Multibody Syst Dyn (2011) 25: 461–483
DOI 10.1007/s11044-010-9234-7

On the effect of multirate co-simulation techniques in the efficiency and accuracy of multibody system dynamics

Francisco González · Miguel Ángel Naya · Alberto Luaces · Manuel González

Received: 3 March 2010 / Accepted: 3 November 2010 / Published online: 7 December 2010

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-simulation is not straightforward due to the limitations enforced by commercial simulation tools used in mechatronics design. This work describes a weakly coupled multirate method intended to be a generic multirate interface between block diagram software and multibody dynamics simulators, arranged in a co-simulation setup. Its main advantage is that it does not enforce equidistant or synchronized communication time-grids and, therefore, it can be easily applied to set up weakly-coupled co-simulations using off-the-shelf commercial block diagram simulators while giving the user a great flexibility for selecting the integration scheme for each subsystem.

The method is first tested on a simple, purely mechanical system with a known analytical solution and variable frequency ratio (*FR*) of the coupled subsystems. Several synchronization schemes (*fastest-first* and *slowest-first*) and interpolation/extrapolation methods (polynomials of different orders and smoothing) have been implemented and tested. Next, the effect of the interface on accuracy and efficiency is assessed making use of a real-life co-simulation setting that links an MBS model of a kart to a thermal engine modelled in Simulink. Results show that the proposed weakly coupled multirate method can achieve considerable reductions in the execution times of the simulations without degrading the numerical solution of the problem.

F. González (✉) · M.Á. Naya · A. Luaces · M. González
Escuela Politécnica Superior, Universidad de A Coruña, Mendizábal s/n, Ferrol 15403, A Coruña, Spain
e-mail: fgonzalez@udc.es

M.Á. Naya
e-mail: minaya@cdf.udc.es

A. Luaces
e-mail: aluaces@udc.es

M. González
e-mail: lolo@cdf.udc.es

Keywords Multibody · Co-simulation · Multirate · Weak coupling · Block diagram simulators

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 modelling of complex mechatronic systems can be accomplished via 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. Reviews about both strategies are provided in [1] and [2].

The weakly coupled strategy has important advantages over the strongly coupled one: specialized modelling 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 modelled and simulated in general-purpose block diagram simulators like Matlab/Simulink from Mathworks,[1] MATRIXx/SystemBuild from National Instruments[2] or the free open source tool Scilab/Scicos from INRIA.[3] Conversely, the behavior of complex mechanical components is better modelled and simulated in specialized tools for multibody system dynamics like MSC.Adams,[4] Simpack,[5] or Recurdyn[6]; these tools also provide interfaces to the aforementioned block diagram simulators, which simplify the setting of weakly coupled simulations. Representative examples of these kinds of co-simulation setups are given in [3] and [4], 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. Similar setups for the co-simulation of mechatronic systems are described in [5] and [6].

An important feature of complex mechatronic systems, derived from their multi-domain nature, is the presence of different time scales, which results in notably different dynamic

[1]The Mathworks, Inc.: MATLAB. http://www.mathworks.com/ (2009).

[2]National Instruments: MATRIXx/SystemBuild. http://www.ni.com/matrixx/what_is_matrixx.htm (2009).

[3]INRIA: Scilab. http://www.scilab.org/ (2009).

[4]MSC.Software Corporation: ADAMS. http://www.mscsoftware.com/ (2009).

[5]AG SIMPACK: SIMPACK. http://www.simpack.com (2009).

[6]Function Bay Inc.: RecurDyn. http://www.functionbay.co.kr/ (2009).

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 be 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 [7]. 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 [8], self-adjusting multirate time stepping strategies [9] and stability analysis of the proposed methods [10, 11].

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 modelled 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 modelling 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 guaranteed in weakly coupled co-simulations if one or more subsystems are integrated with a variable time-step integrator, since the stepsize control algorithms of the different commercial 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, 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 off-the-shelf simulation environments used in the mechatronics industry do not provide yet tools to enable truly multirate integration when they are used in weakly coupled co-simulation setups. Two examples of this situation are pointed: the first one is veDYNA,[7] 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 automobile. Non-mechanical elements, like electronic and control devices, are modelled in Simulink as usual, exchanging input and output data with the mechanical model. veDYNA uses an internal semi-implicit fixed-step

[7] Tesis DYNAWare: veDYNA. http://www.tesis.de/en/index.php?page=544 (2009).

Euler integration scheme to solve the equations of motion of the vehicle, and requires that the Simulink integration 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. A better approach to multirate integration is SIMAT, the interface provided by the multibody simulation software Simpack to perform co-simulation with Simulink. 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 stepsizes for the external communication of the integrators. Other commercial off-the-shelf packages for multibody system simulation have similar limitations.

Research is being carried out 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. [12] adapted the aforementioned Simpack's SIMAT interface to couple SIMPACK with the Modelica/Dymola simulation environment in order to evaluate different interpolation or extrapolation methods, but they enforced communication points that form an equally spaced time grid; in a similar way, Oberschelp and Vöcking [13] investigated the behavior of some multirate techniques in weakly coupled co-simulations using equidistant and synchronized time-grids.

The main goal of this work is to gain insight into the behavior and performance of multirate methods in weakly coupled co-simulation environments when applied to off-the-shelf commercial tools without enforcing equidistant and synchronized communication time-grids. To achieve this, an interface including an algorithm to implement a general multirate method (i.e. not constrained to synchronized time-grids or to a particular integration scheme) able to couple block diagram simulators with external simulation tools, like multibody simulation packages, has been developed. The proposed algorithm can be configured to work in different modes and to use different interpolation and extrapolation methods. Its use is demonstrated in a very simple example, which clearly shows the need for adjusting the interpolation method and the co-simulation strategy as a function of the nature of the mechanical system. The interface is later applied to a more complex example to evaluate the effect of multirate techniques on the efficiency and accuracy of industrial-like multi-domain simulations.

The remainder of the paper is organized as follows: Sect. 2 describes the multirate co-simulation interface created in this research and outlines the coupling strategy it uses. The test of this interface through the use of a simple, purely mechanical example with a known analytical solution is detailed in Sect. 3. In these two sections, several techniques for increasing the accuracy of the simulation are described and the convenience of their use discussed in the light of the found results. In Sect. 4, the interface is used again, this time in the co-simulation of a real-life application, in which the multibody model of a kart is coupled to a Simulink block diagram representing a thermal engine. This example has been used to measure the impact of multirate techniques on the time required to compute the simulation and the precision of the results. Finally, Sect. 5 extracts some conclusions from the work and discusses future lines of research.

2 Multirate co-simulation interface

In order to attain the goals of this paper, a new multirate interface has been designed and implemented, which allows using a weakly coupled co-simulation scheme that combines a general-purpose block diagram package with a multibody simulation software tool.

