Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling.

Peter Palensky, *Senior Member, IEEE, IES,* Arjen A. van der Meer, *Member, IEEE, IES,* Claudio David López, *Member, IEEE,* Arun Joseph, *Student Member, IEEE, IES,* Kaikai Pan, *Student Member, IEEE*

Abstract

Smart Grids link various types of energy technologies such as power electronics, machines, grids, and markets via communication technology, which leads to trans-disciplinary, multi-domain system. Simulation packages for assessing system integration of components typically cover only one sub-domain, while terribly simplifying the others. Co-simulation overcomes this by coupling sub-domain models that are described and solved within their native environments, using specialized solvers and validated libraries. This article discusses the state of the art and conceptually describes the main challenges for simulating intelligent power systems. Part 1 covers fundamental concepts, part 2 applications.

Index Terms

smart grids, co-simulation, power system simulation, HIL, intelligent electrical power system.

I. Introduction

Simulation is fundamental in power engineering because of its merits in the assessment of features such as controllability, reliability and general operability of devices and the power system as a whole; the need to conduct costly and time-consuming laboratory or field experiments is thereby avoided. It helps in predicting the behavior of the power system before an actual contingency (e.g., converter outage, load rejection, line overloading) occurs and to study the effects of necessary control actions to avoid these. Simulation based studies in the traditional power system include a wide range of planning and operational situations such as long term generation and transmission expansion planning, short term operational simulations, and market analysis.

With the advent of wide-spread ICT-based infratructure and power electronics into power systems an important gap is bridged between technologies (e.g., ICT, smart sensing, power electronic equipment, renewable energy sources), disciplines (e.g., consumers and producers inside the liberalised electricity market environment) and domains (e.g., electrical, thermal, telecomunication, storage of energy, market parties). Industrial Electronics is the enabling discipline: Its power electronics, intelligent and distributed algorithms, and automation technology transform the power system from a static, dedicated machine into a flexible, agile platform, where functions are software-defined and where applications are much more complex than just pumping energy from A to B. Enhancing the power system with these Industrial Electronics leads to *cyber-physical* or *intelligent* power systems [1].

Analysis method	Pros	Cons
Real-world experiments	Reliable results, no need to validate models	Only real-time behavior, only for small systems, potentially expensive
Map all system behavior into one modeling domain	Simple software structure	Simplifications and potentially lossy and inaccurate translation for foreign models
Use a universal modeling language (multi-domain)	Flexibility, ease of use	Bad scalability for systems
Couple a heterogeneous set of sub-models	Well-suited tools and languages	Complex software coupling

TABLE I: Alternatives for analyzing intelligent, integrated power systems.

The increasing cyber-physical nature of the power system requires rethinking about the fashion in which its behavior is addressed within planning and operation studies. Traditionally, interactions are addressed by simulating the (sub)system of interest in detail–usually for a particular domain–while simplifying the remaining parts. This is for instance the current approach for power electronic devices. Such treatments lack the level of detail in presenting the characteristics of the overall system, which can be incorrect since underlying interactions are disregarded. In intelligent power systems phenomena hence need to be addressed holistically: the reaction to events in one particular domain are now spread across multiple domains and cannot be safely simplified nor disregarded.

The functionality needs of holistically analyzing intelligent power systems unfortunately clash: On one hand, the system has a *heterogeneous* nature, which requires a methodthat can generically set up and process different type of models representing this heterogeneity. On the other hand, there is a need to *accurately* (i.e. modeling detail) and (computationally) *efficiently* (i.e. time needed) represent each phenomenon of interest.
Heterogeneous systems can be analyzed in a number of approaches (Table I) with individual pros and cons. Experiments are often too expensive in the intelligent power systems domain, and limit the simulation runs to real-time. General purpose *multi-domain* simulation tools have the capability of simulating heterogeneous components and small systems. Once the interactions in intelligent power systems get complex, they will not work correctly anymore (scalability constraints).

The second functionality requirement (i.e. computational efficiency and accuracy) might be met in a specialized, optimized *monolithic* environment. For instance, specialized power system simulations have been developed over the past decades, and have been numerically optimized for particular purposes (e.g., load flow, dynamics, transients). The same holds for telecommunication simulations. Unfortunately, this only holds for non-heterogeneous models. Modeling communication systems in power system simulators (or vice versa) leads to brutal simplifications and is often not possible at all.

It is therefore time to move towards simulation platforms that can handle multi-domain systems with reasonable detail and simulation speed. Coupled simulations or *co-simulations* aim to fulfill the functionality needs by modeling

multi-domain systems across multiple simulation tools, while acting as one integral simulation platform that addresses the study [2].

A supposedly simple example of a hybrid system in the power domain is electric vehicle (EV) charging (Figure 1 lower part). Not only behavioral and electric domains need to be combined, but fundamentally different modeling approaches can also be needed within one domain. Batteries might be best modeled with a universal modeling language like Modelica, the distribution grid has specialized simulators and models, and power electronics again different ones. Squeezing all that into one modeling and simulation software would require simplifications of unknown consequences and a lot of effort.

