LookAhead - A New Approach for Event Handling in Co-Simulation by Predicting State Events

Felix Tischer
VIRTUAL VEHICLE Research Center
Inffeldgasse 21a, 8010 Graz, Austria
Email: Felix.Tischer@v2c2.at

Simon Genser
VIRTUAL VEHICLE Research Center
Inffeldgasse 21a, 8010 Graz, Austria
Email: Simon.Genser@v2c2.at

Daniel Watzenig
VIRTUAL VEHICLE Research Center
Inffeldgasse 21a, 8010 Graz, Austria
Email: Daniel.Watzenig@v2c2.at
Graz University of Technology
Institute of Automation and Control
Inffeldgasse 21/B/I, 8010 Graz, Austria
Email: daniel.watzenig@tugraz.at

Martin Benedikt
VIRTUAL VEHICLE Research Center
Inffeldgasse 21a, 8010 Graz, Austria
Email: Martin.Benedikt@v2c2.at
SETLabs Research GmbH
Elsenheimerstraße 55, 80687 Munich, Germany
Email: Martin.Benedikt@setlabs.de

***Abstract*— Hybrid systems consisting of discrete state events in otherwise continuous systems pose a significant challenge in co-simulation, as these events can lead to large errors depending on when they occur during a coupling time step. The coupling step size thus needed to correctly simulate state events can be significantly smaller than for the continuous part, resulting in an overall reduced coupling step size and longer and more expensive simulations. By dynamically adapting the coupling step size whenever an event is imminent, hybrid co-simulation can be improved significantly when compared to fixed-step co-simulation algorithms. This can be accomplished by predicting upcoming state events in advance. In an example simulation, this improves precision by a factor of 30 and decreases the number of coupling steps by a factor of 7.**

***Key words*—co-simulation, non-iterative co-simulation, hybrid simulation, event handling**

I. Introduction

In simulation, an event is defined as a system change that occurs instantly at a discrete point in time. The manner of change is not specified, and can include changes in state, models, or in the simulation behavior itself. A simulation that incorporates dynamic, continuous changes as well as discrete events is called a hybrid simulation [1] [2].

A physical system described by differential equations that is controlled by an electronic device, for example an autonomous vehicle or a smart power grid, is called a Cyber-Physical System (CPS), and is a prime example for hybrid simulations. The increasing importance of such systems makes it necessary to develop simulation tools that are able to handle them in a reliable and efficient manner [3].

An important tool in industrial development is co-simulation, where individual parts of a larger, complex system are being simulated simultaneously [4]. There, a master algorithm coordinates the simulation of the underlying subsystems by periodically exchanging information between them [5].

These subsystems are model components that contain their own solvers. They can thus have different solver step sizes, but are synchronized at regular intervals, the coupling time step, which is a property of the co-simulation and has a strong influence on the performance and accuracy of the co-simulation.

Events are especially challenging in co-simulation [6], as their time of occurrence is usually not known in advance, and if they occur in a subsystem between coupling time steps, they can introduce a large error in other subsystems as their correct state is only communicated at the next synchronization time.

By the condition of their occurrence, events can be classified as time events and state events. A time event occurs at specific points in time, and its occurrence is thus known in advance. A state event occurs when the model is in a certain state, like a variable reaching some threshold value. The time when a state event occurs is unknown in advance.

In co-simulation, events can also be distinguished by whether they belong to one or more than one subsystem. A private event is an event that happens in a single subsystem and influences the others only indirectly by the changes it introduced in its own subsystem. A shared event is one which belongs to two or more subsystems and affects both directly, like a binary collision. It thus has to occur in all participating subsystems at the same time [7].

The Functional Mock-up Interface (FMI) is an industry standard for co-simulation. The newest version, fmi3.0, added hybrid simulation capabilities for co-simulation FMUs (Func-

 ISBN: 978-1-68558-102-2

tional Mock-up Units) in the form of clock variables whose "ticks" signal events [8]. The terminology and handling of events in this paper is based on the fmi3.0 specification [9].

The proposed algorithm attempts to improve hybrid co-simulation in a specific but common case: a system that is mostly defined by differential equations, but also contains discrete state events that happen rarely but have a large impact on model behaviour. This can be a frictional force changing sign, an autonomous vehicle performing an emergency braking, or as in the example here, collisions in a spring-damper-mass system.

