Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2009
August 30 - September 2, 2009, San Diego, California, USA

DETC2009-86653

WEAK COUPLING OF MULTIBODY DYNAMICS AND BLOCK DIAGRAM SIMULATION TOOLS

Francisco González[*] **Manuel González** **Javier Cuadrado**
Escuela Politécnica Superior, Universidad de A Coruña
Mendizábal s/n, Ferrol, A Coruña, Spain

ABSTRACT

Dynamic simulation of complex mechatronic systems can be carried out in an efficient and modular way making use of weakly coupled co-simulation setups. When using this approach, multirate methods are often needed to improve the efficiency, since the physical components of the system usually have different frequencies and time scales. However, most multirate methods have been designed for strongly coupled setups, and their application in weakly coupled co-simulations is not straightforward due to the limitations enforced by the commercial simulation tools used for mechatronics design.

This work describes a weakly coupled multirate method applied to combine a block diagram simulator (Simulink) with a multibody dynamics simulator in a co-simulation setup. A double-mass triple-spring system with known analytical solution is used as test problem in order to investigate the behavior of the method as a function of the frequency ratio (*FR*) of the coupled subsystems. Several synchronization schemes (*fastest-first* and *slowest-first*) and interpolation/extrapolation methods (polynomials of different order and smoothing) have been tested.

Results show that the *slowest-first* methods deliver the best results, combined with a cubic interpolation (for $FR < 25$) or without interpolation (for $25 < FR < 50$). For $FR > 50$, none of the tested methods can deliver precise results, although smoothing techniques can reduce interpolation errors for certain situations.

1 INTRODUCTION

Modern complex mechatronic systems are made up of multi-domain components of different nature. An automobile is a very representative example of these kinds of systems, involving mechanical components (chassis, suspensions, steering mechanism, powertrain), active control devices (Anti-lock Braking System, Electronic Stability Control, traction control), hydraulic devices (brake circuit) and power sources (internal combustion engine or electric motors). Due to the increasing demand of quality and performance, the traditional design approach based on a sequential design of the components can no longer be applied to such systems: engineers need to model and simulate the dynamic response of the whole system, taking into account the simultaneous interaction phenomena between components.

The modeling of complex mechatronic systems can be accomplished by two different strategies: strongly coupled and weakly coupled. On one hand, the strongly coupled strategy assembles the dynamic equations of each subsystem into a monolithic set of equations, which can be numerically integrated in a single environment. On the other hand, the weakly coupled strategy does not assemble the equations: their numerical integration is performed in parallel by several interconnected environments that exchange information during the integration process, working in a co-simulation configuration. A review about both strategies is provided in [1].

The weakly coupled strategy has important advantages over the strongly coupled one: specialized modeling and simulation tools, familiar to experts in the corresponding field, can be applied to each component. In addition, component models can be modified with minor impact on other components, which results in a better modularity of the whole model. For example, control and hydraulic devices are usually modeled and simulated in general-purpose block diagram simulators like Matlab/Simulink from Mathworks [2], MATRIXx/SystemBuild from National Instruments [3] or the free open source tool Scilab/Scicos from INRIA [4]. Conversely, the behavior of complex mechanical components is better modeled and simulated in specialized tools for multibody system dynamics like MSC.Adams [5], Simpack [6] or Recurdyn [7]; these tools also provide interfaces with the aforementioned block diagram simulators, which simplify the set up of weakly coupled simulations. Representative examples

[*] PhD student and corresponding author, Phone: (+34) 981337400 ext. 3870, Fax: (+34) 981337410, Email: fgonzalez@udc.es

of these kinds of co-simulation setups are given in [8] and [9], where the authors combine a multibody system simulation package (ADAMS and Simpack, respectively) with a block diagram simulator (Simulink) to model a full vehicle equipped with electronic control devices.

Another important feature of complex mechatronic systems, derived from their multi-domain nature, is the presence of different time scales, which results in widely different dynamic response characteristics in terms of frequencies. For example, mechanical components have slow frequency responses compared to fast electronic components. The computational efficiency of dynamic simulations of complex mechatronic systems is quite important, because these models are often used in optimization processes (where each function evaluation involves a complete dynamic simulation) or hardware-in-the-loop settings (where the dynamic simulation must run in real-time). In order to make the numerical integration of the dynamic equations of the whole system as efficient as possible, each component should be integrated with a stepsize adapted to its time scale. This procedure is known as multirate integration.

Research on multirate integration methods for ordinary differential equations (ODE) has been carried out since the late 1970s [10]. The basic idea is to employ two (or more) time grids: a coarse one for the slow components, and a refined one for the fast components; the coupled terms in the slow and fast equation sets are estimated by means of extrapolation or interpolation methods. Many contributions to this subject have been proposed, including advanced techniques like dynamic partitioning of equations with automatic identification of fast and slow components during the integration [11], self-adjusting multirate time stepping strategies [12] and stability analysis of the proposed methods [13].

The application of existing multirate integration methods to mechatronic models obtained by the strongly coupled strategy is straightforward, since they are precisely designed to work on a monolithic set of equations with full control on the integration process. However, if the mechatronic system is modeled according to the weakly coupled strategy, these multirate integration methods cannot be applied directly due to their particular features:

(a) They introduce modifications in the integration schemes, something that is not possible in commercial off-the-shelf modeling and simulation tools used for weakly coupled co-simulation. For example, the aforementioned block diagram simulators and multibody system simulation packages offer their own set of integration schemes that cannot be modified.

(b) They assume that the coarse and refined time-grids are equidistant and synchronized, which means that the large stepsize H is a multiple of the small stepsize h. This condition cannot be guarantied in weakly coupled co-simulations if one or more subsystems are integrated with a variable stepsize integrator, since the stepsize control algorithms of the different simulation environments cannot be synchronized.

(c) They mitigate the unstable behavior caused by the explicit extrapolation of some equation terms by introducing implicit schemes, which involve some kind of iterative process. Again, commercial off-the-shelf simulation tools like block diagram simulators do not allow this kind of iteration with other simulation tools.