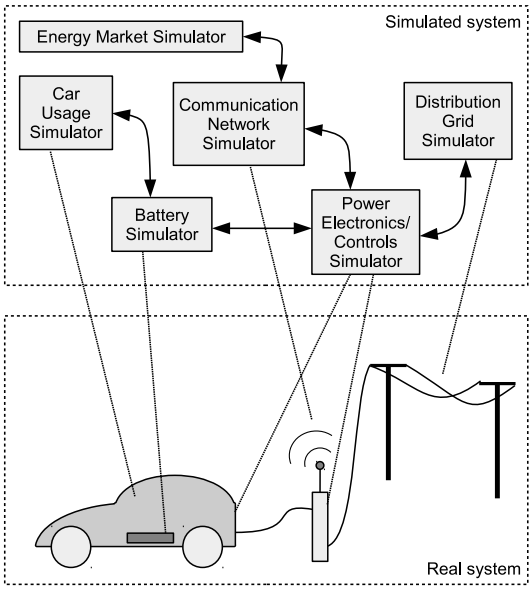

Fig. 1: A typical setup for co-simulating a complex system.

Co-simulation (Figure 1 upper part) on the other hand works with specialized software packages that use validated model libraries and tailor-made solvers. A multi-agent simulator might be the best choice to describe market players and market rules. Power electronics and their controls will be in another platform. The same applies to the distribution grid, or the battery. The choice of tools depends on the required level of detail. If individual driving choices and their interdependencies are important, then a detailed, again multi-agent-based, simulation might be necessary. If only their collective behavior is needed, a statistical model will be sufficient. During the analysis, these requirements can change, which is fully supported by a co-simulation setup. By that, the model choice is not limited by a singular tool, it is rather natural to pick the most appropriate tool for the given simulation questions. Models can even be encapsulated, providing privacy (two market players or two system operators can run joint simulations without sharing internal information). Splitting systems into sub-models also allows for distributed simulation in order to improve performance. Finally the method allows multi-disciplinary teams to work together and combine their knowledge.

This article discusses some of the fundamental aspects involved with co-simulations. The remainder of this paper

is organised as follows. It starts with a basic overview of what a co-simulation is and how it relates to ordinary paradighms such as simulations, numerical solvers, and models. Then it is shown how the co-simulation concept can be sustained by relevant (existing) software and hardware architectures. The paper continues with a discussion of various numerical aspects that co-simulations entail, especially considering the coupling of miscellaneous solvers. In this respect, the paper will conclude with a type of co-simulation that is particularly significant for future intelligent power systems: the obvious coupling between telecommunication networks, residing in the ICT domain, and the power system and its connected devices as such, represented by ordinary physical models.

II. Basic Concepts of Co-Simulation

A co-simulation is composed of a set of coupled simulators that cooperate with each other. Each simulator has its own solver and works simultaneously and independently on its own model. The simulators are coupled by dynamically connecting the models using their input and output variables, so that the outputs of one simulator become the inputs of the other and vice versa. The variable exchange, time synchronization and execution coordination is, in the most general case, facilitated in runtime by a so-called *master* algorithm, which orchestrates the entire co-simulation. The basic composition of a simulation is thus as shown in Figure 2.

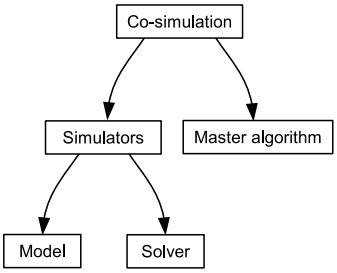

Fig. 2: Basic composition of a co-simulation.

A. The Simulator

As shown in Figure 2 a simulator is defined as a software package that contains the *model* of a system (e.g., a power system or a communication network) and a *solver*, which carries out calculations based on the model and on input variables. Together, the model and the solver, and thus the simulator, allow to predict the behavior of a real system under a set of specified conditions. Simulators typically provide functionality to facilitate the development of the models in the domain they specialize. For example, power system simulators provide a method to describe standard components in terms of a set of physical parameters and a method to define the way components are interconnected. It is the simulator's job to take these descriptions and transform them into equations that can be processed by the solver. This way, any simulator that specializes in a certain domain can implement a modeling method that is most suitable for that specific domain.

The models that can be derived from different subsystems in an intelligent power system can be of the most diverse natures. In the case of the power system, the models can be purely algebraic (in the case of steady-state simulations), purely composed of differential equations (in the case of electromagnetic transients and circuit simulations) or a

combination of algebraic and differential equations (in the case of transient stability simulations that focus on the electromechanical phenomena of rotating generators). Communication networks and markets are in turn modeled as discrete event systems. Other parts might be described with the finite element method or via behavioral models.

B. Co-simulation Master Algorithm

The task of a master algorithm is threefold:

- set up and initialise the simulators (i.e., provide compatible starting conditions);
- synchronize the time of the simulators throughout the simulation;
- exchange variables and events between the simulators.

The master algorithm leads all simulators from the start along the simulation time. Upon start, the models are initialized, and communication links (also referred to as *interfaces*) to the simulators are established. The model time is then in the hands of the master algorithm.

Once the communication links between the master and each simulator have been established and each simulator has been initialized (model parameters, initial conditions of differential equations, etc.), the master assumes the role of stepping each simulator from one so called *communication point* to the next. Individual simulators experience the time steps between communication points (also known as macro time steps) as some kind of full simulation: they get input variables and a simulation duration in the beginning, and start simulating autonomously. At every communication point simulators exchange *inter-model variables*. Then the master algorithm informs all simulators about the next communication point and each simulator proceeds until then. Once the last communication point is reached, the master algorithm ends the co-simulation. The interaction between the co-simulation master and each simulator is depicted in Figure 3.