To hit these events correctly, such systems often have to be co-simulated with a coupling step size that is far smaller than necessary with regards to the continuous behaviour. A lot of calculation time can be saved with an adaptive step size that is large by default, and smaller when a state event is imminent. By evaluating the current state of the system via event indicators, state events are predicted (even though their exact time is still unknown) sufficiently in advance, and the coupling time step is then reduced.

The next section describes how events are defined in the LookAhead algorithm. Section III then describes the algorithm, and Section IV showcases it with an example co-simulation. Section V then summarizes the work and discusses future developments.

II. Definition of Event Indicators

Define an event indicator as the complete set of functions which indicate an event:

$$Z = \{\, z_i(\vec{u}, \vec{y}, \vec{p}) \,|\, E \Leftrightarrow z_i \leq 0 \,\forall\, i\} \tag{1}$$

The elements of this set are functions whose arguments are the inputs $\vec{u}$, the outputs $\vec{y}$, and the parameters $\vec{p}$ of the corresponding subsystem. Each function represents a condition for the event E that is met if the function evaluates to a value less than or equal to zero. Let these functions be called event conditions. Then, E occurring is equivalent to all z_i being less than or equal to zero. We can reformulate this condition as

$$E \Leftrightarrow \max_i z_i \leq 0 \tag{2}$$

To ensure that an event happens only at a single time instant, the functions in Z have to be set up in a way such that at least one function returns a positive value after the event so that its conditions are no longer fulfilled afterwards. As the subsystem cannot control its inputs $\vec{u}$, this generally has to be accomplished by changing its outputs $\vec{u}$. The event can also change the parameters $\vec{p}$ or even the event indicator Z in case of models that support such functionalities.

The z_i are not directly dependent on time but its arguments are; we therefore denote $z_i(\vec{u}(t), \vec{y}(t), \vec{p}(t)) = z_i(t)$. As an event can lead to discontinuities in its arguments, $z_i(t)$ are not guaranteed to be continuous functions; but they are continuous on all intervals where no events occur.

Then, assuming that $z_i(t)$ are continuous right before an upcoming event, $\max_i z_i$ is also continuous, and we can replace (2) with a stricter condition:

$$E \Leftrightarrow \max_i z_i = 0 \tag{3}$$

This means that the event happens as the last of its z_i crosses the x-axis. As the execution of an event changes at least one z_i to be positive afterwards, the event E only happens at these discrete points in time t_E. $\max_i z_i$ is thus always non-negative, and zero exactly at the event times t_E.

III. Description of LookAhead Algorithm

A co-simulation scenario with parallel scheduling is considered, where all subsystems calculate their coupling time steps in parallel and exchange their inputs and outputs afterwards.

This is done by the master algorithm, which orchestrates the co-simulation by signalling the subsystems to calculate time steps, and enables communication between them by requesting outputs and setting inputs. Each subsystem contains a model and, if necessary, handles translating these signals to the standard the contained model uses. The basic co-simulation loop then looks like this:

```
1: while t < t_stop − t_step do
2:     for all subsystem in system do
3:         subsystem.DoStep(t, t_step)
4:     end for
5:     for all connection in connections do
6:         connection.target.Set() ← connection.source.Get()
7:     end for
8:     t ← t + t_step
9: end while
```

Fig. 1. Basic co-simulation loop

In line 3, all subsystems in the co-simulation calculate one coupling time step. Later in line 6, the subsystems exchange data with each other according to their connections. Finally in line 8, the simulation time advances by the coupling step size before commencing to the next.

The LookAhead algorithm consists of two parts. One part is contained in a subsystem (Local LookAhead), while the other one is part of the master algorithm (Main LookAhead).

A. Local LookAhead

A subsystem containing state events can be equipped with a Local LookAhead routine (Algorithm 2) to predict its events and communicate them to the master algorithm. For each event, an event indicator set Z is included in the subsystem. The event conditions z in this set have to be provided manually (for now; see Section V for upcoming work on automatic event indicator construction). These z can only depend on locally accessible variables.