Due to these impediments, commercial state-of-the-art simulation environments used in the mechatronics industry do not yet provide tools to enable multirate integration when they are used in weakly coupled co-simulation setups. Two examples of this situation are provided: the first one is veDYNA [14], a real-time vehicle dynamics simulation environment very popular in the automotive industry, which is based on Matlab/Simulink. veDYNA works as an external simulation tool embedded in Simulink, and provides a library of mechanical elements to model any kind of vehicle. Non mechanical elements, like electronic and control devices, are modeled in Simulink as usual, exchanging input and output data with the mechanical model. veDYNA uses an internal semi-implicit fixed-step Euler integration scheme to solve the equations of motion of the vehicle, and requires that the Simulink integration must be performed with the *ode1* integrator (explicit fixed-step Euler's method) in order to properly synchronize both integrations. This requirement is a strong drawback, since Simulink's *ode1* integration scheme is not suited at all in many situations. Another example of the limitations of currently available simulation environments is SIMAT, the interface provided by the multibody simulation software Simpack to perform co-simulation with Simulink [6]. SIMAT works as a Simulink block that exchanges data between the Simpack model and the Simulink model during the integration. However, its current implementation only allows fixed stepsize integrators with the same stepsize in both simulation environments. The same constraint applies to other packages for multibody system simulation, like ADAMS, which provide interfaces for performing co-simulation with block diagram simulators: none of them support multirate integration.

Research is being carried to introduce multirate methods in weakly coupled co-simulation environments, principally in those which combine general-purpose block diagram simulators with external specialized simulation tools, a common setup in the industry. Busch et al [15] have tested several approaches to improve the aforementioned Simpack's SIMAT interface to support variable stepsizes in both sides of the co-simulation environment; in a similar way, Oberschelp and Vöcking [16] have investigated the behavior of some multirate techniques in weakly coupled co-simulations. However, these works apply multirate methods to solve a particular mechatronic model, and therefore their conclusions cannot be generalized nor extrapolated to other cases.

The goal of this work is to gain insight into the behavior and performance of multirate methods in weakly coupled co-simulations environments. To achieve this, the first contribution of this paper is an algorithm to implement a general multirate method (i.e., not constrained to synchronized time grids or a particular integration scheme) able to couple block diagram simulators with external simulation tools, like multibody simulation packages. The proposed algorithm can be configured to work in different modes and to use different interpolation and extrapolation methods. The second contribution of this paper is to study the accuracy of these coupling techniques in problems of different characteristics, in order to obtain general conclusions about the applicability of multirate methods in weakly coupled co-simulations based on block diagram simulators.

The remaining of the paper is organized as follows: Section 2 describes the methods used to carry out the research: a test problem and the algorithms to implement a general multirate method in a block diagram simulator. Section 3 presents and discusses the results of numerical experiments. Finally, Section 4 provides conclusions and areas of future work.

2 METHODS

A test problem involving two subsystems with fast and slow dynamic responses will be solved by coupling a block diagram simulation tool (to integrate the fast subsystem) with an external software for multibody system simulation (to integrate the slow subsystem). The parameters of the test problem will be adjusted to generate a range of co-simulation situations, which will be used to test different coupling strategies in terms of precision.

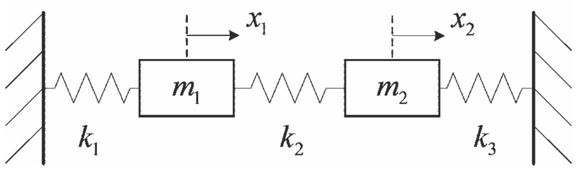

Figure 1: test problem.

2.1 Test problem

The double-mass triple-spring system shown in Figure 1 has been selected as test problem. It is made up of two subsystems represented by masses m_1 and m_2, which are coupled by the spring k_2. This simple, two degree-of-freedom system presents the advantage of having a known analytical solution for its dynamic response, which can be used as a reference in order to measure the accuracy of the coupled multirate numerical integration carried out by any co-simulation scheme. The dynamics of the test problem is governed by Equation (1):

$$\begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix}\begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{Bmatrix}+\begin{pmatrix} k_1+k_2 & -k_2 \\ -k_2 & k_2+k_3 \end{pmatrix}\begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix}=\begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \qquad (1)$$

where x_1 and x_2 measure the horizontal displacement of the masses from their equilibrium position. Equation (1) is a simple second order differential equation whose analytical solution is given by

$$\begin{aligned} x_1(t) &= C_{11}\cdot\cos(\omega_1 t)+C_{12}\cdot\sin(\omega_1 t)+C_{13}\cdot\cos(\omega_2 t)+C_{14}\cdot\sin(\omega_2 t) \\ x_2(t) &= C_{21}\cdot\cos(\omega_1 t)+C_{22}\cdot\sin(\omega_1 t)+C_{23}\cdot\cos(\omega_2 t)+C_{24}\cdot\sin(\omega_2 t) \end{aligned} \qquad (2)$$

where ω_1 and ω_2 are the natural frequencies of the two vibration modes of the system, and the terms C_{ij} are constants that define the amplitude of the vibration. In order to simplify the problem, sinus terms in Equation (2) are removed by setting the initial velocities to zero:

$$\begin{aligned} x_1(t) &= C_{11}\cdot\cos(\omega_1 t)+C_{13}\cdot\cos(\omega_2 t) \\ x_2(t) &= C_{21}\cdot\cos(\omega_1 t)+C_{23}\cdot\cos(\omega_2 t) \end{aligned} \qquad (3)$$

The dynamic response shown in Equation (3) is a function of six independent parameters (ω_1, ω_2, C_{11}, C_{13}, C_{21}, and C_{23}) For the purposes of this study, two of them are set to fixed values in Equation (4), and the rest are presented in a more suitable form in Equation (5):

$$\begin{aligned} \omega_1 &= 1\ \text{Hz} \\ C_{11} &= 1\ \text{m} \end{aligned} \qquad (4)$$

$$\begin{aligned} FR &= \omega_1/\omega_2 \\ AR_{12} &= C_{11}/C_{23} \\ AR_1 &= C_{11}/C_{13} \\ AR_2 &= C_{23}/C_{21} \end{aligned} \qquad (5)$$

From here on, frequencies ω_1 and ω_2 will be identified respectively with the primary frequencies of masses m_1 (fast subsystem) and m_2 (slow subsystem), assuming $\omega_1>\omega_2$, $C_{11}>C_{13}$ and $C_{23}>C_{21}$. The ratios defined in Equation (5) are interpreted as follows:

- The *frequency ratio FR* measures how fast the fast subsystem m_1 is, compared with the slow subsystem m_2.
- The *amplitude ratio* AR_{12} compares the primary amplitudes of both subsystems (C_{11} for m_1 and C_{23} for m_2).
- The *amplitude ratios* AR_1 and AR_2 measure how much the dynamic response of each subsystem is affected by the other subsystem.

