2022 10th Workshop on Modelling and Simulation of Cyber-Physical Energy Systems (MSCPES) | 978-1-6654-6865-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/MSCPES55116.2022.9770106

Qualification of Initialisation Challenges in Co-Simulation setups for Integrated Energy Systems

Arjen A. van der Meer
Faculty of Electrical Engineering,
Mathematics and Computer Science
Delft University of Technology
Delft, the Netherlands
Email: a.a.vandermeer@tudelft.nl

Jan Sören Schwarz
R&D Division Energy
OFFIS
Institute for Information Technology
Oldenburg, Germany
Email: schwarz@offis.de

Kai Heussen
Department of Electrical Engineering
Technical University of Denmark
Kgs. Lyngby, Denmark
Email: kh@elektro.dtu.dk

***Abstract*—Co-simulation is an important tool to capture the complexity of cyber-physical energy systems. The past decade lead us from command-line simulation orchestration to higher readiness-levels in terms of applicability. The configuration and time-domain initialisation of generic co-simulation setups, however, entail a lot of manual activities, especially when simulation components are tightly coupled. This paper provides a qualitative overview on what initialisation challenges crop up and how tools like MOSAIK can tackle these. This is illustrated with a multi-domain co-simulation example, in which the same time loop concept is applied to resolve cyclic dependencies between simulators during initialisation. The paper provides conclusions about the applicability of same time loops for initialisation purposes and will provide directions for further research on this topic.**

I. Introduction

The transition to a carbon-neutral energy system is complex: multiple stakeholders need for instance to draw up governance structures, design operational circumstances, and agree on technology often under uncertain systemic conditions. An energy system, moreover, that will also become more versatile due to the manifold interfaces to other energy carriers (i.e., multi-vector systems) and interfaces to external ICT systems (i.e., digitisation). This calls for interdisciplinary planning and design cycles, of which numerical modelling and simulation are integral parts.

Co-simulations enable us to assess systemic performance indicators by splitting the system under test into subsystems that span one particular engineering domain. These specialised modes are then coupled to each other and exchange information through numerical interfaces during runtime. This concept adheres to typical multidisciplinary requirements such as transparency, model scalability, and design optimisation while leakage of intellectual property is prevented. Applicability, user experience, and numerical performance are considered instrumental to fully utilising the advantages of co-simulations. The current practice, using general purpose tools or employing domain-specific simulations with highly simplified models of parts of the system under test, boast excellent applicability and UX but commonly lack scalability and performance. Key to further rollout of co-simulations is hence the standardisation of typical co-simulation aspects such as numerical interfaces, scheduling algorithms, semantics (model causality, model compatibility), and specification languages.

One particular challenge that needs further consideration is the workflow for configuring, initialising, and running the experimental setup. The goal of this paper is to discuss the issues and to perform a gap analysis around such *initialisation* aspects of co-simulations. Alongside this gap analysis we will introduce how specialised co-simulation tools, notably MOSAIK, are positioned. This is illustrated by a co-simulation experiment on a recently developed multi-energy benchmark system in the H2020 ERIGrid 2.0 project.

The remainder of this paper is organised as follows. We will first introduce and qualify the configuration and initialisation challenges, The paper continues with the current state of art on generic specification of co-simulation setups and scheduling algorithms while concentrating on MOSAIK 3.0. The challenges are illustrated by an example multi-energy case study using the same time loop (STL) approach. The paper ends with conclusions and next steps towards co-simulations as-a-service.

II. Initialization Challenges

Initialisation in co-simulation aims to achieve a coherent initial state across the sub-models. To structure the discussion, we suggest two viewpoints of initialisation: A. the topological conditions for the computation of a coherent initial state (due to model of computation and sub-model interactions) and B. the numerical conditions and processes for achieving coherent starting values.

A. Topological setup and configuration of co-simulations

Co-simulation requires the definition of exchange signals, i.e. the relevant state information shared across simulators. This necessarily implies a directional signal exchange, and thus a sequencing of computation between simulators. Depending on the model of computation (discrete event/continuous time) this ordering can be problematic, and have various different consequences, including issues such as: computational inaccuracy, decreasing convergence speed, and timing inaccuracies.

