TRAJECTORY GENERATION FOR A REMOTELY OPERATED VEHICLE

Sérgio Loureiro Fraga, João Borges Sousa, Fernando Lobo Pereira

{slfraga,jtasso,flp}@fe.up.pt
Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto PORTUGAL
Tel: +351225081539 Fax: +351225081443

Keywords: Trajectory generation, Remotely operated underwater vehicles, Differential flatness, Nonlinear Systems, Modeling.

Abstract

This paper addresses the problem of trajectory generation for a remotely operated vehicle (ROV). The ROV is a nonholonomic vehicle and has limited actuator capabilities. This means that the task of trajectory generation for the inspection of underwater structures is not a trivial one, and that it cannot be done without computer aided design tools. The approach is based on techniques developed for differential flat systems. The ROV model is presented and it is shown that it satisfies the differential flatness property. The paper details the architecture of the computer aided trajectory generation tool.

1 Introduction

In this paper we present our approach to the design of the trajectory specification, generation and tracking modules for the Inspection of Underwater Structures (IES) system (figure 1). IES is a collaborative project that involves the Administração dos Portos de Douro e Leixões (APDL), Faculdade de Engenharia do Porto and Instituto de Sistemas e Robótica - Pólo do Porto. Except for the ROV frame, hull and thrusters, all the other components and systems were designed and implemented at LSTS (Underwater Systems and Technology Laboratory). The IES system comprises the following subsystems: power, computer, motor control, navigation and image. The computer system consists of a PC-104 stack running the real-time operating system QNX, and a Windows based PC connected through an Ethernet cable. The PC-104 stack is housed in the main cylinder of the ROV, and controls the ROV systems through a CAN bus. The PC runs the operator console. The PC-104 computer system runs the command, control and navigation software. Basically, this computer accepts high-level commands from the console and informs the console about the state of the system. The navigation system includes the on-board sensors and an external acoustic navigation network. The on-board sensors are: magnetic compass, inclinometers, Inertial Motion Unit (IMU), Doppler Velocity Logger (DVL), Long Base Line (LBL) acoustic system, magnetic compass and depth cell [2][4].

One of the major innovations of the IES system with respect to commercially available ROV solutions is the consideration of two modes of operation: tele-operation and tele-programming. The IES ROV is able to receive motion commands through an umbilical cable from the console, which is mainly composed of a PC, a TV/Video set, and a joystick. In the tele-operation mode the ROV's pilot uses the joystick to control the vehicle's velocity. In this mode the trajectory generation module interprets, in real time, the joystick commands and transform them into feasible trajectories. In the tele-programming the pilot specifies with the aid of a graphical user interface a trajectory for the vehicle to follow. There are several trajectory specification models including for example waypoints. The trajectory generation module is able to convert the specification into feasible trajectories for ROV to follow. The computation of the trajectory is made at the console and it is sent to the on board computer, where the controller is implemented.

Trajectory generation plays an important role in defining the overall performance of a ROV control system. A properly designed trajectory generation system facilitates the task of the controllers since it should prevent the system from saturating, and thus operating in open-loop. One way to avoid input saturation is to generate feasible trajectories. The design of trajectory generation modules has to take into account the cinematic and dynamic constraints of the ROV in order to generate feasible trajectories.

The problem of trajectory generation for nonlinear systems with nonholonomic constraints [6][5] represents a considerable challenge to control design. In fact, there is no universal technique that can be readily applied to this end. In a differential flat system we can take advantage of techniques from differential geometry for the generation of feasible trajectories. But one thing is trajectory generation, and the other is control design and performance. These are intimately related. We use a three level architecture for trajectory generation and tracking [11] (figure 2):

- User interface for output specification. This module has a graphical user interface to assist the operator in the specification of the desired values for the ROV outputs. Upon validation, the pilot's objectives (represented as system outputs) are sent to the trajectory generation module.
- Trajectory generation. This module computes a feasible trajectory for the ROV, which may be optimized over some

