ECCOMAS Thematic Conference on Multibody Dynamics
July 24 - 28, 2023, Lisbon, Portugal

Offline Learning Control to Improve the Accuracy of Real-time Explicit Co-simulation

Laurane Thielemans[1,3], **Francisco González**[2], **Laurens Jacobs**[3], **Roland Pastorino**[1], **Jan Swevers**[3]

[1] Siemens Digital Industries Software
Interleuvenlaan 68, 3001 Leuven, Belgium
[laurane.thielemans, roland.pastorino]
@siemens.com

[2] Laboratorio de Ingeniería Mecánica
Campus Industrial Ferrol - CITENI,
University of A Coruña
Mendizábal s/n, 15403 Ferrol, Spain
f.gonzalez@udc.es

[3]Department of Mechanical Engineering,
KU Leuven, Flanders Make@KU Leuven
Celestijnenlaan 300, 3001 Leuven, Belgium
[laurens.jacobs, jan.swevers]@kuleuven.be

ABSTRACT

In this paper a novel approach is developed to compensate for inaccuracies caused by direct feedthrough in single-rate explicit co-simulated systems. The method is twofold. First, a time delay is added to the interface variables of the co-simulation in order to break the algebraic loop caused by direct feedthrough. Second, this time delay is corrected by an iterative learning controller (ILC) that learns from previously executed simulation runs of the co-simulated system. The ILC contains an exact model of the time delayed co-simulation manager, but requires no prior knowledge on the dynamics of the subsystems. The method has proven to produce perfect results for examples where the simulation model complies with the standing assumption of using a time delay. For examples that rely on different approximation schemes to break the algebraic loop the developed approach gives good results in practice, but further investigation is required.

Keywords: Explicit co-simulation, Iterative learning control, Hybrid-physical-virtual testing.

1 INTRODUCTION

In industry, the product development methodology is evolving from a test-centric to a model-based development (MBD) approach. By simulating the parts of the system that are not yet physically available, MBD allows to frontload physical testing earlier in the development cycle of a product. Multiple product variants can be tested, flaws and defects can be identified early on, and high-risk real-life situations can be tested in a safe environment. These possibilities enable the MBD approach to reduce the product development time, leading to cost reduction and higher quality products. The technology behind the curtains of MBD is called (real-time) hybrid-physical-virtual testing (HPVT). It enables full system testing using physical and digital twins in R&D laboratories. HPVT relies on recent technical advancements in model-based techniques such as hybrid simulation, hardware-in-the-loop testing, and co-simulation.
The latter technique, co-simulation, constitutes an intuitive approach to model large and complex systems that consider multiple physical phenomena such as thermic, hydraulic, electronic and mechanic effects. The full system is divided into smaller subsystems that can be solved separately, at their own pace. Each subsystem can use a different solver to perform their integrations, in contrast to a monolithic simulation where a numerical solution of the complete model is obtained using a single solver. The subsystems only exchange information at discrete time instants called the **communication points**. In between communication points, the subsystems integrate their dynamics independently. The interval between these time instants is called the **macro time step**.

The **co-simulation manager** manages the discrete-time communication between the subsystems and exchanges the **coupling variables**, the subsystem inputs and outputs, at each communication point. There are many possible scheduling implementations for the co-simulation manager, which can be grouped in two main categories: implicit and explicit co-simulation methods. Implicit co-simulation methods solve the co-simulated problem in an iterative manner, by allowing to re-calculate a macro time step until a certain convergence criterion has been achieved. Explicit or non-iterative co-simulation methods do not allow to re-calculate time steps. Hence, implicit methods have better stability properties [1], [2], but they are infeasible to use when some subsystems are physical components or when there is a limitation on the available time to perform the computations, as is often the case in real-time applications. In contrast, explicit co-simulation methods are more likely to become unstable. These setups often need to resort to approximating or extrapolating the unknown inputs which can cause accuracy loss and stability issues [3]. Accuracy and stability are also affected by other co-simulation configuration parameters, which include the choice of coupling variables [2] and the step size of the co-simulation manager. Macro step sizes are often reduced or adapted either during runtime [4] or offline [5], in order to improve the stability of a co-simulation. However, variable step sizes cannot be applied to co-simulations that include physical subsystems, since they require a fixed step size that is defined by the actuation dynamics. The numerical integration of the continuous time dynamics introduces time delays and discontinuities, leading to a discretisation error in the simulation. However, the error introduced by dividing and coupling the subsystems in a co-simulation, also called the coupling error, is the main influence on the inaccuracy and stability issues in a co-simulation. That is why many researchers such as [4, 6, 7, 8] mainly focus on reducing the coupling error, and we will follow this example. In the remainder of this paper we assume that the discretisation error is negligible compared to the coupling error in the co-simulation. If no information about the subsystem states is given, correcting for the coupling error is only possible by acting on the coupling variables. By representing the co-simulated system in the frequency domain as a sampled-data interconnection, the authors of [6] have observed that the sample-hold process, i.e., the extrapolation, is the main error source. To reduce this extrapolation error they have implemented a H_∞ control algorithm with compensation and smoothing at the interface. Another direction is aiming to preserve the mechanical energy of a co-simulated system, for example with passivity control approaches. In [9] an energy-leak monitoring and correction method has been introduced that modifies the coupling variables to remove the residual energy, but knowledge of subsystem internals is required. The authors of [7] add a nearly energy-preserving coupling element (NEPCE) to the co-simulation to compensate for coupling errors caused by extrapolation. Exact compensation is not possible since this requires a non-causal correction element. The energy-conservation-based co-simulation (ECCO) method defined in [4] introduces the concept of residual power in a co-simulation to provide error estimation and adapt the step size of the co-simulation. This residual power concept is used in [10] to develop an adaptive force correction that can be applied to explicit (real-time) co-simulation. The correction force calculation requires defining a value for weighting component μ that depends on the co-simulation coupling strategy and the system properties.