Numerical experiments performed in Section 3 will use different sets of values for the ratios defined in Equation (5), in order to reproduce diverse co-simulation situations. As example, Figure 2 shows the dynamic response of x_1 for FR = 30, AR_{12} = 0.1, AR_1 = 0.1 and AR_2 = -1000.

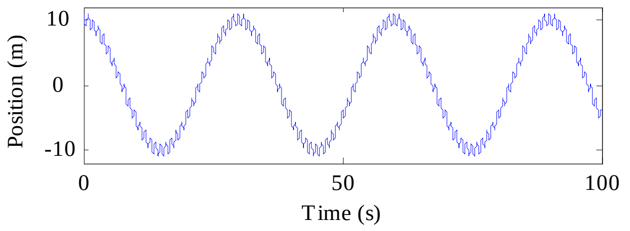

Figure 2: dynamic response of x_1.

After solving the dynamics of the problem in analytical form, the final step is to find the physical parameters of the system (m_1, m_2, k_1, k_2, k_3) and the initial conditions $x_1(t = 0)$

and $x_2(t=0)$ as a function of the response parameters defined in Equations (4) and (5). The resulting expressions will allow adjusting the physical parameters of the test problem in order to generate any desired dynamic response in its two subsystems. Note that the five aforementioned physical parameters can be scaled by the same factor without changing the dynamic response of the system, and therefore one of them must be fixed in advance. The selection of k_2 as fixed parameter greatly simplifies the mathematical manipulations:

$$k_2 = 1 \text{ N/m} \tag{6}$$

The remaining physical parameters can be obtained from the eigenvalue equation:

$$\left(\mathbf{K} - \omega^2 \mathbf{M}\right)\mathbf{A} = \mathbf{0} \tag{7}$$

where **A** is the matrix of modal amplitudes of the system, **K** and **M** are the stiffness and mass matrices shown in Equation (1) and ω stands for the natural frequencies of the system. The characteristic polynomial of the eigenvalue equation leads to a biquadratic equation in ω:

$$\begin{aligned}&\omega^4 m_1 m_2 - \omega^2\left(m_2\left(k_1+k_2\right)+m_1\left(k_2+k_3\right)\right)+\\&+\left(k_1+k_2\right)\left(k_2+k_3\right)-k_2^2=0\end{aligned} \tag{8}$$

which can be analytically solved, giving two equations of the form

$$\omega = f\left(m_1, m_2, k_1, k_2, k_3\right) \tag{9}$$

Two more equations can be obtained by substituting the solution given in Equation (3) in the equations of motion given by Equation (1); as each mode of vibration must satisfy the equations of motion, they lead to:

$$\begin{aligned}\frac{C_{11}}{C_{21}} &= \frac{k_2}{k_{12}-\omega_1^2 m_1} = \frac{k_{23}-\omega_1^2 m_2}{k_2}\\ \frac{C_{13}}{C_{23}} &= \frac{k_2}{k_{12}-\omega_2^2 m_1} = \frac{k_{23}-\omega_2^2 m_2}{k_2}\end{aligned} \tag{10}$$

Equations (9) and (10) form a set of four equations whose solution is expressed using the intermediate parameters a and b:

$$\begin{aligned}a &= C_{11}/C_{21}\\ b &= C_{13}/C_{23}\end{aligned} \tag{11}$$

$$m_1 = \frac{(a-b)}{ab\left(\omega_1^2-\omega_2^2\right)}k_2 \tag{12}$$

$$m_2 = \frac{ab\left(b-a\right)}{ab\left(\omega_1^2-\omega_2^2\right)}k_2 \tag{13}$$

$$k_1 = \frac{a\left(1-b\right)\omega_1^2 + b\left(1-a\right)\omega_2^2}{ab\left(\omega_1^2-\omega_2^2\right)}k_2 \tag{14}$$

$$k_3 = \frac{ab\left(\left(b-1\right)\omega_1^2-\left(1-a\right)\omega_2^2\right)}{ab\left(\omega_1^2-\omega_2^2\right)}k_2 \tag{15}$$

Finally, initial positions can be easily obtained from Equations (3) and (4):

$$\begin{aligned}x_1\left(0\right) &= C_{11}+C_{13} = C_{11}\cdot\frac{\left(1+C_{11}/C_{13}\right)}{C_{11}/C_{13}} = \frac{1+AR_1}{AR_1}\\ x_2\left(0\right) &= C_{21}+C_{23} = C_{11}\frac{\left(1+C_{23}/C_{21}\right)}{\left(C_{11}/C_{23}\right)\cdot\left(C_{23}/C_{21}\right)} = \frac{1+AR_2}{AR_{12}\cdot AR_2}\end{aligned} \tag{16}$$

Equations (11) to (16) provide the values for physical parameters and initial conditions that generate the desired dynamic response of the test problem, described by the parameters in Equation (5). The range of validity of these expressions is limited by the fact that the physical parameters (m_1, m_2, k_1, k_2, k_3) must be positive. This constraint limits the values of the parameters defined in Equation (5) within the following limits:

$$\begin{aligned}FR<1 \quad &\Rightarrow \quad \begin{cases}AR_1\cdot AR_2 < 0\\ AR_{12}/AR_1 < 0\\ AR_{12}\cdot AR_2 > 0\end{cases}\\ FR>1 \quad &\Rightarrow \quad \begin{cases}AR_1\cdot AR_2 < 0\\ AR_{12}/AR_1 > 0\\ AR_{12}\cdot AR_2 < 0\end{cases}\end{aligned} \tag{17}$$

$$\left|AR_{12}\right| > \frac{FR^2-1}{\left|\left(\dfrac{FR^2}{AR_1}-AR_2\right)\right|} \tag{18}$$

$$\left|AR_{12}\right| < \left|\frac{FR^2-\dfrac{1}{AR_1\cdot AR_2}}{FR^2-1}\cdot AR_1\right| \tag{19}$$

2.2 Modeling approach

The proposed test problem is modeled using a weakly coupled co-simulation scheme that combines a general-purpose block diagram simulator with a multibody simulation software, a very common setup in the design and development of mechatronic systems. Simulink [2] has been selected as block diagram simulator, since it is a well-known tool in this field. However, the building blocks and modeling procedures employed in Simulink are also available in other block diagram simulators like SystemBuild and Scicos, and therefore the co-simulation techniques presented in this section are not particular to Simulink and can be implemented in other tools in a straightforward way. The multibody simulation software is a C++ in-house developed code.