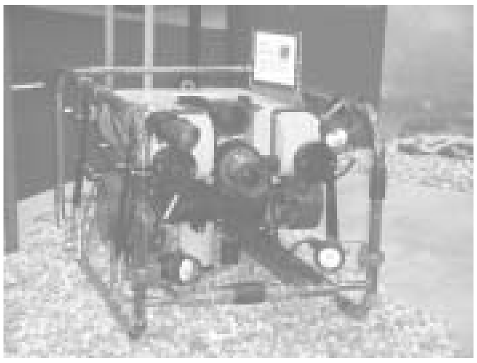

Figure 1: The remotely operated vehicle

factor or variable to improve the overall system performance. The output of this module is a complete description of the desired state and input of the system.

- Controller. This module controls the actuators to track the trajectory defined by the trajectory generation module. The output of this module is a control signal overlaying the input signal, defined by the trajectory generation module, in order to eliminate drifts or perturbations on the structure.

The main focus of this article is the trajectory generation module. For a thorough discussion of the other modules see [4][2].

This paper is organized as follows. Section ***2*** presents the dynamical and kinematical model of the ROV. Section ***3*** introduces differential flat systems. Section ***4*** presents two approaches for trajectory generation for ROVs. Finally, section ***5*** presents some concluding remarks.

2 Modelization of the ROV

The ROV is a rigid body with six degrees of freedom. Its motion can be described using either an inertial coordinate system XYZ or a body fixed coordinate system $X_0Y_0Z_0$ (figure 3). Since the ROV moves slowly in the water, a earth fixed coordinate system can be defined to be inertial. On the other hand, the body fixed coordinate system has its origin in the ROV's centre of mass and its axis' directions coincide with the vehicle's principal inertial axis:

- X_o - longitudinal axis (from aft to fore)
- Y_o - transverse axis (from port side to starboard)
- Z_o - vertical axis (from top to bottom).

The vehicle's position and velocity is represented in the inertial and body fixed coordinate system. We adopted the notation form ”The Society of Naval Architects and Marine Engineers (SNAME)” (1950). According to SNAME, the motion of underwater vehicles in six degrees of freedom can be described by the following vectors:

$$\begin{aligned}
\eta &= \left[\eta_1^T, \eta_2^T\right]^T; \ \eta_1 = [x, y, z]^T; \quad \eta_2 = [\phi, \theta, \psi]^T; \\
v &= \left[v_1^T, v_2^T\right]^T; \ v_1 = [u, v, w]^T; \quad v_2 = [p, q, r]^T; \\
\tau &= \left[\tau_1^T, \tau_2^T\right]^T; \ \tau_1 = [X, Y, Z]^T; \ \tau_2 = [K, M, N]^T.
\end{aligned}$$

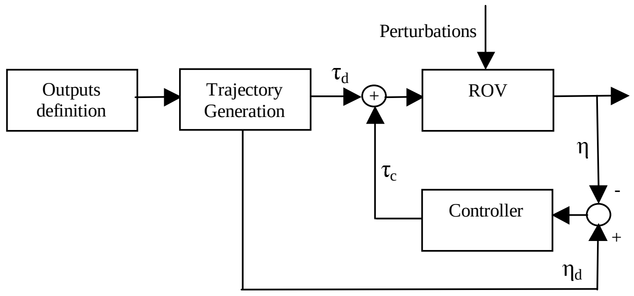

Figure 2: Modules of the trajectory generation system

Thus, η represents the inertial coordinates of the vehicle, ν represents the linear and angular velocities in body fixed coordinates and, finally, τ are the forces and moments applied on the vehicle in body fixed coordinates.

