Proc. of the 1996 SAMCEF User Conference, June 4-5, 1996, Liège, Belgium.

THE DEVELOPMENT OF FEEDBACK CONTROL TOOLS FOR MECANO TO STUDY FLEXIBLE SPACE ROBOTIC SYSTEMS

E. MARTIN E. PAPADOPOULOS J. ANGELES

Dept of Mech. Eng. and Centre for Intelligent Machines,
McGill University, Montreal, Quebec, Canada, H3A 2K6
emartin@cim.mcgill.ca, egpapado@cim.mcgill.ca, angeles@cim.mcgill.ca

Abstract—Space manipulators mounted on an on-off thruster-controlled base are envisioned to assist in the assembly and maintenance of space structures. When handling large payloads, manipulator joint and link flexibility become important, for it can result in payload-attitude controller fuel-replenishing dynamic interactions. The dynamics of these systems is rather complicated, expecially when link flexibility must be modelled, and thus, the most convenient way of dealing with such systems is probably the finite-element method. The commercial package *Mecano*, a module of the general-purpose finite element software *Samcef*, has proven its ability to accurately model systems with flexible elements. However, this package has no built-in functions allowing the use of control techniques to analyze the systems at hand. This paper is intended to introduce the new control tools developped for *Mecano* to model the use of on-off thrusters, time delays, as well as second-order filters and state-estimators. An example is used to illustrate the application of these control techniques.

1 INTRODUCTION

Robotic devices in orbit will play an important role in space exploration and exploitation. The mobility of such devices can be enhanced by mounting them on free-flying bases, controlled by on-off thrusters. Such robots introduce a host of dynamic and control problems not found in terrestrial applications. When handling large payloads, manipulator joint or structural flexibility becomes important and can result in payload-attitude controller fuel-replenishing dynamic interactions. Such interactions may lead to control system instabilities, or manifest themselves as limit cycles [1].

The CANADARM-Space Shuttle system is the only operational space robotic system to date. Its Reaction Control System (RCS), which makes use of on-off thrusters, is designed assuming rigid-body motion, and uses single-axis, thruster switching logic based on phase-plane techniques. This approach is common in the design of thruster-based control systems. However, the flexible modes of this space robotic system have rather low frequencies, which continuously change with manipulator configuration and payload, and can be excited by the RCS activity. The performance degradation of the RCS due to the deployment of a flexible *payload*, with or without the CANADARM, has been studied [2]. A new design for the RCS

was developed to reduce the impact of large measurement uncertainties in the rate signal during attitude control, thereby increasing significantly the performance of the RCS for rigid-body motion [3]. However, the flexibility problem was not addressed. Currently, the method for solving these problems consists of performing extensive simulations. If dynamic interactions occur, corrective actions are taken, which include adjusting the RCS parameter values, or simply changing the operational procedures [2]. The consequences of such interactions can be problematic, since fuel is an unavailable resource in space; hence, classical attitude controllers must be improved to reduce the possibility of such dynamic interactions.

This problem was studied using a single-mode, linear translational mechanical system to approximate the dynamic behaviour of a two-flexible-joint manipulator mounted on a three-degree-of-freedom (dof) base with a constant damping ratio of the system [4], and with a variable one [5]. A state-estimator and design guidelines were suggested to minimize such undesirable dynamic interactions, as well as thruster fuel consumption. These results were validated using a more realistic model with rotational dof [6], but the dynamics of such systems is very complicated, especially when link flexibility must be modelled. A convenient way of dealing with these systems is the finite-element method. The commercial package, *Mecano*, a module of *Samcef*, was chosen due to its ability to accurately model systems with flexible elements. *Samcef* is a finite element software package developed at the Laboratoire de Techniques Aéronautiques et Spatiales (LTAS) of Université de Liège, Belgium. Besides *Mecano*, this finite element package includes modules for the dynamic analysis of rotating structures, thermal analysis, fracture mechanics, analysis of structures in composite materials, analysis of cable strutures, viscoplasticity analysis, and structural optimization. However, this package has no built-in functions allowing the use of control techniques to analyze the systems at hand. Since the main objective of our research is to develop control methods that are intended to reduce the undesired effects of dynamic interactions, the use of control techniques is mandatory. This problem was overcome by programming our own control subroutines via the Samcef user-element. This paper is intended to introduce the new control tools developped for *Mecano* to model the use of on-off thrusters, time delays, as well as second-order filters and state-estimators. The application of these control techniques is illustrated using the model of a spacecraft with a one-link manipulator mounted on it [6]. Simulation results are presented and compared with those obtained using *Matlab*.

2 CONTROL THEORY

2.1 GENERAL BACKGROUND

Generally, a control system is composed of three main parts: the plant, the controller, and the observer or state-estimator. The plant denotes the physical system under control. The plant usually provides output signals measured by sensors, and admits input signals generated by actuators, in order to modify its performance.

The role of the controller is to synthesize the control strategy, i.e., to derive the command signals for the actuators in response to the sensor outputs. The controller may be implemented using analog devices, but digital control has become prevalent, its elements including digital electronic cards and computer software. Finally, when specific outputs are required by the controller, but not readily available by sensors, an observer or state-estimator is included to obtain an estimate of the required output, based on the available signals and a model of the plant.

An important part of a control system implementation is the design of the controller and the state estimator. First, a good model of the plant is required in order to simulate properly the dynamics of the plant with the aid of software. This dynamics, like those of the controller and the observer, are represented mathematically by a set of differential equations. In modern control theory, these differential equations are usually represented with a system of first-order equations of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{1a}$$

while the output takes the form

$$\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}) \tag{1b}$$

where $\mathbf{u}$ and $\mathbf{y}$ are, respectively, the input and output vectors of the system, while $\mathbf{f}$ and $\mathbf{g}$ are, in general, nonlinear functions of the state variables $\mathbf{x}$ of the system and the input $\mathbf{u}$. In classical control theory, where linear systems are involved, the transfer function concept is also widely used. A transfer function is defined as the ratio of the Laplace transform of the output variable to the Laplace transform of the input variable with all initial conditions assumed to be zero. This transfer function represents a relation describing the dynamics of the system under consideration.

