Comparing the Predictive Event Handling Algorithm LookAhead to Rollback and Early Return

Felix Tischer[1] Simon Genser[1] Daniel Watzenig[1, 2] Martin Benedikt[1, 3]

[1]Virtual Vehicle Research GmbH, Inffeldgasse 21a, 8010 Graz, Austria,
{Felix.Tischer,Simon.Genser,Daniel.Watzenig,Martin.Benedikt}@v2c2.at
[2]Institute of Visual Computing, Graz University of Technology, Inffeldgasse 16/2, 8010 Graz, Austria,
daniel.watzenig@tugraz.at
[3]SETLabs Research GmbH, Elsenheimerstraße 55, 80687 Munich, Germany, Martin.Benedikt@setlabs.de

Abstract

LookAhead is a lightweight algorithm that improves event handling in co-simulation by predicting events and adjusting the communication step size beforehand. It operates without requiring subsystem event handling capabilities. This paper compares LookAhead with other event handling methods, namely Rollback and Early Return, from the perspective of performance and applicability. Results from the presented example show that LookAhead performs on par with iterative co-simulation methods and is particularly well-suited for handling shared state events.
Keywords: co-simulation, non-iterative co-simulation, hybrid simulation, event handling

1 Introduction

In co-simulation, the continuousness of signals is critical for performance and accuracy. The coupling error at coupling time steps can introduce discontinuities in a signal that might lead to slower computation, as a subsystem's solver algorithm might have to reset or use smaller steps. Methods like anti-aliasing can be used to smoothen input signals and improve co-simulation results. Such signals usually represent physical quantities that are continuous in nature. In other cases, a signal can actually be discrete, which can pose a particular challenge for co-simulation. This is a common challenge in the simulation of cyber-physical systems, where physical systems interact with or are being controlled by discrete control signals. A discrete change in signals, state or behaviour in a simulation model in general is called an event.
Events can be the result of different types of causes (Cellier and Kofman 2006). An event that is triggered at a certain moment in time, irrespective of the system's state (within limitations), is called a time event. An event that is directly caused by the system reaching a certain state on the other hand is called a state event.
A challenge of discrete signals unique to co-simulation are shared events, where events are being triggered in multiple subsystems by the same cause; if such an event occurs in only some subsystems or at different times, it can lead to incorrect and unphysical behaviour like violating boundary conditions or conservation of energy.
Due to the evolving sophistication of co-simulation techniques and the increasing importance of simulation and co-simulation in industrial development, proper event handling has become more and more important and has been a major focus in the development of the fmi3 standard.

2 Algorithm Description

2.1 LookAhead

For systems that lack sophisticated event handling support, LookAhead can improve the results of a co-simulation by predicting events and adapting the communication step size (Tischer et al. 2023). It does so by defining event indicators, which are sets of functions of a subsystem's inputs, outputs, and parameters. Each function z in an event indicator (called an event condition) is defined in a way such that $z < 0$ means that this condition of the event is fulfilled. An event occurs when all conditions are fulfilled, which is equivalent to when the last condition function crosses zero. Via extrapolation of these event condition functions, an event can be predicted, and the communication time step can be reduced and adapted in advance to hit the event precisely. For a more thorough description, see (Tischer et al. 2023).
As LookAhead only relies on a subsystem's interface, it can be used with any subsystem regardless of its own event handling capability and whether it is a black-box or not. On the other hand, knowledge about its events is needed to a degree that their conditions can be modelled; additionally, the event conditions also need to be able to be modelled in this manner instead of relying on internal variables.

2.2 Rollback

Rollback capability means that a system's state can be reset to a prior point in simulation, usually the previous time step. FMUs with the *canGetAndSetFMUstate* attribute are able to store their state at any point in a simulation and thus support Rollback.
To improve event handling in our use case, the subsystems store their state at every communication time step.

This save state is being stored externally (meaning, outside of the FMU by the importer) for one communication time step, when it is being overwritten by the new one if the step finished successfully. If the step fails, both FMUs in the example system (see following section) are being reset to the beginning of the step, and a new step size is chosen. In our example, if an event occurs in the middle of a communication time step, the simulation is being rolled back to the last time step and then a new communication step size is chosen so that the event occurs at the communication point.

