MULTIBODY DYNAMICS 2009, ECCOMAS Thematic Conference
K. Arczewski, J. Frączek, M. Wojtyra (eds.)
Warsaw, Poland, 29 June–2 July 2009

SIMULATION OF MULTIBODY SYSTEMS WITH THE USE OF COUPLING TECHNIQUES: A CASE STUDY

Paweł Tomulik[⋆], **Janusz Frączek**[⋆]

[⋆]Institute of Aeronautics and Applied Mechanics
Warsaw University of Technology, Nowowiejska 24, 00-665 Warsaw, Poland
e-mails: ptomulik@meil.pw.edu.pl, jfraczek@meil.pw.edu.pl

Keywords: Multibody Dynamics, Co-Simulation, Simulator Coupling, Distributed Integration, Gluing algorithm

Abstract. *Simulation coupling (or co-simulation) techniques provide a framework for the analysis of decomposed dynamical systems with the use of independent numerical procedures for decomposed subsystems. These methods are often seen as very promising because they enable the utilisation of the existing software for subsystem analysis and usually are easy to parallelize, and run in a distributed environment. For example, in the domain of multibody systems dynamics a general setup for "Gluing Algorithms" was proposed by Wang et. al.. It was intended to provide a basis for multi-level distributed simulation environment. The authors presented an example where Newton's method was used to synchronise the responses of subsystem simulators.*

In this paper we discuss some properties of simplified iterative coupling scheme, where subsystems' responses are synchronised at discrete time-points. We use a simple model of double pendulum to investigate the influence of synchronisation parameters on computations. We also try to provide explanation to the oscillatory behaviour of the solutions obtained from this method.

1 INTRODUCTION

The development of computer codes for distributed simulation of dynamical systems is an activity with relatively long tradition. The early efforts towards this direction are dated to late ’70s. The key concepts such as model decomposition, independent integration of separate subsystems and coupling of subsystems can be found in the early simulators for large-scale integrated circuits. These simulators used mainly Timing Simulation technique with relaxation methods for subsystems’ coupling. For example `MOTIS` [1] simulator for `MOS` circuits was the first that used relaxation technique for simulation of decomposed circuits. In the early ’80s Lelarasmee [2, 3] presented the Waveform Relaxation Method and developed a prototype simulator `RELAX` for `VLSI` circuits. The Waveform Relaxation Method introduced the concept of simulator coupling at the differential equation level. The relaxation process was described as a „fixed point iteration in functional space”, i.e. the iterative process searched for a function (wave) rather than for a single point.

Generally, the system decomposition can be applied at different levels of simulator. For example, if an implicit integration method for differential equations is considered, three levels can be identified [2, 4]. The highest one is the *level of differential equations* – the system of differential equations can be decomposed onto subsets that are integrated separately. The numerical integration procedure usually transforms these differential equations into a sequence of algebraic equations. The next (lower) level is thus the *level of nonlinear algebraic equations* resulted from the discretisation of differential equations. An implicit integration routine may solve these algebraic equations using Newton-Raphson or similar method, which transforms nonlinear algebraic equations into sequence of linear problems. The lowest level in this case is the *level of linear equations*, where decomposition-based methods for linear equations can be used.

In the domain of multibody systems significant amount of work has been put into the development of efficient algorithms for dynamic analysis. In most cases the emphasis was placed on an efficient evaluation of right hand sides of differential equations. Examples of algorithms which utilise the system decomposition include the parallel implementation [5] of Bae and Haug recursive algorithm [6, 7], the Divide and Conquer [8, 9] by Featherstone, the Subsystem Synthesis method [10] and the parallel algorithm [11] by Anderson and Duan. All these algorithms concern on formulation and solution of dynamic equations, and are executed at least once per numerical integration step (e.g. when the right-sides of differential equations are evaluated). They can be considered as a lowest-level decomposition algorithms in the simulator. Although the parallel implementations of these algorithms might be very efficient on machines with shared memory, in the distributed case they introduce significant communication footprint as the information interchange between subsystems must be performed frequently.

Approaches were made to apply decomposition at higher equation levels. The aforementioned Waveform Relaxation Method was examined in [12, 13] on mechanical systems. It was shown, for example in [13], that the WR method tends to be divergent when subsystems are coupled by means of kinematic constraints and generally the convergence properties depend on the „stiffness” of coupling forces. This behaviour seems to be inherent to the most of „decoupled” integration algorithms. The divergence was also confirmed in [14, 15] in case of subsystems coupled by constraints. Attempts to replace the constraints with artificial forces were described e.g. by Schiehlen et al. [16], but the results were commented by the authors as unsatisfactory.

Multirate integration methods, that use different time step-sizes in different subsystems, were initiated by the work of Andrus [17]. He decomposed a system of differential equations into two subsets taking into account the response rate of each component. One subsystem with a „slow solution” and a second with a „fast solution” were integrated separately with fourth-order Runge-Kutta integrators using different time-steps. It was suggested that significant savings in computation time can be achieved for systems where the evaluation cost of slow components is greater than the cost of fast components. Wells [18] and Gear [19] developed multirate variant of linear multi-step integration methods and coded an experimental procedure called `MRATE`. The performance of these methods has been shown to be sensitive to the „strength of coupling” between subsystems. A general observation was made in [19] that the automatic step and order selection in multirate methods should lead to balance accuracy in slow and fast subsystems. This is in contrast with the traditional integrators, where fast components (which dictate time-step size for entire system) are integrated with required accuracy and the resultant integration errors in slow components are usually far smaller than the required tolerance (what is paid with unnecessary computational effort in slow subsystems).

Although the multirate methods include traditional integration schemes as the main part of computation algorithm, the stability, convergence, and accuracy of multirate methods are not the same as for base methods used in subsystems. Also these properties can depend on factors that are irrelevant to the traditional integrators. One of such factors, which is absent in regular integration methods, is the error of interpolation and extrapolation of unknown variables used in multirate methods.

An application of multirate integration with „weak coupling” to the simulation of multibody systems was illustrated e.g. in [20, 21]. Here a Jacobi-like time-stepping algorithm was used, in which all subsystems are integrated independently inside of integration macro-step. The states of subsystems are exchanged at synchronisation points. Between the synchronisation points, unknown states of neighbour subsystems are approximated by extrapolation of values obtained at previous synchronisation points (constant, linear and quadratic polynomials were tested). This kind of method is more suitable for parallel implementation than „fastest first” or „slowest first” schemes, which are the synchronisation algorithms of Seidel type. It was pointed out that for a decomposed multibody system, the stability of such methods depends on the „stiffness” of coupling forces between the subsystems and the stability condition gets more restrictive as the couplings become more „stiff”. If the subsystems are coupled by kinematic constraints, then constraints reactions play the role of coupling forces and they are considered as „limit case of infinite stiffness”.

An interesting approach to the co-simulation was that of Gu, Gordon and Asada [22, 23, 24]. The authors presented a simulation coupling algorithms based on the Sliding Mode Control theory [25] (and Discrete-Time Sliding Mode Control). Here the models of coupled subsystems were treated as dynamical objects under control, and the coupling algorithm, which acted as a closed loop controller, was supposed to drive the state trajectories of subsystems towards the constraints manifold in finite time (or rather toward the sliding manifold which approximated it). The effects typical to sliding-mode control problems, such as chattering, were observed in some tests. The convergence of these methods has been shown to be dependent on the properties of subsystems, especially on the rate of change of response.

Modular simulation of multibody systems and modelling based on block representation were presented by Kübler and Schiehlen [26, 27]. Mechanical subsystems, or rather procedures responsible for evaluation of these subsystems, were represented as black-boxes with inputs and outputs defined. Iterative algorithm was proposed as a coupling method for systems with algebraic loops, and it was demonstrated, that non-iterative time stepping method can diverge in the presence of loops.

A concept of platform for simulation of general distributed mechanical systems was described by Wang et. al. [28, 29, 30]. An emphasis has been placed on the independence of simulation methods at different subsystems and the reuse of existing simulation codes (the „gluing algorithm" should be non-intrusive to subsystem simulators). In addition to the system architecture and software techniques to use, the authors proposed the Newton's iterative algorithm for simulators' coupling. In the „gluing algorithm" subsystems are integrated iteratively over single time step until coupling conditions are satisfied. Predefined input quantities, such as reaction forces in joints, are searched by the iterative procedure. It was pointed out that for highly nonlinear systems the standard Newton-Raphson method might be problematic, but no further information on the type of difficulties was provided. In some numerical examples [28] presented by the authors an oscillatory response has been observed where it was not expected. This phenomenon was left as a subject of further investigation.

In this paper we try to investigate more in depth the properties of iterative coupling method similar to that used by „gluing algorithm". The discussion is based primarily on numerical experiments. For the sake of clarity we choose a very simple model of double pendulum, where each body is treated as separate „subsystem" and is integrated independently from the other. It is shown, that in certain conditions the iterative coupling algorithm can give oscillatory results. This behaviour is discussed and the nature of the oscillations is explained. Also we observe, that the convergence of Newton's iteration can be poor in highly-oscillatory cases. Similar weakness is inherent to the classical implicit integrators based on Newton iteration.

The organisation of this paper is as follows. In section 2 the coupling algorithm is explained. In section 3 a test system is described. Section 4 shows numerical experiments. In section 5 experiments' results are discussed. We close the paper with conclusions provided in section 6.