For dynamic feedback systems, where the causal loops are explicitly modelled by signals with integrators, the signal ordering is straightforward. However, physical continuous domain simulation models often include acausal links/coupling points (electrical lines are a prime example), which are solved by domain-specific solvers (e.g. iterative load flow calculation). If a physical system is distributed over separate simulators in a co-simulation setup, the required bidirectional exchange signals lead to one or more algebraic loops across simulators, also referred to as *cyclic dependency*. Algebraic loops need to be resolved by assigning directionality and a computational ordering, which is typically done manually.

In continuous systems, the topological analysis identifies sub-models (equations) of *ordinary (ODE) and differential-algebraic (DAE), explicit algebraic (A), or iterative/implicit algebraic (IA)* nature. In discrete event systems, the ordering is determined by message directionality and time.

To effectively split systems across simulators, acausal exchange variables therefore first need to be identified and, if possible, avoided. If they cannot be avoided, the challenge is to assign a directionality to unavoidable acausal coupling points (e.g., read voltage, write current). Such assignment creates an algebraic loop across simulators, which needs to be handled by the co-simulation orchestrator, as it can affect simulation accuracy or speed significantly. Among other, such cyclic dependencies also lead to initialisation issues, as discussed below.

So, where possible, coupling points are selected where inaccuracies will have minimum impact on the simulation stability, accuracy and performance. Selecting the timing and direction of exchange signals is therefore a key step in designing the co-simulation topology.

B. Domain-specific and Co-simulation Initialisation Approaches

Power system simulations, especially those focusing on quasi-stationary behaviour (RMS, stability), have a research and development stemming from eras in which sequential computation was the standard. These simulations revolve around solving DAEs, in which the differential equations commonly represent the dynamic behaviour of rotating machinery and controls, whereas the algebraic equations encompass the network model and discrete behaviour of connected assets.

Two main solution methods exist here: partitioned and simultaneous methods. The partitioned approach solves differential, model, and network equations sequentially each time step, thereby assuming that the effect of algebraic loops is negligible and can be solved by predictor-corrector iterations. The simultaneous approach aims at combining all network and asset models into one set of equations that is solved by Newton-Raphson iterations each time step. Partitioned methods are applied in most classic powers system simulators for limited memory demand and modularity. Simultaneous methods are more accurate, especially when cyclic dependencies between differential equations and algebraic equations exist (e.g., excitation systems, voltage controllers of power electronic devices), at the cost of a higher computational burden (i.e., Jacobian determination, NR-iterations, storage).

Initialisation is commonly achieved by solving the network equations for a particular scenario (generation, line loading, load). Subsequently, the asset models need to be initialised, which is where tools diverge. Three main approaches exist:

Manual initialisation of models: A common approach is to make the modeller responsible for appropriate initialisation of the asset models. This is common in tools like PSSE and PowerFactory. Both tools use a model framework to define the initialisation order of contained models. PSSE comes with a fixed set of common frameworks that cover most domain-specific setups (e.g., generators, relaying, multi-terminal connections). In PowerFactory, this *composite framework* and hence the initialisation order can be defined by the modeller as well.

Quasi-automatic initialisation: Especially in equation-based tools, the modeller is allowed to specify causal loops inside dynamic models. At initialisation time, these loops iteratively initialise the encapsulated states and algebraic variables. Tools like PSS Netomac/Sincal fall in this category.

Time-based initialisation: Sometimes, the relation between network quantities and model states is not one-to-one. This is for instance the case when 1) different power electronic switch configurations can lead to the same voltage modulation, 2) when discrete states are unknown, 3) for real-time setups. A common approach is then to make an estimation, usually manually, of the algebraic and state variables and let initial phenomena damp out accordingly. Electro-magnetic transients type simulators fall in this category.

The approach common to domain-specific (electrotechnical) simulations is similar for co-simulations. As their character is to construct a modular systemic model of domain-specific models, it is hard to resolve all cyclic dependencies with a simultaneous approach. Hence, sequential methods are most common in scheduling algorithms. MOSAIK version 2, in this respect, is a discrete-time orchestrator that uses `time_shifted` (i.e., delay of one time step) links to resolve causal loops between co-simulation modules. These time shifts need to be specified at configuration time; effectively, this resolution method has the side-effect of numerical convergence effects become visible in the time-domain simulation traces. Further, similar to the partitioned approaches in power system simulations, this can lead to numerical oscillations if such modules are stiffly connected (similar time-constants, algebraic coupling)–an assumption that must be constantly checked by the co-simulation engineer.