The local LookAhead algorithm then gets called at the end of each coupling time step to predict if an event will occur

 ISBN: 978-1-68558-102-2

soon, and does so in a manner independent of the number of event indicators or the sizes of the event indicator sets.

```
1:  for Z in list of event indicators do
2:      t_0 ← −1
3:      for z in Z do
4:          z_prev ← z_now
5:          z_now ← z(t)
6:          z_next ← z_now + d ∗ (z_now − z_prev)
7:          if z_next < 0 then
8:              if z_now > 0 then
9:                  t_0,new ← c ∗ Δt ∗ z_now/(z_prev − z_now)
10:                 if t_0,new < t_min then
11:                     t_0,new ← t_min
12:                 end if
13:             else
14:                 t_0,new ← −1
15:             end if
16:             t_0 ← max(t_0, t_0,new)
17:         else
18:             t_0 ← −1
19:             break
20:         end if
21:     end for
22:     if t_0 ≥ t_min then
23:         append t_0 to t_E
24:     end if
25: end for
26: if t_E not empty then
27:     return min t_E
28: else
29:     return −1
30: end if
```

Fig. 2. Local LookAhead algorithm

For each event indicator, the local LookAhead algorithm takes the current and previous values of the event indicators z (z_{now} and z_{prev}, respectively) to extrapolate the value z_{next} that it will have d (the forecasting factor, $d > 1$) time steps in the future in line 6. For this, the algorithm requires the current coupling step size Δt (which can be smaller than the default value if the previous time step was already shortened by LookAhead) as an input from the master algorithm.

If all z_{next} are below zero, the event is predicted to happen. On the other hand, if one $z_{next} > 0$ (line 17) the event is not predicted to happen, and the algorithm continues on to the next event indicator. If an event indicator is predicted to be negative in the future but is positive at the current time, the time until zero-crossing t_0 gets interpolated in line 9. To prevent overestimation, a safety factor $c \in (0, 1]$ is introduced that reduces t_0.

In case that t_0 is less than the minimal solver step size t_{min}, $t_0 \leftarrow t_{min}$ (line 11). If z_{now} is already negative, t_0 is set to -1 (line 18) as it is not relevant to event time estimation. If all z_{next} are negative, the corresponding event is predicted to happen at the time when the last zero-crossing happens. The event time for the predicted event is then the maximum value of all t_0, as this is the time where (3) is fulfilled, which is then appended to the list of predicted events t_E in line 23.

After iterating through all events of the subsystem, the minimum of all t_E will be returned (line 27). If no event is predicted to happen, LookAhead will return -1 in line 29 to signal that its return value has to be ignored.

B. Main LookAhead

As shown in Algorithm 1, the master algorithm advances its own time by the same constant time step after each iteration of calculating and exchanging outputs. The main LookAhead extends Algorithm 1 to make it possible to adapt this time step.

```
1:  while t < t_stop − t_step,max do
2:      for all subsystem in system do
3:          subsystem.DoStep(t, t_step)
4:      end for
5:      for all connection in connections do
6:          connection.target.Set() ← connection.source.Get()
7:      end for
8:      t ← t + t_step
        list t_next ← {t_step,max}
9:      for all subsystem in system do
10:         if subsystem.supportsLookAhead then
11:             t_E ← subsystem.lookAhead(t_step)
12:             if t_E > 0 then
13:                 append t_E to t_next
14:             end if
15:         end if
16:         t_step ← min t_next
17:     end for
18: end while
```

Fig. 3. Co-simulation loop containing main LookAhead

After completing the conventional co-simulation loop, the master algorithm calls the LookAhead function in every subsystem that supports it. Each returns either -1 if no event is predicted to happen, or returns the predicted time until the next event t_E, in which case this time gets appended to a list of event times t_{next}. The master algorithm then chooses the smallest element of that list as its next time step. If no subsystem returns an event time or if event times are larger than the default time step, this default time step $t_{step,max}$ gets selected.

IV. Example on a spring-damper-mass system

The example is taken from [7] with minimal adaptations.

Two masses are vertically connected in a spring-damper system; the upper mass m_1 is connected to the ceiling with a spring of stiffness k_1 and dampening d_1, and m_1 and the lower mass m_2 are connected together via a second spring of stiffness k_2 and d_2. Each mass has a position x_i and a velocity v_i. The masses are realized as blocks with vertical size $2\Delta x$. Figure 4 illustrates this system.