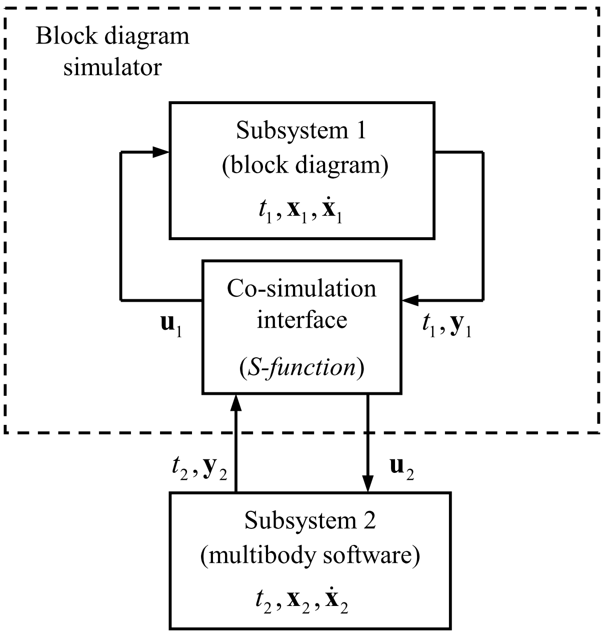

Block diagram simulator
Subsystem 1 (block diagram)
$t_1, \mathbf{x}_1, \dot{\mathbf{x}}_1$
Co-simulation interface (*S-function*)
$\mathbf{u}_1$
$t_1, \mathbf{y}_1$
$t_2, \mathbf{y}_2$
$\mathbf{u}_2$
Subsystem 2 (multibody software)
$t_2, \mathbf{x}_2, \dot{\mathbf{x}}_2$

Fig. 1 Use of the multirate co-simulation interface

This configuration is very common in the design and development of mechatronic systems. Simulink[1] has been selected as block diagram simulator, since it is a well-known tool in this field. However, the building blocks and modelling 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. On the other hand, the multibody simulation software is a C++ in-house developed code, specially optimized for the efficient simulation of dynamic systems [14, 15].

The generic use of the interface is shown in the block diagram model depicted in Fig. 1. The dynamics of the subsystem integrated by the block diagram package is modelled in the upper part of the figure. The states and the outputs of this subsystem are represented by $\mathbf{x}_1$ and $\mathbf{y}_1$, respectively, while t_1 stands for the time inside the block diagram software. The multibody software, in the lower part of the figure, tackles the numerical integration of the second subsystem, which has its own states, outputs, and time $\mathbf{x}_2$, $\mathbf{y}_2$, and t_2. The time-steps of the subsystems are denoted by h_1 and h_2; as the mechanical components in mechatronic devices are usually slower than the rest of the system (electronic devices and control elements, for example); it will be assumed in the following that the block diagram software manages the fastest subsystem in the model, while the external multibody software integrates the slowest one. This condition is equivalent to state that $h_1 < h_2$. The co-simulation interface is responsible for obtaining the inputs for each subsystem ($\mathbf{u}_1$ and $\mathbf{u}_2$) from the outputs supplied by both programs (which can include, but not necessarily, the states of the subsystem and their derivatives) and synchronizing the different time schemes of the subsystems. This interface is embedded in the block diagram simulator, in a block of type *S-function* in Simulink, *UserCode* in SystemBuild or *C/Fortran* block in Scicos. The design and behavior of this block will be described in the following paragraphs.

2.1 Coupling strategy for multirate integration

As explained in the Introduction, the 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 different subsystems independently of the integration schemes and time-steps that apply to each of them. In the proposed coupling scheme, the block diagram simulator (Simulink, in this case) plays the role of *master* integrator, since it is responsible for 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:

$$\begin{aligned}
\mathbf{x}_1^{i+1} &= x_1^i + h_1 \sum_{j=1}^{4} b_j \mathbf{K}_j, \\
\mathbf{K}_1 &= \mathbf{f}\big(t_1^i, \mathbf{x}_1^i\big), \\
\mathbf{K}_2 &= \mathbf{f}\big(t_1^i + h_1/2, \mathbf{x}_1^i + h_1\mathbf{K}_1/2\big), \\
\mathbf{K}_3 &= \mathbf{f}\big(t_1^i + h_1/2, \mathbf{x}_1^i + h_1\mathbf{K}_2/2\big), \\
\mathbf{K}_4 &= \mathbf{f}\big(t_1^i + h_1, \mathbf{x}_1^i + h_1\mathbf{K}_3\big),
\end{aligned} \tag{1}$$

where the b_j coefficients are weight factors whose values are $1/6$ for $j = 1, 4$ and $1/3$ for $j = 2, 3$. 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 $\mathbf{K}_j$. The first evaluation is performed at $(t_1^i, \mathbf{x}_1^i)$, using the states ($\mathbf{x}_1$ in this case) computed in the previous time-step. In block diagram terminology, this evaluation is known as *major time-step*, while the next evaluations (corresponding to $\mathbf{K}_2$, $\mathbf{K}_3$, and $\mathbf{K}_4$) are known as *minor time-steps*.

The *co-simulation interface* block in Fig. 1 manages the evaluation of the dynamic response of the second subsystem 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 software and the external simulator. The structure and behavior of this block are represented in Fig. 2. When the *co-simulation interface* block is evaluated at a given time, it calls its *eval_slave* function in order to get the inputs it needs. The algorithm of this function is represented in pseudo-code in Table 1 and will be described in the next paragraphs.

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 t_1 and outputs $\mathbf{y}_1$ in the block diagram are appended to a dataset that holds the time-history of these values. As it has been mentioned above, these outputs may include or not the states of the block diagram and their derivatives. Data at *minor time-step* evaluations are not stored because they do not correspond to integration points in the timeline.

Step 2 determines whether the external simulator should move ahead in the numerical integration of its subsystem. Two criteria are available to take this decision (steps 2a and 2b), depending on the selected synchronization scheme: *slowest-first* and *fastest-first* [7]. In the *slowest-first* scheme, represented in step 2a, the numerical integration of the slowest subsystem is always ahead of the fastest one. Therefore, when the *co-simulation interface* block is evaluated at $t_1 > t_2$, it calls the external simulator to move ahead in its numerical

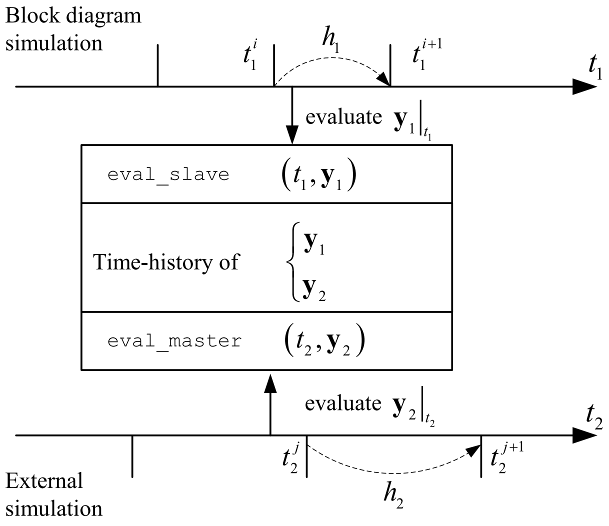


Fig. 2 Working diagram for the co-simulation interface block
Block diagram simulation
t_1^i
h_1
t_1^{i+1}
t_1
evaluate $\mathbf{y}_1|_{t_1}$
eval_slave $(t_1, \mathbf{y}_1)$
Time-history of $\mathbf{y}_1$ $\mathbf{y}_2$
eval_master $(t_2, \mathbf{y}_2)$
evaluate $\mathbf{y}_2|_{t_2}$
t_2
t_2^j
t_2^{j+1}
h_2
External simulation