There are basically two types of controller structures. The first one, the open-loop control, consists of a system where we give a specified input $\mathbf{u}$, which may be based on a desired output of the system and a good mathematical model, but does not use any information of the actual states of the system. This kind of control actions, shown in Fig. 1(a), may not work in the presence of parameter variations in the system, or in the presence of disturbances. To overcome this problem, closed-loop control has been developed. In this case, shown in Fig. 1(b), the output $\mathbf{y}$ of the plant has a direct influence on the control input $\mathbf{u}$, since this control input can be based on the error $\mathbf{e}$ between the desired output $\mathbf{y}_d$ and the actual output. Therefore, closed-loop control is relatively insensitive to parameter variation or disturbances; besides, the overall dynamics of the system can be modified as desired. However, the stability of the system can be affected by the chosen control scheme, the implementation of the controller being more complicated and expensive than its open-loop counterpart.

Mecano, which is based on the finite element method, was not developed to easily design control systems. Its main feature is the modelling of a mechanical systems—the plant— including flexible bodies. External inputs can be supplied to the system to study its dynamic behaviour under different excitations defined by the

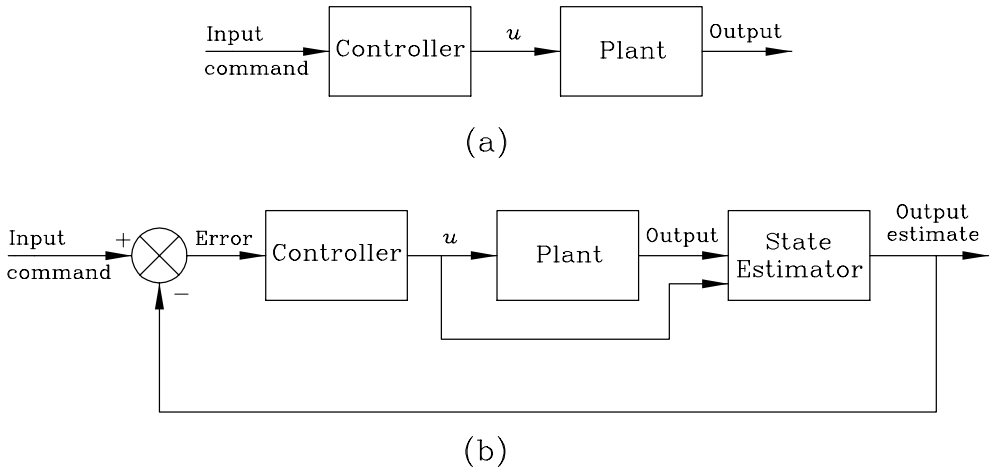

Figure 1: (a) Open-loop control, (b) Closed-loop control.

user. *Mecano* can be readily used for the analysis of open-loop control when the input does not involve any dynamics. However, the analysis of closed-loop system becomes cumbersome and additional work is required. An input that is a function of the output variables of the plant can be given through the user-function of the ".SOL" command. However, this user-function cannot involve any dynamics, the user being left with the general user-element to describe the dynamics that may be contained in the controller or the observer. This problem will be ascertained in Section 3 to see how *Mecano* can be used to simulate a control system involving feedback control and state-estimators. In the next subsection, a control system similar to the one used on the Space Shuttle for the control of its attitude is described as an example.

2.2 ATTITUDE CONTROL WITH ON-OFF THRUSTERS

In this subsection, we present a typical control system to control the attitude of a spacecraft using thrusters. Currently available technology does not allow the use of proportional thruster valves in space, and thus, the classical PD and PID control schemes cannot be used in this application. Therefore, spacecraft attitude and position are controlled by the use of on-off thruster valves, that introduce nonlinearities.

The usual scheme to control a spacecraft with on-off thrusters is based on the error phase plane, defined as that with spacecraft attitude error e and error-rate $\dot{e}$ as coordinates. The on-and-off switching is determined by switching lines in the phase plane and can become complex, as is the case in the phase plane controller of the Space Shuttle [2]. To simplify the switching logic, two switching lines with equations $e + \lambda\dot{e} = \pm\delta$ can be used. The deadband limits $[-\delta, \delta]$ are determined by attitude limit requirements, while the slope of the switching lines, by the desired rate of convergence towards equilibrium and by the rate limits. This switching logic can be represented as a relay with a deadband, where the input is $e + \lambda\dot{e}$, the left-hand side of the switching-line equations [4].

To compute the input to the controller, the attitude and the rate of the spacecraft

are required. Using current space technology, both states can be obtained by sensor readings. However, it can happen that only the attitude is available and then, the velocity must be estimated. In this case, we assume that only the attitude is available from sensors and a state-estimator is used to obtain the required velocity. This attitude signal is passed through a second-order filter of transfer function

$$G_f(s) = \frac{\omega_f^2}{s^2 + 2\zeta_f \omega_f s + \omega_f^2} \tag{2}$$

to eliminate high-frequency noise. The attitude signal is also differentiated while passed through a second-order filter $G_{se}(s)$, as the one defined in Eq. (2), to obtain an estimate for the velocity. A time delay τ has been included to account for the delay between the time a sensor reads a measurement, and the time this measurement is used. Since this delay is more significant than the delay of turning on or off the thrusters, only a sensor time delay is included. This model is presented in Fig. 2, where, clearly, three parts can be identified. The plant is modelled using built-in elements of *Mecano*. However, the controller and the state estimator, which includes dynamics, have to be modelled by means of the user-element of *Mecano*.

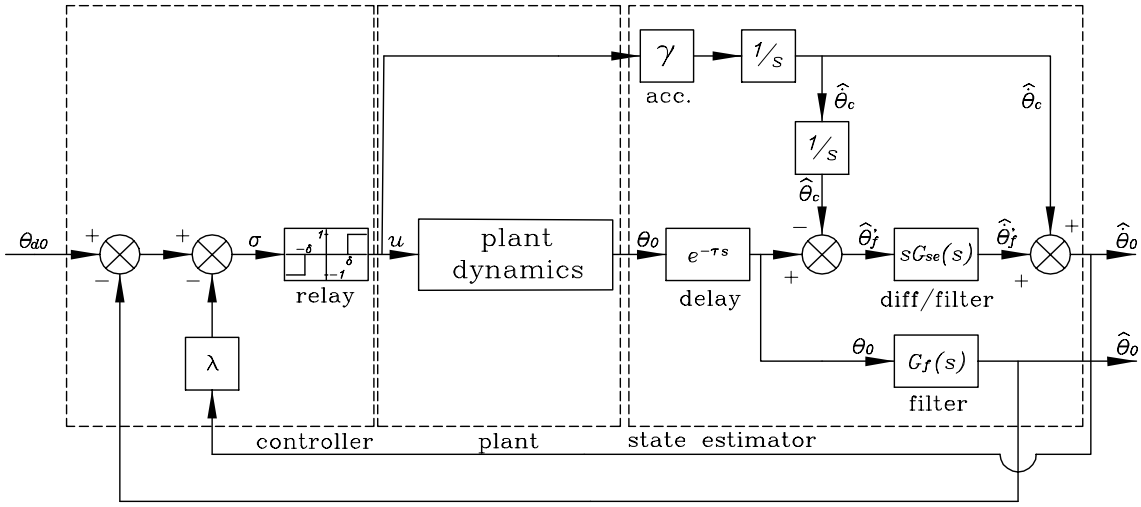

Figure 2: An on-off thruster attitude control system.