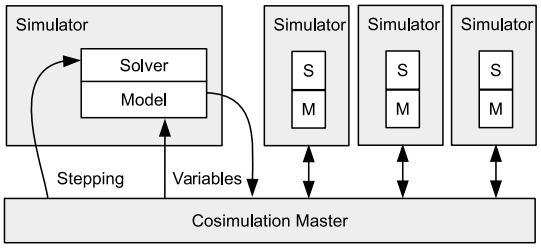

Fig. 3: Interaction between simulator master algorithm and the simulators.

C. Simulator Synchronization

As mentioned above, simulators exchange variables at communication points. The time interval between two communication points is known as *macro time step*. Inside each macro time step, each simulator is allowed to proceed in time as according to its own needs. Simulators can, therefore, carry out a series of *micro time steps* inside a macro time step, as it is shown in Figure 4. In most cases, the macro time step is chosen to be constant, however, it is also possible to control its size at run time in order to increase accuracy and improve computational

performance. The size of the micro time step can differ between simulators (compare simulators 1 and 2 in Figure 4), and can even be variable (see simulator 2 in Figure 4).

The breaks at the communication points and the associated communication take time. The choice of macro step size is therefore crucial for the performance of the co-simulation. If one sub-model has a small micro step size and its variables are relevant for other sub-models, it will slow down all other simulators by imposing a small communication step size.

The synchronization needs for each co-simulation depend on the involved domains and their respective modelling method. Major parameters here are:

- the test criteria that need to be met (e.g. qualitative behaviour, validation of controls);
- the time frame of interest for the interactions between the modeled subsystems. For instance, market or tap changing commands have a much longer time-frame of interest than voltage control and fault ride-through of converter interfaced generation;
- the type of solver in both subsystems (i.e., differential algebraic versus fully discrete or continuous);
- practical considerations like license availability and black-box modeling limit the choice of tools to be applied, and hence restrict the freedom in synchronization alternatives.

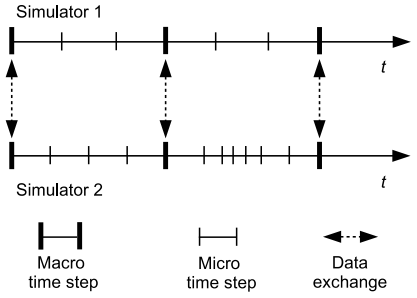

Fig. 4: Micro and macro time steps.

III. A Software Perspective on Co-Simulation

Software packages for simulating future energy systems face several challenges. Two important ones are the potentially large size of the systems (the *scalability* challenge, [3]), and the potentially very different parts of the subsystems (the *heterogeneity* challenge, [4]).

An example of a scalability problem is a vertically integrated view on the power system. The transmission network and its distribution networks are considered as one integral part and consequently modeled in the same system. This leads to a large number of elements and time-consuming dependencies when simulating its behavior. It can be approached by

- simplifying the model (e.g., averaged representation of power electronic devices, network reduction), by
- choosing a faster numerical solver (e.g., adaptive time step-sizes), or by
- providing more computational power (e.g., real time simulation of HVDC systems).

Model simplifications are acceptable as long as the user knows the price to pay for the simplifications. Quantitative figures of lost behavior or frequency limits are needed to decide if a simplification is still acceptable or not for a given study. Reducing an 8^{th} order electric machine model to a 2^{nd} order model, for instance, would remove details that might be important for certain analyses. Another example is the adoption of an averaged grid interface for a converter model, impairing the accuracy of its interactions with the AC side, predominantly during voltage unbalances and valve blocking – crucial information that is lost during model reduction.

Using a faster numerical solver has also limits. Some solvers are specialized for certain problem categories and might even require a special modeling method. Depending on whether the model contains partial differential equations, limiters, or non-linear parts, the choice of solvers narrows and their speed is only determined by their implementation.

One way to increase computational power is to use faster computers, i.e., more memory, cache, higher CPU frequency, etc. The passage of time is on our side in this respect: computers get faster every year. Another way is to separate the problem into sub-problems that can be solved in parallel. This can be an easy task in case of "ridiculously parallel computing": the computational problem consists of doing one and the same analysis many times with different parameters [5]. They can easily be done on separate machines, no communication or synchronization during the simulation runs is needed, however, the problem is more complex if the model itself is split. The submodels have mutual dependencies and dynamic variables might require communication between the models at every time step of the simulation run.

Heterogeneity happens if the power system dynamically interacts with other systems (e.g., heat network, markets, etc.) whose modeling method is fundamentally different to the chosen method of the power system. This is also the case for smart grids: the digital controls require discrete event handling, while the grid infrastructure is probably modeled via differential equations. An instance of such controls are ancillary services and grid support of distributed generation, managed by a centralized controller. Even within one and the same physical domain it is possible to have a heterogeneous setting if parts of the power system are modeled in transient stability and selected parts by electromagnetic transients in order to have a detailed view of some specific components. Heterogeneity of models can be approached by using a universal modeling platform or via co-simulation (see also Table I).