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

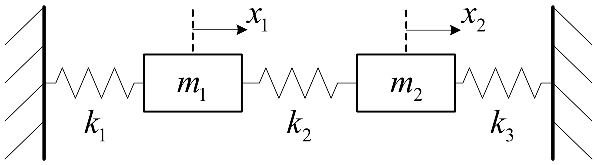
```
1) if $t_1^i$ is a *major time-step*
        store $t_1^i$, $\mathbf{y}_1^i$
        $n = 0$
2a) if (*slowest-first*) then
        while ($t_2^{j+n} < t_1^i$)
            advance integration step in external simulator
            store results ($t_2^{j+n}, \mathbf{y}_2^{j+n}$); $n = n + 1$
     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

integration a certain number of time-steps (represented by counter variable n) until $t_1 < t_2$. After each time-step of the external simulator, the time and the outputs of the slow subsystem, t_2 and $\mathbf{y}_2$, 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 its inputs $\mathbf{u}_2$ at particular instants; these values are interpolated or extrapolated from the time-history of outputs of the fast subsystem $\mathbf{y}_1$ at *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 is always one time-step behind the fastest one.

Finally, in step 3 the values of the inputs to the fast subsystem, $\mathbf{u}_1$, at time t_1, requested by the block diagram simulator are interpolated (or extrapolated, in case of the time t_1 is ahead of the last time at which the outputs of the second subsystem are available) from the time-history of the outputs of the slow subsystem $\mathbf{y}_2$, 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 $t_P, \ldots, t_0$, selected as follows: t_P is the

Fig. 3 Test problem



time-step closest to the evaluation time t that satisfies $t_P > t$ (if there is any time-step ahead of t), and $t_{P-1}, \dots, t_0$ are the previous time-steps stored in the time-history.

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

2.2 Smoothing techniques

For models with very different time scales in their subsystems, interpolation and extrapolation techniques may fail to give correct results in weakly coupled multirate co-simulation. Oberschelp and Vöcking [13] 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 order integrators, which would noticeably increase the elapsed time in computations.

When using smoothing, the interpolation or extrapolation strategies described above 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. When the slow subsystem needs to evaluate its states at time t_2^n, it requests the necessary inputs $\mathbf{u}_2^n$ through a call to the *eval_master* function. The value of these inputs is determined by averaging the buffered values of the outputs of the fast subsystem $\mathbf{y}_1$ in the time-history from time t_2^{n-1} to t_2^n. The averaged value is returned by the *eval_master* function, and considered constant during the integration of the whole time-step of the slow subsystem.

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.

3 Test problem

A test problem involving two subsystems with fast and slow dynamic responses will be solved by coupling a block diagram model in Simulink (to integrate the fast subsystem, 1) with an external multibody model (to integrate the slow subsystem, 2) through the multirate interface already introduced. 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.

The double-mass triple-spring system shown in Fig. 3 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. A similar mechanism has been used by Busch and Schweizer [16] to develop an analytical stability analysis of several co-simulation methods.

The dynamics of the test problem is governed by (2):

$$\begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix} \begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{Bmatrix} + \begin{pmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{pmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}, \tag{2}$$

where x_1 and x_2 measure the horizontal displacement of the masses from their equilibrium position. If initial velocities are set to zero, the analytical solution of these equations is given by

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

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 $\omega_1 > \omega_2$. Equation (3) depends on six parameters. For the purposes of this study, two of them are set to fixed values:

$$\begin{aligned} \omega_1 &= 1\ \text{Hz}, \\ C_{11} &= 1\ \text{m}, \end{aligned} \tag{4}$$

so the motion of the system can be characterized with the following ratios:

$$\begin{aligned} FR &= \omega_1/\omega_2, \\ AR_{12} &= C_{11}/C_{23}, \\ AR_1 &= C_{11}/C_{13}, \\ AR_2 &= C_{23}/C_{21}. \end{aligned} \tag{5}$$

The physical parameters of the system (stiffness and mass) and the initial values of the positions can be determined as a function of the parameters defined in (5), so that the system moves with the desired frequency and amplitude ratios [17].

- 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 Sect. 3.2 will use different sets of values for the ratios defined above, in order to reproduce diverse co-simulation situations. As example, Fig. 4 shows the dynamic response of x_1 for $FR = 30$, $AR_{12} = 0.1$, $AR_1 = 0.1$ and $AR_2 = -1000$.

The block diagram model used for the co-simulation of this test problem is shown in Fig. 5. In this Simulink model, the acceleration of the fast subsystem goes through a double

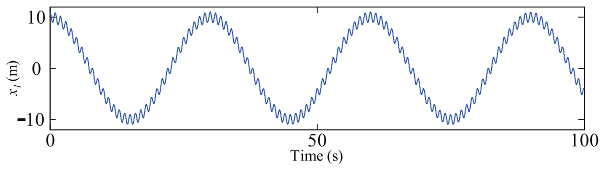


Fig. 4 Dynamic response of x_1

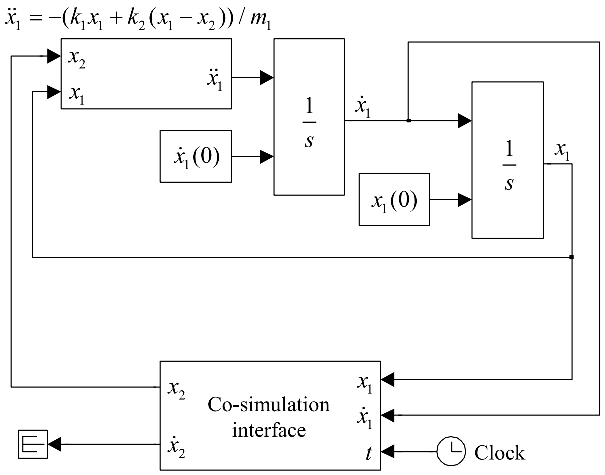


Fig. 5 Simulink model of the test problem

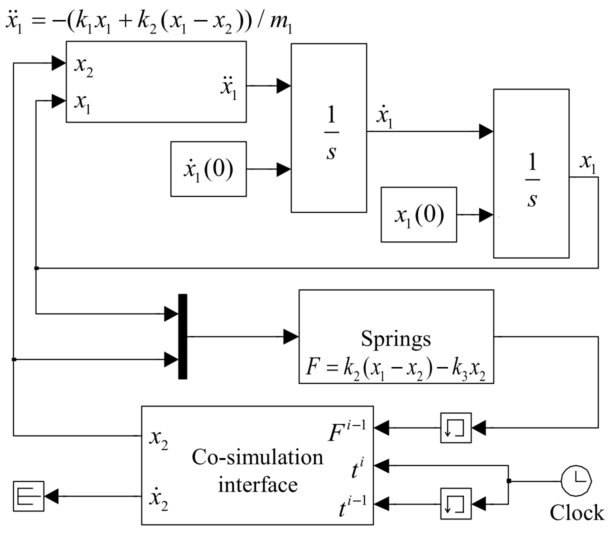