III. How can current co-simulation tools help addressing initialisation challenges

This sections aims to present how current tools help to address the challenges identified regarding the configuration (Section II-A) and scheduling (Section II-B), thereby focusing on the most relevant tools for co-simulation MOSAIK [1], Ptolemy [2], FMI [3], HLA [4], and HELICS [5]. A compar-

features \ tools	MOSAIK 2	MOSAIK 3	Ptolemy 2	FMI 2	FMI 3	HLA	HELICS
scenario data model	x	x	x	-	-	x	x
scriptable scenario	x	x	-	-	-	-	x
discrete event	-	x	x	-	x	x	x
discrete time	x	x	(x)	x	x	x	x
continuous time	-	-	(x)	x	x	x	-
superdense time	-	x	x	-	(x)	-	x
roll-back	-	-	x	-	-	(x)	-

TABLE I: Features available in different tools as relevant to co-simulations. x: available, (x): limited/indirectly available, - not available)

ison of a larger number of co-simulation tools can be found in [6].

Relevant features in these tools are compared in Table I. Regarding the configuration of co-simulation the *scenario data model* and *scriptable scenario* are relevant, which describe capabilities for the specification and automation of a co-simulation scenario. For the execution scheduling, the available time paradigms are important: Namely, *discrete event*, *discrete time*, and *continuous time*. Here, discrete time is the simplest orchestration type for scheduling. Discrete event systems (DEVS) are the most general, as all kinds of simulators generalise to DEVS; continuous time is the most narrow (ODE or DAE) type, but one where specialisation of the interface bears speedup potential (i.e. due to predictive scheduling). Especially for addressing initialisation issues, the more advanced scheduling concept *superdense time* and simulation *roll-back* are of interest.

A. Scenario Configuration Mechanisms

One important aspect to handle the previously introduced initialisation challenges is the scenario management, which provides mechanisms for the configuration of simulation scenarios. The scenario management relies on a data model for the configuration, containing the simulation components, their parametrisation, the definition of data flows between them, and their temporal behaviour. It is the basis for execution of co-simulation and should assist the user in tackling applicability challenges. To find a suitable way of splitting up the system into sub-systems (Section II-A), additional information about the simulation components and scenarios is needed.

For the description of experiments the Holistic Testing Description (HTD) has been developed, which allows a template-based structured process for the planning of experiments in hardware or software [7]. In this process, e.g., test cases, system under test, and initial system states have to be defined to make this information transparent available in a structured way and to use it for the development of a co-simulation experiment.

Also the integration of static Smart Grid Architecture Model (SGAM) and dynamic co-simulation can be an option for improving scenario configuration [8]. An approach for the translation from a Smart Grid system architectures into an executable MOSAIK simulation shows how a detailed description of the system can be utilized for automated simulation configuration [9]. Another approach for the integration of HTD and SGAM into co-simulation with MOSAIK is described in [10]. One part of this approach is also an assisted simulation planning process based on an information model and a catalogue of available co-simulation components, which makes metadata of the components available for choosing and coupling the most suitable ones [11].

For the development of a executable co-simulation scenario different ways exists, like using a GUI, specification files, or scripts. A scriptable configuration allows a flexible definition, which can be helpful especially when the scenario is based on generic topologies and the scenario definition is supposed to be done automatically. In this regards, MOSAIK provides a scenario API for the scenario definition in Python. HELICS boasts similar functionality and allows federate configuration via a web interface.

For MOSAIK, approaches towards automatic simulation configuration are under research. More specifically, the development focuses on a first prototype implementations that allows MOSAIK to assist the user in finding the best co-simulation topology to avoid initialisation issues and tackle also more complicated initialisation issues–notably causal loops crossing multiple subsystems.

B. Co-simulation Scheduling tools

A key task of a co-simulation framework (also: orchestrator, master algorithm) is to execute the different simulation components at the right time, i.e. schedule the simulation. The scheduling can be based on different time paradigms, like discrete time, discrete event, or continuous time [12]. The co-simulation master algorithm and interface specification can offer features to accelerate co-simulation performance or improve simulation accuracy, such as *roll-back*, *lookahead*, or *superdense time*. For this scheduling, cyclic dependencies can be problematic as described in Section II-B. MOSAIK does not allow direct cyclic dependencies, because it would not be clear for the scheduler which connection would be the first to take as an input for a specific simulator (i.e., component of the co-simulation). As a solution, MOSAIK 2 provides asynchronous requests and the `time_shifted` parameter for connections allowing to define which connections of the cycle are executed first. To break the cycle, the output data of the first simulation component will not be available to the dependent component until the next simulation time step. This sequential option