2.3 Early Return

Early Return is an fmi3 functionality (Modelica Association 2024) where an FMU can do a simulation step and, for example when encountering an event, stop and return early from the step. This is different from Rollback in that the step was still finished successfully and does not have to be repeated. The importer has to declare to the FMU that it supports Early Return and can handle subsystems not finishing a whole step. In our example, the FMUs return early when encountering an event to allow for additional exchange of signals.
Early Return can be requested from the outside by the importer as well via callback functions, but this has not been utilised in this setup.

3 Example Setup

3.1 Example System

The example system that the algorithms are being tested on consists of two connected spring-damper-mass systems. The upper mass is connected to a fixed ceiling at height $x = 0$ while the lower mass is connected to the upper one. Both connections have the same stiffness and damping coefficients, $k = 100, d = 1$. This system was introduced in (Dejaco and Benedikt 2017) and was slightly adapted from there.
The system's state is described by four variables, the positions and velocities of both blocks, which can be described by the following system of first order differential equations:

$$\begin{aligned} \dot{x}_1 &= v_1 \\ \dot{x}_2 &= v_2 \\ \dot{v}_1 &= -\frac{2k}{m_1}x_1 - \frac{2d}{m_1}v_1 + \frac{k}{m_1}x_2 + \frac{d}{m_1}v_2 - g \\ \dot{v}_2 &= -\frac{k}{m_2}x_2 - \frac{d}{m_2}v_2 + \frac{k}{m_2}x_1 + \frac{d}{m_2}v_1 - g \end{aligned} \tag{1}$$

with $i \in 1,2$ denoting the upper and lower block and x, v their position and velocity, respectively. The half height of each block is $\Delta x = 0.55$. When splitting the system up into two subsystems, each block takes the other one's position and velocity as inputs and provides its own as outputs.
This simple, continuous-state system is being complicated by two collision events. The upper block can collide with the ceiling, and both blocks can collide with each other. The former is a private event that only affects the upper block, while the latter is a shared event between both blocks.

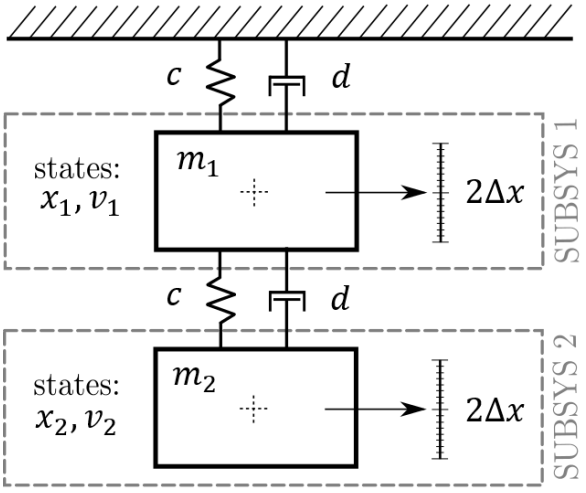

Figure 1. Schematic of the spring-damper-mass system. Figure was taken from (Dejaco and Benedikt 2017) and adapted.

The conditions for the wall block's private event, the collision with the ceiling, are

$$\begin{aligned} -v_1 &< 0 \\ -x_1 &< \Delta x. \end{aligned} \tag{2}$$

The ceiling collision is handled internally and simply by subsystem 1 by flipping its velocity:

$$v_1' = -v_1. \tag{3}$$

Primed variables denote variables after the event. The conditions for the shared event, the collision between the blocks, are

$$\begin{aligned} v_1 - v_2 &< 0 \\ x_1 - x_2 &< 2\Delta x. \end{aligned} \tag{4}$$

The binary collision is handled internally (and seperataly) in both blocks. Each block does this by using the formulas for elastic collisions to calculate the post-collision velocities. For calculating the time steps between collision and the next communication step with the other block, not only the block's own velocity is being calculated but also the other one's to prevent the possibility of false succeeding collisions.
The calculation is being done by these formulas:

$$\begin{aligned} v_1' &= \frac{m_1 - m_2}{m_1 + m_2}v_1 + \frac{2m_2}{m_1 + m_2}v_2 \\ v_2' &= \frac{m_2 - m_1}{m_1 + m_2}v_2 + \frac{2m_1}{m_1 + m_2}v_1. \end{aligned} \tag{5}$$

Even though the equations are the same in both blocks, the results can differ, as each block can only access the other's velocity as an input (that is dependent on

DOI 10.3384/ecp218663

coupling step size and method, and is therefore affected by coupling error), and not the exact state value.