integration to obtain its position. This process is performed by Simulink integrator blocks. The dynamics of the slow subsystem (m_2) is evaluated in the external multibody simulation package and the communication is managed by the co-simulation interface as described in Sect. 2.1.

3.1 Algebraic loops

Block diagram simulators allow creating algebraic loops in the model by connecting the output of a block to its input via direct feedthrough blocks (i.e. without intermediate differentiation or integration blocks). Algebraic loops are a convenient way to model certain problems, but they have two drawbacks. First, they require an iterative solution at each time-step in the numerical integration carried out by the block diagram simulator; as a result, they drastically increase simulation times, which can become unacceptable for weakly coupled co-simulation of mechatronic systems. Secondly, they cause stability problems. Numerical stability of models with algebraic loops may be achieved by iterative coupling techniques (see Busch and Schweizer [16] and Kübler and Schiehlen [18]), but they cannot be applied to co-simulation setups using off-the-shelf commercial block diagram simulators like Matlab/Simulink. Several techniques can be used to avoid algebraic loops: *memory* blocks, which delay the value of a variable one time-step during the integration, are a common example. The outputs of a memory *memory* block are its inputs in the previous integration time-step, as expressed in the following mathematical relation:

$$y_n = u_{n-1} \tag{6}$$

Fig. 6 Simulink model with memory blocks to break algebraic loops



where y_n are the outputs of the block at time-step n and u_{n-1} its inputs in the previous step. It is very convenient to test the proposed multirate method with this modelling technique, since it is often present in block diagram simulations.

In the model shown in Fig. 5, spring forces acting on m_2 are evaluated inside the external simulator. When these forces are transferred to the block diagram simulator, an algebraic loop appears, as shown in Fig. 6: the input force F to the co-simulation interface is connected to its output x_2 through the direct feedthrough block springs. The algebraic loop is broken by placing *memory* blocks in the force and time signals before entering the *co-simulation interface* block. This model will also be used to test the proposed multirate method.

3.2 Numerical experiments and error measurement

Preliminary investigations confirmed that the behavior of the multirate simulation of the test problem is mostly affected by the frequency ratio FR, while the other ratios defined in (5) do not have a significant impact. Therefore, the test problem described in this section has been adjusted with $AR_1 = 0.1$, $AR_2 = -1000$ and $AR_{12} = 0.1$; see Fig. 4 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. Stepsizes h_1 and h_2 have been adjusted to perform 100 time-steps per cycle in each simulator. These time-steps are small enough to keep integration errors very low in both subsystems and, therefore, the error in the numerical solution will be mainly caused by the multirate co-simulation scheme. Each numerical experiment consists of a simulation of 100 cycles of the fastest frequency ω_1, which corresponds to $100/FR$ cycles of the slowest frequency ω_2. As the value of ω_1 is fixed by (4), all the simulations have the same duration in time.

The dynamic response obtained from the weakly coupled co-simulation is compared with the analytical solution of the motion given in (3). The error in the numerical simulation is measured in two ways: position error and energy error. Position error is given by (7)

$$\Delta x = \frac{FR}{N}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - x_i^{\text{exact}}}{x_{\text{rms}}}\right)^2} \tag{7}$$

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 (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 of the analytical solution during the simulation (x_{rms}) instead of by its value at each point, 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 is present. In this way, errors obtained from (7) are comparable through numerical experiments with different FR ratios. If the test problem is fully modelled and solved in Simulink (without co-simulation) with the *ode4* integrator and above mentioned stepsize, the position error given by (7) 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, (7) 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 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

$$\Delta E = \frac{FR}{N}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{E_i - E_0}{E_0}\right)^2} \tag{8}$$

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 Fig. 7) 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 lead to low energy errors 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.

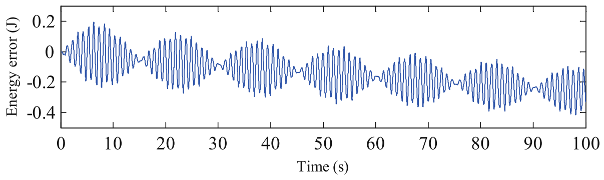


Fig. 7 Time-history of the energy error in the numeric simulation ($FR = 30$), with cubic interpolation

3.3 Effect of damping

Damping forces are typical of controlled multibody system models in mechatronic applications. Therefore, dissipation should be considered in the model in order to better represent real-life co-simulation scenarios.