The block diagram model is shown in Figure 3: the dynamics of m_1 (integrated by Simulink) is modeled in the upper part of the figure, where its acceleration goes through a double integration to obtain its position. On the other hand, the numerical integration of the dynamics of m_2 is performed in the external multibody simulation package embedded in the block named *cosimulation interface* located at the bottom of the figure. This block (of type *S-Function* in Simulink, *UserCode* block in SystemBuild or *C/Fortran* block in Scicos) contains a user function that evaluates the position and velocity of m_2 from the position and velocity of m_1. The design and behavior of this block will be described in the following subsection. Since the proposed test problem has no damping, the velocities are not needed to evaluate the equation terms, but they are represented in Figure 3 for the sake of generality.

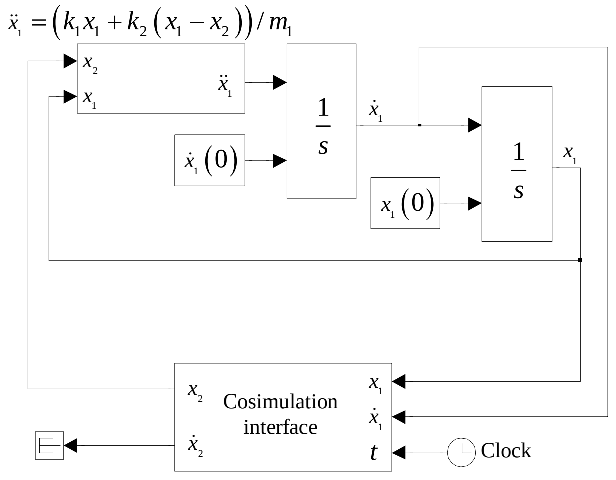

Figure 3: Simulink model of the test problem.

2.3 Coupling strategies for multirate integration

As explained in the Introduction, simulation environments used in weakly coupled co-simulations implement their own set of integration schemes that cannot be modified. Therefore, our purpose is to implement a coupling scheme that enables a multirate integration of m_1 and m_2 independently of the integration schemes and time-steps. In the proposed coupling scheme, the block diagram simulator acts as master integrator, since it is responsible of starting and stopping the numerical simulation. On the other hand, the external simulator acts as slave integrator, working on request.

Without loss of generality, it will be assumed that the block diagram simulator uses the well-known Fourth-Order Runge-Kutta formula, which is known as *ode4* in Simulink:

$$\begin{aligned} x_1^{i+1} &= x_1^i + h_1 \sum_{j=1}^{4} b_j K_j \\ K_1 &= f\left(t_1^i, x_1^i\right) \\ K_2 &= f\left(t_1^i + h_1/2, x_1^i + h_1 K_1/2\right) \\ K_3 &= f\left(t_1^i + h_1/2, x_1^i + h_1 K_2/2\right) \\ K_4 &= f\left(t_1^i + h_1, x_1^i + h_1 K_3\right) \end{aligned} \tag{20}$$

In order to advance a time-step from t_1^i to t_1^{i+1}, the block diagram simulator needs to evaluate all blocks in the model four times, one for each term K_j. The first evaluation is performed at $\left(t_1^i, x_1^i\right)$, using the states ($x_1$ in this case) computed from the previous time-step. In block diagram simulator terminology, this evaluation is known as a *major time-step*, while the next evaluations (corresponding to K_2, K_3 and K_4) are known as *minor time-steps*.

The *cosimulation interface* block in Figure 3 is responsible for evaluating the dynamic response of m_2 at the times required by the block diagram simulator. It contains a set of functions and data structures responsible for synchronizing the numerical integrations in the block diagram simulator and the external simulator. The structure and behavior of this block are represented in Figure 4. When the *cosimulation interface* block is evaluated at a given time, it calls the *eval_slave* function, whose algorithm is represented in Table 1 and will be described in the next paragraphs.

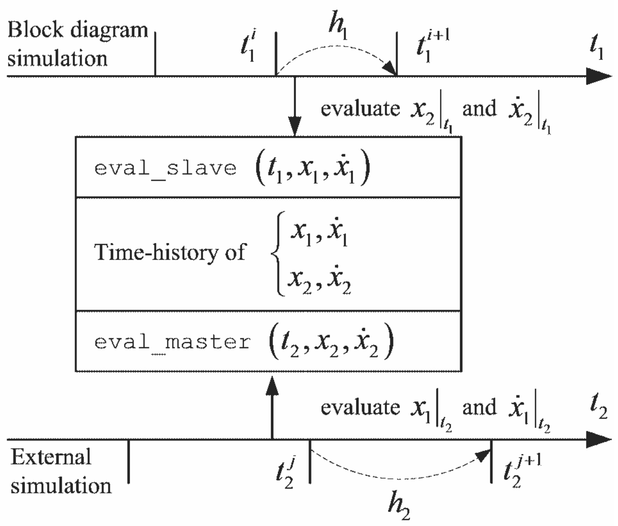
Block diagram simulation
t_1^i
h_1
t_1^{i+1}
t_1
evaluate $x_2\big|_{t_1}$ and $\dot{x}_2\big|_{t_1}$
eval_slave $(t_1, x_1, \dot{x}_1)$
Time-history of $x_1, \dot{x}_1$ / $x_2, \dot{x}_2$
eval_master $(t_2, x_2, \dot{x}_2)$
evaluate $x_1\big|_{t_2}$ and $\dot{x}_1\big|_{t_2}$
t_2
External simulation
t_2^j
t_2^{j+1}
h_2

Figure 4: *co-simulation interface* block working diagram.