Similarly to how the event prediction works in LookAhead, the FMUs use event indicators to trigger their events internally. For an event E_i, we call its event indicator $M_i(z)$, analogously to the naming conventions of the fmi standard. Then, the event E_i occurs if and only if $M_i(z) < 0$. To use the event conditions Equation 2 and Equation 4, we just define the $M_i(z)$ for each event as the minimum of its event conditions:

$$M_i(z) = min_j z_{ij}, \tag{6}$$

where z_{ij} is the j-th event condition for event E_i.

3.2 Co-simulation Setup

The subsystems have been compiled as FMUs, using the Modelica Reference FMUs as a basis. Three versions have been compiled of each subsystem: one without any additional event handling methods, one implementing Rollback, and one implementing Early Return. The simulated time is 20 seconds, the longest time in which the described system shows interesting behaviour. After that, the system reaches a somewhat stable configuration and no more events occur.
The co-simulations have been carried out in Python, using the fmpy package to import and simulate the FMUs. The Early Return FMUs are equipped with Intermediate Update (Modelica Association 2024) support, which is used to log the FMUs' states at every microstep.
In total, simulations with five different setups have been carried out.

- The first, the monolithic simulation, serves as ground truth for our analysis; this was not a co-simulation, but was in fact a simulation of the total system of four equations that the two subsystems consist of. The results have been saved with a time step of $\Delta t = 0.001s$, which is the time step of the fixed step solver integrated in the FMUs.

- The second setup is the standard co-simulation, where no event handling has taken place. This means that collisions are only being handled internally by the subsystems, and the master is only exchanging information between them without any additional handling such as inter- or extrapolation, event triggering, or time step adaptations. It has been carried out with a fixed communication step size of $\Delta t = 0.08s$, which was also the default for the following setups.

- The third setup is similar to the second, except the addition of the LookAhead algorithm turned on, evaluating the event indicators once per simulation loop and adapting the communication step size when an event is being predicted. The minimum step size for LookAhead was equal to the solver step size of $\Delta t = 0.001s$.

- The fourth setup used Rollback. The Rollback FMUs were equipped not only with the capability to reset their state, but also with two additional output signals to indicate if (called EventSensor) and when (called EventTime) an event occurred. If an event occurred at a time that was too far away from a communication point, the co-simulation algorithm would reset the last simulation step and repeat it with the event time as target time.

- The final setup utilises Early Return. The FMUs have been adapted in such a way that, when they are being initialised with $earlyReturnAllowed = True$, they return early immediately before an event occurs. For this, the event indicators $M(z_i)$ have been slightly changed by subtracting a small threshold $m = 0.05$ from their final value, but also changing the condition for triggering the event to $M(z_i) < m$. This makes it so that the event trigger remains unchanged, but the FMU returns early right before the event to allow for exchange of signals before it.

4 Analysis

4.1 Criteria

To compare and analyse the different co-simulation methods, the position error is used as a key metric. This error is calculated as the absolute difference between the position and a reference or ground truth over time, as given in Equation 7, where x_i is the position of block i in the co-simulation and $\hat{x}_i$ is its counterpart in the reference solution. By examining the magnitude and trend of this error, we can assess the accuracy and performance of each co-simulation method.
By taking the root mean square of it, we can get an average error as well. The RMS provides a single scalar value that reflects the overall magnitude of the error, smoothing out noise and highlighting consistent deviations. By evaluating the RMS error up to different points in time, we can observe how the accuracy of the simulation evolves and compare the performance between the different methods. This time-resolved analysis helps to identify when and where the simulation performs well and when it diverges.

As the performance of a hybrid system can depend heavily on how well the events are being handled, the temporal accuracy of events (especially the shared event) is also an important metric that has a direct effect on the position error. Temporal accuracy refers to how closely the timing of key events in the simulation aligns with those in the reference solution. To evaluate temporal accuracy, timestamps of the collision events are compared across simulations, and any deviations are measured as time offsets.