 ISBN: 978-1-68558-102-2

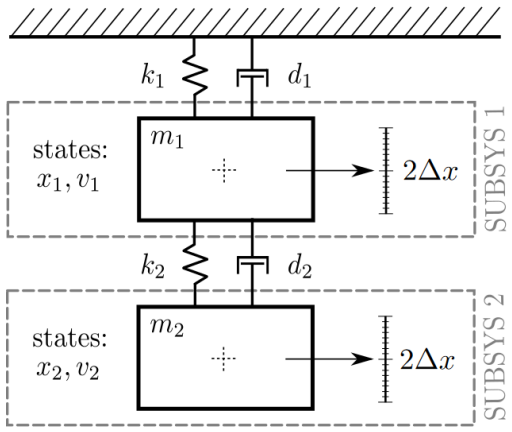

Fig. 4. Setup of the spring-damper-mass system. The figure and example have been taken from [7], with the slight adaptation of changing Δx to $2\Delta x$.

The upper mass can collide with the ceiling and the two masses can collide with each other. These collisions can be realised as state events; they are an instantaneously occurring change triggered by the system being in a certain state.

This system is a good example to demonstrate LookAhead as it is a continuous system that encounters multiple events when the initial conditions are set up correctly, and whose trajectory can diverge greatly between the monolithic simulation and the co-simulation when these events are not handled well enough in the co-simulation.

A. Co-simulation Setup

For co-simulation, the system is split up into upper and lower mass. Each one of the masses takes the other one's position and speed as inputs $\vec{u}$, and provides its own position and speed as outputs $\vec{y}$.

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -\frac{k_1+k_2}{m_1}y_1 - \frac{d_1+d_2}{m_1}y_2 + \frac{k_2}{m_1}u_1 + \frac{d_2}{m_1}u_2 - g \end{aligned} \tag{4}$$

Let event 1 be the collision of the upper mass with the ceiling. Mass 1 then gets reflected downwards (6) when it touches the ceiling and has positive speed (5). Let event 2 be the collision of the two masses. Their reflected velocities depend on the masses (8). Similar to event 1, event 2 occurs when the masses touch and move towards each other (7).

$$Z_1 = \{z_1 = -(y_1 + \Delta x), z_2 = -y_2\} \tag{5}$$

$$E_1 : y_2' = -y_2 \Leftrightarrow \max_{z \in Z_1} z = 0 \tag{6}$$

$$Z_2 = \{z_1 = y_1 - u_1 - 2\Delta x, z_2 = y_2 - u_2\} \tag{7}$$

$$E_2 : y_2' = \frac{m_1 - m_2}{m_1 + m_2}y_2 + \frac{2m_2}{m_1 + m_2}u_2 \Leftrightarrow \max_{z \in Z_2} z = 0 \tag{8}$$

Subsystem 1 simulates the trajectory of the upper mass, and contains its equations of motions (4) as well as both events.

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -\frac{k_1+k_2}{m_2}y_1 - \frac{d_1+d_2}{m_2}y_2 + \frac{k_2}{m_2}u_1 + \frac{d_2}{m_2}u_2 - g, \end{aligned} \tag{9}$$

$$\begin{aligned} Z &= \{z_1 = u_1 - y_1 - 2\Delta x, z_2 = u_2 - y_2\} \\ E_2 : y_2' &= \frac{m_2 - m_1}{m_1 + m_2}y_2 + \frac{2m_1}{m_1 + m_2}u_2 \Leftrightarrow \max_{z \in Z} z = 0 \end{aligned} \tag{10}$$

Subsystem 2 simulates the lower mass, and contains its equations of motions (9) and event 2 (10).

Event 1 is a private state event. As it occurs in two subsystems at the same time, event 2 is a shared state event. The relative velocity between objects changes sign after a collision, thus both events no longer fulfill their second conditions after the event.

For the monolithic solution, the equations of motions are solved as a single system of four equations, and the events are implemented by checking the event conditions after each solver step.

Regarding the setup of the LookAhead routine, the safety and forecasting factors have to be chosen. The value of the safety factor $c = 0.9$ has been chosen based on the behaviour of the variables relevant for event prediction. On the co-simulation step time scale, the positions and velocities of the two masses behave quite linear, meaning that the extrapolation of z inside the LookAhead algorithm only has a small error compared to the real values. Thus, c can be chosen with a high value near 1. If the event conditions z were less predictable, the safety factor would need to be smaller in order to prevent the algorithm from missing events.