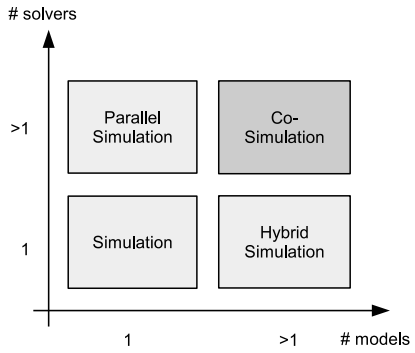

Fig. 5: Four types of simulation: normal (i.e., monolithic), parallel, hybrid, co-simulation.

Figure 5 shows the four fundamental options of solver vs. model. The trivial case involves one solver and one model. Parallel simulation means that the model is still modeled with one tool and one language before it is jigsawed into pieces for parallel solving. The solvers in this case might be identical, or even have different time steps or solving algorithms. A prominent example of parallel simulation is RTDS (Real Time Digital Simulator, [6]): one model is compiled onto multiple computational targets and executed in parallel on dedicated hardware. The case of hybrid simulation involves multiple, different types of modeling environments and languages, in order to make the modeling task easier. However, the models are still forming one monolithic unity that can be solved by one single solver. The only advantage is that the individual parts or aspects of the model get a specialized modeling method (e.g., graphical, different languages and libraries, etc.).

The fourth case combines the advantages - but also the challenges - of the other three: co-simulation.

A. Simulator Interfacing

The simulators that compose a co-simulation need to exchange data with each other during various stages of the simulation workflow (e.g., model instantiation, initialisation, runtime, and data export). The interface between them can be implemented via shared memory if all simulators can access one common memory, or via network communication protocols. Various simulation packages offer a number of semi-standard interfaces, for instance, based on certain proprietary Application Program Interfaces (APIs, e.g. [7]), Object linking and embedding for Process Control (OPC), or transmission control protocol (TCP) socket interfaces. One interface type that seems to be evolving as the standard for coupling physical models and simulators is the Functional Mockup Interface (FMI) [8]. The FMI offers a low-level interface for two purposes:

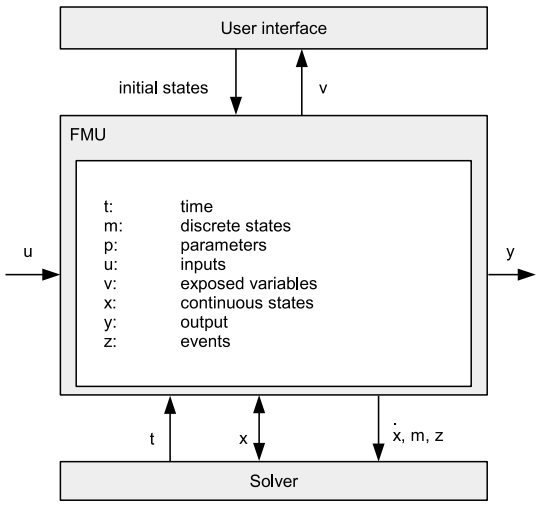

Fig. 6: The Model Exchange variant of the Functional Mockup Interface.

- model exchange;
- co-simulation.

Both are equally useful, depending on what needs to be achieved.

The FMI for Model Exchange (Figure 6) exposes a compiled numerical model to a solver/simulator with a standard interface to initialize the states, execute a time step, determine derivatives, etc. Wrapping a model into such an interface results in a Functional Mockup Unit (FMU): a convenient way how component manufacturers can provide their customers a dynamic, standardized model, of their product, without jeopardizing their intellectual property. Ideally, the customer would plug this FMU into the bigger system model to verify if that particular component interoperates well with the remaining components in the intended system.

The FMI for co-simulation (Figure 7) goes one step further and also packs the solver into the FMU. This FMU acts as a co-simulation slave, orchestrated by a master algorithm, which cares for synchronization, variable exchange, etc. FMI for co-simulation also allows coupling of separate tools as such; in this case the FMU consists of an FMI wrapper around the slave tool (right upper unit in Figure 7), which on its turn contains the model of interest (plus a solver). The former bears the additional advantage that it exempts the user for possessing a dedicated license for the slave simulator.

Two important concepts with FMI are FMU import (where an existing FMU is plugged in and used in a super-model) and export (where a model is compiled into an FMU to be then imported somewhere else). If the model/tool to be wrapped is a black-box tool (closed source, binary only), the wrapping can be computationally very inefficient. In the extreme case, every simulation step requires a re-initialization and re-start of the tool.

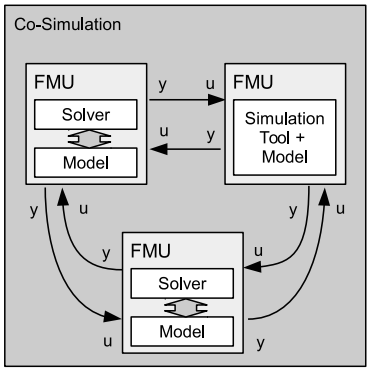

Fig. 7: Usage of the Co-Simulation variant of the Functional Mockup Interface.