This is defined in Equation 8, analogously to the position error, summing up the time differences of all n events that occur during simulation. To get a meaningful result for the time error, all the events in the reference solution have to have a counterpart in the co-simulation. If the co-simulation has events that do not occur in the reference, or any events of the reference do not occur in the co-simulation (or, if it is a shared event, do so only in some of the partaking subsystems) the time error can not be calculated.

$$e_{pos} = |x_1 - \hat{x}_1| + |x_2 - \hat{x}_2| \tag{7}$$

$$e_{time} = \sum_{i}^{n} |t_i - \hat{t}_i| \tag{8}$$

4.2 Results

In Figure 2 we can see the results of the five simulations described in subsection 3.2. One can see that the shared events are the main contributor to the position error; after each collision the deviation from the reference increases.
Table 1 shows the RMS position error for each simulation co-simulation method evaluated at three different time points: 2 seconds, 10 seconds, and at stop time of 20 seconds. The standard co-simulation already has a large error at $t = 2s$ of 0.609 m after just 2 events, and increases by a factor of ten at $t = 10s$. At this point, the simulation is already far off from the reference solution.
In contrast, LookAhead and Rollback maintain consistently low errors across the entire simulation, indicating strong and stable accuracy. Finally, Early Return starts off with the lowest error at $t = 2s$ ($0.071m$), but experiences a significant jump to $2.036m$ at $t = 10s$ and a slightly lower $1.936m$ at $t = 20s$. This accumulating error leads to a diverging behaviour and indicates that the events have not been handled well enough. Overall, LookAhead and Rollback demonstrate the most stable and accurate performance over time, while Early Return is accurate at first, but diverges later in the simulation.
One can see that the Rollback and LookAhead co-simulations performed better than the standard co-simulation and the one using Early Return; we split the result analysis into these two groups.

When looking at the results for the standard and Early Return variants (Figure 3), we can see that not only the trajectories diverge, but also the event times do not match and, at a simulation time of about 12 seconds, an additional collision takes place which changes the following trajectory so much that the position error is no longer a meaningful metric. The issue with Early Return is that, unlike Rollback, it is not possible to reset a simulation, so when one subsystem returns early but the other has completed its step (or stopped at a later time during the step), it is no longer possible to reset the simulation to the point of the event. The early returned subsystem can be fed with updated inputs, but the other one not. Two possible ways to solve this problem is to either schedule the subsystems sequentially, which works when one of them tends to stop earlier, or to allow the FMUs to be stopped externally, which introduces non-deterministic behaviour (Broman 2017). If there is an untreated discrepancy in event times between the involved subsystems, an additional error is introduced, possibly leading to a bigger time discrepancy at the next shared event and explaining the increasing position error.

On the other hand, the Rollback and LookAhead methods (Figure 4) performed much more accurate, with no additional events occurring. From a first impression, one might assume that Rollback and Early Return perform the same; in both cases, the simulation introduces an additional communication step at the event time, and calculating exactly the same outside of any events. The difference is that, as Rollback is an iterative co-simulation technique, the system has more accurate information during subsequent repeated calculation steps, resulting in reduced error — independent of how the event is handled. Unlike Early Return, both subsystems in Rollback calculate to the same event time as their communication step is governed by the co-simulation algorithm, preventing any time discrepancy errors.
LookAhead handles shared events as well as Rollback does, even though it is non-iterative. It accomplishes this by reducing the communication step size just before the event, introducing additional communication steps. Since collisions significantly contribute to position error, the co-simulation improves accuracy by increasing signal exchange when it matters most, while maintaining larger step sizes elsewhere to optimize performance.

Table 1. Root mean square position error at different times.

Variant	RMS@2s	RMS@10s	RMS@end
Standard	0.609	6.060	4.578
LookAhead	0.146	0.100	0.132
Rollback	0.139	0.178	0.159
Early Return	0.071	2.036	1.936

5 Summary, Conclusion and Future Work

We have introduced the LookAhead algorithm, a lightweight event handling algorithm that works by predicting upcoming state events and reducing the communication step size, and compared it to two other event handling methods, the iterative Rollback mechanism and fmi3's Early Return functionality. Based on an example system that has proven itself to be a hard task to simulate correctly due to the shared events having such a large influence on the error, we compared each method's