The kinematical equations relating the state variables between the inertial and body fixed coordinate system are given by:

$$\begin{bmatrix} \dot{\eta}_1 \\ \dot{\eta}_2 \end{bmatrix} = \begin{bmatrix} J_1(\eta_2) & 0_{3\times3} \\ 0_{3\times3} & J_2(\eta_2) \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \Leftrightarrow \dot{\eta} = J(\eta)v \tag{1}$$

with ($t. = \tan(.)M$, $s. = \sin(.)$ and $c. = \cos(.)$)

$$J_1(\eta_2) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{2}$$

and

$$J_2(\eta_2) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix}. \tag{3}$$

The matrix of equation (3) has a singularity for $\theta = \pm\frac{\pi}{2}$ and is not a rotational matrix, i.e. it does not belong to $SO(3)$. This singularity does not bring problems for the ROV's control since it never reaches a state where $\theta = \pm\frac{\pi}{2}$. On the other hand, J_1 matrix is a rotational matrix, i.e. $J_1 \in SO(3)$.

The dynamical model of a rigid body moving in six degrees of freedom is a well known model [7] and may be represented by the following equation (body fixed coordinates):

$$M_{RB}\dot{v} + C_{RB}(v)v = \tau_{RB}. \tag{4}$$

In the preceding equation, $v = [u, v, w, p, q, r]^T$, $C_{RB}(\nu)$ represents the Coriolis forces and centripetal terms, then $\tau_{RB} = [X, Y, Z, K, M, N]^T$ and, finally, M_{RB} represents the inertial matrix of the rigid body. Taking into account the previous definitions about body fixed coordinate system, M_{RB} is given by:

$$M_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_x & 0 & 0 \\ 0 & 0 & 0 & 0 & I_y & 0 \\ 0 & 0 & 0 & 0 & 0 & I_z \end{bmatrix} \tag{5}$$

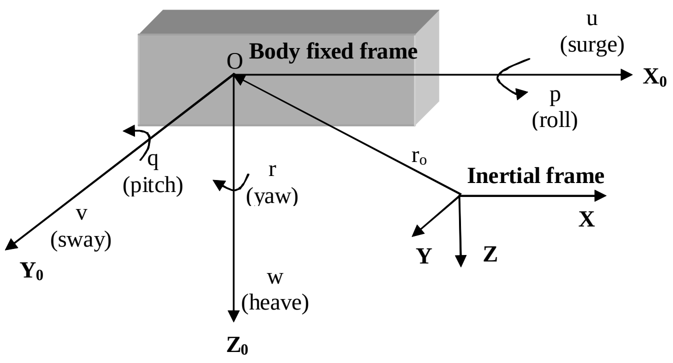

Figure 3: Body fixed and inertial coordinates systems

where m is the vehicle's mass and I_x, I_y, I_z are the inertial moments around the axis x, y, z, respectively. The term $C_{RB}(v).v$ is given by $\begin{bmatrix} \omega^b \times mv^b \\ \omega^b \times I\omega^b \end{bmatrix}$ with $v^b = \begin{bmatrix} u & v & w \end{bmatrix}^T$ and $w^b = \begin{bmatrix} p & q & r \end{bmatrix}^T$ and τ_{RB} are exterior actuations.

The external forces acting on the ROV are included in the right side of equation (4). These forces can be classified in the following categories [1][4]: radiation-induced forces τ_H (added inertia, hydrodynamic damping, restoring forces), environmental forces τ_E (underwater currents, waves, wind) and propulsion forces τ (thrusters and propeller forces, control surfaces and rudder forces).

Considering the previous forces, the resulting force acting on the vehicle is given by:

$$\tau_{RB} = \tau_H + \tau_E + \tau. \qquad (6)$$

The hydrodynamic forces τ_H has the following expression:

$$\tau_H = -M_A\dot{v} - C_A(v)v - D(v)v - g(\eta). \qquad (7)$$

The term $-M_A\dot{v} - C_A(v)v$ models the added mass due to the inertia of the surrounding fluid, while the term $-D(v)v$ represents the total hydrodynamic damping [1]. Finally, the component $-g(\eta)$ represents the restoring forces due to the vehicles' weight and buoyancy and has the following expression:

$$g(\eta) = \mathrm{col}(0, 0, 0, -z_B B \cos\theta \sin\phi, -z_B B \sin\theta, 0), \qquad (8)$$

where z_B represents the distance between the application point of the weight and the buoyancy force and B is the impulsion force, which is equal to the weight W for this ROV.