The FMI originates from the automotive industry and European projects, and is now further developed within a Modelica Association Project (www.fmi-standard.org). It is based on compiled C-code and XML files that describe the interface and the models. FMI is currently supported by 84 simulation tools and the standard is published in a creative commons and Berkeley Software Distribution (BSD) style license.

B. Co-Simulation Orchestration

Once a set of simulations is defined it is the art of co-simulation to coordinate and orchestrate their execution using a master algorithm that propagates events and shared variables and synchronizes model time. A detailed description of co-simulation master algorithms is put forward in [9].

Synchronizing the advancement in model time requires direct access into the integrators or schedulers of the individual simulation packages. Often, one of the packages acts as master (Figure 8), especially if only two simulators are coupled or if the simulators form a *star* topology. The more abstract and general case, however, is when a dedicated master algorithm (Figure 3) coordinates the solvers.

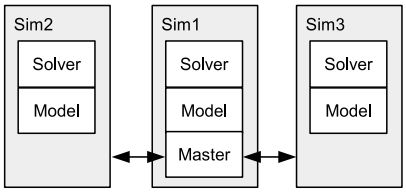

Fig. 8: One central simulator acts as master.

The master algorithm therefore has a number of interfaces. In the case of FMI, the FMI API services are used to perform the time stepping and the variable exchange. Often, the master algorithm also contains the user interface and/or a scripting engine that can automate the simulation experiments. Some co-simulation settings are more sophisticated by allowing the simulators more freedom to proceed in time. If a synchronization need is discovered too late, such simulators must be able to "roll back" in time to synchronize at the correct point in time. Not many legacy simulators have this capability, which is why synchronization needs to be done the "safe" way, i.e., at every step.

The connection between the simulators and the master might be on one and the same machine, using shared memory or messages, or might be network based so that the individual parts of the software setting can be distributed onto separate machines. The High Level Architecture (HLA, [10]) is an example for a complete specification of a central entity (in this case the *Run-Time Infrastructure* RTI) and interaction rules on how to initialize, start, synchronize, and stop potentially distributed simulators (the so-called *federates*). There are several commercial and free HLA implementations available. Using these packages solves many of the issues arising with coupling simulators: synchronization, service lookup, initialization, distributed computing, etc.

The "philosophy" of the master algorithm strongly determines the principles of synchronization. A simple master algorithm would sort and schedule simulation events by time and execute the associated simulator code accordingly while exchanging shared variables. More sophisticated platforms like Ptolemy II [11] or Mosaik [12] provide even more features like hybrid models or scenario handling. Scenarios are sets of parameters that might be varied between multiple simulation runs. The simulation then serves as a utility function of an optimization process (see Figure 9).

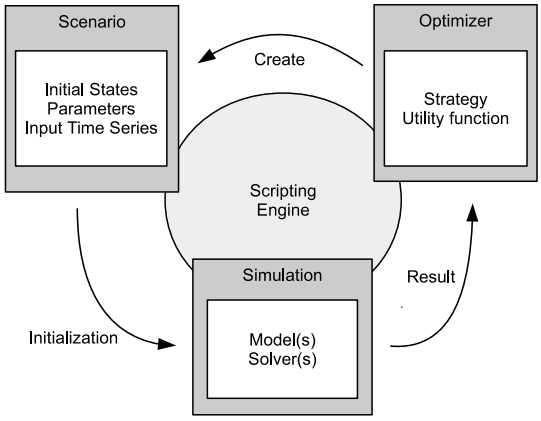

Fig. 9: Flowchart of simulation runs.

Simulations of smart energy systems can be quite computationally expensive and time consuming [13]. If the optimization requires thousands of simulation runs to optimize the smart grid parameters, the entire idea of simulation-supported optimization becoms unfeasible. A way out of this is to enhance the parameter choice during the optimization process. DoE (design of experiments, a method that dates back to the 1930ies) uses statistical methods to support the choice and variation of parameters so that only a small subset of simulations (i.e. experiments) is needed to browse through all potential options [14]. It is thus not sufficient to just couple the simulators, the scripting and optimization details are of equal importance.

C. Practical Considerations

An important aspect with regards to software for co-simulation settings is their openness for interfacing. A simulator can either be a

- black box (i.e. closed source with no co-simulation interfaces) that can at best only be scripted to batch-perform simulations, or have
- proprietary interfaces or APIs (maybe usable for simple co-simulation),
- open interfaces like FMI that allow software integration and optimization, or even be
- open source, so that all details are accessible, where even the solver can be interfaced and controlled.

Often software packages are only available for one particular operating system, so a generalized interface cannot be attained using shared memory. This prompts the user to implement the master interfaces via communication protocols such as the TCP/IP stack. Overhead and latency of these communication protocols can dramatically slow down the system simulation. Closely coupled sub-systems should therefore be hosted on one and the same physical machine, which is often not possible with a mix of heterogeneous closed source software. The ideal case is a combination of the last two variants: standardized interface in an open-source software package.

Another limitation of black-box, legacy simulators is that they usually do not offer a sophisticated API for simulation control: Step sizes can not be chosen and rollback is practically impossible. In the worst case they can only be used in an initialize-simulate-terminate fashion: The individual communication steps are full simulation

runs for them, and system states need to be stored and restored in between, which is the slowest way possible of interfacing other simulators.