DOI 10.3384/ecp218663

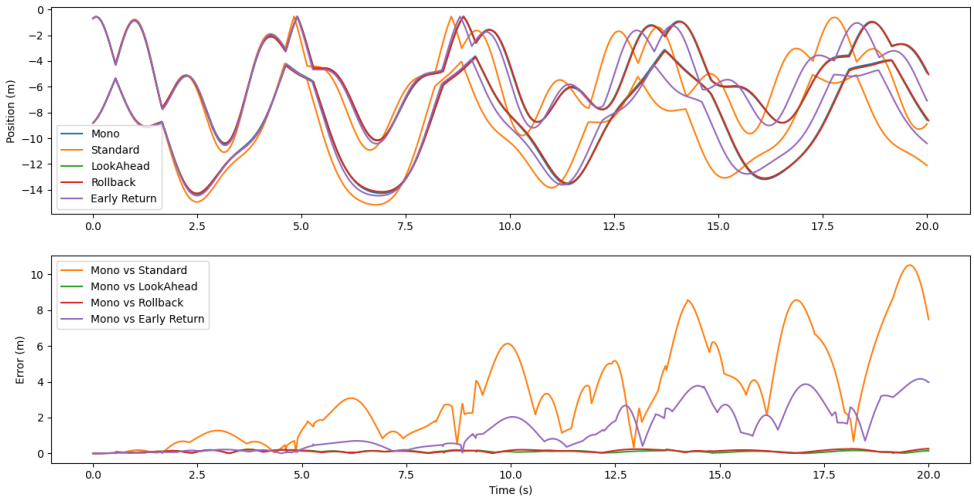

Figure 2. An overview over all 5 co-simulation results. The upper panel shows the trajectories of both blocks, while the lower panel shows the position error e_{pos} as calculated in Equation 7.

performance to the monolithic reference solution, a co-simulation algorithm without any event handling, and to each other. All three methods have their strengths, each one having the high ground in different systems.

Rollback can significantly enhance the accuracy of a co-simulation with comparatively small programming effort. This is achieved by enabling a subsystem to access and restore its internal state, and by providing a mechanism to notify the importer when such operations should occur. (We assume here that the importer supports this functionality for all three methods.) However, the main drawback lies in the potential computational cost: for complex models, saving and restoring system states can be computationally expensive. If states are stored thousands of times for an event that occurs only a few times, the overhead may outweigh the benefits. This limitation can be mitigated by storing the system state less frequently and, if a reset is required, reverting to an earlier point in time. Furthermore, if not all subsystems support Rollback, sequential coupling must be employed. In this case, the Rollback-capable subsystems are executed first, while the remaining components proceed only after a successful communication step has been completed.

Early Return has its strengths in handling private events. It not only enables recording the exact time and state at which an event occurs, but also allows this information to be propagated to other system components, thereby improving overall accuracy. To maximise this advantage, similarly to Rollback, Early Return FMUs can be calculated first in a sequential coupling scheme. The problem that we encountered in this setup is that two components utilising Early Return can give contradicting information on when an event takes place, leading to time discrepancies. This problem arises only under specific circumstances, but can lead to uncertain credibility in systems with events in more than one subsystem. Another big advantage of Early Return is its versatility in how it can be used, as aspects such as the ability to trigger Early Return from the outside (Modelica Association 2024) have not been explored in this setup.

LookAhead operates at the importer level, managing the communication step size of the entire co-simulation. While this global approach may be less optimal for a single component, it is unaffected by multiple concurrent events. When LookAhead predicts an event, it sets the next communication point and thus ensures that all subsystems are synchronized. This method remains effective regardless of the number of events or event sources, making it robust in managing both shared and simultaneous events. However, its main limitation lies in the types of events it can handle. Events must be predictable from the component's interface (its inputs and outputs) which can be challenging for events that occur deep in complex subsystems. Even though LookAhead can be applied to any black-box subsystem, knowledge about the system's events is necessary to implement its event detection. Additionally, the event must be reliably predictable: sudden state events triggered by rapidly evolving conditions can be difficult to anticipate and may indicate the need to reduce the communication step size of the continuous portion of the model.

Future work aims to improve usability by supporting internal event indicators and devising a method to automatically associate signals with events. On a technical level, LookAhead is being expanded to work with other coupling schemes than the standard parallel coupling, such as sequential coupling and parallel or sequential coupling for different communication step sizes. LookAhead is be-