2 COUPLING METHOD

In this paper we assume, that a complete multibody system is composed of subsystems described by Ordinary Differential Equations. The subsystems are interconnected with others by kinematic constraints. Complete set of equations describing the interconnected subsystems is written as

$$\dot{\mathbf{x}}_i = \mathbf{f}_i(t, \mathbf{x}_i, \mathbf{u}), \quad i = 1, \ldots, n \tag{1}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{x}) \tag{2}$$

where $\mathbf{x}(t) = \begin{bmatrix} \mathbf{x}_1^T & \ldots & \mathbf{x}_n^T \end{bmatrix}^T$ are the state variables, $\mathbf{u} = \mathbf{u}(t)$ contains input variables to the subsystems, $\mathbf{g}(\mathbf{x})$ represents the coupling error (e.g. violation of kinematic constraints), and i stands for the subsystem number. We treat subsystems separately and synchronise them at discrete points. The synchronisation procedure, we adapted from [28], might be described as an iterative solution of the following system of equations

$$\mathbf{0} = \mathbf{g}^{k+1}(\mathbf{U}^k) \tag{3}$$

with respect to some set of parameters $\mathbf{U}^k$. The symbol $\mathbf{g}^{k+1}$ represents violation of coupling equations at the $k+1$-st discrete time point. The values of $\mathbf{U}^k$ determine the shape of input functions $\mathbf{u}(t)$ for $t \in [t^k, t^{k+1})$. The algorithm searches for the values of $\mathbf{U}^k$, such that the $\mathbf{g}(\mathbf{x}(t))$ at the end of time interval $[t^k, t^{k+1})$ is driven to zero. The evaluation of $\mathbf{g}^{k+1}(\mathbf{U}^k)$ involves the time-integration of subsystems (1) within this interval.

We use piecewise constant functions as $\mathbf{u}(t)$. The values of $\mathbf{u}(t), t \in [t^k, t^{k+1})$ are determined by the parameters $\mathbf{U}^k$. The parameters $\mathbf{U}^k$ are chosen to have dimension equal to the size of g^{k+1} in (3). If the size of $\mathbf{U}^k$ is same as the number of inputs $\mathbf{u}$ we define the input variables as $\mathbf{u}(t) = \mathbf{U}^k, t \in [t^k, t^{k+1})$. If there is more parameters $\mathbf{U}^k$ than inputs $\mathbf{u}(t)$, we divide the interval $[t^k, t^{k+1})$ into sub-intervals and split the parameters $\mathbf{U}^k$ into subsets assigning each subset to one of the sub-intervals. This is demonstrated in the figure 1 for system with single input u. The figure on the left illustrates a case with one input and one parameter, the figure on the right shows a case with one input and two parameters. For example, in some experiments we add velocity constraints to the coupling equations (2) and then the size of g^{k+1} and $\mathbf{U}^k$ grows twice, but the number of inputs $\mathbf{u}$ remains unchanged.

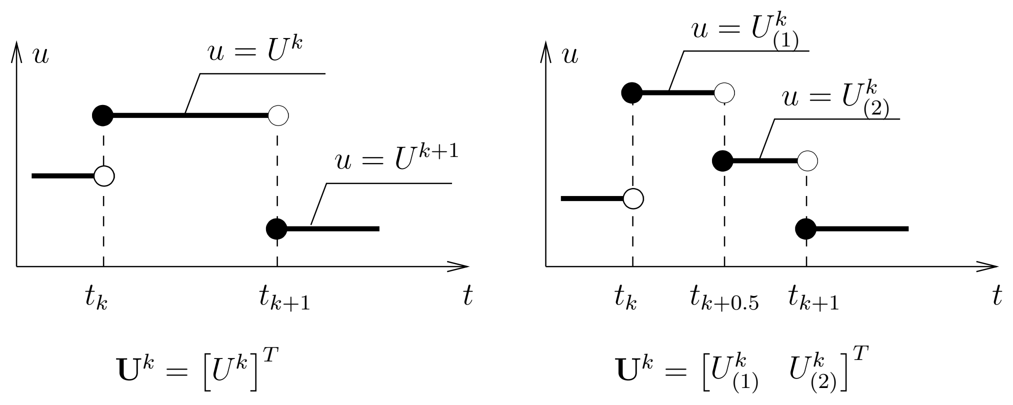

Figure 1: Interpolating input variables. Left figure: one parameter U^k defines one input u over $[t^k, t^{k+1})$. Right figure: two parameters: $\mathbf{U}^k$ define one input variable u on sub-intervals.

In context of mechanical systems coupled by constraints, the input values $\mathbf{u}(t)$ usually approximate unknown Lagrange's multipliers. From the perspective of subsystems' models they act as external forces applied to the system (they must be defined as so, because the subsystems know nothing about coupling constraints). From the perspective of „gluing agent" they approximate reactions of coupling constraints.

The above algorithm is one of the simplest variant of „gluing algorithms" outlined in [28]; other variants are also possible. For example, we use „single-step" synchronisation scheme whereas „multi-step" variants can also be constructed. Piecewise constant interpolation is used for $\mathbf{u}$, but linear or polynomial interpolation might be tried as well. We use finite-difference method for the approximation of a Jacobian matrix in Newton iteration because it is difficult to provide it in analytical form. The task of deriving analytical expressions for entries of the Jacobian would involve not only the differentiation of equations of motions, but should also include the discretisation rules used by the subsystems' integrators and would lead to recursive formulae. An approach based on analytical Jacobian would also cause the coupling method to be subsystem-intrusive, as the subsystem evaluation and integration routines would have to be re-coded to provide additional terms necessary for Jacobian evaluation.

It can be observed that in this computational process the coupling equations are guaranteed to be satisfied only at the discrete points t_k. *Between the points, the motion of subsystems isn't forced to be consistent, thus the constraints* $\mathbf{g}(\mathbf{x}(t))$ *might oscillate crossing or touching zero only at discrete points* t_k. This effect can be overlooked if only the values g^k at discrete points t_k are watched. In practise, these oscillations can significantly influence computations. They propagate to the input variables $\mathbf{u}$ and pose an obstacle in proper prediction of initial guess $\mathbf{U}^{k[0]}$ to the iterative solution of equation (3). We'll show that in numerical experiments in section 5.

The presented method has some features in common with the methods based on sliding mode control. There are discontinuities in the inputs $\mathbf{u}(t)$ in both methods (sliding mode control incorporates discontinuities intentionally). The discontinuity points are isolated in time, thus a chattering appears (as it is in a discrete-time sliding mode control). The differences between two methods are, that (a) the prescribed control law has been replaced with iterative algorithm for the determination of $\mathbf{u}(t)$ and (b) there is no reaching phase in the iterative algorithm - the "sliding-like mode" is achieved immediately as the simulation starts.

3 THE TEST SYSTEM

We chosen planar double pendulum as a test system for the coupling method. The double pendulum is composed of single pendulum $\mathbf{1}$ of length L_1 and a single body $\mathbf{2}$ of length L_2 (figure 2). Equations of motion for assembled double pendulum (all-at once) can be derived

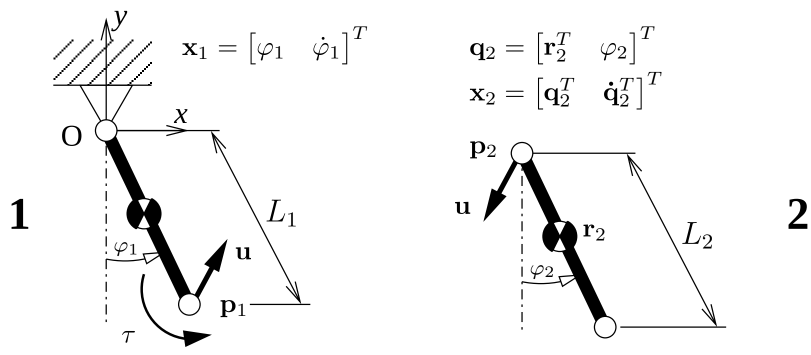
y
$\mathbf{x}_1 = [\varphi_1 \quad \dot\varphi_1]^T$
$\mathbf{q}_2 = [\mathbf{r}_2^T \quad \varphi_2]^T$
$\mathbf{x}_2 = [\mathbf{q}_2^T \quad \dot{\mathbf{q}}_2^T]^T$
x
O
$\mathbf{p}_2$
1
L_1
u
$\mathbf{r}_2$
L_2
2
φ_1
u
φ_2
$\mathbf{p}_1$
τ

Figure 2: The decomposed model of double pendulum.

using independent coordinates φ_1 and φ_2 as:

$$\overbrace{\begin{bmatrix} J_1 + m_2 L_2^2 & \frac{1}{2} m_2 L_1 L_2 \cos(\varphi_2 - \varphi_1) \\ \frac{1}{2} m_2 L_1 L_2 \cos(\varphi_2 - \varphi_1) & J_2 + \frac{1}{4} m_2 L_2^2 \end{bmatrix}}^{\overline{\mathbf{M}}} \overbrace{\begin{bmatrix} \ddot\varphi_1 \\ \ddot\varphi_2 \end{bmatrix}}^{\ddot{\boldsymbol{\varphi}}} =$$

$$= \underbrace{\begin{bmatrix} \tau(t) + \frac{1}{2} m_2 L_1 L_2 \sin(\varphi_2 - \varphi_1)\dot\varphi_2^2 - g\,(\frac{1}{2} m_1 L_1 + m_2 L_2)\sin\varphi_1 \\ -\frac{1}{2} m_2 L_1 L_2 \sin(\varphi_2 - \varphi_1)\dot\varphi_1^2 - \frac{1}{2}\, g\, m_2 L_2 \sin\varphi_2 \end{bmatrix}}_{\overline{\mathbf{Q}}} \quad (4)$$

where J_1 is the moment of inertia of pendulum $\mathbf{1}$ with respect to point O, m_2 is the mass of body $\mathbf{2}$, J_2 is the moment of inertia of body $\mathbf{2}$ w.r.t its mass centre, L_1 and L_2 are the lengths of the pendulum $\mathbf{1}$ and body $\mathbf{2}$ respectively, $\tau(t)$ is an external applied torque, and g is the gravity. Parameter values used in all tests are given in Table 1.