IV. Numerical Solution of Coupled Models

Each model in a co-simulation is solved independently using a different numerical solver. This gives rise to a series of numerical challenges that have to do with lower results accuracy and numerical errors that continuously increase as the co-simulation progresses in time. To illustrate the numerical challenges that arise when coupling continuous models (e.g., physical components, systems, and controls) by having them exchange their outputs, let us consider a case with two subsystems modeled through differential algebraic equations (DAEs)

$$\dot{\mathbf{x}}_1 = \mathbf{f}_1(\mathbf{x}_1, \mathbf{u}_1, t), \qquad \dot{\mathbf{x}}_2 = \mathbf{f}_2(\mathbf{x}_2, \mathbf{u}_2, t), \tag{1}$$

$$\mathbf{y}_1 = \mathbf{g}_1(\mathbf{x}_1, \mathbf{u}_1, t), \qquad \mathbf{y}_2 = \mathbf{g}_2(\mathbf{x}_2, \mathbf{u}_2, t), \tag{2}$$

where $\mathbf{u}_i$ are the inputs vectors, $\mathbf{x}_i$ are the vectors of state variables, $\mathbf{y}_i$ are the output vectors, $\mathbf{f}_i$ and $\mathbf{g}_i$ are vector-valued functions and t represents the time, for subsystems $i = 1, 2$. If both subsystems are coupled into one system so that the outputs of each subsystem become the inputs of the other, i.e., that the coupling conditions

$$\mathbf{u}_1 = \mathbf{y}_2, \qquad \mathbf{u}_2 = \mathbf{y}_1, \tag{3}$$

are fulfilled, two main approaches can be followed to simulate the coupled system: In the traditional simulation approach equations (1) and (2) are composed into one system of DAEs according to the coupling equations (3) using the chosen simulation tool and are then solved with only one numerical solver so that system outputs are calculated for a set of discrete points in time $\mathbf{t} = \{t_1, t_2 \ldots t_k, t_{k+1} \ldots t_K\}$. In this case, equations (1) to (3) are *strongly coupled* and are fulfilled at every point in $\mathbf{t}$ [15]. When subsystems are strongly coupled, the numerical stability of the simulation (i.e., the assurance that the local truncation error of the numerical solution remains constrained as simulation time progresses) depends exclusively on the properties of the model and the chosen solver [16].

In a co-simulation, equations (1) to (3) are *weakly coupled* since each subsystem is solved independently and only outputs are exchanged between subsystems at the communication points in time defined in $\mathbf{t}$. Within each macro time step $t_k \rightarrow t_{k+1}$, each subsystem can be solved using several micro time steps that do not lead to output exchanges but that do contribute to the accuracy of the calculated outputs at t_{k+1}. When subsystems are weakly coupled, only the fulfillment of the coupling equations (3) is guaranteed at the macro time steps defined in $\mathbf{t}$, but not that of equations (1) and (2). This can manifest itself through either numerical instabilities or inaccurate results [15]. Numerical stability is regarded in this context as the stability of the solution for a non-zero integration step.

The reason for these numerical instabilities and/or inaccuracies lays on the appearance of algebraic loops between subsystems and the methods used to overcome them; in fact, zero-stability of co-simulations is guaranteed as long as no algebraic loops exist [16]. Zero-stability refers to a solution being stable when the integration step approaches zero. Figure 10 shows a block diagram of the coupled subsystems described by equations (1), (2) and (3) that makes the algebraic loop visible. For subsystem 1 to advance from t_k to t_{k+1} its solver requires input values at t_{k+1}, which are only available if subsystem 2 has already produced outputs for t_{k+1}. However, subsystem 2 is unable to produce these outputs since it also depends on the outputs of subsystem 1 at t_{k+1} in order to do so.

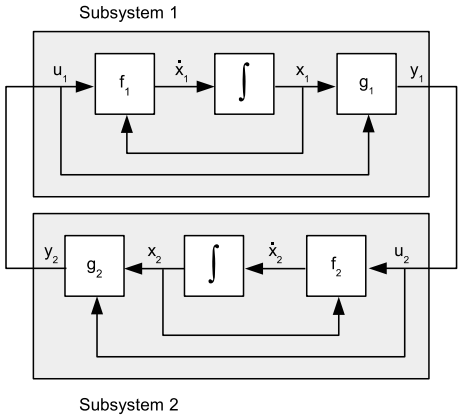

Fig. 10: Algebraic loop in the co-simulation of the two subsystems from equations (1), (2) and (3).

Several methods for overcoming algebraic loops in co-simulations exist, which in most cases have to do with the sequence in which the subsystems exchange values with each other, the choice of variables to exchange, and the placement of dynamics in the coupling equations. However, these methods may severely affect the quality of the results and trade-offs between computational performance and accuracy or stability are to be made.

A. Communication Sequences

Communication sequences, describe the order in which simulators exchange values. These sequences are typically classified in parallel and serial depending on whether the simulators that compose the co-simulation can run in parallel or if they must be executed one after the other. The consequences of choosing one type or the other go beyond the computational performance of the co-simulation, since each communication sequence overcomes algebraic loops in a slightly different way.