3 CONTROL TOOLS FOR MECANO

3.1 TRANSFORMATIONS OF EQUATIONS

In this subsection, it is shown how to transform a general control system with feedback in a form that can be analyzed with *Mecano*. In order to apply the user-element of *Mecano*, the system of equations must be cast in the form of a vector second-order equation, namely,

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{g}^{int} = \mathbf{g}^{ext} \tag{3}$$

The output signals of the user-element Fortran subroutine are the inertial forces $\mathbf{f}_{ine}$, the internal forces $\mathbf{f}_{int}$ and the iteration matrix $\mathbf{S}$, which are given by

$$\mathbf{f}_{ine} = \mathbf{M}\ddot{\mathbf{q}} \tag{4}$$

$$\mathbf{f}_{int} = \mathbf{g}^{int} \tag{5}$$

$$\mathbf{S} = \alpha_1 \mathbf{K}_T + \alpha_2 \mathbf{M} + \alpha_3 \mathbf{C} \tag{6}$$

where α_i, for $i = 1, 2, 3$, are constants defined in the *Mecano* user-element, and $\mathbf{K}_T$ and $\mathbf{C}$ are, respectively, the stiffness and the damping matrices, defined as

$$\mathbf{K}_T = \frac{\partial \mathbf{f}_{int}(\mathbf{q})}{\partial \mathbf{q}}, \qquad \mathbf{C} = \frac{\partial \mathbf{f}_{int}(\mathbf{q})}{\partial \dot{\mathbf{q}}} \tag{7}$$

As mentioned in the previous section, control systems generally involve transfer functions and first-order equations to represent the dynamics of the system. Most of the time, controllers and estimators are designed using linear differential equations. Therefore, first-order equations, as those of Eqs. (1a & b), can be written in the usual state-space form as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{8a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{8b}$$

Then, a representation of this system in transfer-function form is readily available, namely,

$$\mathbf{G}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} \tag{9}$$

where $\mathbf{I}$ denotes the $n \times n$ identity matrix, $\mathbf{G}(s)$ being generally a matrix composed of transfer functions, and is, hence, called the matrix transfer function. Therefore, considering a block diagram representing a control system, we can first write the state-space equations in transfer-function form, then, by block-diagram transformation [7], the transfer function mapping a desired input variable u to a desired output variable y can be obtained. Generally, for a single-input single-output system (SISO), this transfer function takes the form

$$G(s) = \frac{b_n s^n + b_{n-1} s^{n-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} = \frac{Y(s)}{U(s)} \tag{10}$$

where $U(s)$ represents the Laplace transform of the input variable and $Y(s)$ that of the output variable. Note that the order of the numerator is, in general, smaller or equal to n. In cases, where it is equal to m, with $m < n$, we have $b_i = 0$ for $i = m+1, \cdots, n$. From Eq. (10), a nth-order differential equation is readily obtained, namely,

$$a_n y^{(n)} + a_{n-1} y^{(n-1)} + \cdots + a_1 y^{(1)} + a_0 y = b_n u^{(n)} + b_{n-1} u^{(n-1)} + \cdots + b_1 u^{(1)} + b_0 u \tag{11}$$

Usually, the j-th derivative of the input u is not known. However, by introducing a proper change of variable, this nth-order equation can be written in a more suitable form. Let us consider the equation

$$a_n x^{(n)} + a_{n-1} x^{(n-1)} + \cdots + a_1 x^{(1)} + a_0 x = du \tag{12}$$

Then, we can write

$$y = c_0 x + c_1 x^{(1)} + \cdots + c_{n-1} x^{(n-1)} + c_n u \tag{13}$$

Therefore, if the coefficients c_i, $i = 0, \cdots, n$, of Eq. (13) can be determined, then the output variable $y(t)$ is obtained as a linear combination of the time response $x(t)$ of Eq. (12), and its i-th time derivative, for $i = 1, \cdots, (n-1)$. In order to determine these coefficients, let us consider the i-th derivative of y, for $i = 1, \cdots, n$,

$$\begin{aligned}
y^{(1)} &= c_0 x^{(1)} + c_1 x^{(2)} + \cdots + c_{n-1} x^{(n)} + c_n u^{(1)} \\
y^{(2)} &= c_0 x^{(2)} + c_1 x^{(3)} + \cdots + c_{n-1} x^{(n+1)} + c_n u^{(2)} \\
&\vdots \\
y^{(n-1)} &= c_0 x^{(n-1)} + c_1 x^{(n)} + \cdots + c_{n-1} x^{(2n-2)} + c_n u^{(n-1)} \\
y^{(n)} &= c_0 x^{(n)} + c_1 x^{(n+1)} + \cdots + c_{n-1} x^{(2n-1)} + c_n u^{(n)}
\end{aligned} \tag{14}$$

Substituting Eqs. (14) into Eq. (11), we obtain

$$\begin{aligned}
&a_n c_0 x^{(n)} + a_n c_1 x^{(n+1)} + \cdots + a_n c_{n-1} x^{(2n-1)} + a_n c_n u^{(n)} \\
&+ a_{n-1} c_0 x^{(n-1)} + a_{n-1} c_1 x^{(n)} + \cdots + a_{n-1} c_{n-1} x^{(2n-2)} + a_{n-1} c_n u^{(n-1)} \\
&\qquad\qquad \vdots \\
&+ a_1 c_0 x^{(1)} + a_1 c_1 x^{(2)} + \cdots + a_1 c_{n-1} x^{(n)} + a_1 c_n u^{(1)} \\
&+ a_0 c_0 x + a_0 c_1 x^{(1)} + \cdots + a_0 c_{n-1} x^{(n-1)} + a_0 c_n u = \\
&\qquad b_n u^{(n)} + b_{n-1} u^{(n-1)} + \cdots + b_1 u^{(1)} + b_0 u
\end{aligned} \tag{15}$$