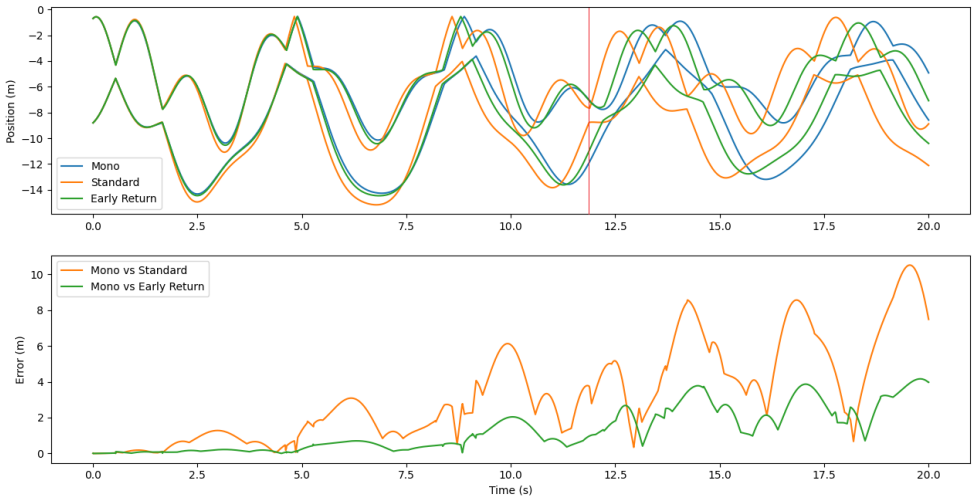

Figure 3. Comparison of the standard and Early Return co-simulation results. Although both systems diverge from the monolithic reference solution, Early Return keeps the position error much lower.

ing implemented in the co-simulation tool ICOS, developed at Virtual Vehicle Research GmbH and included in the co-simulation platform Model.CONNECT™by AVL List GmbH (AVL 2023).

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 by the Austrian Federal Ministry for Innovation, Mobility and Infrastructure (BMIMI), Austrian Federal Ministry for Economy, Energy and Tourism (BMWET), 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

AVL (2023). *Model.CONNECT™, Co-simulation tool for the coupled simulation of multiple models and different tools as well as the integration with real-time systems.* URL: https://www.avl.com/en/simulation-solutions/software-offering/simulation-tools-z/modelconnect (visited on 2023-08-24).

Broman, David (2017). *Hybrid Simulation Safety: Limbos and Zero Crossings*. arXiv: 1710.06516 [cs.CY]. URL: https://arxiv.org/abs/1710.06516.

Cellier, François E and Ernesto Kofman (2006). *Continuous system simulation*. Springer Science & Business Media.

Dejaco, Daniela and Martin Benedikt (2017). "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. Madrid, Spain: SCITEPRESS - Science and Technology Publications, Lda, pp. 288–295. ISBN: 9789897582653. DOI: 10.5220/0006440402880295. URL: https://doi.org/10.5220/0006440402880295.

Modelica Association (2024). *Functional Mock-up Interface Specification*. URL: https://fmi-standard.org/docs/3.0.2/ (visited on 2025-04-10).

Tischer, Felix et al. (2023). "LookAhead - A New Approach for Event Handling in Co-Simulation by Predicting State Events". In: *SIMUL 2023 Proceedings*. SIMUL 2023. Valencia, Spain.

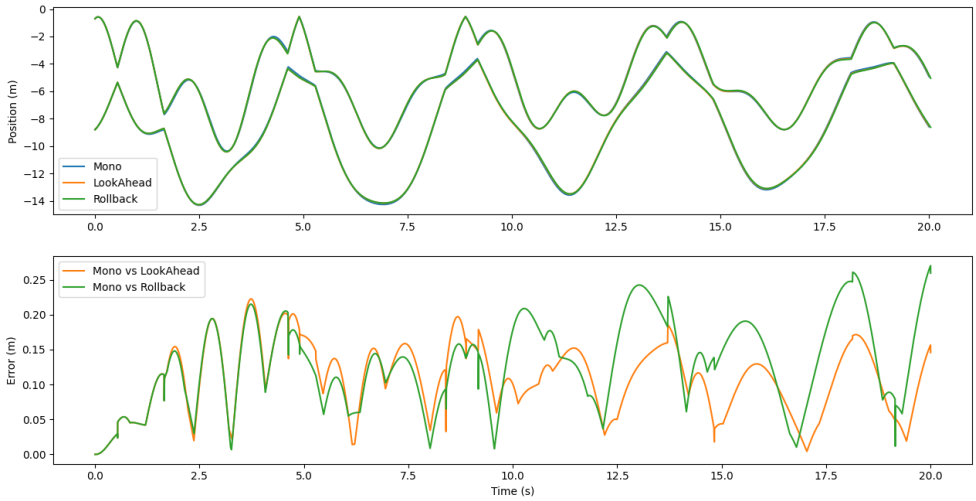

Figure 4. Comparison of the Rollback and LookAhead co-simulation results. They follow the trajectory of the reference solution very closely and keep the error at a constant magnitude. Note that the scale of the error graph is much smaller than in Figure 2 and Figure 3.