Table 1: *eval_slave* function algorithm, in pseudo-code.

```
1) if $t_1$ is a major time-step
        store $\left(t_1^i, x_1^i, \dot{x}_1^i\right)$
        $n=0$
2a): if (slowest-first) then
        while $\left(t_2^{j+n} < t_1\right)$
                advance integration step in external simulator
                store results $\left(t_2^{j+n}, x_2^{j+n}, \dot{x}_2^{j+n}\right)$; $n=n+1$
        end
2b): if (fastest-first) then
        while $\left(t_2^{j+n} + h_2 < t_1\right)$
                advance integration step in external simulator
                store results $\left(t_2^{j+n}, x_2^{j+n}, \dot{x}_2^{j+n}\right)$; $n=n+1$
        end
3) Interpolate or extrapolate $x_2(t_1), \dot{x}_2(t_1)$ at $t_1$
```

In step 1, if the evaluation is performed in a *major time-step* (block diagram simulators provide routines to determine this condition), the input time and states (x_1 and $\dot{x}_1$ in this case) are appended to a dataset that holds the time-history of these values. Input states at *minor time-step* evaluations are not stored because they do not correspond to integration points in the timeline.

Step 2 determines if the external simulator should move ahead in the numerical integration of m_2. Two criteria are available to take this decision (steps 2a and 2b), depending on the selected synchronization scheme: *slowest-first* and *fastest-first* [10]. In the *slowest-first* scheme represented in step 2a, the numerical integration of the slowest subsystem (m_2, in the external simulator) is always ahead of the fastest subsystem (m_1, in the block diagram simulator). Therefore, when the *cosimulation interface block* is evaluated at $t_1 > t_2$, it calls the external simulator to move ahead in the numerical integration of m_2 a certain number of time-steps (represented by counter variable n) until $t_1 < t_2$. After each time-step, the states of m_2 (x_2 and $\dot{x}_2$) are appended to a dataset that holds the time-history of these values. In this process, the integration scheme of the external simulator will need the values of x_1 and $\dot{x}_1$ at particular instants: these values are interpolated or extrapolated from the time-history of *major time-steps* (stored in step 1) by the *eval_master* function. The *fastest-first* scheme represented in step 2b is very similar, but the numerical integration of the slowest subsystem m_2 is always one time-step h_2 behind the fastest subsystem m_1.

Finally, in step 3 the values of x_2 and $\dot{x}_2$ at t_1 requested by the block diagram simulator are interpolated or extrapolated from the time-history of the numerical integration of m_2, stored in step 2.

The interpolation or extrapolation of states in the *eval_slave* and *eval_master* functions is performed using order P polynomials. The user can select the value of P from 0 to 4. The polynomials are built with $P+1$ time-steps t^P ... t^0, selected as follows: t^P is the time-step closest to the evaluation time t that satisfies $t^P > t$ (if there is any time-step ahead of t), and t^{P-1} ... t^0 are the previous time-steps stored in the time-history.

The functions and data structures of the *cosimulation interface* have been implemented as a C/C++ library independent of the external simulator and the number of shared states. The external simulator only needs to provide two functions: a function to move ahead a time-step in the numerical integration and return the resulting states, and a user routine to hook the *eval_master* function. Most dynamics simulation tools can satisfy these requirements.

2.4 Smoothing techniques

For models with very different time scales in their subsystems (high values of *FR*), interpolation and extrapolation techniques may fail to give correct results in weak coupled multirate co-simulation. Oberschelp [16] described a smoothing technique to overcome this problem. A similar strategy has been tested in this work. Smoothing is expected to improve the global precision of the simulation, avoiding the need of raising the number of integration time-steps per cycle, or using higher degree integrators, which would noticeably increase the elapsed time in computations.

When using smoothing, the interpolation or extrapolation strategies described in the previous Section are replaced by an averaging of the values of the fast subsystem during the last time-step of the slow one. This averaging is performed on the basis of a *fastest-first* method, with the integration of the fast subsystem being performed in advance with respect to the slow one. It should be noted that the use of extrapolation techniques is still required during the calls to the *eval_slave* function, for the computation of the states of the slow subsystem at the times required by the fast one.

When the slow subsystem needs to evaluate its states at time t_2^n, it requests the states of the fast subsystem at this time, x_1^n, and their derivatives, through a call to the *eval_master* function. The value of the states of the fast subsystem at time t_2^n is determined by averaging the buffered values of x_1 in the time-history from time t_2^{n-1} to t_2^n, and the same operation is done with their derivatives. The averaged value, $\overline{x}_1^n$, is returned by the *eval_master* function and considered constant during the integration of the time-step of the slow subsystem.

2.5 Algebraic loops

Block diagram simulators allow creating algebraic loops in the model by connecting the output of a block to its input throughout direct feedthrough blocks (i.e., no differentiation or integration blocks). Algebraic loops are a convenient way to model certain problems, but they require an iterative solution at each time-step in the numerical integration. As a result, they drastically increase simulation times, which is usually unacceptable for weakly coupled co-simulation of mechatronic systems. Several techniques can avoid algebraic loops: *delay* and *memory* blocks, which delay the value of a variable one time-step, are examples. It is very convenient to test the proposed multirate method with this modeling technique, since it is often present in block diagram simulations.

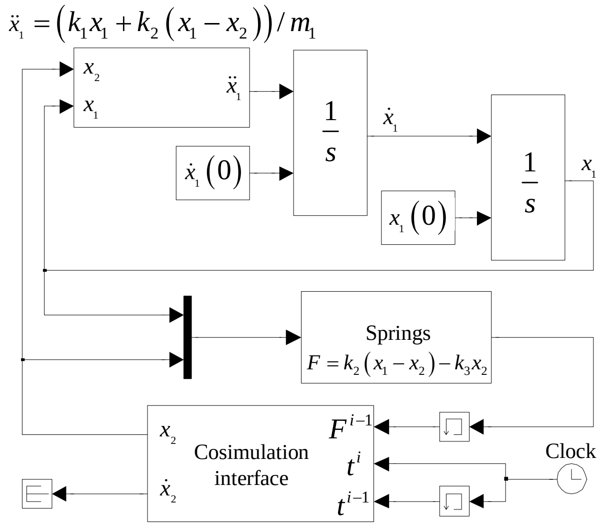

Figure 5: Simulink model with memory blocks to avoid algebraic loop.