Differentiating Eq. (12) m times, we derive

$$a_n x^{(m+n)} + a_{n-1} x^{(m+n-1)} + \cdots + a_1 x^{(m+1)} + a_0 x^{(m)} = du^{(m)} \tag{16}$$

Therefore, using a suitable value for m, Eq. (15) reduces to

$$\begin{aligned}
&c_0 du + c_1 du^{(1)} + \cdots + c_{n-1} du^{(n-1)} \\
&+ a_n c_n u^{(n)} + a_{n-1} c_n u^{(n-1)} + a_1 c_n u^{(1)} + a_0 c_n u = \\
&\qquad b_n u^{(n)} + b_{n-1} u^{(n-1)} + \cdots + b_1 u^{(1)} + b_0 u
\end{aligned} \tag{17}$$

which can be written as

$$\begin{aligned}
&a_n c_n u^{(n)} + (a_{n-1} c_n + c_{n-1} d) u^{(n-1)} \\
&+ \cdots + (a_1 c_n + c_1 d) u^{(1)} + (a_0 c_n + c_0 d) u = \\
&\qquad b_n u^{(n)} + b_{n-1} u^{(n-1)} + \cdots + b_1 u^{(1)} + b_0 u
\end{aligned} \tag{18}$$

By equating terms of the same powers, we obtain

$$c_n = b_n/a_n \tag{19a}$$

$$c_i = \frac{1}{d}\left(b_i - \frac{a_i}{a_n}b_n\right), \qquad i = 0, 1, \cdots, n-1 \tag{19b}$$

Therefore, if an equation of the form of Eq. (11) is to be integrated for a given input u, then Eq. (12) can be integrated and the desired time response $y(t)$ is obtained using Eq. (13) with the coefficients c_i, $i = 0, \cdots, n$, of Eqs. (19a & b).

Moreover, since *Mecano* can only handle second-order equations, we must write Eq. (12) as a set of second-order equations. We can introduce the following change of variables

$$z_i = x^{(2i-2)}, \qquad i = 1, 2, \cdots, \lceil n/2 \rceil \tag{20}$$

with $\lceil \cdot \rceil$ defined as the *roof function* of its *natural* argument $(\cdot)$, i.e., in our case, as the integer that is closest to $n/2$ from above.

Differentiating Eq. (20), we obtain

$$\dot{z}_i = x^{(2i-1)}, \qquad i = 1, 2, \cdots, \lceil n/2 \rceil \tag{21}$$

$$\ddot{z}_i = x^{(2i)}, \qquad i = 1, 2, \cdots, \lceil n/2 \rceil \tag{22}$$

For $i = 1, 2, \cdots, (\lceil n/2 \rceil - 1)$, we have

$$\ddot{z}_i = z_{i-1} \tag{23}$$

which leads to $(\lceil n/2 \rceil - 1)$ equations. Moreover, substituting the z_i, $\dot{z}_i$ and $\ddot{z}_i$ expressions in Eq. (12), and assuming an even n, we obtain

$$a_n \ddot{z}_{\lceil n/2 \rceil} + a_{n-1}\dot{z}_{\lceil n/2 \rceil} + a_{n-2} z_{\lceil n/2 \rceil} + a_{n-3}\dot{z}_{\lceil n/2 \rceil - 1} + a_{n-4} z_{\lceil n/2 \rceil - 1} + \cdots + a_1 \dot{z}_1 + a_0 z_1 = du \tag{24}$$

In vector form, we finally obtain

$$\mathbf{M}\ddot{\mathbf{z}} + \mathbf{C}\dot{\mathbf{z}} + \mathbf{K}\mathbf{z} = \mathbf{f} \tag{25}$$

where

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{\lceil n/2 \rceil - 1} \\ z_{\lceil n/2 \rceil} \end{bmatrix}, \qquad \mathbf{f} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ du \end{bmatrix}, \qquad \mathbf{M} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & a_n \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ a_1 & a_3 & \cdots & a_{n-3} & a_{n-1} \end{bmatrix}, \qquad \mathbf{K} = \begin{bmatrix} 0 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 \\ a_0 & a_2 & a_4 & \cdots & a_{n-4} & a_{n-2} \end{bmatrix}$$

In cases where n is odd, Eq. (25) is exactly the same, except that we have $a_n = 0$, and the mass matrix becomes semi-definite. This semi-definiteness is not a problem, since, in the integration scheme used in *Mecano*, there is no need to invert the mass matrix; only the iteration matrix $\mathbf{S}$ is inverted. Therefore, in order to apply the user-element of *Mecano*, we directly have

$$\mathbf{f}_{ine} = \mathbf{M}\ddot{\mathbf{z}} \tag{26a}$$

$$\mathbf{f}_{int} = \mathbf{C}\dot{\mathbf{z}} + \mathbf{K}\mathbf{z} \tag{26b}$$

$$\mathbf{S} = \alpha_1\mathbf{K} + \alpha_2\mathbf{M} + \alpha_3\mathbf{C} \tag{26c}$$

3.2 APPLICATION: ATTITUDE CONTROL

We show, in this subsection, how to use *Mecano* to simulate the model depicted in Fig. 2 by introducing the techniques presented in the previous subsection.

In that system, the plant dynamics block represents the model of a spacecraft with a space manipulator mounted on it, where we try to control the attitude of the base in a planar motion. This plant is modelled using the standard-elements library of *Mecano*. Moreover, even if *Mecano* can give the velocity and the acceleration of the base, we assume that only the attitude is available by sensors and therefore, only the angular displacement of the node representing the centre of mass (CM) of the spacecraft is considered. A time delay τ is implemented by writing a Fortran subroutine that is called by the user-element subroutine and that stacks values of the attitude for previous time steps in a vector array. The desired value of the attitude for the current time step is obtained by linearly interpolating the value in that vector corresponding to the attitude at τ seconds before.