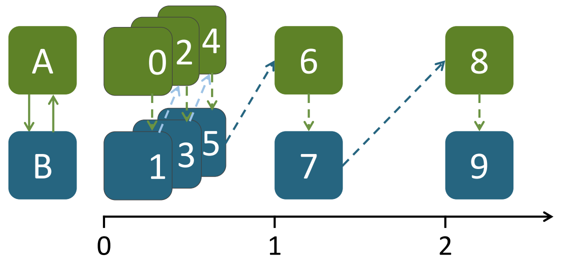


Fig. 1: Visualization of superdense time (i.e., same time loops) in MOSAIK [13].

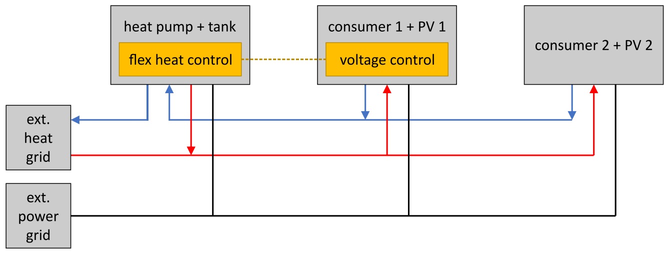


Fig. 2: Overview of the system configuration in *"Benchmark 2 – Multi-Energy Networks"* scenario [17].

is commonly used in power system simulations and works adequately for small perturbations but is known to cause issues when severe system events occur [14]. This solution also implies that for the time-shifted connections no value would be available for the first simulation time step. As a result, the usage of the `time_shifted` parameter explicitly requires an initialisation value for the shifted attributes. The time shifting of values can be problematic especially in the beginning of a simulation, because oscillation of values may arise until (time-domain based) initialisation of all simulation components is finished, which comes with a huge computational penalty.

A more profound approach for solving the problem of cyclic dependencies is the concept of superdense time, which is a common concept in simulation tools [12], [15] and was introduced in MOSAIK 3.0 together with a new event-based scheduling [16]. Superdense time allows a simulation to do multiple simulation steps without incrementing the simulation time as visualised in Figure 1.

With superdense time, it is possible to implement a initialisation phase between the different co-simulation components for the start of simulation to find an accurate initial state. This is related to the challenge of starting values (Section II-B) as it can help to find a stable state of the models. As the most inputs of the models have to be set somehow at the beginning of the simulation, unstable behaviour can occur. Thus, the first steps of a simulation are often used only to find a stable state of the system and the results are thrown away.

IV. Illustration of Co-Simulation Initialisation by a Case Study

As explained in the previous section, superdense time is a promising concept for the identified initialisation challenges and was applied in a case study to show how new MOSAIK 3.0 features can be used to address these. This case study is based on a benchmark scenario developed in the ERIGrid 2.0 project [17].

A. General description of benchmark scenario

The *"Benchmark 2 – Multi-Energy Networks"* scenario describes a typical multi-energy setup with an low-voltage (LV) electricity grid and a district heating network [17]. As shown in Figure 2, the system additionally contains a heat pump with a tank and a controller, and two aggregated consumer households with PV systems. The optional voltage controller observes the voltage and can adjust the heat pumps power limit in times of voltage violations. In this way, the scenario provides an example of *sector coupling* of electricity and heat. The benchmark simulator components are all written as discrete-time simulators, but represent various model types: explicit algebraic (A, e.g. heat pump) and implicit algebraic (IA, e.g. power flow) models, continuous dynamic (ODE, e.g. hot water tank), and discrete event type (DE, e.g. voltage controller).

The interdependence of these two sectors, which each consist of multiple components, leads to initialisation issues when studied using co-simulation. As there are many parameters and system states, which have to be defined for the start of the simulation run, it can not be guaranteed that the system can start in a steady state. Thus, in the original version of the benchmark scenario the first day of simulation is just done for the initialisation purposes (time-based initialisation) and the results are thrown away. This case study investigates possible applications of superdense time for initialisation, to avoid or reduce this time consuming "warm-up" period that needs to be chopped off from the results anyway.

