last generation. In Figure 4.1 the numbers at the top report, as an example, the total cost associated with every node;
8. individuate the optimal policy, by starting from the optimal son of the last generation and run the tree backward.

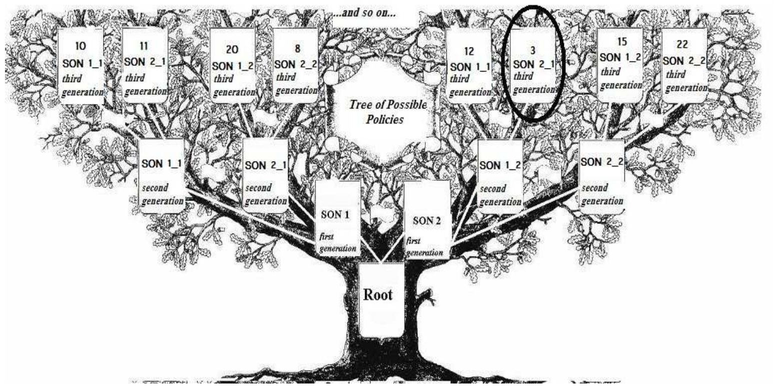
...and so on...
10 SON 1_1 third generation
11 SON 2_1 third generation
20 SON 1_2 third generation
8 SON 2_2 third generation
Tree of Possible Policies
12 SON 1_1 third generation
3 SON 2_1 third generation
15 SON 1_2 third generation
22 SON 2_2 third generation
SON 1_1 second generation
SON 2_1 second generation
SON 1_2 second generation
SON 2_2 second generation
SON 1 first generation
SON 2 first generation
Root

Figure 4.1: Tree of Possible Policies

From the software point of view each son-node is represented by a 7 elements array:

1. value of the curvilinear abscissa s (this value is the same for every node of the same generation);
2. value of the velocity along the path $\dot{s}$;
3. value of the acceleration along the path $\ddot{s}$ (“control”);
4. time to reach the node from the previous one;
5. cost associated to the last branch;
6. total time (from beginning of the path) to run until that point;
7. index (identifier of the father).

By keeping memory of the father for every node it is possible to quickly run the tree backwards once the final condition is reached in order to find the best $\ddot{s}$ profile. The number of generations N_L (that is, the number of segments in which the path is divided) and the number N_C of admissible acceleration values $\ddot{s}$ can be selected by the user.

The required memory and computation time increase very quickly with the number of levels, as the maximum number of nodes to be evaluated is $N_C^{N_L}$.

The tree exploration is the most time consuming portion of the algorithm. A pruning approach is implemented in order to increase the computation efficiency. The new nodes of a generation are analyzed while building the tree and pruned if the remaining length of trajectory to be run is not sufficient to brake down with the minimum available acceleration in order to reach the final point with zero velocity.

5 Simulation Results

Two sample simulation cases are here presented in order to show the main features of the proposed approach.

5.1 Simulation Test Case 1

The first simulation shows the behavior of the proposed algorithm in the computation of a quasi-optimal docking maneuver as a function of the number of path segments (number of generations. or levels, of the dynamic programming algorithm). The parameters of the maneuver are reported in Table 5.1. A cubic path ($y = \left(y_0/x_0^3 \right) x$) is taken as reference.

Parameter	Units	Value
Height above the Earth surface	km	480
Maximum (minimum) tangential acceleration $\ddot{s}_{\max}$ $(-\ddot{s}_{\min})$	$\frac{m}{s^2}$	$5 \cdot 10^{-4}$
Weighting factor$k$	$\frac{m^2}{s^4}$	0
Initial velocity $\dot{s}_0$	$\frac{m}{s}$	0.2
Initial position (x_0, y_0, z_0)	m	(300, 400, 0)
Number of path segmentsN_L	-	5
Number of possible tangential acceleration valuesN_C	-	3

Table 5.1: Numerical values for cubic manoeuvre

The same constrained path has been correctly obtained for any number of levels, as the nature of the algorithm implies. This result is reported in Figure 5.1.