Now, in order to implement the state estimator, a transfer function mapping the filtered attitude signal $\hat{\theta}_0$ to the attitude signal θ_0 must be derived, as well as a transfer function mapping the estimated attitude rate $\dot{\hat{\theta}}_0$ to the attitude signal θ_0 and the command of the thruster u. For the attitude, we readily have

$$\hat{\theta}_0 = G_f(s)\theta_0 \tag{27}$$

Using the definition of $G_f(s)$ given by Eq. (2), we have

$$\hat{\theta}_0 = \frac{\omega_f^2}{s^2 + 2\zeta_f\omega_f s + \omega_f^2}\theta_0 \tag{28}$$

which leads to

$$\ddot{\hat{\theta}}_0 + 2\zeta_f\omega_f\dot{\hat{\theta}}_0 + \omega_f^2\hat{\theta}_0 = \omega_f^2\theta_0 \tag{29}$$

Comparing this equation with Eq. (11), and using $n = 2$, since we have a second-order differential equation, we have

$$a_0 = b_0 = \omega_f^2, \quad a_1 = 2\zeta_f\omega_f, \quad a_2 = 1, \quad b_1 = b_2 = 0 \tag{30}$$

which is in a form readily integrable by *Mecano*. Note that θ_0 is not an external input to the user-element, but the position of the node describing the CM of the spacecraft represented as a rigid body in *Mecano*. Therefore, the user-element must be composed of 2 nodes, the actual CM of the spacecraft, and a second node that will be the desired estimated attitude $\hat{\theta}_0$ obtained by solving the differential equation of Eq. (29).

For the attitude rate, we have, by examining Fig. 2,

$$\begin{aligned}
\hat{\dot{\theta}}_0 &= \hat{\dot{\theta}}'_f + \hat{\dot{\theta}}_c \\
&= sG_{se}(s)\hat{\theta}'_f + \hat{\dot{\theta}}_c \\
&= sG_{se}(s)(\theta_0 - \hat{\theta}_c) + \hat{\dot{\theta}}_c \\
&= sG_{se}(s)(\theta_0 - \frac{1}{s}\hat{\dot{\theta}}_c) + \hat{\dot{\theta}}_c \\
&= sG_{se}(s)\theta_0 + [1 - G_{se}(s)]\,\hat{\dot{\theta}}_c \\
\hat{\dot{\theta}}_0 &= sG_{se}(s)\theta_0 + [1 - G_{se}(s)]\,\frac{1}{s}\gamma u
\end{aligned} \tag{31}$$

where γ is the angular acceleration impinged to the system by the thrusters.

Using the definition of the second-order filter, as given in Eq. (2) for $G_f(s)$, we have

$$\begin{aligned}
\hat{\dot{\theta}}_0 &= \frac{\omega_{se}^2 s}{s^2 + 2\zeta_{se}\omega_{se}s + \omega_{se}^2}\theta_0 + \left(1 - \frac{\omega_{se}^2}{s^2 + 2\zeta_{se}\omega_{se}s + \omega_{se}^2}\right)\frac{1}{s}\gamma u \\
\hat{\dot{\theta}}_0 &= \frac{\omega_{se}^2 s}{s^2 + 2\zeta_{se}\omega_{se}s + \omega_{se}^2}\theta_0 + \frac{s + 2\zeta_{se}\omega_{se}}{s^2 + 2\zeta_{se}\omega_{se}s + \omega_{se}^2}\gamma u
\end{aligned} \tag{32}$$

Moreover, we define

$$\psi = \hat{\dot{\theta}}_0 \tag{33}$$

to obtain, from Eq. (32),

$$\ddot{\psi} + 2\zeta_{se}\omega_{se}\dot{\psi} + \omega_{se}^2\psi = \omega_{se}^2\dot{\theta}_0 + \gamma\dot{u} + 2\zeta_{se}\omega_{se}\gamma u \tag{34}$$

Since this equation is linear, we have, by superposition,

$$\ddot{\psi}_1 + 2\zeta_{se}\omega_{se}\dot{\psi}_1 + \omega_{se}^2\psi_1 = \omega_{se}^2\dot{\theta}_0 \tag{35a}$$

$$\ddot{\psi}_2 + 2\zeta_{se}\omega_{se}\dot{\psi}_2 + \omega_{se}^2\psi_2 = \gamma\dot{u} + 2\zeta_{se}\omega_{se}\gamma u \tag{35b}$$

$$\psi = \psi_1 + \psi_2 \tag{35c}$$

Comparing Eq. (35a) with Eq. (11), we have

$$a_0 = b_1 = \omega_{se}^2, \quad a_1 = 2\zeta_{se}\omega_{se}, \quad a_2 = 1, \quad b_0 = b_2 = 0 \tag{36}$$

Since the input to Eq. (35a) is $\dot{\theta}_0$ and only θ_0 is assumed available, we must resort to the technique introduced in the previous subsection. Let us consider the equation

$$\ddot{x}_1 + 2\zeta_{se}\omega_{se}\dot{x}_1 + \omega_{se}^2 x_1 = \omega_{se}^2\theta_0 \tag{37}$$

Comparing this equation with Eq. (12), we have

$$d = \omega_{se}^2 \tag{38}$$

The desired time response $\psi_1(t)$ is thus obtained by integrating Eq. (37) in a user-element and then using Eq. (13) with c_0, c_1 and c_2 given by Eq. (19a & b), namely,

$$c_0 = 0, \quad c_1 = 1, \quad c_2 = 0 \tag{39}$$

Now, comparing Eq. (35b) with Eq. (11), we have

$$a_0 = \omega_{se}^2, \quad a_1 = 2\zeta_{se}\omega_{se}, \quad a_2 = 1, \quad b_0 = 2\zeta_{se}\omega_{se}\gamma, \quad b_1 = \gamma, \quad b_2 = 0 \tag{40}$$

Resorting to the same technique just introduced above, we can consider the equation

$$\ddot{x}_2 + 2\zeta_{se}\omega_{se}\dot{x}_2 + \omega_{se}^2 x_2 = \omega_{se}^2 \gamma u \tag{41}$$

which gives

$$d = \omega_{se}^2 \gamma \tag{42}$$

The desired time response $\psi_2(t)$is thus given by integrating Eq. (41) and by using Eq. (13) with c_0, c_1, and c_2 given by Eq. (19a & b), namely,

$$c_0 = \frac{2\zeta_{se}}{\omega_{se}}, \quad c_1 = \frac{1}{\omega_{se}^2}, \quad c_2 = 0 \tag{43}$$