Substituting equation (6) into (4) together with equation (7), we obtain the vehicle's dynamic model:

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau_E + \tau \qquad (9)$$

where $M = M_A + M_{RB}$ and $C(v) = C_{RB}(v) + C_A(v)$. A full description of these matrices can be found in [1].

It was not considered the model of ROV's actuators, namely the model of the thrusters and the electric motors. It was not also considered the model of environment forces, such as aquatic currents, waves and wind. Despite their influence in the vehicle's behaviour, not including these models doesn't affect the desired results for this work. On the other hand, this simplification reduces considerably the implementation of software equations allowing more control on the trajectory generation results.

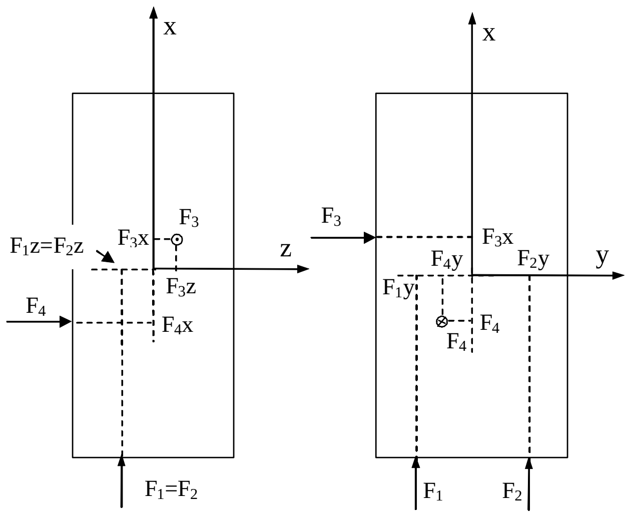

Figure 4: Applied forces on the ROV by the thrusters

The dynamical model of the ROV is presented in equation (9) in body fixed coordinates and can also be represented in the inertial coordinate system by using the kinematical equation (1):

$$M_\eta(\eta)\ddot{\eta} + C_\eta(v,\eta)\dot{\eta} + D_\eta(v,\eta)\dot{\eta} + g_\eta(\eta) = \tau_\eta. \qquad (10)$$

where

$$\begin{aligned} M_\eta(\eta) &= J^{-T}(\eta) M J^{-1}(\eta) \\ C_\eta(v,\eta) &= J^{-T}(\eta)\left[C(v) - MJ^{-1}(\eta)\dot{J}(\eta)\right]J^{-1}(\eta) \\ D_\eta(v,\eta) &= J^{-T}(\eta) D(v) J^{-1}(\eta) \\ g_\eta(\eta) &= J^{-T}(\eta) g(\eta) \\ \tau_\eta(\eta) &= J^{-T}(\eta)\tau. \end{aligned} \qquad (11)$$

After presentation of dynamic and cinematic equations it is convenient to show how the ROV's thrusters affect the variable τ. In figure 4are represented the forces applied in the ROV by the thrusters. The ROV does not have rudders so its motion is only affected by thrusters. The relationship between the forces applied by the thrusters and its effect in variable τ is given by:

$$\begin{bmatrix} X \\ Y \\ Z \\ K \\ M \\ N \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -F_3z & F_4y \\ F_1z & F_2z & 0 & -F_4x \\ -F_1y & -F_2y & F_3x & 0 \end{bmatrix} \cdot \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}. \qquad (12)$$

3 Differential flatness

In this section, we show the utility of differential flatness proprieties for trajectory generation [9][10]. A system is differentially flat if it is possible to find a set of variables (equal in number of the inputs), called flat outputs, such that all state variables and input become defined from them without integration. Let $\eta \in R^n$ be the system state and $\tau \in R^m$ its input, then the system is flat if it is possible to find flat outputs $z \in R^m$ in the form:

$$z = \mathbf{Z}(\eta, \tau, \dot{\tau}, ..., \tau^{(l)}) \tag{13}$$

such that

$$\begin{aligned} \eta &= \boldsymbol{\eta}(z, \dot{z}, ..., z^{(q)}) \\ \tau &= \boldsymbol{\tau}(z, \dot{z}, ..., z^{(q)}). \end{aligned} \tag{14}$$

where $\mathbf{Z}, \boldsymbol{\eta}, \boldsymbol{\tau}$ are mapping functions between flat outputs, state variables and input.