In this paper we propose a novel, two-stage approach to correct for coupling errors in single-rate explicit co-simulation. First, instead of using an extrapolation method inside a subsystem to approximate unknown inputs, we choose to add a time delay to the outputs of this subsystem, which is easier to correct. Second, we design an iterative learning control (ILC) law that counteracts this time delay, by repeating the co-simulation multiple times and learning from the tracking error (the time delay) in the previous run. Since we target hybrid-physical-virtual applications that require real-time execution, we only consider explicit co-simulation with a fixed time step. Additionally, we assume that no information is available about the subsystem dynamics, except for the coupling variables.

The paper is organized as follows. Section 2 explains the concept of explicit co-simulation and discusses the example of a co-simulation where one of the subsystems has direct feedthrough. Section 3 describes the developed twofold methodology to improve the accuracy of explicit co-simulation: first the general principle, second a more detailed explanation for a co-simulation with two subsystems. Results are discussed in Section 4, and Section 5 concludes on the current status of this research and lists possible next steps.

2 EXPLICIT CO-SIMULATION

Explicit co-simulation does not allow rollback or iterating over a time step, such that each time step is only executed once. This choice of coupling approach, together with the order in which the subsystems evaluate their state and output at the next time step, influences the accuracy and stability properties of the co-simulation. The co-simulation manager can schedule the subsystem evaluations in parallel or in series. These schemes are known as Jacobi and Gauss-Seidel, respectively. The latter allows each next scheduled subsystem to use the already updated coupling variables from the previously scheduled subsystem. This way, Gauss-Seidel scheduling generally delivers more accurate results, although this property is shown to depend on the chosen order of subsystems [3]. This paper only considers the parallel scheduling of subsystems in single-rate co-simulation since it allows for higher computational performance in real time.
Fig. 1 schematically presents the Jacobi scheme for a single-rate co-simulation with two linear time-invariant (LTI) subsystems. At each macro step k, three steps are executed: In the first step,

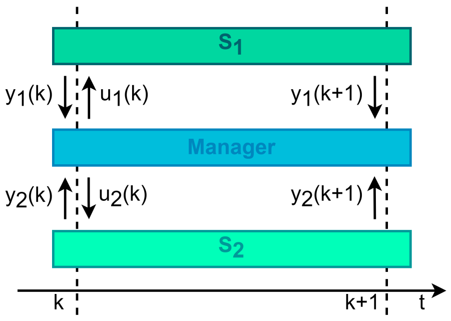


Figure 1: Single-step, single-rate Jacobi co-simulation scheme

the two subsystems send their outputs at time k to the manager which, in the second step, forwards these as the inputs to the other subsystem. In the third and final step the subsystems solve the state-space equations for the next time step $k+1$. Equation (1) prescribes the LTI state-space representation of Subsystem 1.

$$\begin{aligned} x_1(k+1) &= A_1 \cdot x_1(k) + B_1 \cdot u_1(k) \\ y_1(k+1) &= C_1 \cdot x_1(k+1) + D_1 \cdot u_1(k+1) \end{aligned} \tag{1}$$

In some co-simulation cases, depending on the chosen coupling variables, the output at time $k+1$ cannot be calculated since $u(k+1)$ is not available yet. Hence, an approximation $\tilde{u}(k+1)$ is necessary. Some example scenarios where this problem occurs are implicit integration formulas, multi-rate co-simulation schemes, nonlinear subsystems or subsystems with **direct feedthrough**. We will only discuss the latter in detail. We say that a subsystem has direct feedthrough when its outputs directly, instantaneously depend on its inputs. This might result in an algebraic loop which can only be solved iteratively or by approximating the unknown inputs, shown in Equation (2).