Therefore, the estimate of the attitude rate is given by

$$\hat{\dot{\theta}}_0 = \psi = \psi_1 + \psi_2 = \dot{x}_1 + \frac{2\zeta_{se}}{\omega_{se}} x_2 + \frac{1}{\omega_{se}^2}\dot{x}_2 \tag{44}$$

The last item required to simulate the whole system of Fig. 2 is a user-element for the controller. From that figure, we have that

$$\sigma = \theta_{d0} - \hat{\theta}_0 - \lambda\hat{\dot{\theta}}_0 \tag{45}$$

$$= e + \lambda\dot{e} \tag{46}$$

where e end $\dot{e}$ are, respectively, the error on the attitude and that of the attitude rate, as defined in Section 2. The parameter σ thus represents the left-hand side of the switching-line equations. The required command of the thruster u is thus simply the output of a relay with a dead zone whose input is σ. Thus, we have the algorithm

```
if σ > δ then
    u = 1
elseif σ < −δ then
    u = −1
else
    u = 0
endif
```

This algorithm can be easily implemented in a user-element.

3.3 INPUT-OUTPUT PROBLEMS IN MECANO

As mentioned previously, *Mecano* was not developed to simulated control systems, but only to analyze dynamical systems. Therefore, the inputs and outputs allowed in the user-element subroutine are sufficient for most cases, but result in some problems when dealing with control systems. For example, there is no input to the subroutine for the resulting forces applied to a node, and there is no output to allow the application of an external force to a node. These two input-output drawbacks became a challenge when modelling the system analyzed in the previous subsection. For example, the input in Eq. (41) is the command of the thruster u. Since this information is not readily available in the subroutine, we had to define a rigid body composed of one node with the same inertia of the spacecraft. Then, when the thruster force was applied to the spacecraft, the same force was also applied to this rigid body. The node representing the CM of this rigid body was thus part of the user-element to integrate Eq. (41), and by considering the acceleration of the node, the command of the thruster u, which is either $+1$, 0 or -1, was extracted.

Another challenge was the application of the force produced by the thruster to the spacecraft. Since that force is obtained in a user-element, it was not possible to apply directly this force to the node representing the CM of the spacecraft because only inertial and internal forces can be specified. To overcome this problem, the thruster force was defined as the inertial force of this node used by the user-element. This force was defined in the opposite direction of the thruster force required. The effects of this inertial force, when combining it with the overall motion of the node representing the CM of the spacecraft, was an acceleration in the desired direction.

Apparently, the addition of a few arguments in the inputs and outputs of the user-element would decrease significantly the workload needed when developing a user-element to simulate control systems.

4 SIMULATION RESULTS AND COMPARISON

The relations derived in Subsection 3.2 for the model of Fig. 2 have been implemented in four user-elements; one to integrate each of Eqs. (29, 37, and 41), and one to implement the thruster switching logic introduced at the end of Subsection 3.2. Moreover, we consider for the plant, the planar one-flexible-joint manipulator on the 3-dof spacecraft of Fig. 3, with the parameters of Table 1, where $x_c = (2712 + 15.5\beta m_0)/(320 + \beta m_0)$. These parameter values have been chosen in [8, 6] to approximate the Space Shuttle/CANADARM system when the arm is fully extended. Moreover, this system has been studied in [6], the reader being referred to that paper for the derivation of the mathematical model. The spacecraft and the link are modelled using rigid elements in *Mecano*, while the flexible joint is modelled using a hinge-joint element with elastic locking and viscous damping. The spring stiffness k and damping coefficient c of the flexible joint were chosen to match the first natural frequency and the damping ratio of the same Space Shuttle/CANADARM

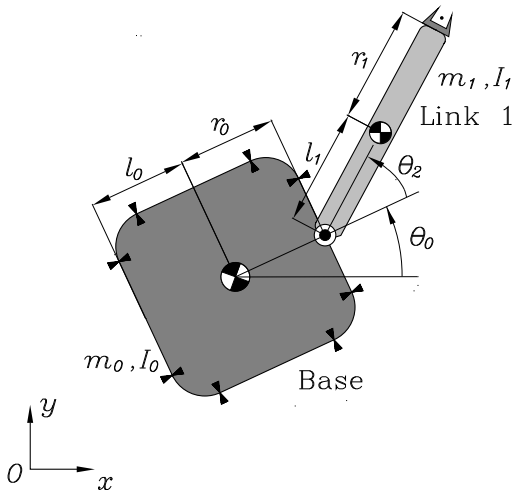

Figure 3: A planar free-flying manipulator.

Table 1: Shuttle, simplified 1-link manipulator and payload parameter values.

Body	l_i (m)	r_i (m)	m_i (kg)	I_i (kg m^2)
0		1	75,000	1,635,937
1	x_c	$15.5 - x_c$	$320 + \beta m_0$	$(320 + \beta m_0)x_c^2 - (5423.5 + 31\beta m_0)x_c$ $+31424.11 + 240.25\beta m_0$

system, in a specific configuration, namely

$$k = 123,985 \text{ Nm/rad} \tag{47}$$

$$c = 6,166 \text{ Nms/rad} \tag{48}$$

As done in [6], the joint is locked in a specific configuration θ_1, with θ_1 being the angular position of the rotor. Moreover, using the parameters in Table 2, which are based on available space-manipulator data, the system was simulated using both *Mecano* and *Matlab*. In this table, β is the ratio of the mass of the payload to the mass of the spacecraft, λ is the slope of the switching lines, δ is the spacecraft attitude limit, γ_0 is the nominal angular acceleration of the spacecraft, i.e $\gamma_0 = n/I_0$ where n is the available thruster torque, τ is the time delay, and ω_f, ω_{se}, ζ_f, and ζ_{se} are the parameters of the second-order filters. Simulations results for an initial base angular error of 0.05 rad are shown in Fig. 4 when using *Matlab* and in Fig. 5 when using *Mecano*.