Equation (4) will be written for brief notation as

$$\overline{\mathbf{M}}\ddot{\boldsymbol{\varphi}} = \overline{\mathbf{Q}} \quad (5)$$

Numerical solution to (5) will be used later as reference trajectory to examine the results from coupling method.

For decomposed subsystems we write equations of motion as follows. The motion of subsystem $\mathbf{1}$ is described by single coordinate φ_1 and the equations of motion can be written as

Subsystem	Parameter	Value
1	L_1	$1\ m$
	m_1	$3\ kg$
	J_1	$1\ kg\,m^2$
2	L_2	$1\ m$
	m_2	$3\ kg$
	J_2	$0.25\ kg\,m^2$

Table 1: Values of parameters for double pendulum

$$J_1\ddot{\varphi}_1 = \underbrace{\tau(t) - \frac{1}{2}L_1 m_1 g \sin\varphi_1 + L_1\left(u_{(1)}\cos\varphi_1 + u_{(2)}\sin\varphi_1\right)}_{Q_1(t,\varphi_1;\mathbf{u})} \tag{6}$$

Corresponding ODEs in standard form, used by ODE integration procedure, are written as

$$\underbrace{\begin{bmatrix}\dot{x}_{(1)}\\ \dot{x}_{(2)}\end{bmatrix}_1}_{\dot{\mathbf{x}}_1} = \underbrace{\begin{bmatrix}x_{(2)}\\ J^{-1}Q(t,x_{(1)};\mathbf{u})\end{bmatrix}_1}_{\mathbf{f}_1(t,\mathbf{x}_1;\mathbf{u})} \tag{7}$$

or for brevity

$$\dot{\mathbf{x}}_1 = \mathbf{f}_1(t,\mathbf{x}_1;\mathbf{u}) \tag{8}$$

The motion of subsystem **2** is described using absolute coordinates $r_{2(x)}, r_{2(y)}, \varphi_2$:

$$\underbrace{\begin{bmatrix}m_2 & 0 & 0\\ 0 & m_2 & 0\\ 0 & 0 & J_2\end{bmatrix}}_{\mathbf{M}_2}\underbrace{\begin{bmatrix}\ddot{r}_{2(x)}\\ \ddot{r}_{2(y)}\\ \ddot{\varphi}_2\end{bmatrix}}_{\ddot{\mathbf{q}}_2} = \underbrace{\begin{bmatrix}-u_{(1)}\\ -u_{(2)} - m_2 g\\ \frac{1}{2}L_2\left(u_{(1)}\cos\varphi_2 + u_{(2)}\sin\varphi_2\right)\end{bmatrix}}_{\mathbf{Q}_2(t,\varphi_2;\mathbf{u})} \tag{9}$$

Similarly as with (6) we transform equations (9) to the standard form

$$\underbrace{\begin{bmatrix}\dot{\mathbf{x}}_{(1:3)}\\ \dot{\mathbf{x}}_{(4:6)}\end{bmatrix}_2}_{\dot{\mathbf{x}}_2} = \underbrace{\begin{bmatrix}\mathbf{x}_{(4:6)}\\ \mathbf{M}^{-1}\mathbf{Q}(t,x_{(3)};\mathbf{u})\end{bmatrix}_2}_{\mathbf{f}_2(t,\mathbf{x}_2;\mathbf{u})} \tag{10}$$

and write them in brief form as

$$\dot{\mathbf{x}}_2 = \mathbf{f}_2(t,\mathbf{x}_2;\mathbf{u}) \tag{11}$$

The subsystems **1** and **2** are coupled by kinematic constraints

$$\mathbf{0} = \underbrace{\begin{bmatrix}L_1\sin\varphi_1\\ -L_1\cos\varphi_1\end{bmatrix}}_{\mathbf{p}_1} - \underbrace{\begin{bmatrix}r_{2(x)} - 0.5L_2\sin\varphi_2\\ r_{2(y)} + 0.5L_2\cos\varphi_2\end{bmatrix}}_{\mathbf{p}_2} \tag{12}$$

Equations (12) simply reflect the fact, that the points $\mathbf{p}_1$ and $\mathbf{p}_2$ should coincide. We write the constraints in short form as

$$\mathbf{0} = \mathbf{\Phi}(\mathbf{q}) \tag{13}$$

where $\mathbf{q} = \left[\varphi_1, \mathbf{q}_2^T\right]^T$.

The coupling equations (2) can be constructed in several ways. We have examined two formulations based on constraints equations (13). In the first approach we used kinematic constraints at „displacement level" only, that is we defined $\mathbf{g}$ in (2) as:

$$\mathbf{g}(\mathbf{x}) \equiv \mathbf{\Phi}(\mathbf{q}) \tag{14}$$

The second approach is based on position and velocity constraints. We use as the coupling terms the constraints $\mathbf{\Phi}$ and their time derivatives $\dot{\mathbf{\Phi}}$ as follows

$$\mathbf{g}(\mathbf{x}) \equiv \begin{bmatrix} \mathbf{\Phi}(\mathbf{q}) \\ \dot{\mathbf{\Phi}}(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix} \tag{15}$$

Note that in the second case, i.e. in equations (15), there are four scalar coupling conditions whereas in (14) there were just two. Consequently in the coupling algorithm (3), the set of unknown parameters $\mathbf{U}^k$ must be extended to four components even if there are only two input variables $\mathbf{u}(t)$ in (1). Alternatively we could reformulate the equations of motion, using for example `GGL`-formulation [31] with additional multipliers in order to extend the $\mathbf{u}(t)$ to four components, but we assumed that the subsystem models (8) and (11) are provided as they are, and inputs $\mathbf{u}$ can not be redefined even if we want to refine the coupling conditions (3).

4 NUMERICAL EXPERIMENTS

We have performed series of experiments in order to examine the behaviour of the method in certain circumstances. We ran computations for two particular cases:

1. Double pendulum under the gravity only (non-stiff case):

$$\tau(t) = 0 \tag{16}$$

2. Double pendulum with oscillatory, rapidly decaying torque applied (stiff case):

$$\tau(t) = 200\,\exp(-t/0.02) \cdot \sin(200 \cdot 2\pi t) \tag{17}$$

For each case we performed one simulation with displacement-level coupling defined by equation (14) and one with displacement&velocity-level coupling as defined by (15). For all experiments we synchronise the subsystems with the tolerance $|g_{(j)}^{k+1}(\mathbf{U}^k)| \leq 1 \cdot 10^{-10}$.

5 RESULTS AND DISCUSSION

5.1 Non-stiff case

5.1.1 Displacement-level synchronisation

In the figure 3 are shown kinematic quantities φ_i, and $\dot{\varphi}_i$ obtained from simulation of non-stiff case $\tau(t) = 0$ with synchronisation time-step $H = 0.01s$ and displacement-level-only coupling (14). Both subsystems are integrated with Runge-Kutta method using Matlab procedure `ode45` [32]. The simulation results from „simulator coupling" (solid line) are compared to „all-at-once" simulation results $\overline{\varphi}_1$, $\dot{\overline{\varphi}}_1$, $\overline{\varphi}_2$, $\dot{\overline{\varphi}}_2$ obtained by numerical integration of equations (4) (dashed line).

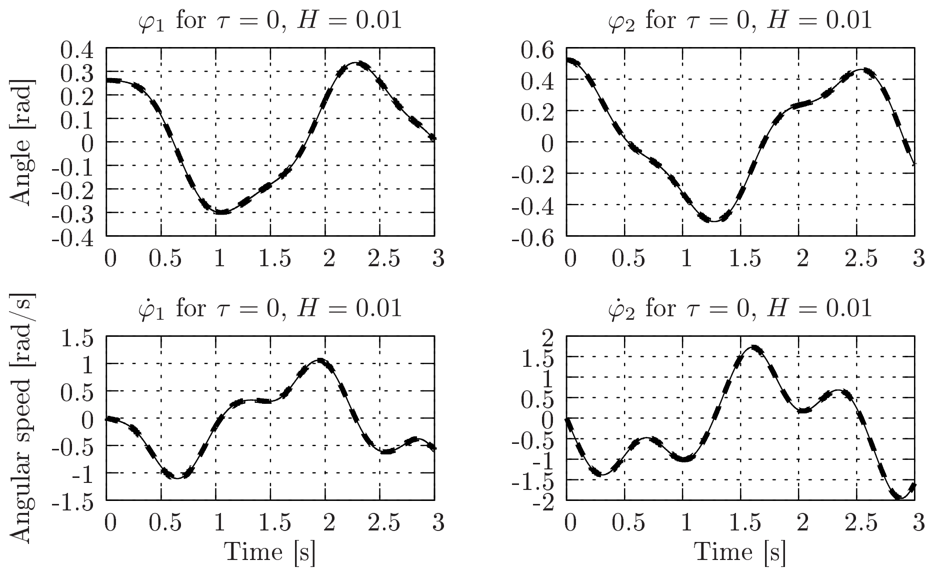

Figure 3: Simulation results for the non-stiff case $\tau(t) = 0$ with $H = 0.01s$ and displacement-level synchronisation. Solid line: simulator coupling, dashed: all-at-once integration.