The forecasting factor of $d \leq 2$ is chosen similarly.

B. Simulation Results

In the upper part of Figure 5, the trajectories of three simulations are compared. The solid blue lines are the trajectories of the co-simulation where both subsystems contain a local LookAhead instance. The green dashed lines are the trajectory of a co-simulation without LookAhead capability, and the red dotted lines are the monolithic solution as a reference. The upper mass collides with the ceiling when its position is $x_1 = -\Delta x$, indicated by the horizontal line. The lower part of Figure 5 shows the absolute error

$$e_{pos} := |x_1 - \hat{x}_1| + |x_2 - \hat{x}_2|$$

for both co-simulations, where $\hat{x}_i$ are the positions of the reference solution. It is easy to see how LookAhead improves the performance of the co-simulation. The root mean square of this error,

$$e_{pos,rms} := \sqrt{\frac{1}{n}\sum_{i=0}^{n}(|x_{1,n} - \hat{x}_{1,n}| + |x_{1,n} - \hat{x}_{1,n}|)^2},$$

is $e_{rms} = \mathbf{0.12}$ and $e_{rms} = \mathbf{3.65}$, with and without LookAhead respectively. The default time step of the co-simulation was $t_{step} = 0.08s$, calculating to $t_{stop} = 20s$ in

 ISBN: 978-1-68558-102-2

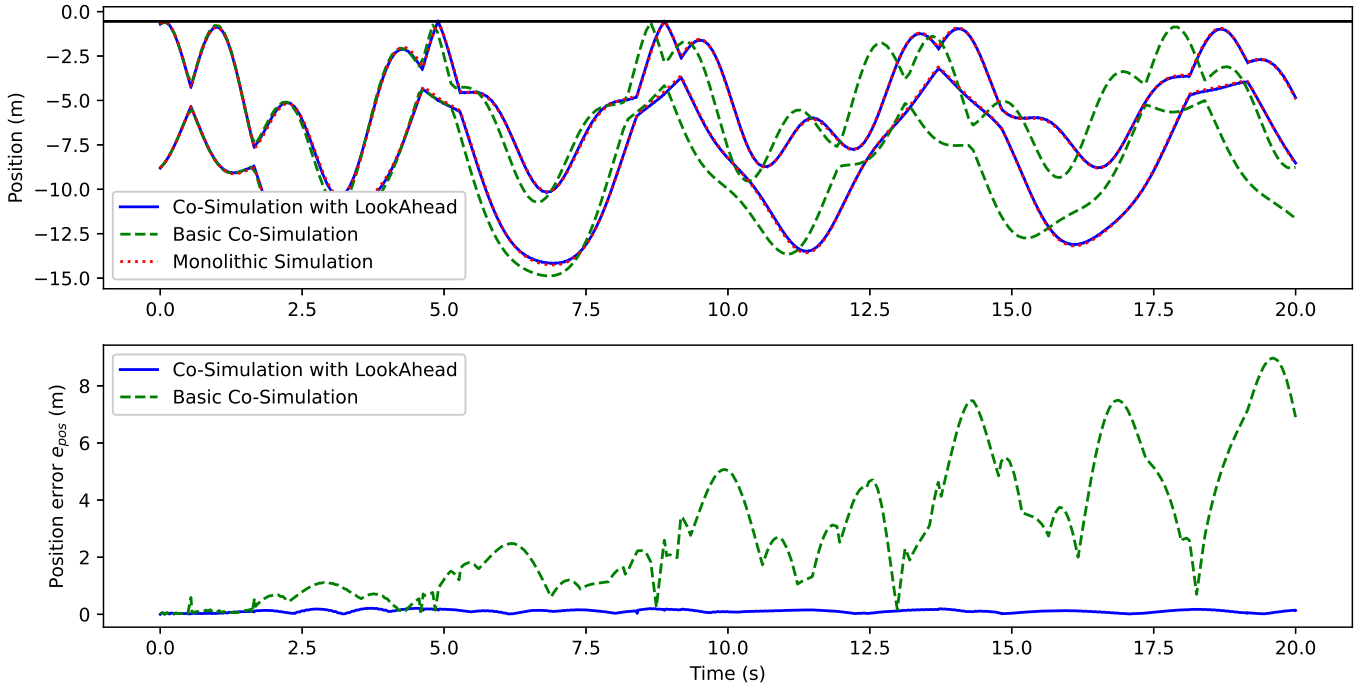

Fig. 5. Comparison of trajectories of the two masses. Top: total positions of the two masses in each simulation case. Bottom: Position error with respect to the monolithic simulation result.