Differentially flat systems are interesting when flat outputs represent the desired behaviour of the system and trajectory planning can be made from these variables. This approach is especially interesting when the flat outputs have a physical meaning and are not an arbitrary combination of state variables and input. The problem is that there is no systematic method to determine the flat outputs. Hence, finding flat outputs with physical meaning for a particular system can be a hard task [8]. Usually, the dimension of the flat outputs is lower than system dimension, which means an improvement on the steering efficiency. For differentially flat systems we are able to transform the system such that the equations of motion for the flat outputs variables become trivial. For instance, in the case of a rigid body with six degrees of freedom and three inputs, if the flat outputs can be found then it would be possible to steer the body just as if it was a point. Since flat outputs trajectories are completely free, the only constraints that should be imposed refer to the desired initial and final configuration of the system and bounds on the derivatives of the trajectories. Other constraints, such as, input saturation can be converted to the flat outputs space by imposing limitations on the curvature of the trajectories or bounds on the higher order derivatives.

Next, we show that ROV's model is differentially flat. As stated before, its model in inertial coordinates is given by:

$$M_\eta(\eta)\ddot{\eta} + C_\eta(v,\eta)\dot{\eta} + D_\eta(v,\eta)\dot{\eta} + g_\eta(\eta) = \tau_\eta. \tag{15}$$

The system state η represents the ROV's inertial positions. We can easily check that the state variables are also the flat outputs since the system input (τ_η) becomes defined from those variables without integration [8][3].

As pointed out before the number of flat outputs must be equal to the dimension of the input space. This is not the case of our ROV. The IES ROV is under-actuated. We have actuation in four degrees of freedom. In this situation, the trajectory generation is based on the fact that roll an pitch are stable (due to the large distance between the application point of the weight and the buoyancy forces) and assuming that it is not a requisite to define a trajectory for these degrees of freedom, so their trajectories are always assumed to be zero. The actuation on these two degrees of freedom is also negligible since the components F_{ij} in equation (12) affecting the variables K, M are smaller when compared to the actuation on the others degrees of freedom. Hence, we compute the desired actuation in all degrees of freedom but we only apply four forces on the ROV. We expect the actuation on roll and pitch to be negligible. We confirmed this hypothesis from the simulation runs. We do not consider

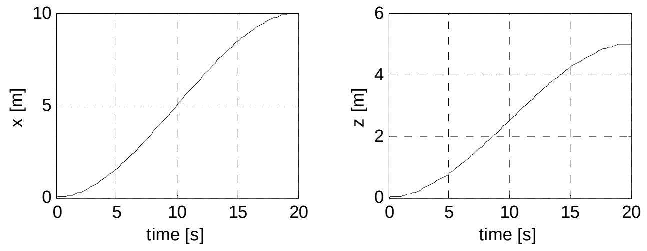

Figure 5: Flat outputs parametrization

K, M in the computation of these forces. This may result in some performance degradation. The computation of the forces to apply to each propeller is given by:

$$\begin{bmatrix} X \\ Y \\ Z \\ N \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -F_1 y & -F_2 y & F_3 x & 0 \end{bmatrix} \cdot \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}. \tag{16}$$

The effect of the actuators on ϕ, θ is small but we will present a technique that minimize the actuation in these two degrees of freedom. We expect a better performance since the equation (16) becomes a better approximation of the physical reality. To sum up, the ROV's flat outputs are the state variables and an approximation is made when computing the input from those flat outputs.