The results obtained from simulator coupling and „all-at-once" integration are very similar – the differences are shown in the figure 4. The relative differences calculated as

$$\delta\varphi_i = \frac{\max\limits_{0\le t\le 3} |\varphi_i(t) - \overline{\varphi}_i(t)|}{\max\limits_{0\le t\le 3} |\overline{\varphi}_i(t)|} \cdot 100\% \qquad \delta\dot{\varphi}_i = \frac{\max\limits_{0\le t\le 3} |\dot{\varphi}_i(t) - \dot{\overline{\varphi}}_i(t)|}{\max\limits_{0\le t\le 3} |\dot{\overline{\varphi}}_i(t)|} \cdot 100\% \tag{18}$$

are $\delta\varphi_1 = 0.0000952\%$, $\delta\varphi_2 = 0.00001353\%$, and $\delta\dot{\varphi}_1 = 0.03014\%$, $\delta\dot{\varphi}_2 = 0.03408\%$.

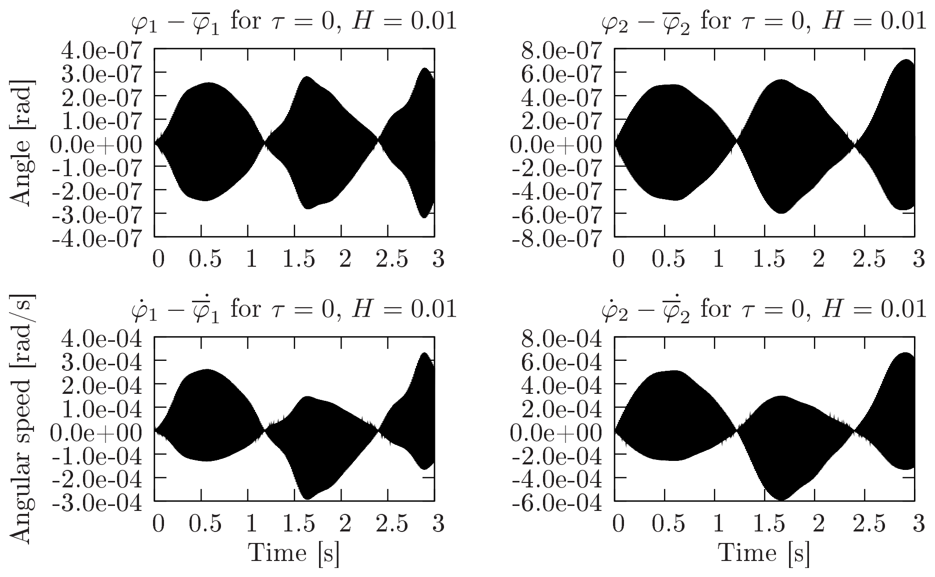

Figure 4: The difference between simulation coupling and all-at-once integration for non-stiff case with $H = 0.01$ and displacement-level synchronisation.

The obtained values of $\mathbf{u}(t)$ for the simulation of non-stiff case are shown in the figure 5 (solid lines). They seem to be very similar to the reaction forces computed using the solutions from „all-at-once" integration (dashed line). However the reaction forces are continuous functions whereas the computed inputs $\mathbf{u}(t)$ are discontinuous, piecewise constant functions (figure 7).

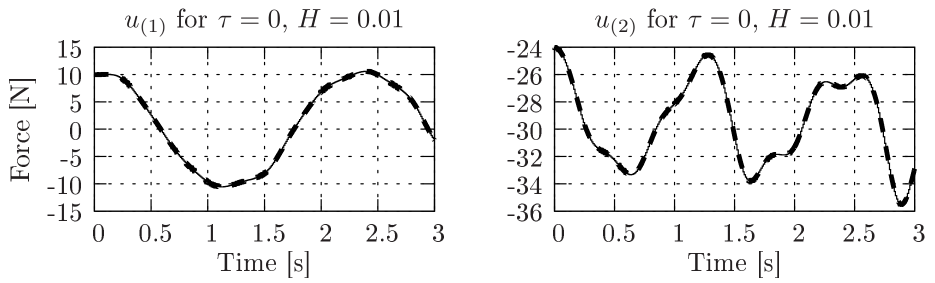

Figure 5: Input variables (reaction forces) for the simulation of non-stiff case with $H = 0.01$ and displacement-level synchronisation. Solid line: simulator coupling, dashed line: all-at-once integration.

Constraints violation errors are shown in the figure 6. The displacement error in revolute joint has order of magnitude $1 \cdot 10^{-6}$ and the errors in velocity constraints are of order $1 \cdot 10^{-3}$.

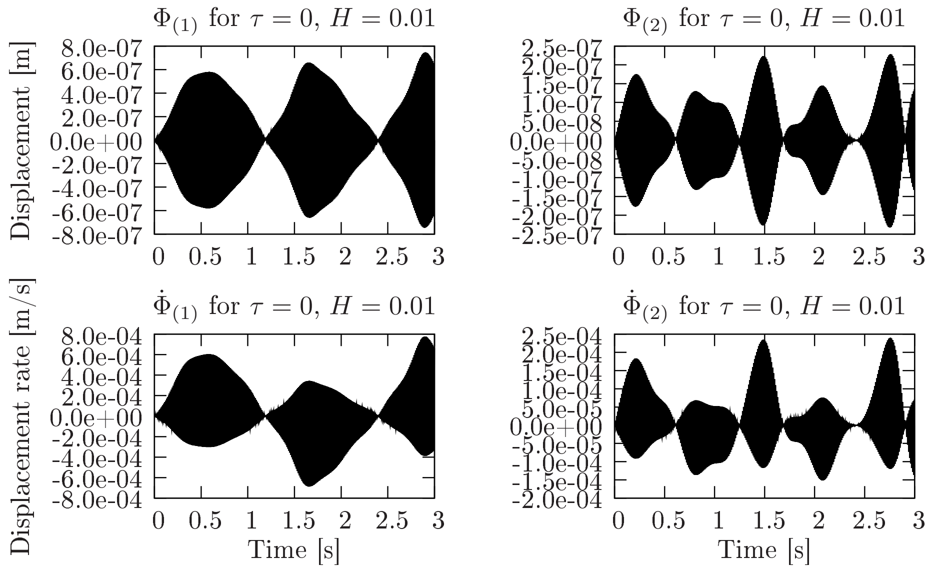

Figure 6: Constraints error in the coupling joint for the simulation of non-stiff case with $H = 0.01$ and displacement-level synchronisation.

It can be observed that the resultant displacement errors are much larger than the prescribed tolerance $|g_{(j)}^{k+1}(\mathbf{U}^k)| \leq 1 \cdot 10^{-10}$ we used for synchronisation. It is simply explained by the fact, that we place the synchronisation conditions only at discrete points t_k but plot the values of $\mathbf{g}(\mathbf{x}(t))$ also between the synchronisation points. The kinematic quantities $\mathbf{x}_1$ and $\mathbf{x}_2$ can be inconsistent between these points and the errors of coupling constraints can get much larger here. The situation is illustrated in the figure 7, where the horizontal displacement error $\Phi_{(1)}$, and the first derivative $\dot{\Phi}_{(1)}$ are visualised over a very short period of time $t \in [0.0, 0.1]$. This plot also shows what kind of responses we should expect from this method of simulator coupling.

It is noticeable, that piecewise constant function $\mathbf{u}(t)$ can give very rough approximation of the „nominal" reaction force (dashed line) for larger synchronisation steps H or for faster systems. The displacement error $\Phi_{(1)}$ is near zero at each synchronisation point and it oscillates around zero as it was expected. First derivative $\dot{\Phi}_{(1)}$ switches from rising to falling and vice-versa each time the plot of $u_{(1)}$ crosses the „nominal" curve. An interesting observation is, that for this particular simulation the $\Phi_{(1)}$ crosses zero not only at synchronisation points t_k (circles on the plot), but also inside of each interval (t_k, t_{k+1}).