B. Implementation of superdense time for initialisation

The benchmark scenario is available as a MOSAIK scenario and in Figure 3 the data flows in this co-simulation are shown. The boxes represent the simulation components in MOSAIK, and the two figures represent two variations of the simulation scenario to evaluate the effects of assigning different subsets. Blue arrows represent direct and the orange arrows time shifted data flows in the original scenario. The simulation components coupled in a same time loop (STL) using superdense time are highlighted in blue and the execution order is indicated by the number in their top right corner. In the upper-left corner, the respective model type and simulator mode are indicated. It can be seen that multiple cyclic dependencies exists, as there are many time shifted data flows (highlighted in orange). For all of these time-shifted connections, initial values need to be provided in the scenario as there is no simulation data available for the first step.

To improve this initialisation process of this scenario, the simulation components were extended to support an initialisation phase based on superdense time. As indicated by the green arrow circles, the scenario contains many options for im-

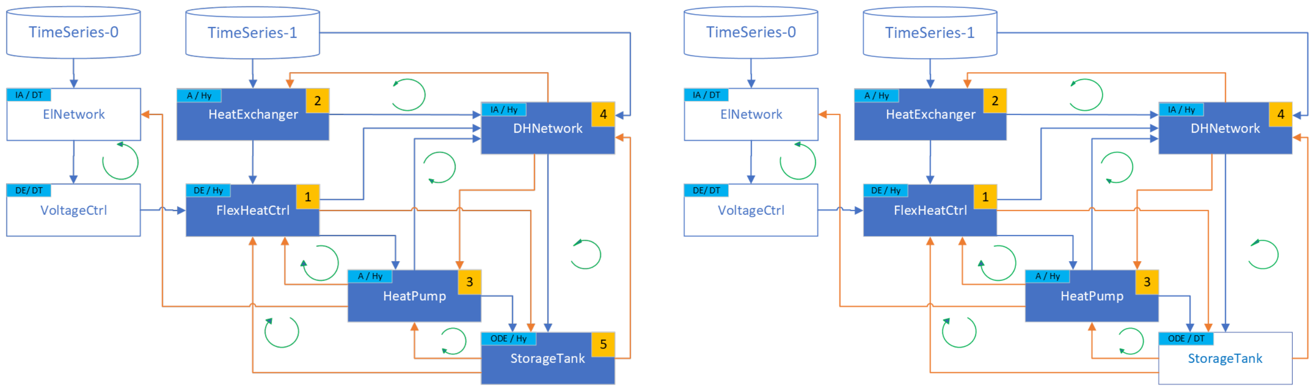


Fig. 3: Visualisation of MOSAIK simulation scenario with two variants of the same time loops (STL). The orange numbers identify the ordering within the STL; the light blue boxes indicate model type (IA,A,ODE,DE) and MOSAIK simulator type (DT,Hy).

plementing STL with subsets of components. As starting point we implemented STL with combinations of two components, but due to the interdependence the system state was still not stable. Thus, all components of the heat domain were added to the STL. The heat domain was chosen as it contains many systems affecting each other with direct mass hence energy flows.

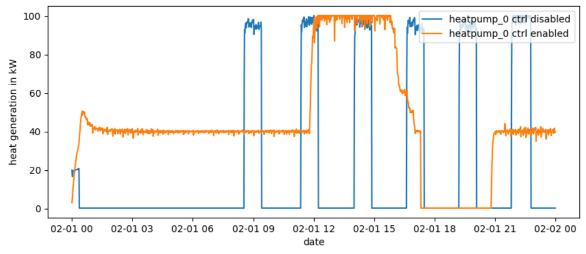
For the implementation, the simulators had to be slightly adapted to be compatible with the MOSAIK 3 API and super-dense time. To activate (and also stay in) a STL, the simulator has to provide its 'cyclic' attribute(s) via the `get_data` function indicating as output time the current step time as long as the STL should stay active. Thus, an additional attribute was added to all simulators indicating if the initialisation is still in progress or the simulation can proceed. The control of this was added in the domestic heating network and the flex heat controller. For this control function, the values of all relevant attributes in the STL are cached and checked for convergence. As the STL poses a cyclic dependency, which can not be resolved by MOSAIK automatically, one connection has to be set as `weak`, to indicate that it will be the last one to be executed. Additionally, the type of the simulation components was changed to the new `hybrid` type [16]. This was chosen because some connections can still use the `time-based` paradigm, and by adding the relevant attributes as `trigger`, the new MOSAIK event-based features were available.