The effect of damping has been considered adding dissipative elements (c_1 and c_3, in parallel with springs k_1 and k_3, respectively) to the test mechanical system represented in Fig. 3. The analytical solution of the resulting mechanism is also known, and the same definition of position error given in (7) has been used in to measure the accuracy of the set of simulations. The value of c_3 has been adjusted to a fraction ζ of the critical damping that the second mass would have in a one degree-of-freedom independent system. In the performed simulations, the value of the damping ratio ζ ranges between 0 and 0.5. The value of c_1 is adjusted proportionally:

$$c_3 = 2\zeta m_2 \omega_2; \qquad c_1 = 2\zeta m_1 \omega_1 / FR. \tag{9}$$

The division of c_1 by FR in (9) is justified by the fact that the first subsystem performs 100 cycles in each simulation, whereas the slow one performs only $100/FR$. Introducing this factor will make the relative damping of both subsystems comparable at any instant during the motion.

3.4 Results and discussion

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 (7) and (8), have been measured for each interpolation method for a span of FR ranging from 1.5 to 100. Results for the test problem without damping can be seen in Fig. 8.

The first conclusion that can be drawn from the performed simulations is 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 at the beginning of this 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 simulation losing track of the reference solution and, therefore, has not helped the reduction of the errors. This result is consistent with the conclusions of [11]. *Fastest-first* (*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 Fig. 8 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 upward 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

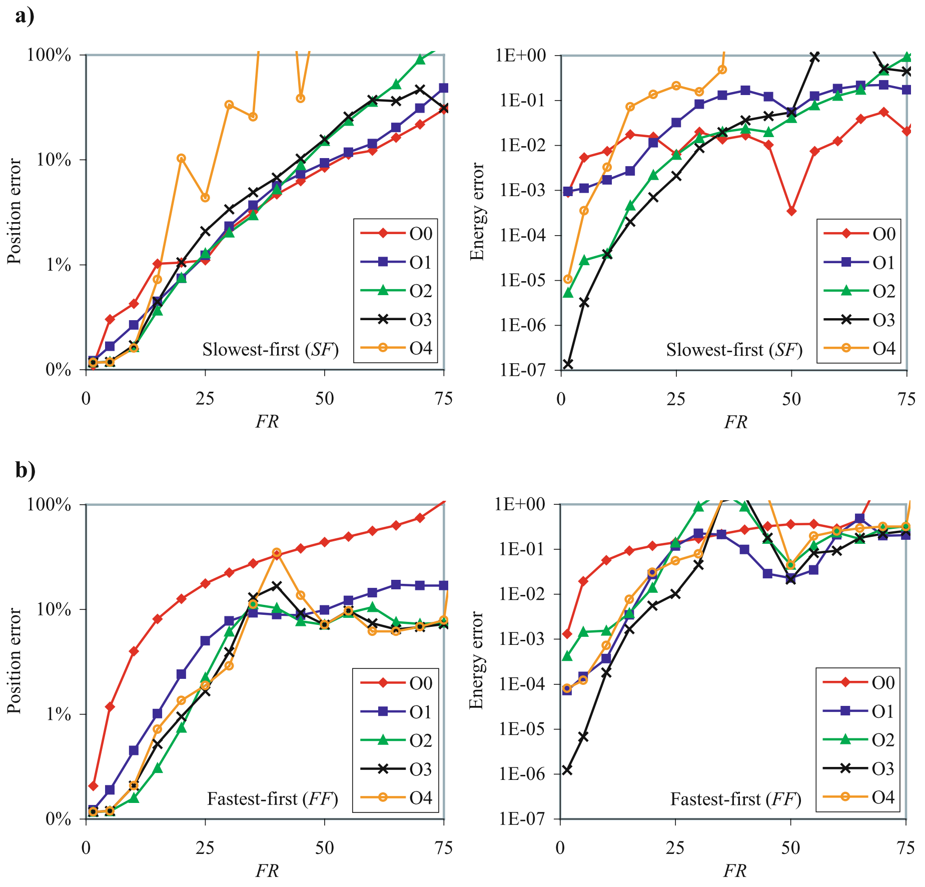


Fig. 8 Position error in x_1 (*left*) and energy error (*right*) for different interpolation polynomial orders as a function of *FR*, for *slowest-first* (**a**) and *fastest-first* (**b**) schemes

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 (7) and (8), and used as indicators of the correctness of the solution, are not enough for determining the suitability of a co-simulation method for solving every 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, on the contrary, in 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.

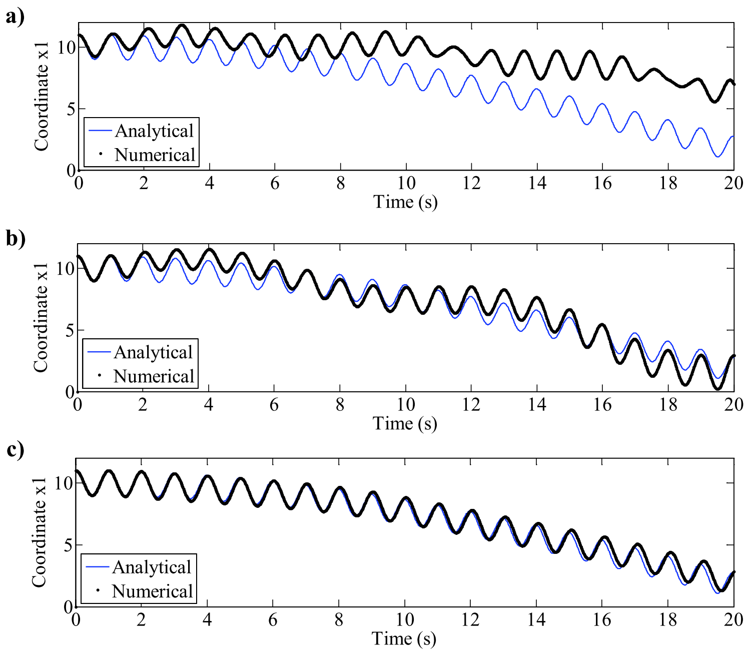


Fig. 9 Response to 20 simulation cycles of the fast subsystem for $FR = 90$; (**a**) *Slowest-first* with $O0$. (**b**) *Fastest-first* with $O3$. (**c**) Smoothing with $O3$

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 between the time-steps grows.

A comparison of the co-simulation results for $FR = 90$ without damping can be seen in Fig. 9. 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 ($O0$) has been used; in the central graphic, $O3$ interpolation has been used in *eval_master* and *eval_slave* functions, together with *FF* strategy. The lower image shows the better accuracy obtained using the smoothing technique 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 from. 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 the equivalent model with an algebraic loop and *memory* blocks, depicted in Fig. 6, the obtained results have been practically equivalent to those of the original model of Fig. 5.

In most simulations, it has been observed that the accumulated error grows as the simulation time increases. This fact is expected to be mitigated in real-life complex multiphysics systems for two reasons. First, real systems use to have dissipative elements like dampers that soften the effect of vibrations. In the second place, most co-simulated systems include

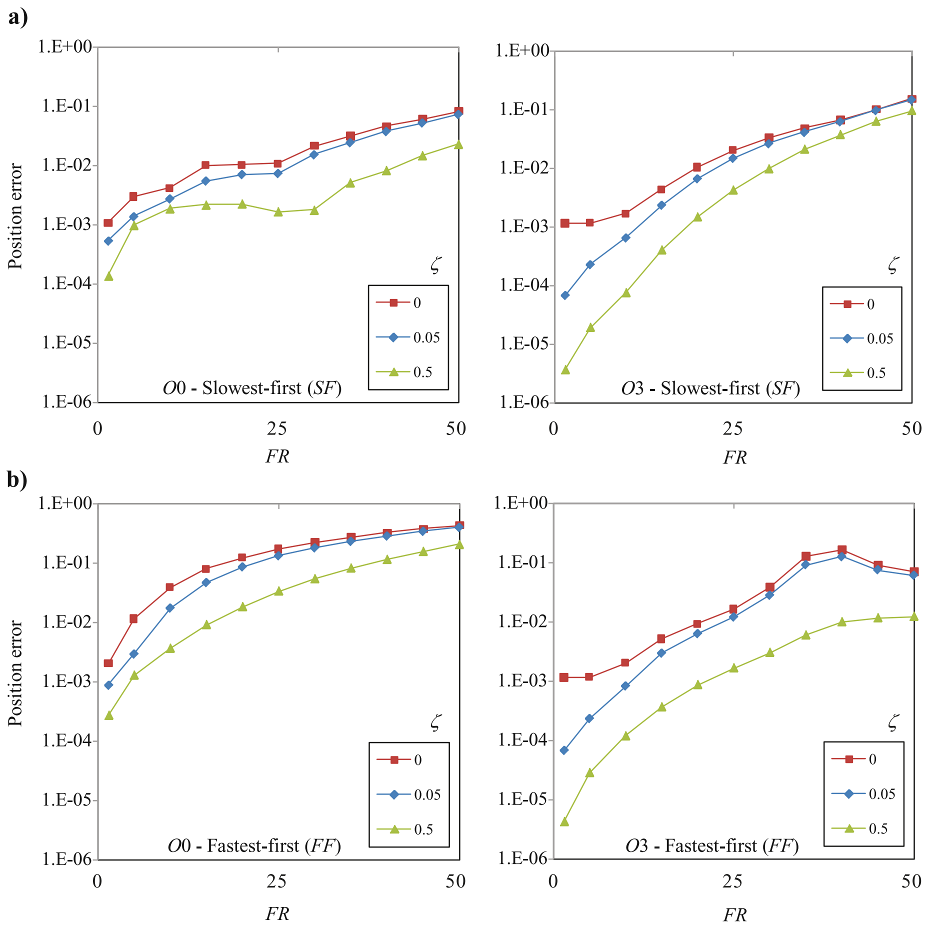


Fig. 10 Position error in x_1 for $O0$ (*left*) and $O3$ (*right*) for different values of the damping ratio (ζ) as a function of *FR*, for *slowest-first* (**a**) and *fastest-first* (**b**) schemes

control elements oriented to reference tracking, which make the whole system less sensitive to error accumulation.

In order to investigate the behavior of the multirate co-simulation algorithm in the presence of dissipative elements, the numerical experiments have been repeated using the test problem with damping described in Sect. 3.3. Results of this series of simulations are summarized, for representative orders of the interpolation polynomials, in Fig. 10.

Two conclusions can be extracted from the numerical experiments with damping elements: the first one is that, as expected, the introduction of damping in the system helps to reduce the position errors in the simulations. The reduction in position errors consistently decreases as larger values of the damping are used in the simulations: as a rough measure, a damping ratio ζ of 50% reduces the position errors around an order of magnitude. In the second place, for the range of damping coefficients and interpolation polynomials used, the addition of damping does not change qualitatively the performance of the co-simulation interface as a function of *FR*. This means that the errors in the tracking of the reference solution (amplification or attenuation of oscillations) described in the previous paragraphs appear at

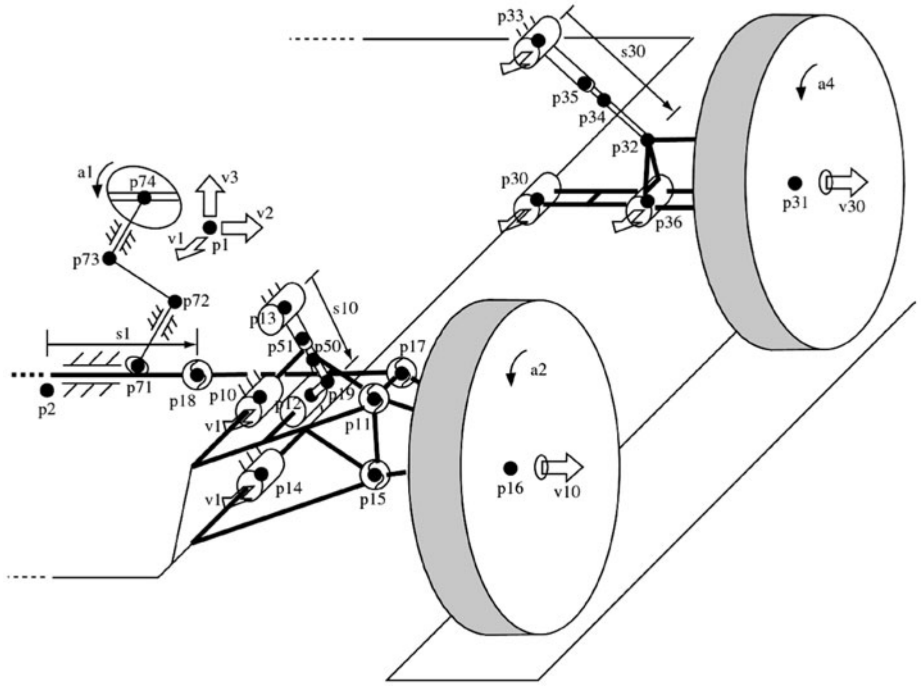


Fig. 11 Multibody model of the vehicle used in simulations

the same frequency ratios regardless of the addition of damping, although mitigated in case of high values of the damping ratio ζ.