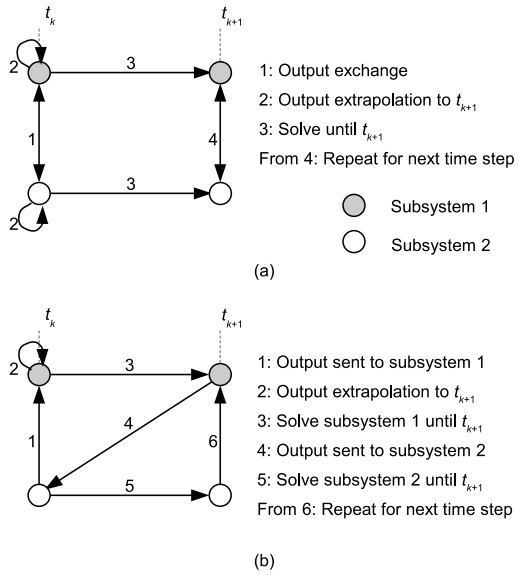

Fig. 11: Communication sequences for co-simulation (weakly coupled) for one macro time step $t_k \rightarrow t_{k+1}$: (a) Parallel (Jacobi), (b) Serial (Gauss-Seidel).

1) Parallel Sequence: In parallel communication sequences, also known as *Jacobi* type, all the subsystems that compose the co-simulation are solved in parallel and simultaneously exchange values at discrete points in time. A diagram depicting this sequence for two subsystems is shown in Figure 11 (a). Since each subsystem requires the output of the other at t_{k+1} in order to advance from t_k to t_{k+1}, the outputs of each subsystem at t_{k+1} are extrapolated from the outputs at t_k. The simplest way to achieve this is through zero-order hold extrapolation , that is, the outputs at t_{k+1} are assumed to be the same as at t_k. Among the extrapolation methods typically found in the literature are zero, first and second order hold extrapolation [17] and polynomial extrapolation [15].

The advantage of the parallel sequence is that simulators can be executed on distributed computers, which can aid the overall computational performance of the co-simulation. The main disadvantage is that each simulator must predict future outputs of the remaining simulators by means of extrapolation, making the accuracy and stability of the co-simulation completely dependent on the accuracy of this prediction [18].

2) Serial Sequence: In serial communication sequences, also known as *Gauss-Seidel* type, subsystems are solved one after another. A diagram depicting this sequence for two subsystems is shown in Figure 11 (b). Since subsystem 1 requires the output of subsystem 2 at t_{k+1} in order to advance from t_k to t_{k+1}, the outputs of subsystem 2 at t_{k+1} are extrapolated from its outputs at t_k. Once subsystem 1 has advanced to t_{k+1}, its outputs become available to subsystem 2 which can now advance to t_{k+1} without performing extrapolations. The fact that this sequence requires only one extrapolation usually results in slightly more accurate results than those obtained with the parallel sequence, with the disadvantage that since simulators are executed one after another, more coupled simulators results in longer executions times [15].

3) Iterative Sequences: Iterative sequences can be derived from both parallel and serial sequences. Figure 12 and Figure 13 show diagrams of the iterative version of each sequence type. Although only two iterations are displayed

in the figure, these sequences can be extended to any number of iterations. In practice, iterations are carried out until a convergence criterion is fulfilled. Depending on how strict the convergence criterion is, strong coupling between subsystems can be enforced through iterative sequences which yields much more accurate results than non-iterative sequences. Iterative sequences have also been shown to be zero-stable as long as zero-stable solvers are used for each subsystem [16]. It is easy to note that although iterative sequences have better accuracy and numerical stability properties than their non-iterative counterparts, a much higher computational effort is required to run the co-simulation.

Due to the higher accuracy of serial sequences [15], iterative serial sequences have a higher convergence rate than parallel iterative sequences [19]. The practical implementation of iterative sequences imposes the requirements that all the simulation tools involved in the co-simulation must have a time rewinding mechanism so the same time step can be solved more than once. However, it is uncommon to find commercial simulators that offer this feature.

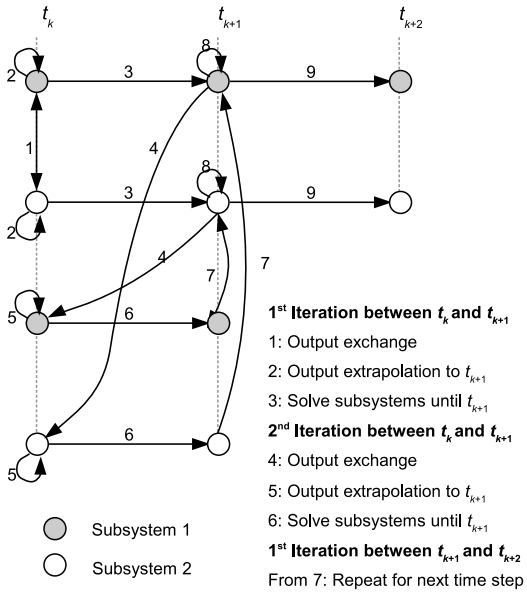

Fig. 12: Iterative (strongly coupled) communication sequences for co-simulation, parallel (Jacobi) type, for one macro time step $t_k \rightarrow t_{k+1}$.