4 Trajectory generation for the ROV

In this section, we present two trajectory generation methods based on differential flatness [10]. The problems addressed in this paper are finite horizon and anti-causal, i.e. some information about the trajectory's future is necessary. This information can be the complete form of the trajectory or waypoints. The temporal scheme of the trajectory generation architecture defines the desired trajectory for all states and inputs at a lower rate than that of the controllers. These, in turn, track the defined trajectory at a higher rate ($10ms$).

We implemented the algorithms in ANSI C and used the libraries from the software package "C-Routines for Trajectory Generation for Flat and Approximately Flat Systems" [9] developed at "California Institute of Technology".

In this paper we consider the problem of finding a trajectory for the state variables and the system inputs from the flat outputs trajectory, in a finite time horizon $[t_0, t_f]$. The trajectories of the flat outputs will be approximated by polynomials to compute its derivatives symbolically to enhance the computational efficiency. The model used for this problem is the one presented in equation (15) since, in this case, it is convenient to define the trajectory in the inertial coordinate system to facilitate the task of the pilot.

The first method is the simplest one since it refers to guide the vehicle from a point of its state space to another one. In this problem we are given the entire state at t_0 and at t_f. Assuming the inputs and its derivatives are also defined at both times, we can compute the flat outputs for these instants (equation (14)). After determination of the initial and final values of the

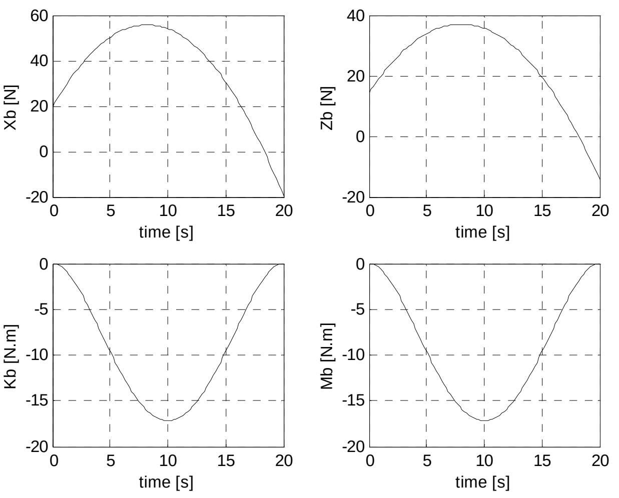

Figure 6: Computed inputs

flat outputs it is possible to parameterize them for an interval $[t_0, t_f]$ as follows:

$$z_i(t) = \sum_j A_{ij}\phi_j(t) \tag{17}$$

where $\phi_j(t)$ represents a basis of polynomial functions, which through a linear combination given by the matrix A allows the computation of the flat outputs at $[t_0, t_f]$. The values of the A_{ij} coefficients can be obtained by solving the following system of equations:

$$\begin{array}{ll} z_i(t_0) = \sum_j A_{ij}\phi_j(t_0) & z_i(t_f) = \sum_j A_{ij}\phi_j(t_f) \\ \cdots & \cdots \\ z_i^{(l)}(t_0) = \sum_j A_{ij}\phi_j^{(l)}(t_0) & z_i^{(l)}(t_f) = \sum_j A_{ij}\phi_j^{(l)}(t_f). \end{array} \tag{18}$$

Let p be the number of polynomials of basis ϕ_j. Then it is necessary to specify the initial and final values for the flat outputs until the l derivative ($p = 2(l+1)$) in order to determine all A_{ij} coefficients. Thus, the number of coefficients to be computed in matrix A will be $2(m(l+1))$.

Since a trajectory for the flat outputs has already been defined between the instants t_0, t_f by the previous parameterization, now it is necessary to compute the state and input of the system from that trajectory. This can be done by choosing several points along the flat outputs trajectory in order to compute the corresponding state and input. Increasing the number of computed points N, means an improvement on the system accuracy since the discretization is made with a higher frequency. In the ROV, the flat outputs coincide with vehicle's state so it is only necessary to compute its input using equation (15). However, it should be noted that a trade-off has to be resolved because increasing the number of computed points will increase the computation time, which is also determined by the time necessary to perform the flat outputs parameterization. The parameterization time is much smaller on systems where nonlinear systems of equations must be resolved to compute the state and the input. In the ROV system, it is not necessary to resolve systems of equations since analytical expressions exist to obtain the system input.