$$y_1(k+1) = C_1 \cdot x_1(k+1) + D_1 \cdot \tilde{u}_1(k+1) \tag{2}$$

Extrapolation techniques are often used to approximate the unknown input at time $k+1$. The most frequent approach is the zero order hold extrapolation (ZOH), that is, $\tilde{u}(k+1) = u(k)$, where the last evaluated value at time k is used to calculate the output at time $k+1$.

$$y_1(k+1) = C_1 \cdot x_1(k+1) + D_1 \cdot u_1(k) \tag{3}$$

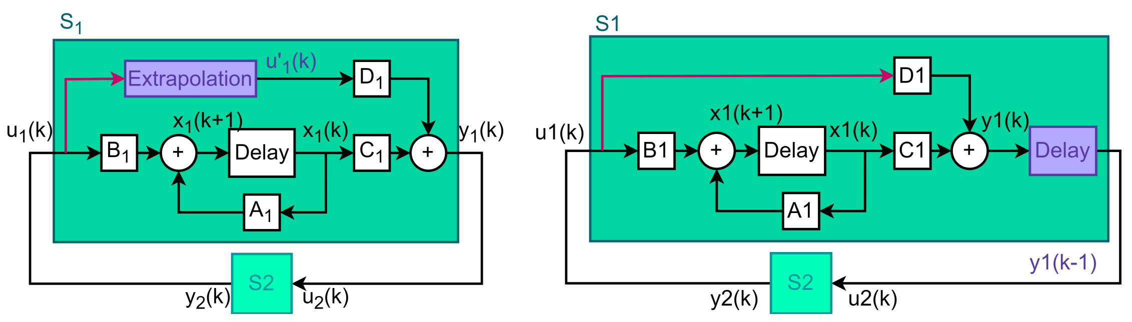
ZOH extrapolation helps to stabilize the co-simulation when direct feedthrough is present, but generally has a negative impact on its accuracy. Higher extrapolation orders will reduce the coupling error, but can have other effects on stability and robustness, because of a higher sensitivity to discontinuities [2]. Other methods, such as least squares approximation and polynomial approximation methods can be used to extrapolate the unknown inputs as well, without prior knowledge on the internal dynamics [11].
To summarize this section, the co-simulation coupling error depends on the magnitude of the time step, the chosen coupling variables, the co-simulation scheduling scheme and the extrapolation method. Reducing the coupling error is the goal of this paper and the following section, Section 3, will explain the methodology we propose for this purpose.

3 OFFLINE LEARNING CONTROL APPLIED TO CO-SIMULATION

Complete elimination of the coupling error in a co-simulation is very challenging, since in most cases the available information about the subsystems is insufficient. This paper assumes that the subsystems do not provide information about their states, such that only the coupling variables can be used to calculate and apply the coupling error correction. Single-rate explicit co-simulation is considered with Jacobi scheduling and a fixed macro time step. At least one of the subsystems has direct feedthrough.

3.1 General Methodology

The methodology we propose to correct for coupling errors in a single-rate explicit co-simulation is twofold. For the sake of brevity, without loss of generality, the methodology is explained using a co-simulation with two subsystems, where the first one has direct feedthrough, such that its outputs directly depend on its inputs. The first key contribution lies in the iterative nature of our approach, consisting in performing a sequence of consecutive co-simulation runs. A correction is added to the co-simulation after every run, based on the information collected during the run. This yields a correction approach that does not require implicit methods, adapting the communication step size, or modifying interface variables during runtime, rendering it compatible with real-time execution. This iterative control method, referred to as Iterative Learning Control (ILC) [12], is well known in control engineering. The goal of ILC is to iteratively learn a control input from previous test runs to improve the tracking of a specified reference. The tracking error and control inputs of the entire test are memorized and fed back offline into the ILC algorithm to improve the control input for the next test run. In the case of co-simulation, however, a reference is not available, because the analytic and the monolithic (numerical) solutions, which could be used to determine the correctness of co-simulation results, are not available in most applications of interest. As a consequence, appropriate assumptions have to be made to provide a valid reference for the ILC in co-simulation experiments.