In the model shown in Figure 3, spring forces acting on m_2 are evaluated inside the external simulator. If these forces are evaluated in the block diagram simulator, an algebraic loop can appear, as shown in Figure 5: the input force to the *cosimulation interface* is connected to its output x_2 throughout the direct feedthrough block *springs*. The algebraic loop is avoided by placing *memory* blocks in the force and time signals before entering the *cosimulation interface* block. This model will also be used to test the proposed multirate method.

2.6 Numerical experiments and error measurement

Preliminary investigations confirmed that the behavior of the multirate method is mostly affected by the frequency ratio *FR*, while the other ratios defined in Equation (5) do not have a significant impact. Therefore, the test problem described in Section 2 has been adjusted with AR_1 = 0.1, AR_2 = -1000 and AR_{12} = 0.1; see Figure 2 for an example of the dynamic response of x_1. A sweep of frequency ratios *FR* is performed in order to evaluate how this parameter affects the co-simulation process.

In the block diagram simulator (Simulink), the *ode4* integrator is used, while the multibody simulator uses the trapezoidal rule. Step-sizes h_1 and h_2 have been adjusted to perform 100 time-steps per cycle in each simulator. These step-sizes are small enough to keep integration errors very low in both subsystems. Each numerical experiment consists on a simulation of 100 cycles of the fastest frequency ω_1, which corresponds to 100/*FR* cycles of the slowest frequency ω_2.

The dynamic response obtained from the weakly coupled co-simulation is compared with the analytical solution of Equation (3). The error in the numerical simulation is measured in two ways: position error and energy error. Position error is given by Equation (21)

$$\Delta x = \frac{FR}{N}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - x_i^{exact}}{x_{rms}}\right)^2} \tag{21}$$

where x_i is position at time t_i obtained in the numerical simulation, x_i^{exact} is the position at the same time obtained from the analytical solution in Equation (3), and n is the number of points of time in the time-history of the solution (n = 10,000). To obtain a relative error, the absolute error in position is divided by the quadratic mean in the simulation (x_{rms}) instead of x_i^{exact} to avoid singularities when the analytical solution takes values close to zero. N = 100 is the number of simulated cycles of the fast subsystem, and the factor *FR/N* is introduced to correct the accumulation of errors when a high number of cycles of the slow subsystem are present. In this way, errors obtained from Equation (21) are comparable between numerical experiments with different *FR* ratios. If the test problem is fully modeled and solved in Simulink (without co-simulation) with the *ode4* integrator and abovementioned step-size, the position error given by Equation (21) is in the order of 10^{-8}, which corresponds to an almost exact solution. Position errors below 10% still correspond to a good numerical solution, which cannot be distinguished from the analytical solution at first glance.

However, Equation (21) gives high position errors when the numerical solution presents a small delay compared to the analytical solution, even when the phase difference is very small and the numerical solution can be still considered very good. Therefore, this position error can mislead about the precision in certain situations. To overcome this limitation, an additional measurement of the energy error can be used, as the system is fully conservative. Thus, the energy error is defined as

$$\Delta E = \frac{FR}{N}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{E_i - E_0}{E_0}\right)^2} \tag{22}$$

being E_0 the initial value of the energy of the system (that should be constant during the simulation), and E_i the energy at time t_i obtained in the numerical simulation. The oscillations that have been observed in the energy history of the system (see Figure 6) justify the use of a norm-2 error instead of a simple comparison between the initial and final energy levels of the system.

It has been observed that some numerical simulations deliver a low energy error despite the position time-history is obviously incorrect: the numerical integration conserves the system energy but gives a wrong solution after a few cycles. Therefore, both errors (position and energy) should be considered to determine the precision of the obtained numerical solutions.

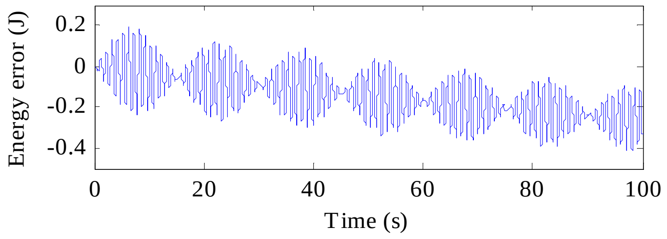

Figure 6*:* Time-history of the energy error in the numeric simulation (*FR = 30*), with cubic interpolation.

3 RESULTS AND DISCUSSION

3.1 Performed simulations

Both *fastest-first* and *slowest-first* approaches have been tested. They will be referred to in the following as *FF* and *SF*. In addition, the interpolation orders used in *eval_master* and *eval_slave* functions can be different and one of the following: zero (constant value, designed as *O0*), linear (*O1*), quadratic (*O2*), cubic (*O3*) and fourth order (*O4*).

The position error for x_1 and the energy error, defined in Equations (21) and (22), have been measured for each interpolation method for a span of *FR* ranging from 1.5 to 100. Results can be seen in Figure 7.