A qualitative comparison of Figs. 4 and 5 shows that the results are identical. Figures 4(a) and 5(a) show the error phase-plane that determines the switching logic, while Figs. 4(b) and 5(b) show the attitude of the spacecraft vs. its attitude rate. The final attitude and attitude rate of the spacecraft are included in Table 3. We can

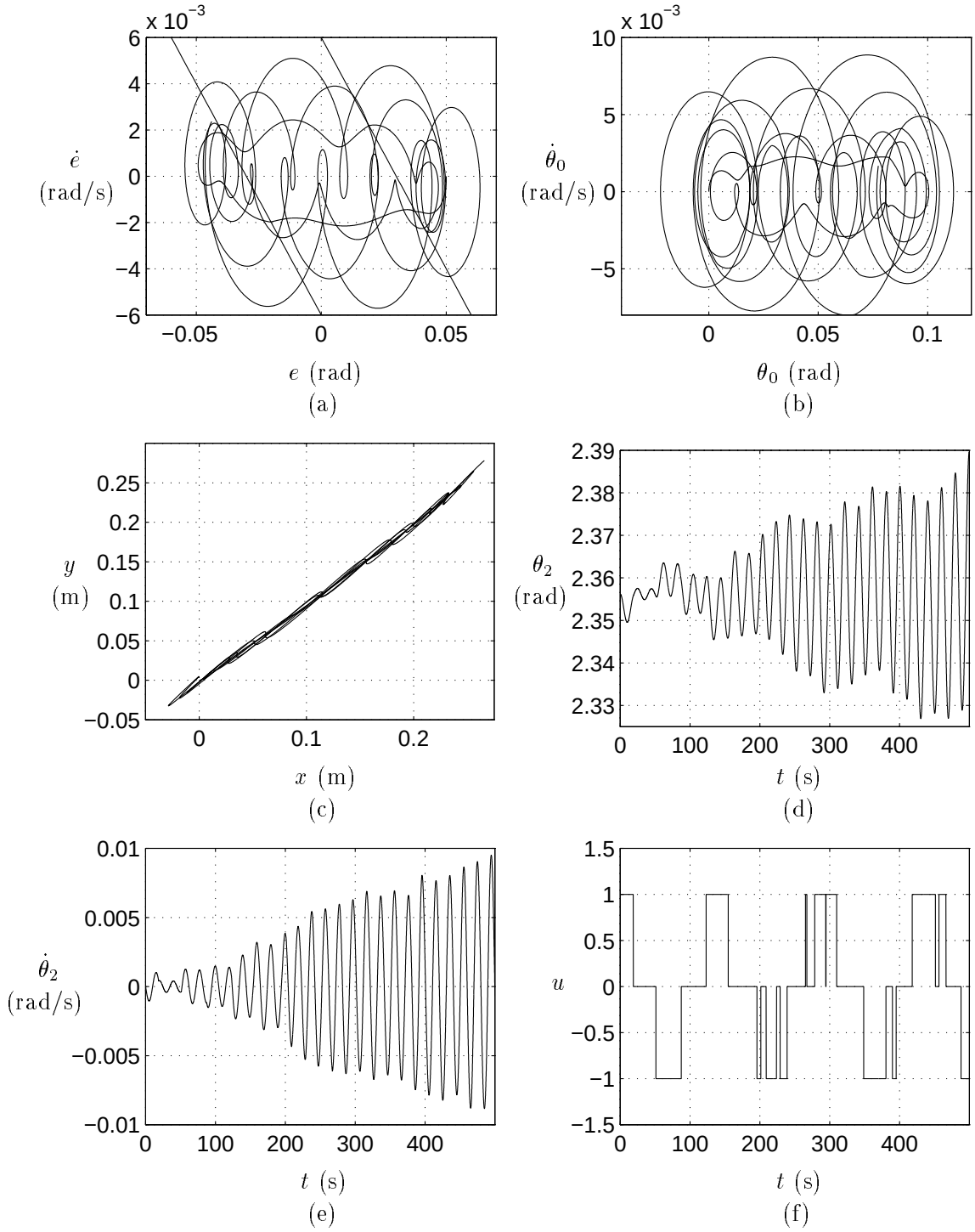

Figure 4: Simulation results using *Matlab*: (a) Error phase plane; (b) Attitude phase plane; (c) x-y trajectory; (d) joint angle history; (e) Joint rate history; and (f) Thruster command history.