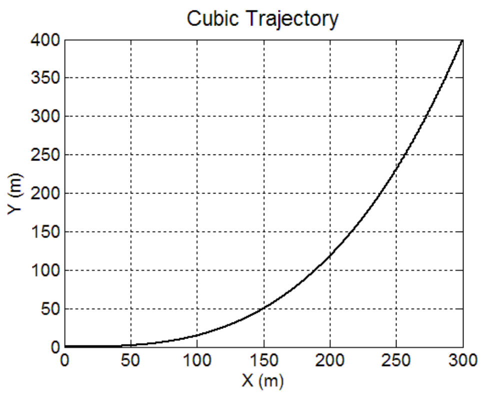

Figure 5.1: Cubic path tracked by the chaser

Figure 5.2 shows the fuel consumption and time required for the manoeuvre, Figure 5.3 shows the CPU time (on a Pentium IV machine). The independent variable is the number of levels.

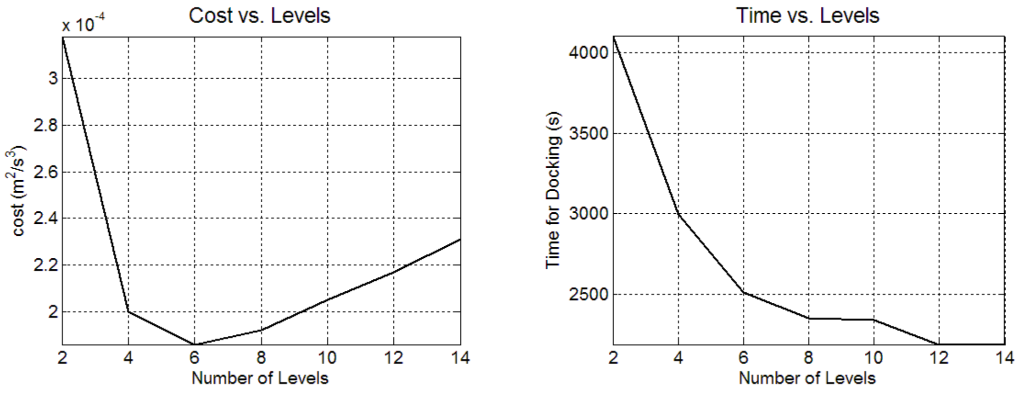

Figure 5.2: a) Fuel vs. levels; b) Time for manoeuvre vs. levels

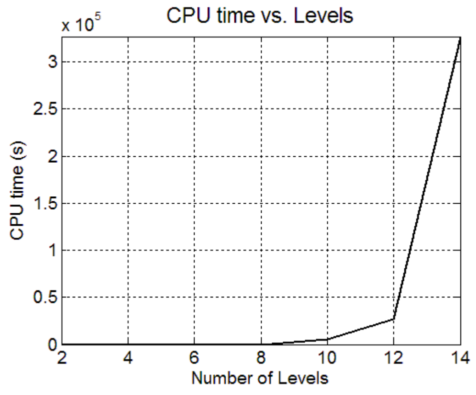

Figure 5.3: CPU time vs. n. of levels

It is apparent in Figure 5.2 that increasing the number of levels means a larger tree to build and explore, i.e. higher CPU resources. Figure 5.3 is worth some comment on the goal of the proposed technique. The proposed approach gives a sub-optimal solution. In particular, by dividing the trajectory in a finite number of segments, and imposing a limited set of values for the command acceleration, we are clearly introducing considerable discretization into the problem. Therefore the solution is heavily dependent on the number of levels. For the presented simulation case and for several other tests we have run, it results that the algorithm gives the best result for a number of levels between 4 and 6, as indicated in Figure 5.2 (a). The CPU time is also reasonable within that range. Figure 5.4 reports the controls profile for three different numbers of levels.