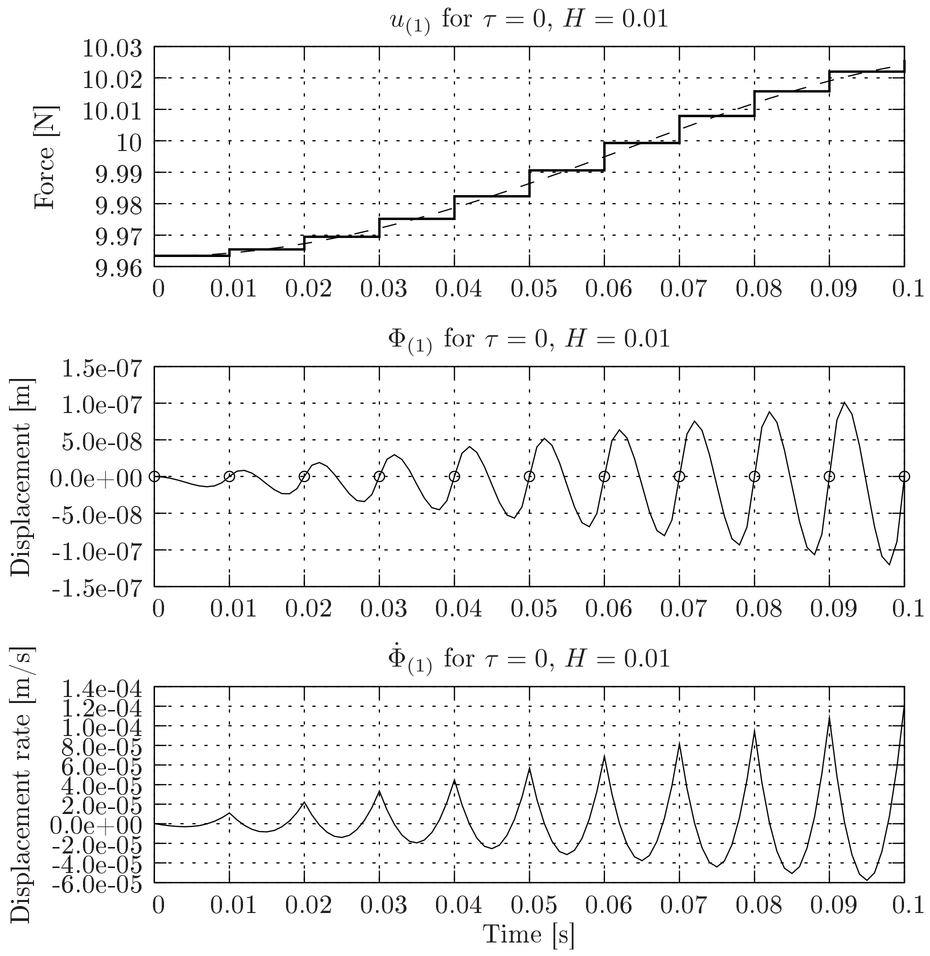

Figure 7: „Zoomed" input variable $u_{(1)}$ (reaction force), constraint error $\Phi_{(1)}$ and its derivative $\dot{\Phi}_{(1)}$ for the simulation of non-stiff case with $H = 0.01$ and displacement-level synchronisation. Solid lines: simulator coupling, dashed line: all-at-once simulation, circles: synchronisation points.

5.1.2 Displacement&velocity-level synchronisation

What can be observed from previous results is, that usually the velocity constraints behave worse than position constraints. It has been shown in the figure 4 that for simple simulation the velocity errors were three orders of magnitude larger than displacement errors. Additionally, the displacement errors can change quite rapidly in the vicinity of the synchronisation point t_k if their derivatives (errors of velocity constraints) are allowed to have non-zero values here. In effect, the displacement errors can grow to relatively high values inside of synchronisation interval. In order to reduce this effect, we have tried to include the velocity constraints into coupling equations. Our intent is to make the „wave" of displacement errors $\Phi_{(j)}$ „tangent" to t axis at synchronisation points t_k with the hope, that this will reduce the growth of constraints inside of the interval (t_k, t_{k+1}).

In this experiment we defined coupling equations according to (15). The resultant constraints errors for this experiment are shown in the figure 8. It can be seen, that the constraints errors have been reduced by the factor of about two as compared to displacement-level-only synchronisation.

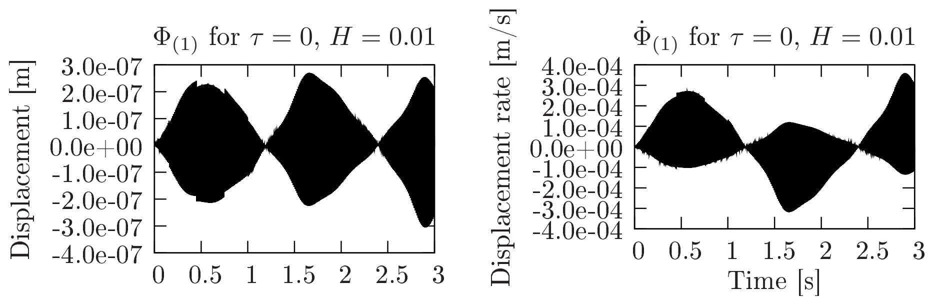

Figure 8: Constraints error in the coupling joint for the simulation of non-stiff case with $H = 0.01$ and displacement&velocity-level synchronisation.

The input variable $u_{(1)}$ and constraint errors $\Phi_{(1)}$, $\dot{\Phi}_{(1)}$ are visualised in the figure 9 in close-up. It is emphasised, that the coupling conditions are applied to displacement and velocity constraints (circles on the graphs). Also it can be seen, that the step length in $u_{(1)}$ is a half of this from the figure 7. This is because we have to „switch" the function $\mathbf{u}$ two times faster than previously, according to the number of coupling equations and parameters $\mathbf{U}^k$ (it was previously explained –see the figure 1).

5.2 Stiff case

5.2.1 Displacement-level synchronisation

The solutions φ_i and $\dot{\varphi}_i$ for the simulations of stiff case with $H = 0.01$ and displacement-level synchronisation are shown in the figure 10. It becomes apparent, that the oscillations are likely to appear also in the angular velocities $\dot{\varphi}_i$. Still the oscillations are not visible on the plots of angular positions. This is because they were „filtered-out" by the integration procedure.

It can be seen from figure 11, that the resultant input variables $\mathbf{u}(t)$ oscillate around „nominal" trajectories with considerable amplitude. Here the amplitudes are about $15\,N$ for $u_{(1)}$ and $5\,N$ for $u_{(2)}$, thus the oscillations are of same order of magnitude as the values of inputs $\mathbf{u}(t)$.

Constraints errors for stiff-case simulation with displacement-level synchronisation are shown in the figure 12. Again we see significant oscillations here, with amplitude reaching the values $0.4\,mm$ for displacements and $0.2\,m/s$ for velocity errors.

The input variable $u_{(1)}$, constraint $\Phi_{(1)}$ and constraint rate $\dot{\Phi}_{(1)}$ is presented in close view in the figure 13. It is illustrated how the computed variables $u_{(j)}$ behave, if the original system is faster than the synchronisation process.

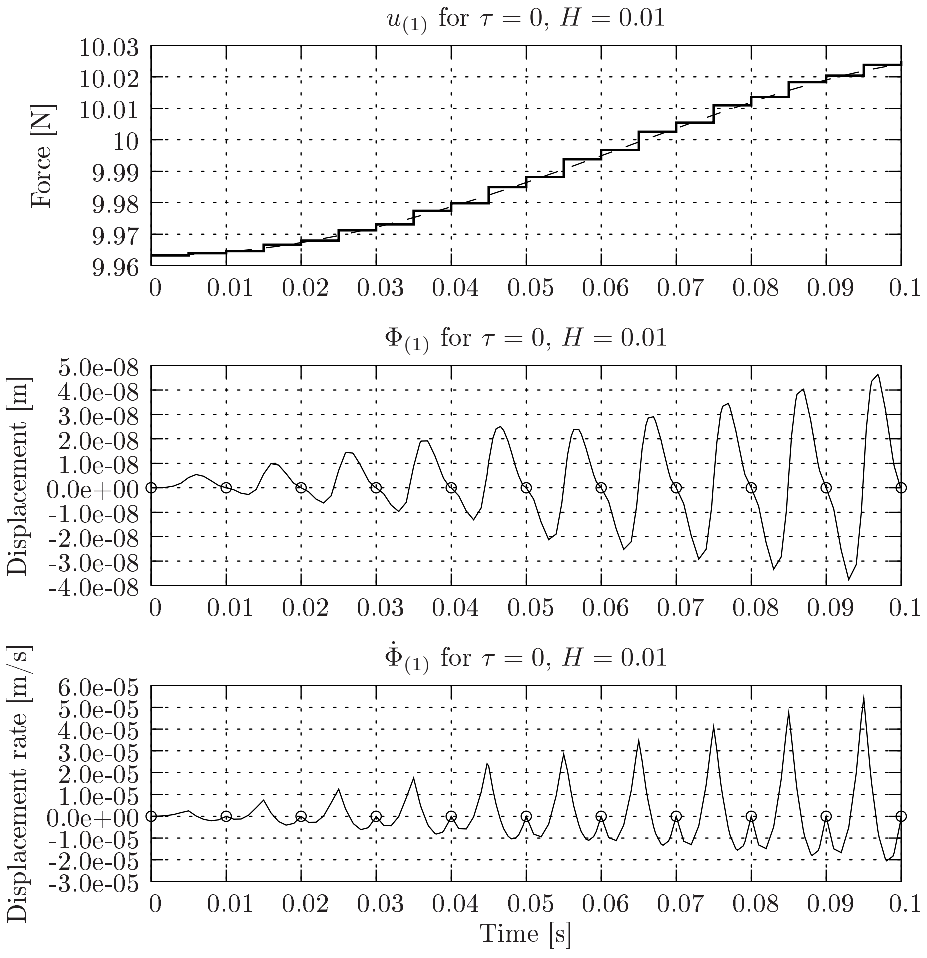

Figure 9: „Zoomed" input variable $u_{(1)}$ (reaction force), constraint error $\Phi_{(1)}$ and its derivative $\dot{\Phi}_{(1)}$ for the simulation of non-stiff case with $H = 0.01$ and displacement- and velocity-level synchronisation. Solid lines: simulator coupling, dashed line: all-at-once simulation, circles: synchronisation points.