4 Application to a multiphysics problem

The use of the multirate co-simulation interface described in Sect. 2 is intended to reduce the computational effort associated with the numerical integration of the equations of motion of a multiphysics system, keeping at the same time the errors derived from interpolation and extrapolation under reasonable limits. It is desirable thus to quantify both the increase of efficiency and the introduced numerical errors, in the context of the simulation of a non-academic, real-life test problem. To this end, the multirate interface and co-simulation simulation methods described above have been applied to the solution of the dynamics of a vehicle, in this case a kart. This multiphysics model is divided into two subsystems: a multibody model of the mechanical components of the vehicle, including the steering column, tyres and suspensions, and a thermodynamic model of a four-cylinder spark ignition engine.

The model of the mechanical components of the kart can be seen in Fig. 11; the figure represents only half of the model, the actual one includes the suspension of the four wheels and the whole chassis. The number of variables of the multibody system is 163, and the motion is integrated making use of the well-known index-3 augmented Lagrangian formulation with projection of velocities and accelerations. This formalism uses the trapezoidal rule as numerical integrator, and has been described by Cuadrado et al. in [19]. The multibody code is implemented in Fortran and its configuration is detailed in [20].

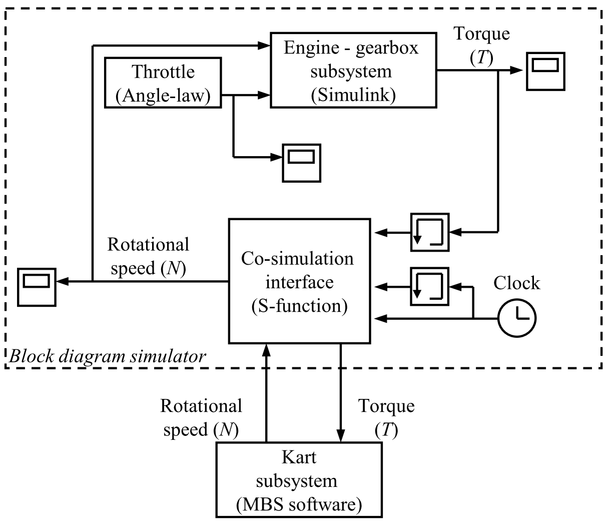


Fig. 12 Joint model of the engine and the kart

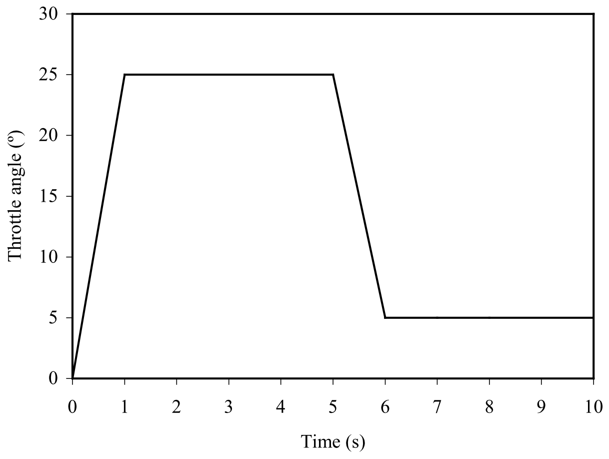


Fig. 13 Throttle angle during simulations

The engine is modelled in Simulink, following the description given in [21], using conventional diagram blocks and adding an automatic gearbox to link it to the transmission. The block diagram model that corresponds to this system is shown in Fig. 12. The upper part of the graphic represents the Simulink model of the engine and gearbox, which also includes the *co-simulation interface*, described in Sect. 2. *Memory* blocks are used to avoid the closing of an algebraic loop. The code for the simulation of the mechanical components of the kart is compiled as a library and invoked from the co-simulation interface. The model undergoes a maneuver in which the angle of the throttle varies following the law depicted in Fig. 13. The pitch angle of the vehicle (ψ) is taken as control variable, to check if the setting behaves in an adequate way. This variable is closely related to the acceleration of the vehicle.

The Simulink part of the model is integrated with *ode4*, and the nature of the system it models requires using a time-step of $h_1 = 10^{-4}$ s. The multibody subsystem can be integrated with trapezoidal rule with time-steps as big as $h_2 = 10^{-2}$ s without significant errors.

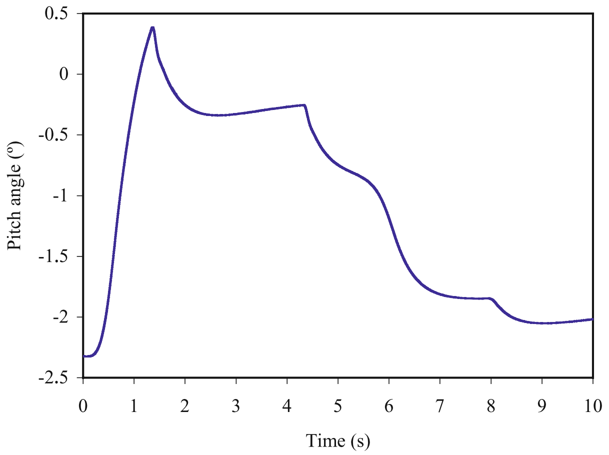


Fig. 14 Pitch angle ψ in reference case

Table 2 Elapsed time in calculations and maximum difference in pitch angle ($\Delta\psi$) with respect to the reference case, for different values of *FR*, with *SF* and $O0$ in both subsystems

	$FR=1$ (Ref.)	$FR=5$	$FR=10$	$FR=50$	$FR=100$
Elapsed time (s)	158.4	44.8	30.4	19.0	17.1
$\Delta\psi$ (°)	0	0.0031	0.0055	0.0252	0.0398

A direct co-simulation scheme with the same time-step in both subsystems would be forced to use the smallest one to keep numerical accuracy in the fast component, leading to a considerable increase in the total computation time. In the performed simulations, the time-step in the Simulink subsystem has been kept constant, and the time-step of the multibody subsystem has been varied from $h_2 = 10^{-4}$ s to $h_2 = 10^{-2}$ s in order to measure the effect of using multirate integration on the accuracy and efficiency of the simulation. The case in which both subsystems are integrated with the same time-step $h_1 = h_2 = 10^{-4}$ s and constant interpolation $O0$ is taken as the reference solution; the pitch angle in this case is shown in Fig. 14. The shape of the pitch angle curve in this graphic agrees with the angle law for the throttle depicted in Fig. 13. The sudden drops in the angle between seconds 4 and 5 and at second 8 correspond to the moments when the gear of the vehicle is changed by the automatic gearbox.

The total computing time of the 10 s simulation under the reference conditions $h_1 = h_2 = 10^{-4}$ s exceeds 150 s. The use of multirate co-simulation is expected to reduce the total computing time; however, it is also reasonable to expect divergences to occur in the results with respect to the reference solution. In order to measure the impact of multirate simulation in the elapsed time in computations and the deviations from the reference value, simulations at different values of *FR* have been carried out. It must be noted that the meaning of *FR*, for complex multiphysics problems like the one here discussed, does not correspond to the ratio between the natural frequencies of the subsystems (which may not be easy to identify), but it must be substituted by the relation between the time-steps used to integrate them. For this first set of simulations, constant interpolation ($O0$) and *slowest-first* strategy have been used. Besides the computation time, the maximum deviation in pitch angle ψ with respect to the reference case during the motion has been measured.