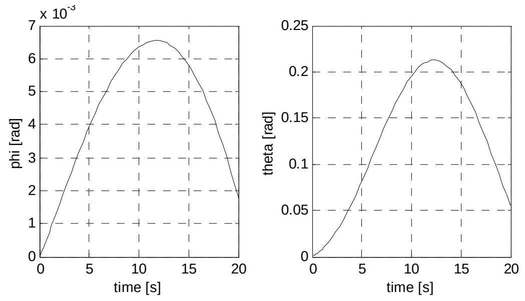

Figure 7: ROV's roll and pitch behaviour

In figure 5 we show a parameterization (with $p = 6$) of the flat outputs from two given points: an initial point $(x, y, z, \phi, \theta, \psi) = (0, 0, 0, 0, 0, 0)$ at $t_0 = 0s$ and a final point $(x, y, z, \phi, \theta, \psi) = (10, 0, 5, 0, 0, 0)$ at $t_f = 20s$. At these points the initial and final velocity is defined to be zero. It is only shown the trajectories for x, z because the others variables are always zero. After performing the flat outputs parameterization, it is now possible to compute the input, which is able to implement the parameterized trajectory. The number of points used to perform that task were $N = 100$ and the computed inputs are depicted in figure 6 (in body coordinates). Again, Y, N is not presented since these inputs are always zero.

It should be noted that this trajectory will be the input of the control module mentioned in section ***1***.

As it is possible to verify in figure 6, the desired values for K, M are slightly different of zero. This means a degradation of performance as discussed in section ***4***. As a result of computing thrusters forces using equation (16), the ROV's roll and pitch will have the shape depicted in figure 7. As expected, the approach brings loose of performance in roll and pitch, which is a consequence of ROV's under actuation.

In order to minimize this drawback, it is possible to generate trajectories minimizing the actuation effort on K, M. To implement this concept, trajectories will be generated through the following minimization cost function:

$$\min_A \int_{t_0}^{t_1} (z_d(s) - A\phi(s))^T * (z_d(s) - A\phi(s)) + \lambda P(K, M) ds \tag{19}$$

where $z_d(t)$ represents the desired trajectory for the flat outputs, for instance the trajectory defined by the previous algorithm, $z(t) = A\phi(s)$ is the output trajectory ($z(t)$ still be a linear combination of a polynomial basis with $p = 10$ polynomials) and P represents a function which penalizes the actuation effort on the variables K e M (in this example $P = K^2 + M^2$). The trajectory z is a trade-off solution between trajectory tracking of $z_d(t)$ and the actuation effort in K, M, which is obtained by a suitable choice of λ. If λ is chosen to be close to zero, this means that we give more preference to trajectory tracking. Otherwise, we prefer to reduce the actuation effort.

The main design issue is the choice of a suitable λ. This choice is left to the pilot who will be able to go through an iterative

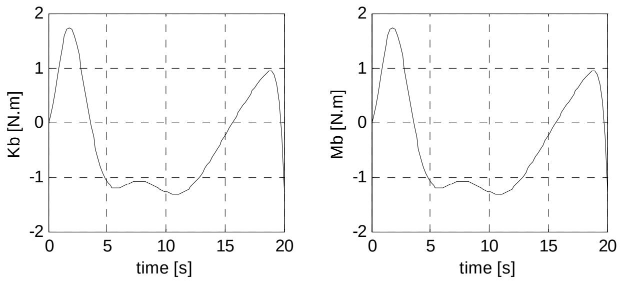

Figure 8: Results of input minimization on K, M

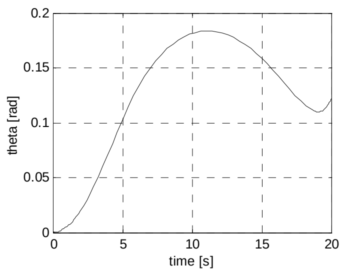

Figure 9: Pitch behaviour after minimizing actuation on K, M

procedure with the user interface. Detailed information about the output trajectory with a chosen λ should be given to the pilot before applying the trajectory on the vehicle. This way the ROV's pilot can refine as much as we want the value λ, and obtain a better overall performance.