C. Simulation results

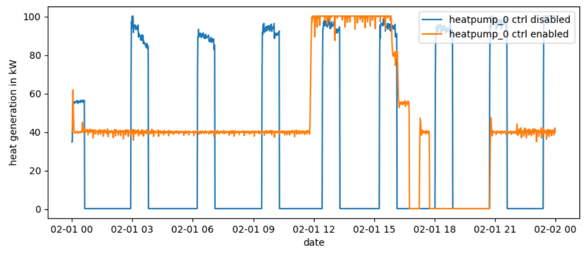
The results of a simulation of one day are described in the following. In Figure 4 the power consumption of the heat pump is depicted. Comparing the both version with and without the STL, it can be seen that with the enabled voltage controller there was just a slight effect erasing the swing in the beginning and fining earlier a stable state. The results with the disabled voltage controller show that the normal cycle of 'on' and 'off' times of the heat pump is reached earlier.

Figure 5 shows the temperature in the tank with disabled voltage control. It can be seen, that without the STL the initial temperature is quite high and equivalent in all layers of the tank. Due to the STL, the layers have directly different temperatures and the starting value is already lower, which leads to an earlier stable state with the normal heat pump cycle as shown in Figure 4.

The execution time of the case study with STL was compared to the execution time without. Though absolute execution times are relatively high (1 to 10 seconds), which is owing to the scripting nature of Python, it is more significant to see what the relative computational burden of the STL is. For the one-day simulation this was 19.85 - 28.68% for voltage control disabled and enabled respectively. For the week simulation time, this relative difference was only 3.68 - 5.13 % for the voltage control enabled/disabled respectively.

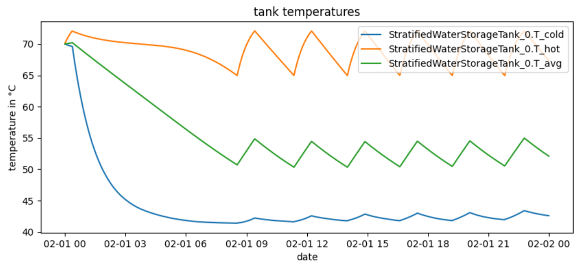


(a) Without STL.

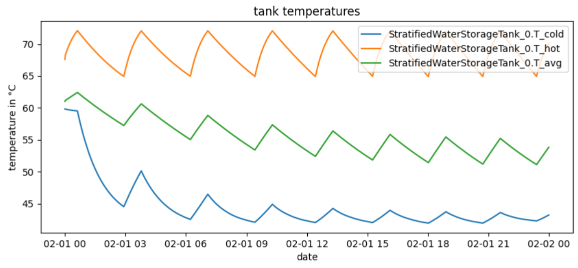


(b) With STL.

Fig. 4: Heat pump power consumption (one day simulation)



(a) Without STL.



(b) With STL.

Fig. 5: Tank temperatures (one day simulation).

V. Conclusion

This paper highlighted a couple of challenges related to the configuration and initialisation procedure of co-simulations. It was discussed how cyclic dependencies between co-simulation components complicate the execution of domain-specific simulations (e.g., algebraic loops) and co-simulations (causality of signals available to the orchestrator). Subsequently, the features of MOSAIK to resolve some of these challenges have been discussed and illustrated with a multi-domain co-simulation scenario (power & heat).

Based on the qualitative analysis of tool features, it can be concluded that the applicability of co-simulations can profit hugely from automated scenario generation (at configuration time) and automatic detection of cyclic dependencies. In case of MOSAIK, same time loops enable the initialisation of complex co-simulation setup in a systemic way. Next steps in this respect include a decision-making algorithm in the co-simulation configuration phase that intelligently designates the same time loop simulators to make optimally use of this functionality.

The simulation results show a more optimal initial state of the simulated system when same time loops were applied in MOSAIK. The computational penalty introduced by the application of superdense time is assumed acceptable for the accuracy gains expected in more complex scenarios. That is, the additional computational burden of superdense time is comparable to the gains in terms of time-domain based initialisation overhead. The scalability of this approach (system complexity and size) and further optimisation in terms of computation burden are topics for further research.

Acknowledgment

This work is supported by the European Community's Horizon 2020 Program (H2020/2020-2024) under project "ERIGrid 2.0" (Grant Agreement No. 870620). Further information is available at the corresponding website www.erigrid2.eu.