(a) Using extrapolation (b) Using a time delay

Figure 2: Control loops used to represent a co-simulation environment with an algebraic loop

This is actually the second part of the proposed approach: instead of using an extrapolation method as shown in Fig. 1a, the co-simulation manager delays the output of the subsystem with direct feedthrough with one time step, as shown in Fig. 1b. This is because the extrapolation used in Equation (2) generates an approximation of the output of the subsystem with direct feedthrough which cannot be further corrected if the state of this subsystem is not available. That is why we prefer to add a time delay to the output of this subsystem, as given in Equation (4).

$$\begin{aligned} x_1(k+1) &= A_1 \cdot x_1(k) + B_1 \cdot u_1(k) \\ y_1(k+1) &= C_1 \cdot x_1(k) + D_1 \cdot u_1(k) \end{aligned} \tag{4}$$

Because this delay leads to a larger error in the uncorrected co-simulation, this approach may seem counterintuitive at first sight. However, as we will argue next, the resulting system is easier to correct with an ILC scheme. When an output is time-delayed, its desired value, which will become the ILC reference, is the output without this delay. This means that we need a time prediction of one time step. Put in other words, we need a non-causal controller. It is precisely in this type of situation where we can fully exploit the strengths of ILC. Since the ILC algorithm memorizes control inputs and outputs from the previous run, these can be shifted ahead in time, giving insight into future time steps. As such, the ILC can accurately compensate the addition of a delay to a subsystem during the next run. If, instead, the co-simulation is using an extrapolation method, for example ZOH extrapolation, the exact subsystem output is not available anymore as a reference signal for the ILC scheme. From ILC theory point-of-view, the sequence of co-simulations will still converge to a zero tracking error, albeit to a wrong reference. Hence, it is not very suitable in the considered application. Calculating another ILC reference that exactly compensates for the ZOH error is generally not possible, since the subsystem internals are not accessible.

3.2 Iterative Learning Control for a Co-simulation with Two Subsystems

In order to apply the offline learning principle to a co-simulated problem with two subsystems, we start from a standard feedforward control scheme, as given in [12]. The goal of feedforward control is to track a given reference as closely as possible, with the feedforward control signal compensating for the dynamics that cause the tracking error. Iterative learning control is a discrete-time feedforward control technique that learns from previously executed iterations, allowing for feedback in the iteration domain. Therefore, the formulas we state in the remainder of the paper are all given at a time step k for an iteration j. For our co-simulation the tracking error $e^j(k)$ is the difference between the reference signal $r^j(k)$, which is the output of Subsystem 1 without the time delay, and what is available as input to Subsystem 2, $u_2^j(k)$.

$$e^j(k) = r^j(k) - u_2^j(k) \tag{5}$$

This difference, without feedforward correction, corresponds to a time delay of one macro time step. Hence, the plant P considered by the learning control algorithm is a time delay with transfer function $P(z) = z^{-1}$ in the discrete-frequency domain. The ILC provides a correction signal (or control input) for the next iteration $c^j(k+1)$, which depends on the tracking error and correction signal from the current iteration $c^j(k)$. The learning function $L(q)$, with q the forward time-shift operator, relates the correction signal with the tracking error. Since the plant is rather simple, we propose to use a plant-inversion type ILC, where the learning function depends on a model of the inverted dynamics of the plant. Now the discrete-time ILC algorithm between consecutive iterations j and $j+1$ of the co-simulation is given by $c^{j+1}(k) = c^j(k) + P^{-1}(q)e^j(k)$. We can make the learning function L causal by making use of $qx(k) \equiv x(k+1)$ and rewriting the previous formulation as

$$c^{j+1}(k) = c^j(k) + q^{-1}P^{-1}(q)e^j(k+1) \tag{6}$$

The learning function then becomes $L(q) = q^{-1}P^{-1}(q)$, which equals one.
The iterative learning control scheme applied to a co-simulation with two subsystems is visualized in Fig. 3. It shows the closed-loop co-simulation for time step k with the ILC algorithm for iteration

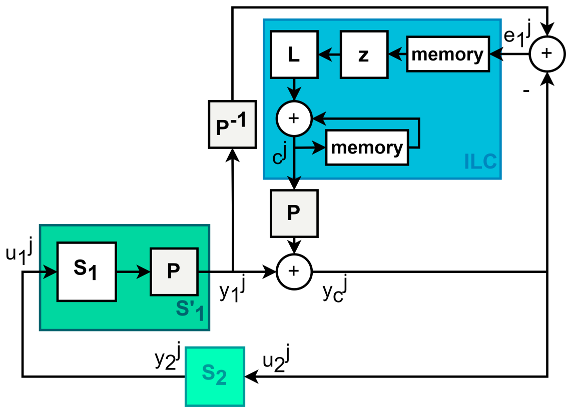


Figure 3: ILC to correct for time delay in a single-rate explicit co-simulation with two subsystems

j. The blue block (ILC) contains the offline calculation of the ILC algorithm, the light green block (S_2) represents Subsystem 2, and the dark green block (S_1') represents the adapted Subsystem 1. The latter is split up into two blocks, the first one of which represents the original dynamics of Subsystem 1 (S_1), and the second one represents the time delay (P) that is used to approximate the unknown output of S_1. It is clear that the ideal tracking reference of the ILC, which is the output of the first subsystem without time delay, is not available in real-time. Since the ILC algorithm is computed offline, the delayed output of Subsystem 1 can be shifted forward in time using an inverse time delay, providing us with the ideal tracking reference for the ILC,