5.2.2 Displacement&velocity-level synchronisation

With velocity constraints added to the coupling equations, the method gives much better solutions than with displacement-level-only synchronisation. It can be seen from the figure 14 that the constraints errors at the beginning of simulation are about two times smaller than in displacement-level synchronisation and they disappear as time progresses. This is opposite to the previous results, where the oscillations in $\mathbf{\Phi}(t)$, and $\dot{\mathbf{\Phi}}(t)$ remained constant, even when the causing factor $\tau(t)$ practically disappeared.

The same applies to the computed inputs $\mathbf{u}(t)$. The first component $u_{(1)}$ shown in the figure 15 oscillates with significant amplitude through relatively short period of time and the oscillations decay as the $\tau(t)$ disappears. This can be observed more precisely on the figure 16.

5.3 Comparison: displacement coupling vs displacement&velocity coupling

Table 2 summarises the constraints errors for previous experiments. Mean and maximum values of the position and velocity constraints (magnitudes) are presented. The mean and maximum values are calculated for the entire simulation interval $t \in [0, 3]$. It can be seen, that for

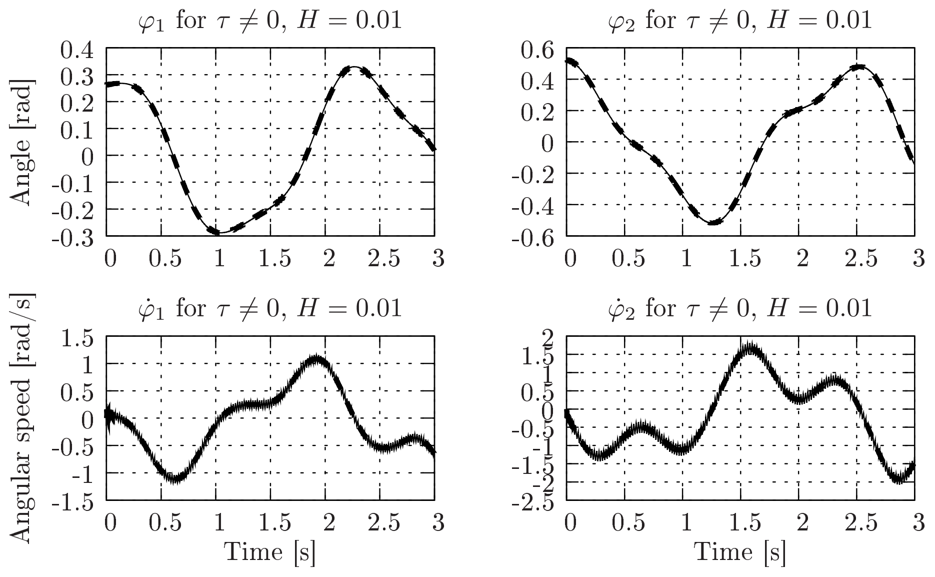

Figure 10: Simulation results for the stiff case $\tau(t) = 200\exp(-t/0.02)\sin(200 \cdot 2\pi t)$ with $H = 0.01s$ and displacement-level synchronisation. Solid line: simulator coupling, dashed: all-at-once integration.

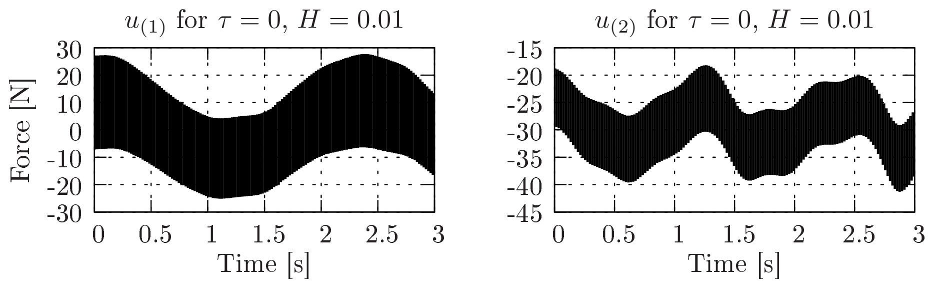

Figure 11: Input variables (reaction forces) for the simulation of stiff case with $H = 0.01$ and displacement-level synchronisation.

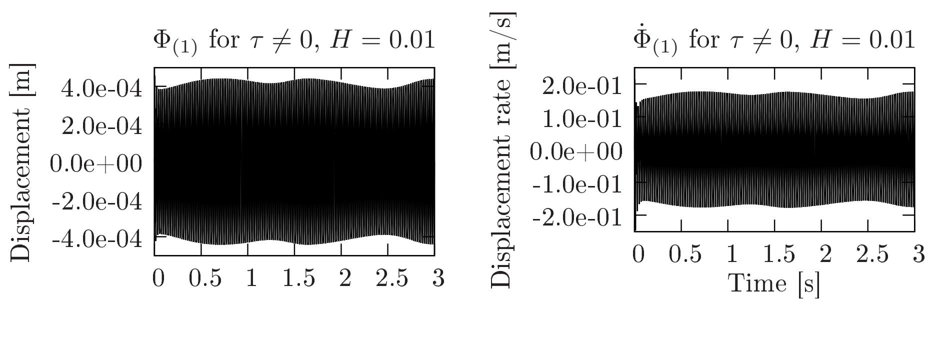

Figure 12: Constraints error in the coupling joint for the simulation of stiff case with $H = 0.01$ and displacement-level synchronisation.

each case, the accuracy of displacement&velocity-level coupling is better than displacement-level only. The „quotient" value in table 2 is calculated as a quotient of error value obtained with displacement-only synchronisation and the value obtained with displacement&velocity synchronisation. The value greater than one means that for particular case the displacement&velocity-

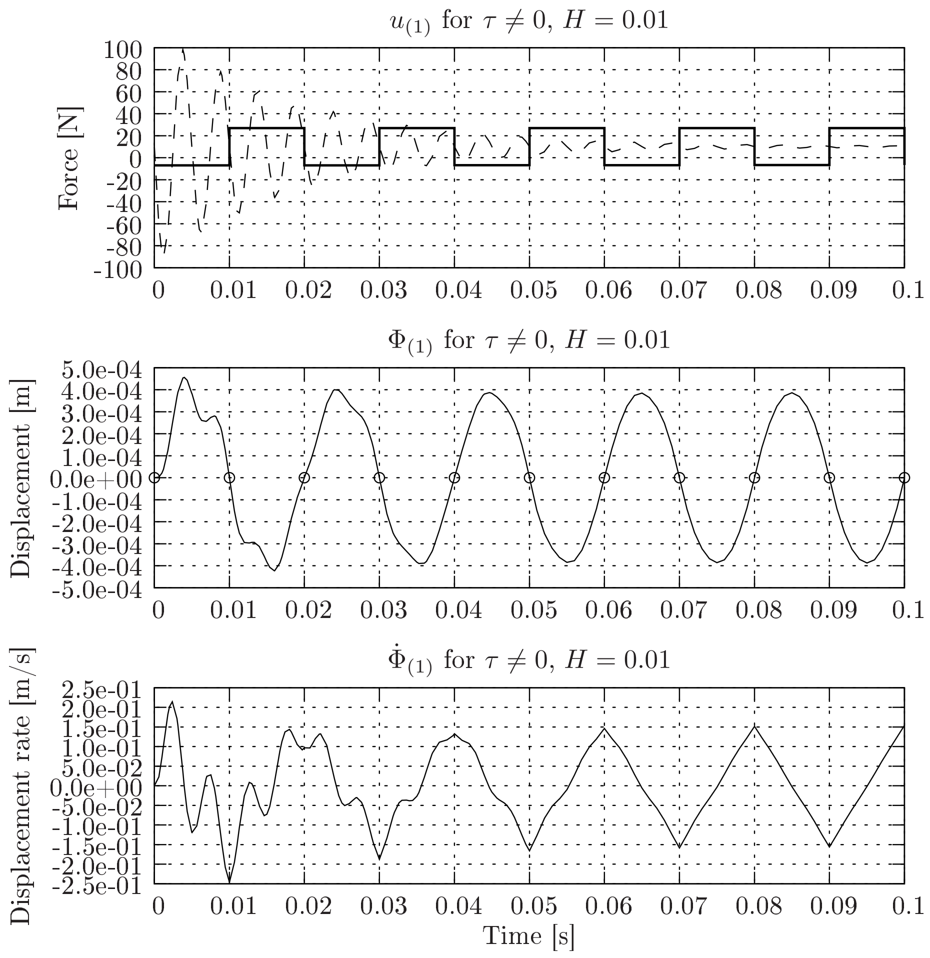

Figure 13: „Zoomed" input variable $u_{(1)}$ (reaction force), constraint error $\Phi_{(1)}$ and its derivative $\dot{\Phi}_{(1)}$ for the simulation of stiff case with $H = 0.01$ and displacement-level synchronisation. Solid lines: simulator coupling, dashed line: all-at-once simulation, circles: synchronisation points.

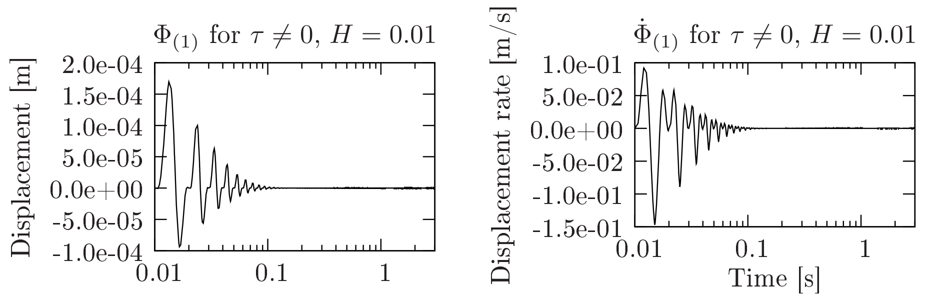

Figure 14: Constraints error in the coupling joint for the simulation of stiff case with $H = 0.01$ and displacement&velocity-level synchronisation.

level synchronisation resulted with accuracy better than displacement-only synchronisation.