250 steps. LookAhead introduced only 32 additional steps. This means that, with the same base coupling step size, the error improves by $\mathbf{96.7}\%$ or a factor of $\mathbf{30.3}$ while only taking 12.8%. more steps.

To achieve an error equal or less than that without LookAhead, $t_{step} = 0.01$ is required, taking 2000 steps instead with an error of $e_{rms} = 0.07$, meaning that, for a desired error limit, LookAhead reduces the number of steps by $\mathbf{85.9}\%$ or a factor of $\mathbf{7.1}$.

Figure 6 shows two detail views of the monolithic and LookAhead simulations. Approaching the event in the top image, coupling step size gets increasingly smaller until the event occurs, after which the step size returns to default. In the bottom half no event occurs, but as their extrapolated trajectories intermittently cross each other, LookAhead decreases step size for a few steps before returning to default once it is clear no event will happen.

V. Summary, Conclusion and Future Work

A. Summary

By defining functions that indicate the occurrence of state events, it is possible to estimate imminent state events a few time steps prior to their occurrence by interpolating these event condition functions. Then, the coupling time step can be adapted to hit these event times more precise.

Compared to a constant time step simulation, where events can happen at random points between time steps, this can significantly improve the error, especially in chaotic systems like the one shown in the example.

B. Conclusion

The proposed LookAhead algorithm vastly improves the performance of hybrid systems where only a small number of events occur during co-simulation, while keeping calculation time low. The coupling step size is only reduced when necessary, and the additional overhead for event prediction is kept as small as possible. In this manner, the error was reduced by a factor of 30 with the same step size. To achieve the same error without implementing LookAhead in the master algorithm, the co-simulation takes 7 times more steps.

This is much more efficient than choosing a smaller coupling step size in general or taking an iterative approach [10], as many models do not support this. If they do, saving and (re)setting states is computationally expensive as well, especially with growing complexity of the models or the co-simulation, while LookAhead's complexity only depends on the number of events.

C. Future Work

Future work is mostly focused on usability. In the example, the event indicators have to be set up manually and are independent of their actual implementation in the subsystems, as these are treated as black-boxes. It would be favourable if this step could be automated in cases where many events are present or it is not known how they depend on the inputs.

Also, there exist other scheduling paradigms besides parallel execution. Thus, for general coupling schemes the main LookAhead algorithm has to be adapted, depending on how and when signal exchange takes place.