$$r^j(k) = P(q)^{-1} y_1^j(k) \tag{7}$$

The ILC correction signal, multiplied with the time delay, is added to the delayed output of Subsystem 1, and the sum is applied as the input to Subsystem 2,

$$u_2^j(k) = y_c^j(k) = y_1^j(k) + P(q)c^j(k) \tag{8}$$

Since ILC is an iterative approach, it is useful to check how the error evolves over iterations, so as to analyse the convergence properties of the correction mechanism. A standard ILC algorithm assumes an iteration invariant reference, while in the co-simulation this reference changes over iterations, since all subsystems influence each other. For an ILC algorithm with changing reference, we need to analyse how the ILC error evolves in the iteration domain considering the fact that Subsystems 1 and 2 form a closed-loop connection. Because of its effective characterization, we perform this analysis in the frequency domain. Starting from the scheme in Fig. 3 we see that the output of Subsystem 1 and its corrected output can be expressed as $y_1^j(z) = S_1(z)S_2(z)y_c^j(z)$ and Equation (8) respectively. The ILC error is defined in Equation (5) as the difference between the desired S_1 output and the corrected output. Filling them in further gives

$$e^j(z) = P^{-1}(z)y_1^j(z) - y_c^j(z) = P^{-1}(z)\Big(P(z)S_1(z)S_2(z)y_c^j(z)\Big) - y_c^j(z) \tag{9}$$

The next step is to express the corrected S_1' output in terms of the ILC correction signal,

$$y_c^j(z) = \frac{P(z)}{I - P(z)S_1(z)S_2(z)} c^j(z) \tag{10}$$

where I represents the identity matrix. Combining Equations (9) and (10) leads to the influence of the ILC correction on the error:

$$e^j(z) = \frac{(S_1(z)S_2(z) - I)P(z)}{I - P(z)S_1(z)S_2(z)} c^j(z) \triangleq \tilde{P}(z)c^j(z) \tag{11}$$

The update law of the ILC gives

$$c^{j+1}(z) = P(z)(c^j(z) + e^j(z)) \tag{12}$$

The goal is to check stability and convergence of the ILC error, over iterations, which becomes, for iteration $j+1$

$$e^{j+1}(z) = (1 + \tilde{P}(z)z)e^j(z) \tag{13}$$

since $P(z) = z^{-1}$ and $L(z) = zP^{-1}(z) = 1$, giving us the recursive error propagation between iterations. Based on the convergence and stability properties of ILC [12], we can conclude that a sufficient condition for the ILC for a co-simulation with two subsystems, where one subsystem contains a time delay to break the algebraic loop to be asymptotically stable and converging is

$$\gamma \triangleq \|1 + \tilde{P}(z)z\|_\infty < 1 \tag{14}$$

In addition, (14) also guarantees the ILC is monotonically convergent, since

$$\|e_\infty(z) - e_{j+1}(z)\|_\infty < \gamma \|e_\infty(z) - e_j(z)\|_\infty \tag{15}$$

This derived analysis proves the theoretical convergence analysis of the ILC applied to a co-simulation with two LTI subsystems that includes a time delay to break the algebraic loop caused by direct feedthrough. Yet it has become clear that that the ILC convergence, as well as the convergence rate depends on the dynamics of the co-simulated subsystems. In practice, subsystem dynamics are usually not available, making it impossible to certify convergence.

4 RESULTS

To demonstrate our novel two-stage approach, we use a popular benchmark problem [10], shown in Fig. 4: a linear oscillator composed of two masses that are connected to each other and to the ground with ideal springs and dampers, without external excitation inputs. The subsystems exchange force and displacement, making it a force-displacement coupling where the subsystem with force output has direct feedthrough. The parameters of the two linear oscillator examples are given in Table 1. The macro step size of the co-simulation is 10^{-3} s. The examples are run with a simulation duration of 20 s per iteration.

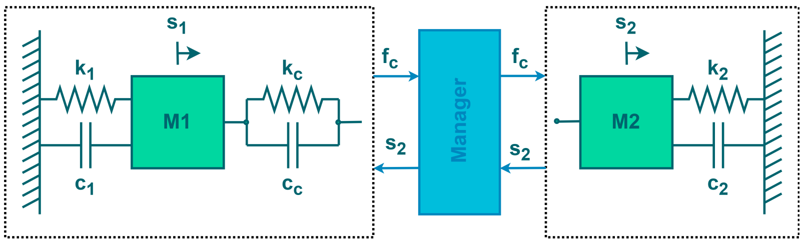


Figure 4: The benchmark problem of the linear oscillator with force-displacement coupling