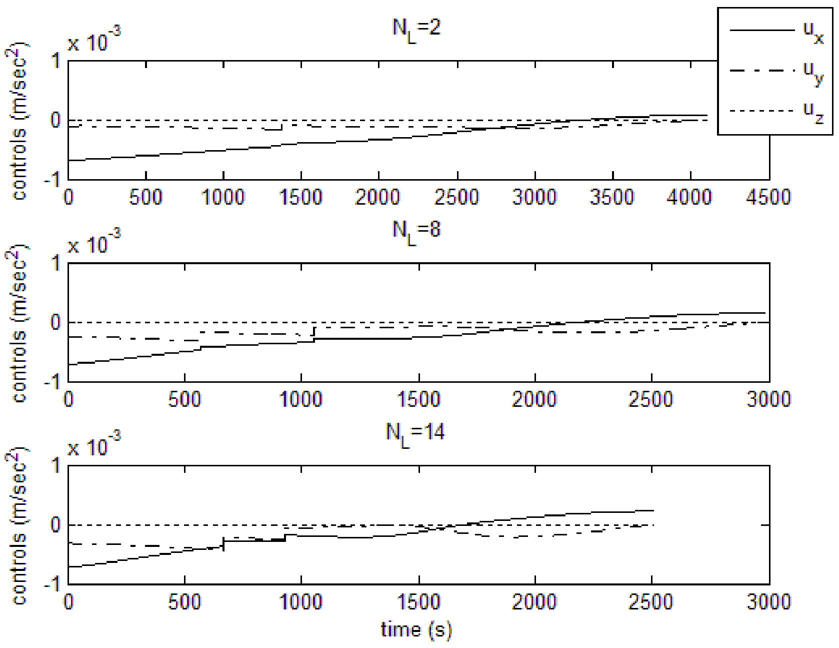

Figure 5.4: Controls vs. Time, increasing N_L

5.2 Simulation Test Case 2

The second simulation considers the final straight line maneuver (along the y direction) bringing a chaser spacecraft to dock with its target, as previously studied in [9]. We show here how the tree strategy is able to generate similar results to [9], but in a quicker way. Furthermore, initial and final conditions are exactly satisfied. The initial conditions here adopted are the intermediate conditions (breakpoint between two stages) reported in [9]:

Parameter	Units	Value
Height above the Earth surface	km	480
Maximum (minimum) acceleration $\ddot{s}_{\max}$ $(-\ddot{s}_{\min})$	$\frac{m}{s^2}$	$5 \cdot 10^{-4}$
Weighting factor k	$\frac{m^2}{s^4}$	0
Initial velocity $\dot{s}_0$	$\frac{m}{s}$	0.2
Initial position $(x_0,\, y_0,\, z_0)$	m	(0, 300, 0)
Number of levels N_L	-	5
Number of possible tangential acceleration values N_C	-	3

Table 5.2: Numerical values for straight line maneuver

Figure 5.5 shows that the trajectory is perfectly tracked:

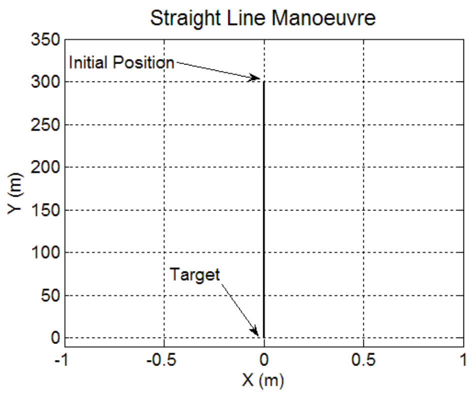

Figure 5.5: Straight line trajectory

The controls, acceleration and velocity profile are reported on Figure 5.6 and Figure 5.7.