Additionally, for the simulation of stiff case we have compared the results taking into account only the period $t \in [0.5, 30]$. This way we exclude the beginning jump in oscillations amplitude (figure 14), and concern only on the characteristics for the rest of simulation. The mean and max values of constraints errors in this interval are shown in table 3. It can be seen, that the inclusion

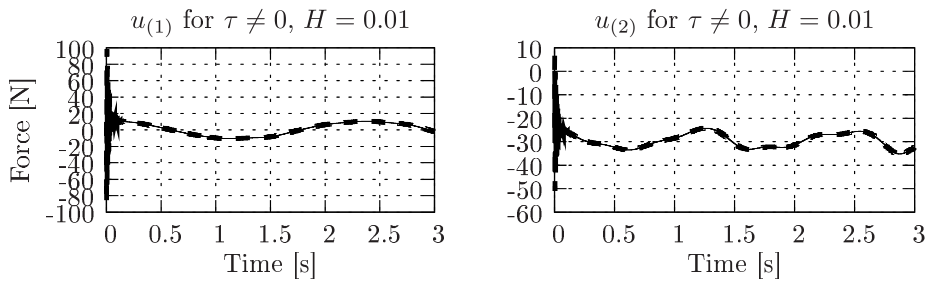

Figure 15: Input variables (reaction forces) for the simulation of stiff case with $H = 0.01$ and displacement&velocity-level synchronisation. Continuous line: simulator coupling, dashed line: all-at-once integration.

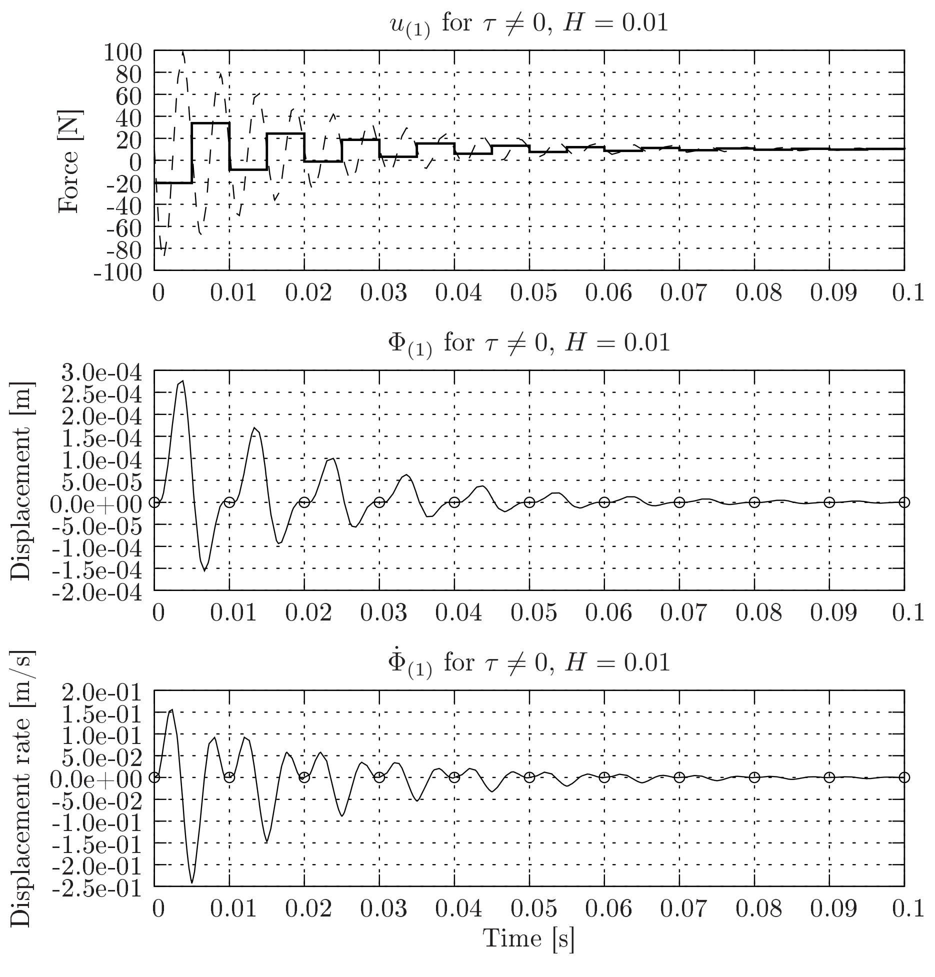

Figure 16: „Zoomed” input variable $u_{(1)}$ (reaction force), constraint error $\Phi_{(1)}$ and its derivative $\dot{\Phi}_{(1)}$ for the simulation of stiff case with $H = 0.01$ and displacement- and velocity-level synchronisation. Solid lines: simulator coupling, dashed line: all-at-once simulation, circles: synchronisation points.

of velocity constraints into coupling equations have reduced the oscillations of constraints by the factor of about three orders of magnitude.

Case	Coupling	mean$\|\mathbf{\Phi}\|$	$\max\|\mathbf{\Phi}\|$	mean$\|\dot{\mathbf{\Phi}}\|$	$\max\|\dot{\mathbf{\Phi}}\|$
$\tau(t) = 0$	Displ.	$2.46 \cdot 10^{-7}$	$7.47 \cdot 10^{-7}$	$1.59 \cdot 10^{-4}$	$7.77 \cdot 10^{-4}$
	Displ.&Vel.	$8.78 \cdot 10^{-8}$	$3.05 \cdot 10^{-7}$	$6.98 \cdot 10^{-5}$	$3.60 \cdot 10^{-4}$
	Quotient	2.80	2.45	2.28	2.16
$\tau(t) \neq 0$	Displ.	$2.84 \cdot 10^{-4}$	$4.73 \cdot 10^{-4}$	$8.61 \cdot 10^{-2}$	$2.55 \cdot 10^{-1}$
	Displ.&Vel.	$2.54 \cdot 10^{-6}$	$2.86 \cdot 10^{-4}$	$3.50 \cdot 10^{-2}$	$2.51 \cdot 10^{-1}$
	Quotient	112	1.65	2.41	1.01

Table 2: Constraints errors for two types of coupling. The mean and max are computed for the period $t \in [0, 3]$.

Case	Coupling	mean$\|\mathbf{\Phi}\|$	$\max\|\mathbf{\Phi}\|$	mean$\|\dot{\mathbf{\Phi}}\|$	$\max\|\dot{\mathbf{\Phi}}\|$
$\tau(t) \neq 0$	Displ.	$2.86 \cdot 10^{-4}$	$4.54 \cdot 10^{-7}$	$8.68 \cdot 10^{-2}$	$1.82 \cdot 10^{-1}$
	Displ.&Vel.	$9.11 \cdot 10^{-8}$	$2.99 \cdot 10^{-7}$	$7.07 \cdot 10^{-5}$	$3.53 \cdot 10^{-4}$
	Quotient	$3.14 \cdot 10^{3}$	$1.52 \cdot 10^{3}$	$1.23 \cdot 10^{3}$	$5.15 \cdot 10^{2}$

Table 3: Constraints errors for two types of coupling. The mean and max are computed for the period $t \in [0.5, 3]$.

5.4 Extrapolation of unknown variables

In the case, when the oscillations of $\mathbf{u}(t)$ become significant, a new problem can appear. The overall procedure gets inefficient, due to poor initial guesses provided to the iterative coupling algorithm. At each synchronisation step, the Newton-Raphson procedure must be provided with guess values of $\mathbf{u}(t_k)$ (and $\mathbf{u}(t_{k+0.5})$ for displacement&velocity synchronisation). These values are generated through an extrapolation of previous values $\mathbf{u}(t_l), l < k$. If the function $\mathbf{u}(t)$ is oscillatory, a „traditional” extrapolation based on consecutive points can give initial guesses that are too far from final solution. For these cases we tried in our experiments to extrapolate separately two envelopes of the oscillatory $u_{(j)}(t)$. The method is illustrated in the figure 17.

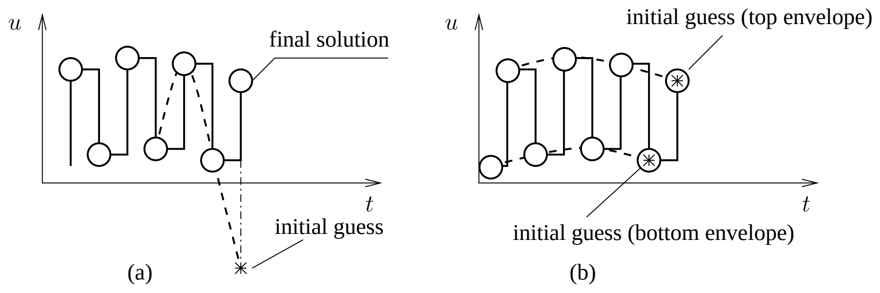

Figure 17: To methods of extrapolation: (a) – extrapolation based on consecutive points, (b) – separate extrapolation of bottom and top envelope

We have tried this extrapolation „trick” for the simulation of stiff case with displacement-level coupling. The effect of this special extrapolation technique is illustrated in the figure 18. It is clear, that in this particular simulation, the extrapolation method based on envelopes gives better initial guesses. In table 4 is shown the total number of iterations performed within the entire three-second simulation with synchronisation step $H = 0.01s$ (300 time-steps).