Table 1: Parameters of the linear oscillator examples

Example	M1 [kg]	M2 [kg]	k_1 [N/m]	k_2 [N/m]	k_c [N/m]	c_1 [Ns/m]	c_2 [Ns/m]	c_c [Ns/m]
1	1	1	10	1000	100	0.1	0.1	0.1
2	1	1	10	1000	100	0	0	0

We study the ILC performance for two different scenarios: (a) when co-simulation uses a time delay to eliminate the algebraic loop (the time delay assumption holds), and (b) when the co-simulation uses ZOH extrapolation (the time delay assumption does not hold). We investigate how the same control algorithm behaves in both situations, and how it compares to the residual power method in [10] and to the uncorrected co-simulation. To verify the accuracy of the results, the co-simulated solution is verified against its monolithic numerical counterpart using the mechanical energy of the system as a metric. Since the ILC only tries to correct for the coupling error caused by the time delay approximation, and not for the discretisation error, the monolithic solution is the best achievable tracking result for the ILC. It must be noted that neither the monolithic solution nor the mechanical energy are usually available in applications of practical interest, so they are only used to verify the simulation results, but not as sources of information for the ILC.

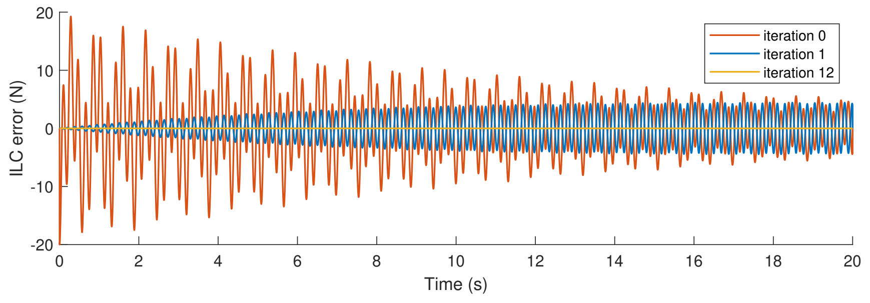


Figure 5: Convergence of the ILC error for the damped linear oscillator

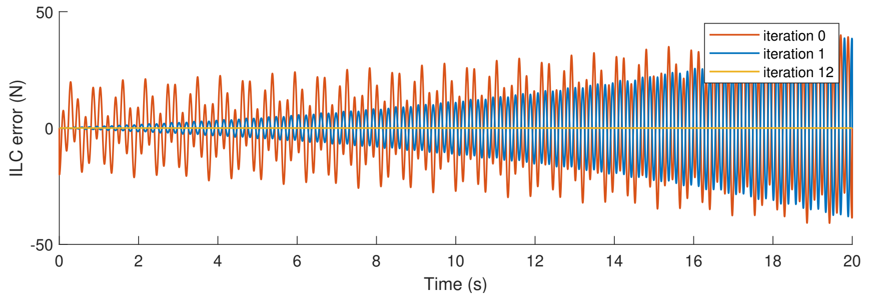


Figure 6: Convergence of the ILC error for the undamped linear oscillator

For discussing the results both undamped and damped linear oscillator examples are considered simultaneously. Figs. 5 and 6 depict the error of the ILC for three co-simulation iterations. Iteration 0 gives the original error that is given to the ILC, before any correction has taken place. In iteration 1 a first ILC correction has been calculated and added to the co-simulation coupling variables, affecting the output of that iteration. Iteration 12 shows that the ILC algorithm has converged and the error has been reduced to zero with an accuracy of 10^{-7} N for both examples. Equation (14) has been satisfied for the damped linear oscillator, while for the undamped example, which is marginally unstable, it does not hold. In practice, for both examples, the 2-norm of the error has been reduced to 10^{-7} N. The difference in force (input variable of Subsystem 2) with the monolithic solution to the linear oscillator is plotted on Figs. 7 and 9. The left subfigure compares the developed ILC approach with time delay approximation to the uncorrected co-simulation with ZOH extrapolation, and a co-simulation with ZOH extrapolation that is corrected using the residual power approach of [10]. The right subfigure gives a more detailed view and shows the developed ILC approach when ZOH extrapolation is used, so the time delay assumption does not hold. Similar plots are created for the mechanical energy of the co-simulation (Figs. 8 and 10).
These four figures (with eight plots) confirm that that the developed twofold methodology both stabilizes the co-simulation and matches the monolithic solution with an accuracy of 10^{-6}, considering that the subsystems are implemented as expected, that is, using a time delay to break the algebraic loop. When the time delay assumption is not satisfied, in contrast, a perfect correction is not possible. The ILC results in reasonably good results and an accuracy comparable to the residual power approach is obtained. However, for this case, our two-stage approach that only