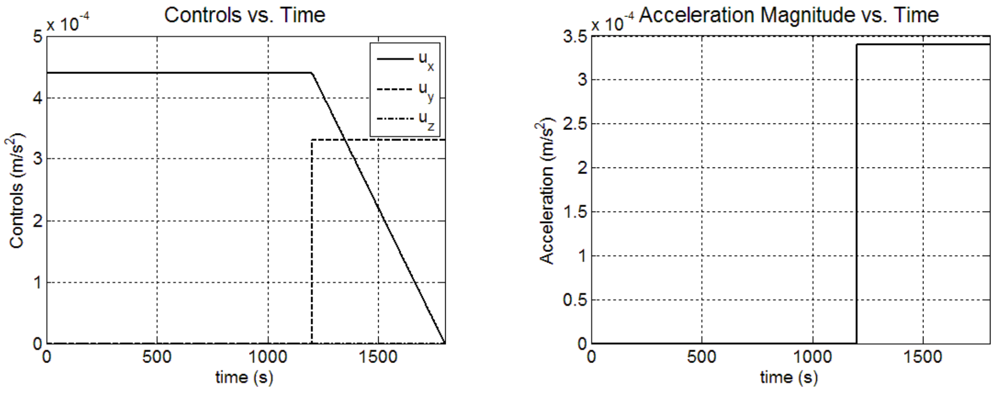

Figure 5.6: a) Controls vs. time; b) Acceleration vs. time

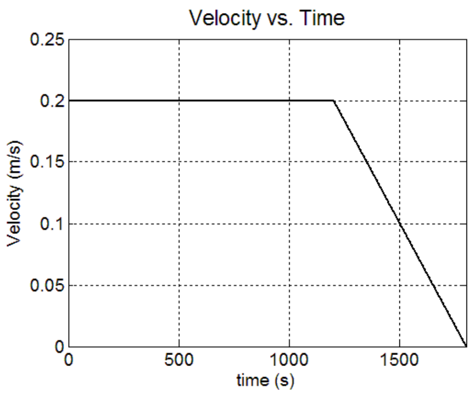

Figure 5.7: Velocity vs. time

The time required to complete the docking is $1800\,s$, with a total cost of $1.566 \cdot 10^{-4}\,\dfrac{m^2}{s^3}$, while in Ref. 8 the corresponding values are: $1885\,s$, $1.538 \cdot 10^{-4}\,\dfrac{m^2}{s^3}$. Discrepancies in cost and time are due to the fact that in Ref. 8 the final boundary conditions, are not exactly matched, whereas we are here maneuvering between two specified and accurately achieved positions and velocities.

6 Real-Time Validation

For the above reported simulations, the optimization algorithm was coded in Matlab-Simulink.

Furthermore, validation of the capability of our proposed approach to run in real time has been experimentally obtained. In particular, a desktop computer, running the Mathworks-XPC target operative system, has been used as target hardware by downloading the compiled version of the optimization software.

Figure 6.1 shows the Simulink model. It is composed by two main Embedded Functions. The Planner generates a sequence of nodes following the near-optimal $\ddot{s}$ profile. This sequence is feed forwarded to the dynamic inversion function (called "control generation"). On a Pentium II 800 Mhz machine, the algorithm run in real time at a sample frequency of $5\,Hz$.

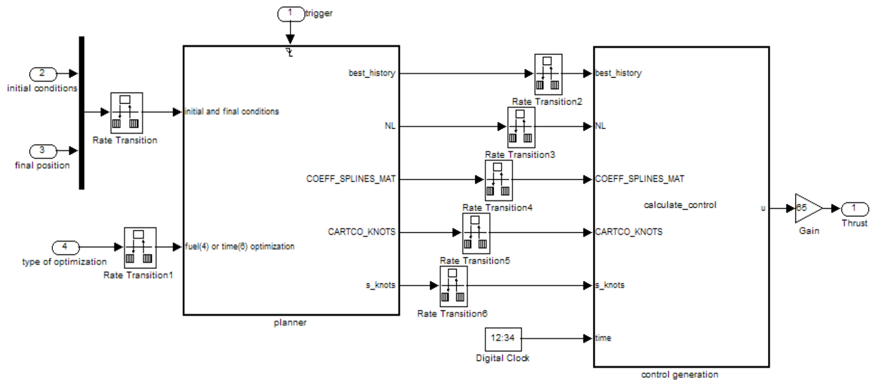

Figure 6.1: Simulink Model for Real Time testing

7 Conclusion

The present work uses a direct approach for real time sub-optimal control of spacecraft rendezvous and docking along a constrained path. The Hill-Clohessy-Wiltshire equations are used as basis for a dynamic inversion computation and the path is parameterized through cubic B-splines having the curvilinear abscissa as parameter. Dynamic programming operating on the curvilinear acceleration is proposed as optimization method in order to find the fuel optimal policy to drive the chaser spacecraft to he target spacecraft. The reliability of the algorithm has been assessed for different number of levels in the dynamic programming tree, and the results have been compared to an example reported in recent literature. Real Time implementation of the optimization algorithm has been tested on a Pentium II machine with a sample time up to 0.2 seconds.