The results summarized in Table 2 show a dramatic reduction in computing time as the value of *FR* increases. Regarding to the differences in the pitch angle ψ, these are never

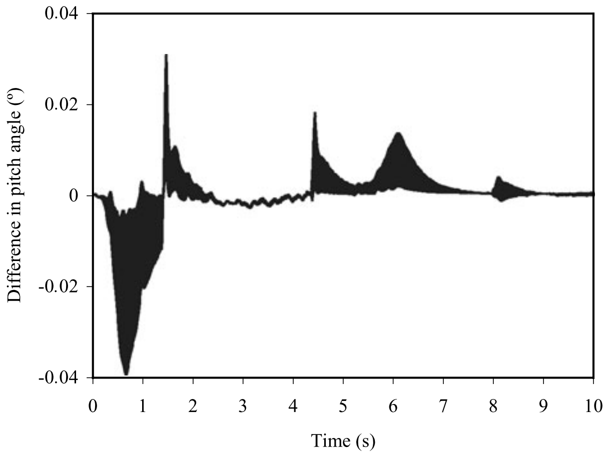


Fig. 15 Difference in pitch angle ($\Delta\psi$) with respect to the reference case, with $FR = 100$, SF and $O0$ in both subsystems

Table 3 Maximum difference in pitch angle ($\Delta\psi$) with respect to the reference case for $FR = 100$. Only representative interpolation strategies are represented

	SF	FF	FF	FF	FF
Simulink interpolation	$O0$	$O0$	$O0$	$O0$	$O0$
MBS interpolation	$O0$	$O0$	$O1$	$O2$	$O3$
$\Delta\psi$ (°)	0.0398	0.0385	0.0078	0.0086	0.0303

higher than 0.04° in absolute value, for a variable that oscillates between $-2.5°$ and 0.5°. This means that direct co-simulation, with the use of $O0$ polynomials, is able to simulate the system without significant deviations in the results, with values of FR up to 100. The plots of the pitch angle for the different values of FR overlap Fig. 14, so they are indistinguishable in practice. A graphical representation of the deviation of the control variable with respect to that of the reference case has been chosen instead, and it can be seen in Fig. 15 for $FR = 100$. The sudden variations of the measured deviation make the results in this graphic look like a solid region, but a line is actually represented.

Figure 15 shows another relevant feature of the behavior of the co-simulated system: the divergences in pitch angle increase when sudden variations of the variable happen, but the error gets close to zero when the angle varies slowly. This stable behavior of the whole system agrees with the conclusions stated in Sect. 3.4.

As it was shown in the previous section, it is not possible to determine beforehand whether the use of higher order interpolation polynomials or other co-simulation techniques will enhance the obtained results. More simulations have been performed to gain insight into this subject; the most relevant ones are summarized in Table 3 for $FR = 100$. The elapsed time is not shown, as there are not significant differences between the methods. Other configurations have been tested (alternative combinations of orders in interpolation polynomials and smoothing techniques) but their use has not improved the precision of the simulation. As it can be drawn from the table, there is no gain in rising the degree of the polynomials beyond one, as the linear case yields the best results for this problem.

The time-history of the pitch angle in the case that performs best in Table 3 is represented in Fig. 16. The comparison of this graphic to the one in Fig. 15 highlights the benefits of using the *fastest-first* configuration and linear polynomials for the interpolation of the data from the MBS software in this particular case.

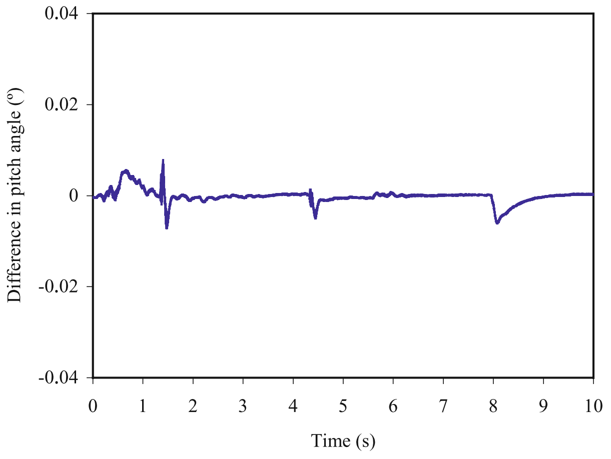


Fig. 16 Difference in pitch angle ($\Delta\psi$) with respect to the reference case, with $FR = 100$, FF, $O0$ interpolation in Simulink and $O1$ interpolation in MBS

5 Conclusions and future work

In this research, the effect of multirate techniques in the efficiency and accuracy of weakly coupled co-simulated settings has been assessed. To this end, a general multirate co-simulation interface for coupling block diagram simulators and external tools has been built, and a synchronization algorithm has been designed, in order to coordinate the exchange of information between both software packages. The developed interface avoids techniques which are not available in block diagram simulators (iteration or modification in the integration schemes) and does not enforce equidistant or synchronized communication time-grids. Therefore, it can be easily applied to set up weakly-coupled co-simulations using off-the-shelf commercial block diagram simulators while giving the user a great flexibility for selecting the integration scheme for each subsystem.

The way in which the interface operates is based on interpolation and extrapolation of inputs and outputs between simulators using polynomial approximations, and two synchronization schemes are available: *slowest-first* and *fastest-first*. This interface allows the user to select different co-simulation settings, such as the order of the interpolation polynomials, and incorporates additional techniques to improve the behavior of the simulation under certain conditions.

The algorithm has been implemented in C/C++ and tested in the co-simulation of the dynamics of a simple, purely mechanical system by coupling the well-known simulation tool Simulink with an in-house developed multibody dynamics simulator. The accuracy of the method was tested against the frequency ratio *FR*, which is equivalent to the ratio between the time-step sizes used in the two coupled simulators. The first test battery of the designed interface has revealed that the adjustment of the co-simulation settings is strongly dependent of the physical characteristics of the simulated subsystems. As a consequence, the co-simulation parameters must be adapted as a function of the particular features of the problem, and a general configuration, valid for any situation, cannot be found. In some cases, the use of smoothing is required in order to find a stable solution to the problem.