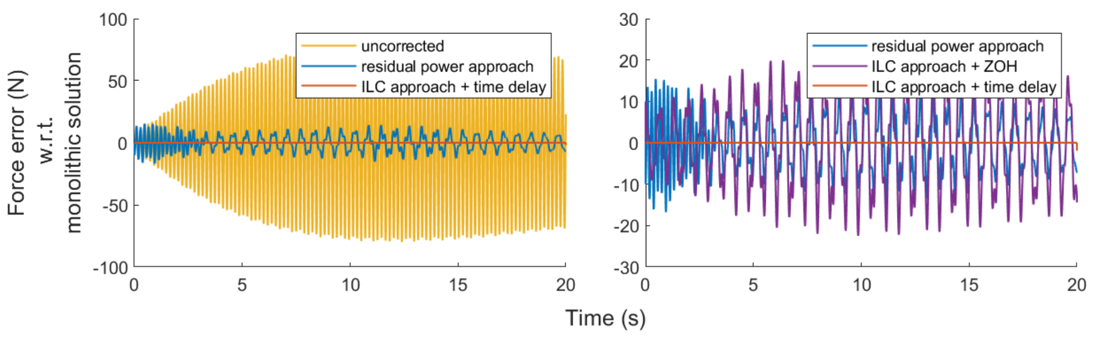


Figure 7: Comparing the input force of Subsystem 2 to the monolithic solution for the damped linear oscillator

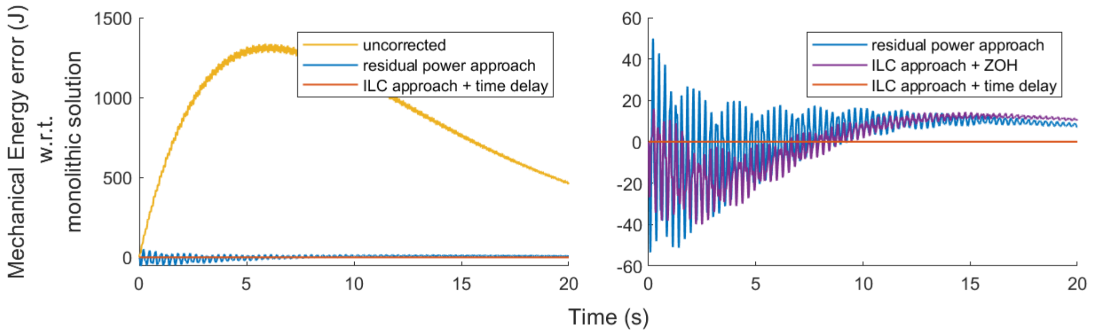


Figure 8: Comparing the mechanical energy to the monolithic solution for the damped linear oscillator

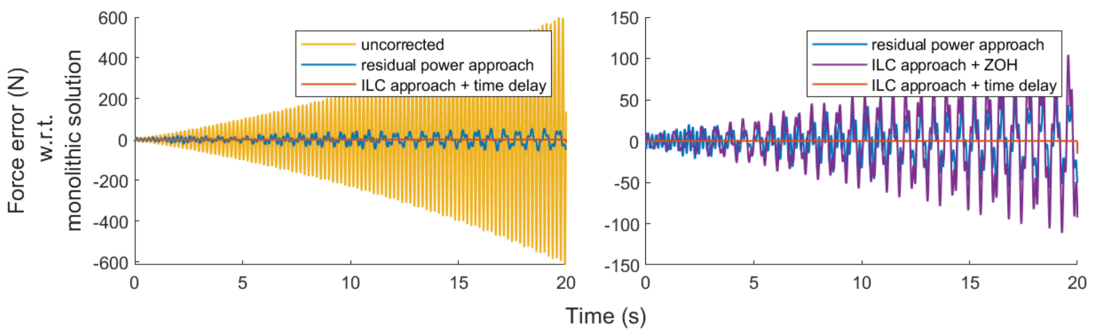


Figure 9: Comparing the input force of Subsystem 2 to the monolithic solution for the undamped linear oscillator

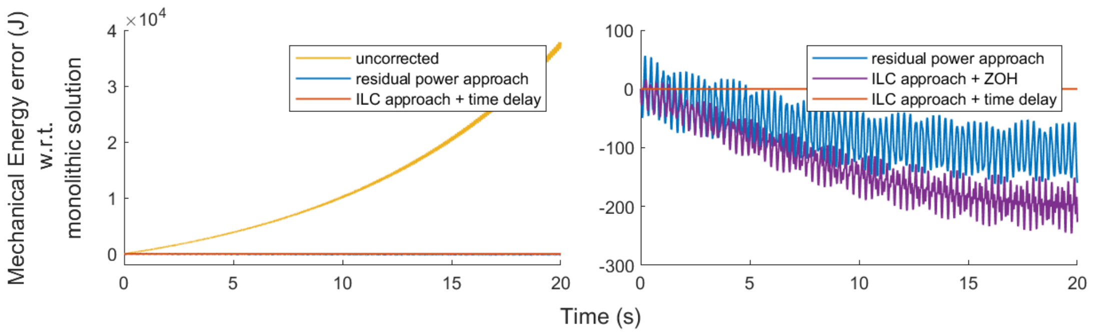


Figure 10: Comparing the mechanical energy to the monolithic solution for the undamped linear oscillator