Further developments include modifying the path by moving the splines control points in order to optimize it, and adding control bounds. Furthermore experiments are going to be performed on an hardware-in-the-loop laboratory test-bed ([19]).

References

[1] Clohessy, W. H., and Wiltshire, R. S. Terminal Guidance System for Satellite Rendezvous. *Journal of the Aerospace Sciences* **Vol. 27**(No. 9) (1960) 653–658.

[2] Tschauner J. Elliptic orbit rendezvous. *AIAA journal* **Vol. 5**(No. 6) (1967) 1110–1113.

[3] Inalhan, G., Tillerson, M., How, J. P. Relative dynamics and control of spacecraft formations in eccentric orbits. *Journal of Guidance, Control and Dynamics* **Vol. 1** (January-February) (2002).

[4] Izzo, D. Formation Flying linear modelling. *Proceedings of the* 5^{th} *International Conference On Dynamics and Control of Systems and Structures in Space* Cambridge, UK (14-18 July 2002) 283–289.

[5] Izzo, D., Sabatini, M., Valente, C. A new linear model describing formation flying dynamics under J2 effects. *AIDAA, XVII Congresso Nazionale* Roma (15-19 September 2003).

[6] Schaub, H., Alfriend, K.T. J2 invariant relative orbits for spacecraft formations. *In Goddard Flight Mechanics Symposium* (May 18-20 1999) paper no. 11.

[7] Vadali, S. R. An analytical solution for relative motion of satellites *Proceedings of the* 5^{th} *International Conference on Dynamics and Control of Structures and Systems in Space* (2002).

[8] Sabatini M., Bevilacqua R., Pantaleoni M., Izzo D. Numerical Search of Bounded Relative Satellite Motion *Journal of Nonlinear Dynamics and Systems Theory* **Vol. 6** (No. 4) (2006) 411–419.

[9] Guelman, M., Aleshin, M. Optimal Bounded Low-Thrust Rendezvous with Fixed Terminal-Approach Direction *Journal of Guidance, Control and Dynamics* **Vol. 24**(No. 2) (2001).

[10] Campbell, M. E. Planning Algorithm for Multiple Satellite Clusters. *AIAA Journal of Guidance, Control and Dynamics* **Vol. 26**(No. 5) (2003).

[11] Vadali, S.R., Vaddi, S.S. and Alfriend, K.T. An Intelligent Control Concept for Formation Flying Satellite Constellations. *International Journal of Nonlinear and Robust Control, dedicated to Formation Flying* **Vol. 12** (2002) 97–115.

[12] Shiller, Z., Dubowsky, S. Robot Path Planning with Obstacles, Actuator, Gripper, and Payload Constraints *The International Journal of Robotics Research* **Vol. 8**(No. 6) (1989) 3–18.

[13] Bernelli-Zazzera, F., Romano, M. Time-Optimal Motion for Robotic Manipulators with Obstacles Avoidance. *Proceedings of* 4^{th} *International Conference on Dynamics and Control of Structure in Space* Cranfield University, Cranfield, UK (May 1999).

[14] Yakimenko, O. A. Direct Method for Rapid Prototyping of Near-Optimal Aircraft Trajectories. *Journal of Guidance, Control, and Dynamics* **Vol. 23**(No. 5) (2000) 865–875.

[15] Bellman, R. *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, (1957).

[16] Cruciani, I., De Divitiis, N., De Matteis, G., Filippone, E. Autonomous Guidance for a Sub-Orbital Re-Entry Vehicle *Paper IAC-03-A.7.06, 54th International Astronautical Congress* Bremen, Germany, (September 2003).

[17] Miles, D. W. *Real Time Dynamic Trajectory Optimization with Application to Free-Flying Space Robots* Stanford University Ph. D. Thesis (1997).

[18] Mortensons, M.E. *Geometric Modeling.* John Wiley & Sons: New York (1985).

[19] Romano, M., Friedman, D. A. and Shay, T. J.. Laboratory Experimentation of Autonomous Spacecraft Approach and Docking to a Collaborative Target. *AIAA Journal of Spacecraft and Rockets* **Vol. 44**(No. 1) (2007).