Next, the interface has been applied to the co-simulation of the multibody model of a real kart, simulated in a Fortran MBS code, powered by a thermal engine modelled in Simulink. Results show that the use of multirate techniques has been able to reduce the computation time required by the simulation in one order of magnitude, within reasonable margin of error. In this case, the use of first order interpolation polynomials ($O1$) has contributed to alleviate the deviations of the motion with respect to the reference solution. The example is very representative of the co-simulation of complex mechatronic systems, where the dynamic simulation of the mechanical components in a multibody software consumes around 60%–90% of the CPU-time, while the remaining time is consumed by the block diagram simulator. In these circumstances, increasing the stepsize in the multibody dynamics simulator by a factor of 50 can reduce the time needed to complete the simulations in a factor ranging from 2.4 to 8.5.

Currently, two lines of future research can be pointed out. First, a general numerical indicator is desirable, in order to measure in a practical and easy way the deviation of a solution with respect to a reference. And second, a way of determining the optimal co-simulation strategy before running the simulation would be very helpful, as it would remove the need of performing several trials to adjust the interface to the particular conditions of the simulated system.

Acknowledgements This research has been sponsored by the Spanish MEC (Grant No. DPI2006- 15613-C03-01 and the FPU Ph.D. fellowship No. AP2005-4448).

References

1. Samin, J.C., Brüls, O., Collard, J.F., Sass, L., Fisette, P.: Multiphysics modeling and optimization of mechatronic multibody systems. Multibody Syst. Dyn. **18**, 345–373 (2007)
2. Arnold, M.: Numerical methods for simulation in applied dynamics. In: Arnold M., Schiehlen W. (eds.) Simulation Techniques for Applied Dynamics. CISM Courses and Lectures, pp. 191–246. Springer, Wien/New York (2008)
3. Liao, Y.G., Du, H.I.: Cosimulation of multi-body-based vehicle dynamics and an electric power steering control system. Proc. Inst. Mech. Eng. K, J. Multibody Dyn. **215**, 141–151 (2001)
4. Vaculin, O., Kruger, W.R., Valasek, M.: Overview of coupling of multibody and control engineering tools. Veh. Syst. Dyn. **41**, 415–429 (2004)
5. Mikkola, A.: Utilization of coupled simulation in the fatigue loads prediction of a hydraulically driven log crane. In: ASME Biennial 2001, Pittsburgh, Pennsylvania (2001)
6. Teppo, J., Rouvinen, A., Mikkola, A., Kurronen, P., Salminen, P., Pyrhönen, O.: Coupled simulation of electrically driven machine system. In: Burrows C.R., Edge K.A. (eds.) Bath Workshop on Power Transmission and Motion Control (PTMC, 2001), pp. 103–116. Professional Engineering Publishing, London (2001)
7. Gear, C.W., Wells, D.R.: Multirate linear multistep methods. BIT **24**, 484–502 (1984)
8. Engstler, C., Lubich, C.: Multirate extrapolation methods for differential equations with different time scales. Computing (Vienna/New York) **58**, 173–185 (1997)
9. Savcenco, V., Hundsdorfer, W., Verwer, J.G.: A multirate time stepping strategy for stiff ordinary differential equations. BIT Numer. Math. **47**, 137–155 (2007)
10. Verhoeven, A., Ter Maten, E.J.W., Mattheij, R.M.M., Tasic, B.: Stability analysis of the BDF slowest-first multirate methods. Int. J. Comput. Math. **84**, 895–923 (2007)
11. Arnold, M.: Stability of sequential modular time integration methods for coupled multibody system models. J. Comput. Nonlinear Dyn. **5**, 031003 (2010)
12. Busch, M., Arnold, M., Heckmann, A., Dronka, S.: Interfacing SIMPACK to Modelica/Dymola for multi-domain vehicle system simulations. SIMPACK News **11**, 1–3 (2007)
13. Oberschelp, O., Vöcking, H.: Multirate simulation of mechatronic systems. In: LCM '04: Proceedings of the IEEE International Conference on Mechatronics, pp. 404–409 (2004)
14. González, M., González, F., Dopico, D., Luaces, A.: On the effect of linear algebra implementations in real-time multibody system dynamics. Comput. Mech. **41**, 607–615 (2008)
15. González, F., Luaces, A., Lugrís, U., González, M.: Non-intrusive parallelization of multibody system dynamic simulations. Comput. Mech. **44**, 493–504 (2009)
16. Busch, M., Schweizer, B.: Numerical stability and accuracy of different co-simulation techniques: analytical investigations based on a 2-DOF test model. In: Proc. of the 1st Joint International Conference on Multibody System Dynamics, Lappeenranta, Finland (2010)
17. González, F.: Efficient implementations and co-simulation techniques in multibody system dynamics. Ph.D. thesis, University of La Coruña (2010)

18. Kübler, R., Schiehlen, W.: Modular simulation in multibody system dynamics. Multibody Syst. Dyn. **4**, 107–127 (2000)
19. Cuadrado, J., Cardenal, J., Morer, P., Bayo, E.: Intelligent simulation of multibody dynamics: space-state and descriptor methods in sequential and parallel computing environments. Multibody Syst. Dyn. **4**, 55–73 (2000)
20. Naya, M.A., Dopico, D., Perez, J.A., Cuadrado, J.: Real-time multibody formulation for VR-based design and evaluation of automobile controllers. J. Multibody Dyn. **221**, 261–276 (2007)
21. Crossley, P.R., Cook, J.A.: A nonlinear engine model for drivetrain system development. In: International Conference on Control (Control 91), vols. 1–2, pp. 921–925, Edinburgh, Scotland (1991)