Although, this trick should not be considered as a general enhancement to every simulation that gives oscillatory responses. Note that this is not a rule that the oscillatory behaviour of $\mathbf{u}$ must always be the same. Here the input quantities $\mathbf{u}$ oscillate as fast as they can, but some of our other experiments have shown, that the oscillations can have frequencies lower than

Case	Coupling type	Consecutive points	Envelope extrap.
$\tau(t) \neq 0$	Displacement	827	302
	Displacement&Velocity	859	532

Table 4: Total number of iterations for two extrapolation methods.

the maximum possible. For such cases there is more than two discrete points inside of a single oscillation period and a more elaborate technique of „smart" extrapolation should be developed.

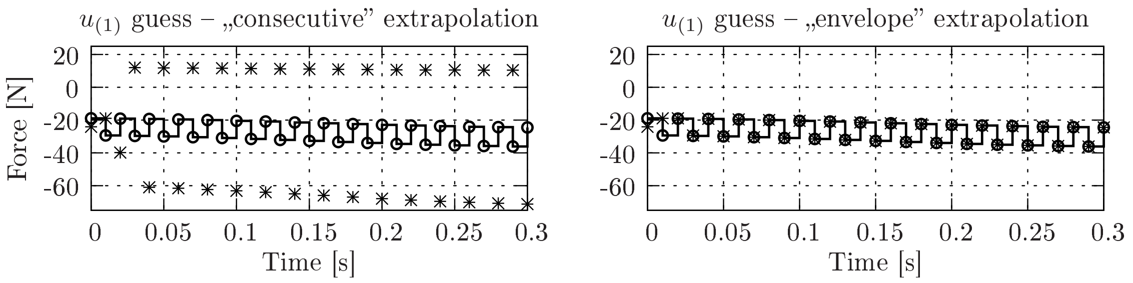

Figure 18: The effect of two methods of extrapolations of unknown input values $\mathbf{u}(t)$. Left plot: cubic extrapolation based on consecutive points, right plot: separate cubic extrapolations of top and bottom envelopes (even and odd points). Solid line: the values of input function $u_{(1)}$(t), circles: the final solutions from iterative algorithm, stars: initial guesses to iterative algorithm.

6 CONCLUSIONS

In this paper we presented a case study of simulator coupling in multibody system dynamics. Some properties of discrete-time coupling method were discussed. Our observations show that this method may give oscillatory responses. This phenomena can be treated as an analog of chattering seen in the sliding mode control algorithms. The numerical experiments suggest, that the chattering can be partially suppressed by the inclusion of the velocity-level constraints into coupling equations. Also the efficiency penalties caused by the chattering can be reduced by using separate extrapolations of the envelopes of oscillating $\mathbf{u}(t)$ for the initial guess instead of „direct" extrapolation based on consecutive points. For general case more elaborate extrapolation method should be developed.

7 ACKNOWLEDGEMENTS

The work is supported by European Regional Development Fund, Operational Programme – Innovative Economy – grant PO IG.01.02.01-00-014/08-00 as a part of project PROTEUS.

REFERENCES

[1] Basant R. Chawla, Hermann K. Gummel, and Paul Kozak. MOTIS – an MOS timing simulator. *IEEE Transactions on Circuits and systems*, cas-22(12):901–910, 1975.

[2] Ekachai Lelarasmee. *The Waveform Relaxation Method for Time Domain Analysis of Large Scale Integrated Circuits: Theory and Applications*. PhD thesis, 1982.

[3] Ekachai Lelarasmee, Albert E. Ruehli, and Alberto L. Sangiovanni-Vincentelli. The waveform relaxation method for time-domain analysis of large scale integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1(3):131–145, 1982.

[4] Gary D. Harchel and Alberto L. Sangiovanni-Vincentelli. A survey of third-generation simulation techniques. In *Proceedings of the IEEE*, volume 69, pages 1264–1280, Oct 1981.

[5] Dae Sung Bae, , Jon G. Kuhl, and Edward J. Haug. A recursive formulation for constrained mechanical system dynamics: Part III: Parallel processor implementation. *Mechanics of Structures and Machines*, 16(2):249–269, 1988.

[6] Dae Sung Bae and Edward J. Haug. A recursive formulation for constrained mechanical system dynamics: Part I: Open loop systems. *Mechanics of Structures and Machines*, 15(3):383–393, 1987.

[7] Dae Sung Bae and Edward J. Haug. A recursive formulation for constrained mechanical system dynamics: Part II: Closed loop systems. *Mechanics of Structures and Machines*, 15(4):481–506, 1987.

[8] Roy Featherstone. A divide-and-conquer articulated-body algorithm for parallel O(log(n)) calculation of rigid-body dynamics. part 1: Basic algorithm. *The International Journal of Robotics Research*, 18(9):867–875, 1999.

[9] Roy Featherstone. A divide-and-conquer articulated-body algorithm for parallel O(log(n)) calculation of rigid-body dynamics. part 1: Trees, loops, and accuracy. *The International Journal of Robotics Research*, 18(9):876–892, 1999.

[10] Kim Sung-Soo. A subsystem synthesis method for efficient vehicle multibody dynamics. *Multibody System Dynamics*, 7:189–207, 2002.

[11] Kurt S. Anderson and S. Duan. Hihgly parallelizable low order dynamics simulation algorithm for multi-rigid-body systems. *Journal of Guidance, Control and Dynamics*, 23(2):355–364, 2000.

[12] B. Leimkuhler. Estimating waveform relaxation convergence. *SIAM Journal of Scientific Computing*, 14:872–889, 1993.

[13] Ben Leimkuhler. Relaxation techniques in multibody dynamics. *Transactions of Canadian Society for Mechanical Engineering*, 17(4A):459–471, 1993.

[14] Fang-Chung Tseng. *Multibody Dynamics Simulation in Network-Distributed Environments*. PhD thesis, 2000.

[15] Fan-Chung Tseng, Zheng-Dong Ma, and Gregory Hulbert. Efficient numerical solution of constrained multibody dynamics systems. *Computational Methods in Applied Mechanical Engineering*, 192:439–472, 2003.

[16] W. Schiehlen, A. Rükgauer, and TH. Schirle. Force coupling versus differential algebraic description of constrained multibody systems. *Multibody System Dynamics*, 4:317–340, 2000.

[17] J. F. Andrus. Numerical solution of systems of ordinary differential equations separated into subsystems. *SIAM Journal on Numerical Analysis*, 16(4):605–611, 1979.

[18] Daniel Raymond Wells. *Multirate linear multistep methods for the solution of systems of ordinary differential equations*. PhD thesis, Champaign, IL, USA, 1982.

[19] C. W. Gear and D. R. Wells. Multirate linear multistep methods. *BIT*, 24(4):484–502, 1984.

[20] Gerhard Hippmann, Martin Arnold, and Marcus Schittenhelm. Efficient simulation of bush and roller chain drives. In *Multibody Dynamics 2005, ECCOMAS Thematic Conference*, Madrid, Spain, June 2005.

[21] Martin Arnold. Multi-rate time integration for large scale multibody system models. In *IUTAM Symposium on Multiscale Problems in Multibody System Contacts*, volume 1, pages 1–10, 2007.

[22] Bei Gu, W. Gordon, and H. Harry Asada. Co-simulation of coupled dynamic subsystems: A differential-algebraic approach using singularly perturbed sliding manifolds. In *Proceedings of the American Control Conference*, Chicago, Illinois, June 2000.

[23] Bei Gu and H. Harry Asada. Co-simulation of algebraically coupled dynamic subsystems. In *Proceedings of the American Control Conference*, Arlington, VA, June 2001.

[24] Bei Gu and Harry H. Asada. Co-simulation of algebraically coupled dynamic subsystems without disclosure of proprietary subsystem models. *ASME Journal of Dynamic Systems, Measurement and Control*, 126(1):1–13, Mar 2004.

[25] Vadim Utkin, Jürgen Guldner, and Jingxin Shi. *Sliding Mode Control in Electromechanical Systems*. CRC Press, 1996.

[26] R. Kübler and W. Schiehlen. Virtual assembly of multibody systems. *Stability and Control: Theory and Applications*, 3(3):223–233, 2000.

[27] R. Kübler and W. Schiehlen. Modular simulation in multibody system dynamics. *Multibody System Synamics*, 4:107–127, 2000.

[28] Jinzhong Wang, Zheng-Dong Ma, and Gregory Hulbert. A gluing algorithm for distributed simulation of multibody systems. *Nonlinear Dynamics*, 34:159–188, 2003.

[29] Jinzhong Wang, Zheng-Dong Ma, and Gregory Hulbert. Gluing for dynamic simulation of distributed mechanical systems. *Advances in Computational Multibody Systems*, 5:69–94, 2005.

[30] Jinzhong Wang, Zheng-Dong Ma, and Gregory Hulbert. A distributed mechanical system simulation platform based on a „gluing algorithm”. *Journal of Computing and Information Science in Engineering*, 5:71–76, Mar 2005.

[31] C. W. Gear, G. K. Gupta, and B. Leimkuhler. Automatic integration of Euler-Lagrange equations with constraints. *Journal of Computational and Applied Mathematics*, 12:77–90, 1985.

[32] The MathWorks, Inc. *MATLAB function reference*.