 ISBN: 978-1-68558-102-2

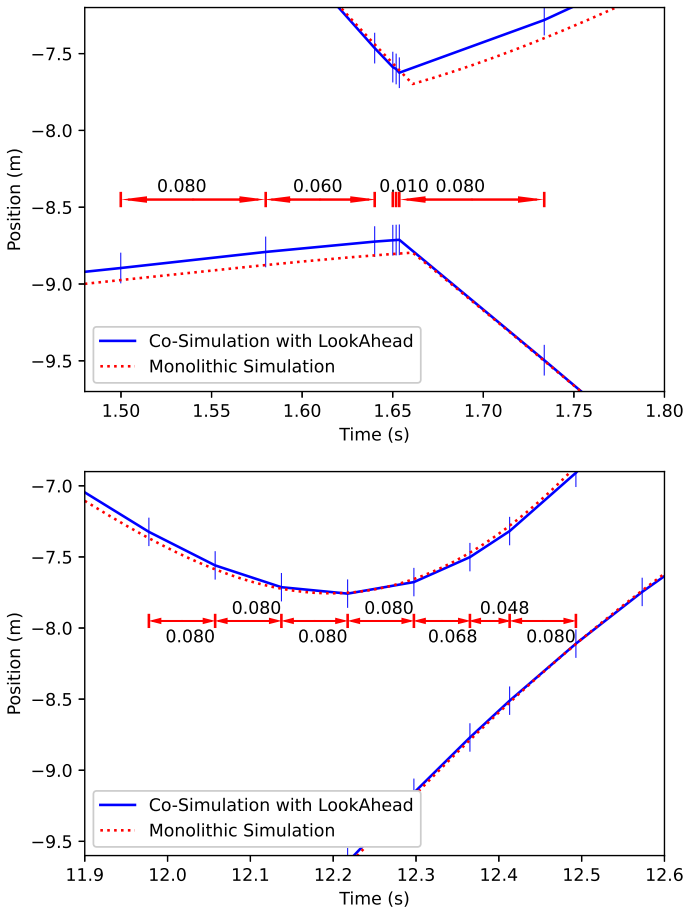

Fig. 6. Detailed views of LookAhead predicting events. The vertical lines mark the synchronization time points in the co-simulation. The lines get denser as they approach an event, and return to the default communication step size afterwards. Top: A shared event. Bottom: Prediction of an event that does not happen.

This could be combined with an example of an industrial application, where the performance of the LookAhead algorithm can be compared with different coupling methods, iterative co-simulation, and other methods for improving co-simulation results.

Finally, LookAhead is planned to be implemented in the ICOS co-simulation tool developed at Virtual Vehicle Research GmbH, that is contained in the Model.CONNECT™ co-simulation platform developed by AVL List GmbH [11].

Acknowledgment

The publication was written at Virtual Vehicle Research GmbH in Graz, Austria. The authors would like to acknowledge the financial support within the COMET K2 Competence Centers for Excellent Technologies from the Austrian Federal Ministry for Climate Action (BMK), the Austrian Federal Ministry for Labour and Economy (BMAW), the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorised for the programme management.

References

[1] N. Mustafee, J. Powell, S. Brailsford, S. Diallo, J. Padilla, and A. Tolk, "Hybrid simulation studies and hybrid simulation systems: Definitions, challenges, and benefits," in *2015 Winter Simulation Conference (WSC)*, 12 2015.

[2] A. Pinto, A. Sangiovanni-Vincentelli, L. P. Carloni, and R. Passerone, "Interchange formats for hybrid systems: Review and proposal," vol. 3414, pp. 526–541, Springer Verlag, 2005.

[3] G. Schweiger, G. Engel, J.-P. Schöggl, I. Hafner, T. S. Nouidui, and C. Gomes, "Co-simulation - an empirical survey: Applications, recent developments and future challenges," *SNE Simulation Notes Europe*, vol. 30, pp. 73–76, 6 2020.

[4] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: A survey," *ACM Computing Surveys*, vol. 51, 4 2018.

[5] R. Kübler and W. Schiehlen, "Two methods of simulator coupling," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 6, pp. 113 – 93, 2000.

[6] F. Cremona, M. Lohstroh, D. Broman, E. A. Lee, M. Masin, and S. Tripakis, "Hybrid co-simulation: it's about time," *Software and Systems Modeling*, vol. 18, pp. 1655–1679, 6 2019.

[7] D. Dejaco and M. Benedikt, "A novel approach for handling discontinuities in non-iterative co-simulation," in *Proceedings of the 7th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, SIMULTECH 2017, (Setubal, PRT), p. 288–295, SCITEPRESS - Science and Technology Publications, Lda, 2017.

[8] D. Broman, L. Greenberg, E. A. Lee, M. Masin, S. Tripakis, and M. Wetter, "Requirements for hybrid cosimulation standards," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, (New York, NY, USA), p. 179–188, Association for Computing Machinery, 2015.

[9] C. Gomes, T. Blochwitz, C. Bertsch, K. Wernersson, K. Schuch, P. R., O. Kotte, I. Zacharias, M. Blesken, T. Sommer, M. Najafi, and A. Junghanns, "The fmi 3.0 standard interface for clocked and scheduled simulations," *Proceedings of 14th Modelica Conference 2021, Linköping, Sweden, September 20-24, 2021*, vol. 181, pp. 27–36, 9 2021.

[10] F. Cremona, M. Lohstroh, D. Broman, M. D. Natale, E. A. Lee, and S. Tripakis, "Step revision in hybrid co-simulation with fmi," pp. 173–183, Institute of Electrical and Electronics Engineers Inc., 12 2016.

[11] AVL, "Model.connect™, co-simulation tool for the coupled simulation of multiple models and different tools as well as the integration with real-time systems.." https://www.avl.com/en/simulation-solutions/software-offering/simulation-tools-z/modelconnect. Accessed: 2023-08-24.

 ISBN: 978-1-68558-102-2