compensates for a time delay is not capable of formally guaranteeing any error reduction. More simulation examples would be required to confirm this reasoning. The residual power approach is non-iterative and gives good results without repeating the co-simulation, after its parameters have been properly tuned, but is less accurate than our approach. As one can intuitively expect, this example demonstrates that there is a trade-off between achieving highly accurate results and having to repeat the co-simulation several times.

5 CONCLUSIONS

Coupling errors in a co-simulation caused by direct feedtrough, nonlinearities or multi-rate simulation can result in inaccurate and possibly unstable simulation results. In the quest of eliminating these unwanted effects, this paper has investigated a novel twofold approach to remove the errors coming from approximating the unknown inputs in single-rate explicit co-simulation examples. First, a time delay is added to the subsystems with direct feedthrough instead of using an extrapolation method. Second, an iterative learning control (ILC) algorithm is implemented that compensates for the error caused by this time delay. The approach has been shown to deliver accurate results when the co-simulation complies with the standing time delay assumption, at the cost of having to run the co-simulation several times. For cases that rely on different approximation schemes or without direct feedthrough, the developed approach can possibly give good results in practice despite the ILC tracking a wrong reference. Further investigation for these cases is required, nevertheless. Next to that, future work will revolve around applying the method to nonlinear or multi-rate co-simulations, in addition to developing strategies to determine the existence of direct feedthrough in the subsystems, which is necessary to apply the method satisfactorily.

ACKNOWLEDGMENTS

The financial support from the Flemish Agency for Innovation and Entrepreneurship (VLAIO project HBC.2020.2880) is gratefully acknowledged.

REFERENCES

[1] Kübler, R., Schiehlen, W.: Two Methods of Simulator Coupling. Mathematical and Computer Modelling of Dynamical Systems **6**(2) (2000) 93–113

[2] Schweizer, B., Li, P., Lu, D.: Explicit and implicit cosimulation methods: Stability and convergence analysis for different solver coupling approaches. Journal of computational and nonlinear dynamics **10**(5) (2015) 051007

[3] Li, P., Yuan, Q.: Influence of coupling approximation on the numerical stability of explicit co-simulation. Journal of Mechanical Science and Technology **34**(6) (2020) 2289–2298

[4] Sadjina, S., Kyllingstad, L.T., Skjong, S., Pedersen, E.: Energy conservation and power bonds in co-simulations: non-iterative adaptive step size control and error estimation. Engineering with Computers **33**(3) (2017) 607–620

[5] Inci, E.O., Croes, J., Bosmans, J., Vermaut, M., Desmet, W.: Offline adaptation of co-simulation time steps by leveraging past co-simulation runs in multi-level mechatronic systems. Mechanism and Machine Theory **171** (2022) 104740

[6] Chen, W., Ran, S., Wu, C., Jacobson, B.: Explicit parallel co-simulation approach: analysis and improved coupling method based on H-infinity synthesis. Multibody System Dynamics **52**(3) (2021) 255–279

[7] Benedikt, M., Watzenig, D., Zehetner, J., Hofer, A.: NEPCE - A nearly energy-preserving coupling element for weak-coupled problems and co-simulation. In: Proceedings of NAFEMS World Congress. (2013) 1–12

[8] Tamellin, I., Richiedei, D., Rodríguez, B., González, F.: Eigenstructure assignment and compensation of explicit co-simulation problems. Mechanism and Machine Theory **176** (2022) 105004

[9] González, F., Arbatani, S., Mohtat, A., Kövecses, J.: Energy-leak monitoring and correction to enhance stability in the co-simulation of mechanical systems. Mechanism and Machine Theory **131** (2019) 172–188

[10] Rodríguez, B., Rodríguez, A.J., Sputh, B., Pastorino, R., Naya, M.A., González, F.: Energy-based monitoring and correction to enhance the accuracy and stability of explicit co-simulation. Multibody System Dynamics **55**(1) (2022) 103–136

[11] Busch, M.: Continuous approximation techniques for co-simulation methods: Analysis of numerical stability and local error. ZAMM - Journal of Applied Mathematics and Mechanics **96**(9) (2016) 1061–1081

[12] Bristow, D., Tharayil, M., Alleyne, A.: A survey of iterative learning control. IEEE Control Systems Magazine **26**(3) (2006) 96–114