The results of trajectory computation with $\lambda = 0,05$, with $z_d(t)$ being the same trajectory presented in figure 5, are depicted in figure 8, where it is possible to see a significant reduction on K, M values. This reduction causes a slightly degradation in trajectory tracking, namely in x, z, but, as expected, results in a lower perturbation on θ (figure 9).

The main drawback of the previous approach is, definitively, the computational power required to perform the minimization presented in equation (19). This fact may prevent the application of this technique in real-time applications.

5 Conclusions

This paper has reported an approach to the problem of trajectory generation for a ROV. The architecture of the trajectory generation and control system consists of three modules: 1) output specification; 2) trajectory generation; 3) controller. The main advantage of this architecture is its modularity: we developed the three modules separately, and further improvements can be implemented independently.

The main focus of this article is the design of the trajectory generation module. This module takes into account ROV's dynamical and kinematical model to steer the vehicle without violating its constraints. It is shown that ROV's model is differentially flat under some mild assumptions: the stability of ROV in the pith and roll modes and the negligible influence of the thrusters on these degrees of freedom.

We presented two trajectory generation techniques that take advantage of the differential flatness of the ROV model. The first one steers the vehicle from an initial state configuration to a final one, while the second one steers the vehicle minimizing the effect on K, M variables. We present simulation results that confirm the validity of our assumptions, namely in what concerns the low sensitivity of the ROV's roll and pitch when we use the second technique. However, this improvement requires more computational time, which may not be feasible for real time applications.

6 Acknowledgments

This material is based upon work funded by the Programa PRAXIS XXI - Medida 3.1b) (IES project) and by Ministério da Defesa, Portugal. Sérgio Fraga was funded by the Fundação para a Ciência e Tecnologia (FCT) and the European Social Fund.

References

[1] Thor I. Fossen. *Guidance and Control of Ocean Vehicles*. John Wiley and Sons, 1995.

[2] Sérgio Loureiro Fraga, João Sousa, Anouck Girard, and Alfredo Martins. An automated maneuver control framework for a remotely operated vehicle. In *MTS/IEEE Oceans 2001*. IEEE, 2001.

[3] Sérgio Loureiro Fraga, João Borges Sousa, and Fernando Lobo Pereira. A framework for the automation of a remotely operated vehicle. In *10th IEEE Mediterranean Conference on Control and Automation*, Junho, 2002.

[4] Rui Manuel Ferreira Gomes. Modeling and control of underwater vehicles. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2002.

[5] Jean-Claude Latombe. *Robot Motion Planning*. KAP, 1993.

[6] Jean-Paul Laumond. *Robot Motion Planning and Control*. Springer, 1998.

[7] Richard Murray, Zexiang Li, and S. Sastry. *Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[8] Richard Murray, Muruhan Rathinam, and Willem Sluis. Differential flatness of mechanical control systems: A catalog of prototype systems. In *ASME International Mechanical Engineering Congress and Exposition*, 1995.

[9] Michiel J. Van Nieuwstadt. *Trajectory Generation for Nonlinear Control Systems*. PhD thesis, California Institute of Technology, Pasadena, California, 1997.

[10] Michiel Van Nieuwstadt and Richard Murray. Approximate trajectory generation for differentially flat systems with zero dynamics. In *IEEE Conference on Decision and Control, New Orleans*, 1995.

[11] Pravin Varaiya. Towards a layered view of control. In *36th IEEE Conference on Decision and Control*. IEEE, 1997.