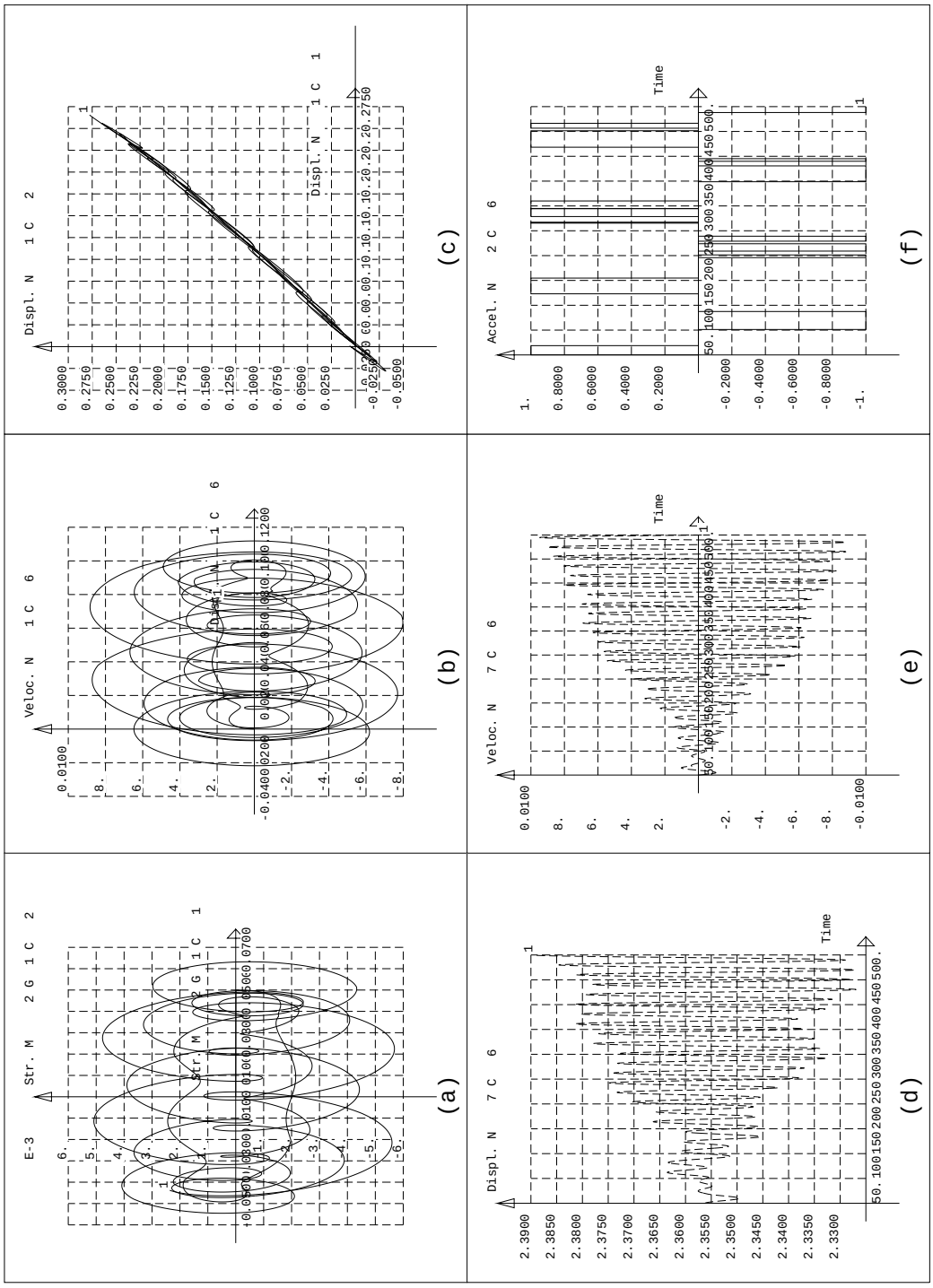

Figure 5: Simulation results using *Mecano*: (a) Error phase plane; (b) Attitude phase plane; (c) x-y trajectory; (d) joint angle history; (e) Joint rate history; and (f) Thruster command history.

Table 2: Parameter values for simulations.

θ_1	β	λ (s)	δ (rad	γ_0	τ (s)	ω_f (rad/s)	ω_{se} (rad/s)	ζ_f	ζ_{se}
135°	0.3	5	0.03	$0.02°/s^2$	0.1	0.2513	0.2513	0.707	0.707

Table 3: Attitude and attitude rate at $t = 500$ s.

	Matlab	*Mecano*	Error
θ_0	7.745817×10^{-2}	7.743437×10^{-2}	0.03%
$\dot{\theta}_0$	1.676471×10^{-3}	1.655943×10^{-3}	1.2%

see that, after 500 seconds, the difference between the attitude calculated with both packages is only 0.03%, while this error is 1.2% for the attitude rate. Figures 4(c) and 5(c) show the x and y position of the CM of the spacecraft, while Figs. 4(d) and 5(d) and Figs. 4(e) and 5(e) show the joint angle and joint rate, respectively. Finally, Figs. 4(f) and 5(f) show the history of the thruster command u.

It is interesting to note that, if the same system is simulated for a longer period, say 2000 s, a large limit cycle will develop, thus resulting in a continuous operation of the thrusters and a high fuel consumption. Results for a 2000 s run are reported in [6] using *Matlab*.

From this good matching of the simulation results using *Mecano* and *Matlab*, we can conclude that *Mecano* can be used effectively to simulate control systems. Since *Matlab* has built-in functions for control-system design, it could be used first to design a control system based on an approximate model of the plant. Then, *Mecano* could be used to simulate the same control system, but with a more detailed model of the plant to give more insight on the behaviour of the actual physical plant. In our case, *Mecano* will be used to model a 6-link manipulator mounted on a spacecraft where both joint and link flexibility will be considered.

5 CONCLUSIONS

This work examined the capacity of *Mecano* to simulate control systems. Various techniques were applied to transform the equations describing a controller or an observer in a form suitable for implementation in the user-element of *Mecano*. A typical spacecraft attitude control system was employed as an example to experience these techniques, and to perform a comparison with results obtained using *Matlab*. We have shown that *Mecano* could be effectively used to simulate control systems, thus allowing an extended analysis of a system with a detailed model of the plant. Moreover, it was pointed out that a few additional arguments in the inputs and outputs of the user-element Fortran subroutine would decrease significantly the

workload needed when developing a user-element to simulate control systems.

ACKNOWLEDGEMENTS

The support of this work by the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR), and by the Natural Sciences and Engineering Council of Canada (NSERC) is gratefully acknowledged. The first author is an NSERC Class-of-67 Scholar. The support of Quebec's *Ministère des affaires internationales* under the Quebec-Wallonia Scientific Collaboration Agreement is highly acknowledged.

REFERENCES

[1] Millar, R. A., and Vigneron, F. R., "Attitude Stability of a Pseudorate Jet-Controlled Flexible Spacecraft," *J. of Guid., Cont., and Dyn.*, Vol. 2, No. 2, 1979, pp. 111–118.

[2] Sackett, L. L., and Kirchwey, C. B., "Dynamic Interaction of the Shuttle On-Orbit Flight Control System with Deployed Flexible Payload," *Proc. of the AIAA Guid. and Cont. Conf.*, San Diego, CA, 1982, pp. 232–245.

[3] Kubiak, E. T., and Martin, M. W., "Minimum Impulse Limit Cycle Design to Compensate for Measurement Uncertainties," *J. of Guid., Cont., and Dyn.*, Vol. 6, No. 6, 1983, pp. 432–435.

[4] Martin, E., Papadopoulos, E., and Angeles, J., "On the Interaction of Flexible Modes and On-off Thrusters in Space Robotic Systems," *Proc. of the 1995 Int. Conf. on Intelligent Robots and Systems, IROS'95*, Vol. 2, Pittsburgh, PA, 1995, pp. 65–70.

[5] Martin, E., Papadopoulos, E., and Angeles, J., "Towards Reducing Thruster-Flexibility Interactions in Space Robots," *To appear in the Proc. of the Eleventh CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, Udine, Italy, 1996.

[6] Martin, E., Papadopoulos, E., and Angeles, J., "Control System Design for Reduced Thruster-Flexibility Interactions in Space Robots," *To appear in the Proc. of the 1996 Canadian Society for Mechanical Engineering (CSME) Forum*, Hamilton, Canada, 1996.

[7] Dorf, R. C., *Modern Control Systems*, Addison-Wesley Publishing Company, 1986.

[8] E. Martin, "Interaction of Payload and Attitude Controller in Space Robotic Systems," Master Thesis, Dept. of Mech. Eng., McGill University, Montreal, Canada, 1994.