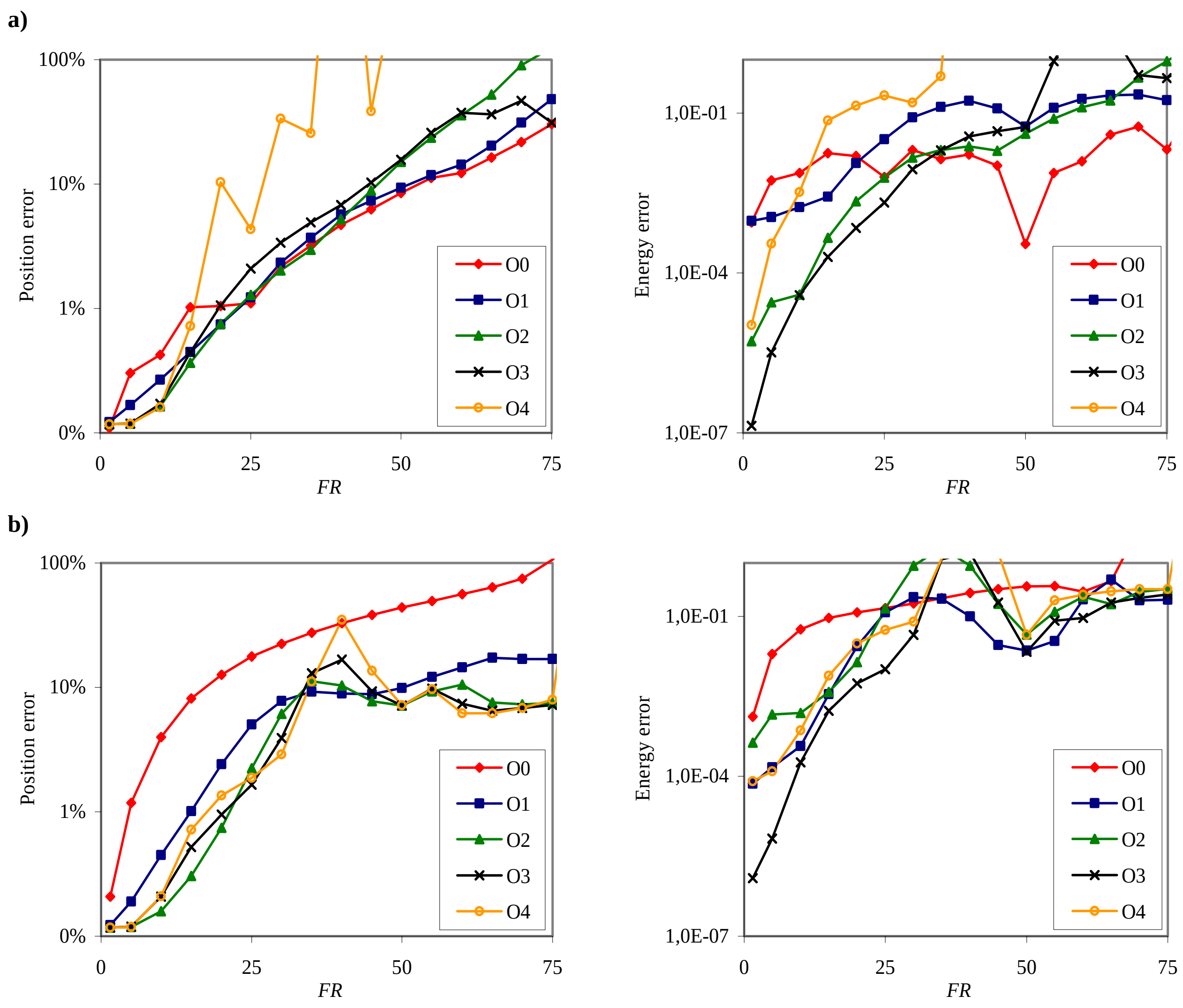

Figure 7*:* Errors in the position of x_1 (left) and energy (right) for different interpolation techniques as a function of *FR*, with *SF* (a) and *FF* (b) strategies.

The performed simulations show that it is not possible to find an optimal "general purpose" co-simulation method, even for such a simple test problem as the one described in the previous Section.

For *FR* < 25, *slowest-first* (*SF*) integration combined with cubic interpolation (*O3*) shows the best performance, attaining good position and energy error levels. The use of higher order interpolation polynomials suffers from instabilities, which results in the losing of the reference solution, and therefore has not helped the reduction of the errors. *FF* techniques, on the other hand, attain very low error levels in the integration of the position of x_2, as it was expected, because the integration of the slow subsystem is performed on the basis of already evaluated values of x_1; however, this improvement is made at the cost of worsening the energy levels and the shape of the time-history of x_1.

For 25 < *FR* < 50, *SF* integration without interpolation (*O0*) seems to be the most suitable strategy. The use of *FF* strategies in this range of frequency ratios leads to a numerical instability that translates into the amplification of the oscillations in x_1, and can be visualized in Figure 7 as a peak in the error graphics around *FR* = 40.

For *FR* > 50, the position errors with *SF* strategies are always over 10% and they follow an upwards trend; among them, the use of no interpolation (*O0*) gives the best results in position and energy. On the other hand, *FF* techniques seem to stabilize the position error in this region under 10% with reasonable levels of energy errors, at least with *O2* and higher interpolation degrees. However, the analysis of the position history shows that this is a consequence of the attenuation of the fast oscillations of the first subsystem, m_1. In fact, when *FR* grows to values of 80 and higher, the inverse effect takes place and the oscillations are amplified, leading to great errors in position and energy. In both cases -amplification and attenuation- the results cannot be considered valid, even when low error levels in both position and energy are attained.

Two consequences can be inferred from the exposed:

- The errors defined in Equations (21) and (22), and used as indicators of the correctness of the solution, are not enough for determining the suitability of a co-simulation method for solving a particular problem.
- The use of *FF* strategies can lead to the rising of numerical instabilities, resulting in amplified oscillations in the solution of the problem or well, on the contrary, to the filtering of small oscillations, with the loss of the contribution of the fast frequency ω_1 to the solution.

For values of *FR* > 90, even *SF* with O0 configuration is affected by a sudden growth of the errors and every interpolation degree fails completely to follow the analytic reference solution.

The use of smoothing techniques can help the reduction of the error for relatively high values of *FR*, increasing the ability of the simulation to track the reference solution. In order to attain acceptable results, the polynomial fitting interpolation methods for the evaluation of the states of the slow subsystem can be substituted with least squares approximations. This can help to *filter* the stiff variations in velocities that arise when the difference of time-steps grow.

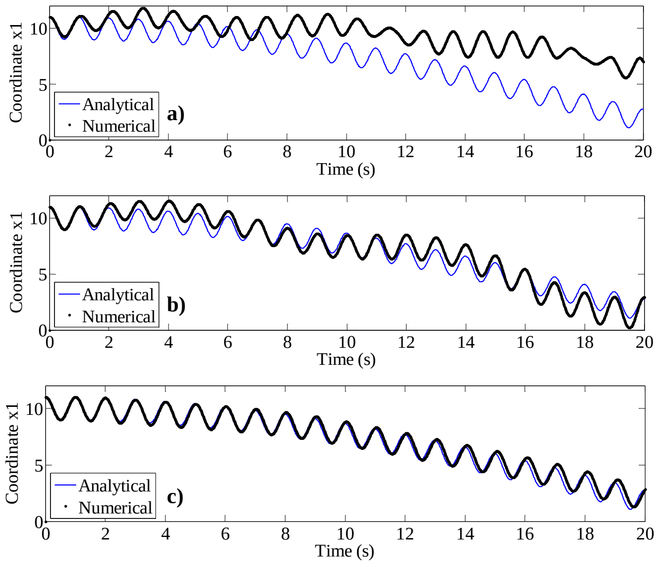

Figure 8: Response to 20 simulation cycles of the fast subsystem. a) *Slowest-first* with *O0*. b) *Fastest-first* with *O3*. c) Smoothing with *O3*. *FR* = 90.