References

[1] C. Steinbrink, M. Blank-Babazadeh, A. El-Ama, S. Holly, B. Lüers, M. Nebel-Wenner, R. Ramírez Acosta, T. Raub, J. S. Schwarz, S. Stark, A. Nieße, and S. Lehnhoff, "CPES Testing with mosaik: Co-Simulation Planning, Execution and Analysis," *Applied Sciences*, vol. 9, no. 5, p. 923, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/5/923

[2] J. Davis, II, M. Goel, C. Hylands, B. Kienhuis, P. I. Edward A. Lee, J. Liu, X. Liu, L. Muliadi, S. Neuendorffer, J. Reekie, N. Smyth, J. Tsay, and Y. Xiong, "Overview of the Prolemy Project," University of California, Berkeley, California, Tech. Rep. 4, 2014.

[3] Modelica Association Project "FMI", "Functional Mock-up Interface for Model Exchange and Co-Simulation," Tech. Rep., 2021. [Online]. Available: https://github.com/modelica/fmi-standard/releases/download/v2.0.3/FMI-Specification-2.0.3.pdf

[4] "Department of Defense High Level Architecture," in *Winter Simulation Conference Proceedings*, 1997, pp. 142–149.

[5] B. Palmintier, D. Krishnamurthy, P. Top, S. Smith, J. Daily, and J. Fuller, "Design of the HELICS High-Performance Transmission-Distribution-Communication-Market Co-Simulation Framework," in *2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, no. September, 2017, pp. 1—-6.

[6] M. Vogt, F. Marten, and M. Braun, "A survey and statistical analysis of smart grid co-simulations," *Applied Energy*, vol. 222, no. March, pp. 67–78, 2018.

[7] K. Heussen, C. Steinbrink, I. F. Abdulhadi, V. H. Nguyen, M. Z. Degefa, J. Merino, T. V. Jensen, H. Guo, O. Gehrke, D. Esteban, M. Bondy, D. Babazadeh, F. P. Andrén, and T. I. Strasser, "ERIGrid Holistic Test Description for Validating Cyber-Physical Energy Systems," *Energies*, vol. 12, no. 14, pp. 1–31, 2019. [Online]. Available: https://www.mdpi.com/1996-1073/12/14/2722

[8] "CEN-CENELEC-ETSI smart grid coordination group smart grid reference architecture," Nov. 2012.

[9] C. Binder, M. Fischinger, L. Altenhuber, D. Draxler, G. Lastro, and C. Neureiter, "Enabling architecture based Co-Simulation of complex Smart Grid applications," *Energy Informatics*, vol. 2, no. 1, 2019.

[10] R. Elshinawy, R. P. Ramirez, J. S. Schwarz, and S. Lehnhoff, "Structured Planning of Hardware and Software Co-simulation Testing of Smart Grids," in *Proceedings ofthe 10th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, no. SIMULTECH 2020, 2020, pp. 197–208.

[11] J. S. Schwarz, C. Steinbrink, and S. Lehnhoff, "Towards an Assisted Simulation Planning for Co-Simulation of Cyber-Physical Energy Systems," in *7th Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, Montreal, 2019, pp. 1–6.

[12] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: State of the art," 2017. [Online]. Available: http://arxiv.org/abs/1702.00686

[13] T. Raub, J. S. Schwarz, A. Ofenloch, and T. Brandt, "Mosaik Scheduling Figures," 11 2021. [Online]. Available: https://figshare.com/articles/figure/Mosaik_Scheduling_Figures/16988437

[14] A. A. van der Meer, R. L. Hendriks, M. Gibescu, J. A. Ferreira, M. A. M. M. van der Meijden, and W. L. Kling, "Combined simulation method for improved performance in grid integration studies including multi-terminal vsc-hvdc," in *Proc. Renewable Power Generation conference*, 2011.

[15] 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, no. 3, pp. 1655–1679, 2017.

[16] A. Ofenloch, J. S. Schwarz, D. Tolk, T. Brandt, R. Eilers, R. Ramirez, T. Raub, and S. Lehnhoff, "MOSAIK 3.0: Combining Time-Stepped and Discrete Event Simulation," in *1st International workshop on "Open Source Modelling and Simulation of Energy Systems"*, 04 2022, in press.

[17] AIT CES and E. Widl, "ERIGrid2/benchmark-model-multi-energy-networks: v1.0," Nov. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5735005