A comparison of the co-simulation results can be seen in Figure 8. The co-simulated output for variable x_1 is compared to the analytical solution of the motion (thin continuous line); in the upper image no interpolation has been used, in the central graphic, *O3* interpolation has been used in *eval_master* and *eval_slave* functions. The lower image shows the better accuracy obtained using the smoothing technique for the same problem with *O3* interpolation in *eval_slave* function. However, it must be noted that smoothing is subject to the same filtering or amplifying problems that *fastest-first* implementations suffer. As a consequence, smoothing has only shown an acceptable performance for certain combinations of *FR* and the interpolation (or approximation) algorithm used for the slow subsystem.

Regarding to the equivalent model with an algebraic loop, depicted in Figure 5, the obtained results have been practically equivalent to those of the original model of Figure 3. The use of *memory* blocks has yielded a better performance than the equivalent model with *delay* blocks.

In most simulations, it has been observed that the accumulated error grows as simulation time increments. This is not expected to happen in real systems for two reasons. First, real systems use to have dissipative elements like dampers that soften the effect of small vibrations. In the second place, most co-simulated systems include control elements, oriented to reference tracking, which make the whole system less sensitive to error accumulation.

4 CONCLUSIONS

Several common requirements for co-simulation involving a general-purpose block diagram simulator and an external simulation tool have been identified. A multirate co-simulation interface for block diagram simulators has been designed and assessed on a simple test problem. This interface allows the use of different integrators and time scales in each subsystem of the whole model. Two co-simulation strategies, *fastest-first* and *slowest-first*, have been programmed, and several orders of interpolation and approximation methods have been implemented and compared in order to find the most suitable co-simulation configuration as a function of the different time scales of the subsystems. Additionally, smoothing techniques have also been implemented and tested.

Results suggest that the configuration of the interface must be changed depending on the frequency ratio *FR* between subsystems, in order to attain the best performance. The interpolation polynomial degree and the *fastest-first*/*slowest-first* configuration should be adjusted as a function of the most critical subsystem, in which lower errors are allowed. In general lines, the following conclusions can be pointed:

- For low frequency ratios (*FR* < 50), *SF* is the best co-simulation strategy. *FF* can be better for certain values of *FR*, especially for *FR* > 50, but it is subject to numerical problems that can lead to the degeneration of the solution.
- For *FR* < 25, cubic interpolation (*O3*) with *SF* has yielded the best results. In the range 25 < *FR* < 50, no interpolation (*O0*) with *SF* is recommended.
- For *FR* > 50, energy and position (for the variable x_1) errors grow up to high levels with every co-simulation strategy. The use of smoothing techniques can help the reduction of these errors for certain values of *FR*.

Currently, research is being carried out in order to find a suitable indicator of the ability of a co-simulation method to simulate a certain problem. Future research is needed for extending the range of valid frequency ratios the interface is able to tackle, particularly with the development of techniques that overcome the problems that appear for frequency ratios over *FR* = 50.

ACKNOWLEDGMENTS

This research has been sponsored by the Spanish MEC (grant No. DPI2006-15613-C03-01 and the F.P.U. Ph.D. fellowship No. AP2005-4448).

REFERENCES

[1] Samin, J. C., Bruls, O., Collard, J. F., Sass, L., and Fisette, P., 2007, "Multiphysics Modeling and Optimization of Mechatronic Multibody Systems," Multibody System Dynamics, **18**(3), pp. 345-373.

[2] The Mathworks, Inc., 2009, "MATLAB," http://www.mathworks.com/.

[3] National Instruments, 2009, "MATRIXx/SystemBuild," http://www.ni.com/matrixx/what_is_matrixx.htm.

[4] INRIA, 2009, "Scilab," http://www.scilab.org/.

[5] MSC.Software Corporation, 2004, "ADAMS," http://www.mscsoftware.com/.

[6] Intec GmbH, 2009, "SIMPACK," http://www.simpack.de/.

[7] Function Bay Inc., 2004, "RecurDyn," http://www.functionbay.co.kr/.

[8] Liao, Y. G., and Du, H. I., 2001, "Cosimulation of Multi-Body-Based Vehicle Dynamics and an Electric Power Steering Control System," Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics, **215**(3), pp. 141-151.

[9] Vaculin, O., Kruger, W. R., and Valasek, M., 2004, "Overview of Coupling of Multibody and Control Engineering Tools," Vehicle System Dynamics, **41**(5), pp. 415-429.

[10] Gear, C. W., and Wells, D. R., 1984, "Multirate Linear Multistep Methods," Bit, **24**(4), pp. 484-502.

[11] Engstler, C., and Lubich, C., 1997, "Multirate Extrapolation Methods for Differential Equations With Different Time Scales," Computing (Vienna/New York), **58**(2), pp. 173-185.

[12] Savcenco, V., Hundsdorfer, W., and Verwer, J. G., 2007, "A Multirate Time Stepping Strategy for Stiff Ordinary Differential Equations," BIT Numerical Mathematics, **47**(1), pp. 137-155.

[13] Verhoeven, A., Ter Maten, E. J. W., Mattheij, R. M. M., and Tasic, B., 2007, "Stability Analysis of the BDF Slowest-First Multirate Methods," International Journal of Computer Mathematics, **84**(6), pp. 895-923.

[14] Tesis DYNAWare, 2009, "veDYNA," http://www.tesis.de/en/index.php?page=544.

[15] Busch, M., Arnold, M., Heckmann, A., and Dronka, S., 2007, "Interfacing SIMPACK to Modelica/Dymola for Multi-Domain Vehicle System Simulations," SIMPACK News 11[2], 1-3.

[16] Oberschelp, O., and Vocking, H., 2004, "Multirate Simulation of Mechatronic Systems," LCM '04: Proceedings of the IEEE International Conference on Mechatronics 2004, pp. 404-409.