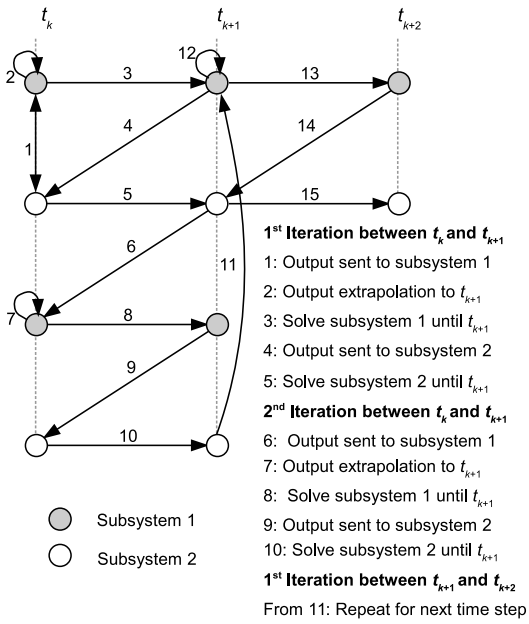

Fig. 13: Iterative (strongly coupled) communication sequences for co-simulation, serial (Gauss-Seidel) type, for one macro time step $t_k \rightarrow t_{k+1}$.

B. Coupling Variables and Models

The choice of coupling variables (i.e., the variables that are exchanged between models), or equivalently, the way a system is partitioned into subsystems, influences the accuracy and stability of the co-simulation. For example, in the case of mechanical systems the choice of different combinations of force and displacement variables and the numerical consequences of each combination have received much attention [15], [20].

As discussed in the previous subsection, when subsystems are coupled as described by equations (1), (2) and (3), extrapolation of the inputs is required, which introduces additional errors into the numerical solver of each subsystem. If the subsystems are coupled so that the coupling variables exhibit *weak dynamic interactions* at the chosen coupling point, the effect of the errors introduced by extrapolations on the overall co-simulation accuracy can be minimized. This is exemplified in [21] with the linear circuit shown in Figure 14 (a).

Let us consider a case where R_2 and C_2 are large. The voltage v_2 cannot change quickly and would therefore have little influence on the way current i_2 changes, so the behavior of i_2 would be mostly influenced by v_1. The same can be said about the influence of i_2 on v_2. Since the dynamic interaction between i_2 and v_2 is weak, having i_2 and v_2 as coupling variables would be beneficial for mitigating the influence of extrapolation errors on the results. In this case, the monolithic model from Figure 14 (a) could be co-simulated as in Figure 14 (b). The model employed to couple both subsystems is composed of a voltage source of value v_2 and a current source of value i_2, shown in dashed lines. In this case v_2 is determined by the subsystem on the right and sent to the voltage source on the left while i_2 is determined by the subsystem on the left and sent to the current source on the right. This can be done following any of the communication sequences discussed in the previous subsection.

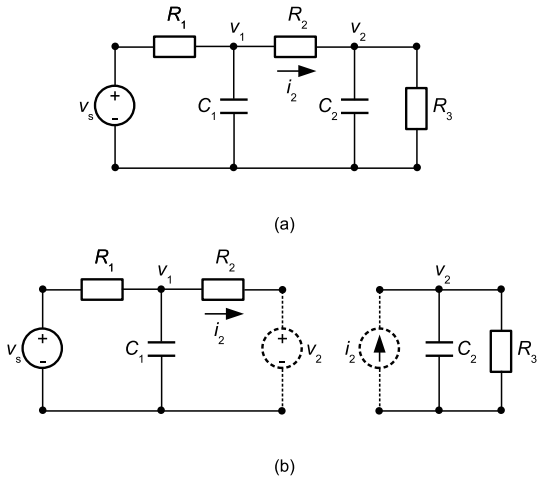

Fig. 14: Subsystem coupling at a point of weak dynamic interaction [21]: (a) Monolithic model, (b) Co-simulated subsystems.

In [16] the possibility of coupling subsystems by adding artificial dynamics to the coupling equations is discussed. A simple way to achieve this is to add a delay Δt in the coupling equations, so equations (3) become

$$\mathbf{u}_1(t) = \mathbf{y}_2(t-\Delta t), \qquad \mathbf{u}_2(t) = \mathbf{y}_1(t-\Delta t). \tag{4}$$

This is equivalent to using zero-order hold extrapolation of the inputs if Δt is chosen to be the size of one macro time step.

Although artificial dynamics break existing algebraic loops, they modify the dynamic behavior of the original model. To avoid these modifications, an alternative is to transfer some of the dynamics of each subsystem to the coupling equations, which gives rise to more sophisticated coupling models than the one shown in Figure 14 (b). One of the classical methods for achieving this is the use of the transmission line method [22]. The idea behind this method is to take advantage of the delay associated with waves traveling through a transmission line to absorb the delay required to break algebraic loops between subsystems. Although it is straightforward to apply this concept to power systems with long transmission lines, it can be used for other types of physical systems as well [23].

C. Error Estimation and Step Size Control

The ability to estimate the local truncation error of a co-simulation at run time gives insight into the quality of the results and allows accuracy and performance improvements through the implementation of macro step size control mechanisms.

Most of the error estimation methods that have been proposed for co-simulation are inspired by traditional error estimation methods for numerical differential equation solvers, and require the comparison of two solutions, each with a different level of accuracy. In [24] a method based on *Richardson extrapolation* is employed. Here, two consecutive macro time steps of size h are carried out: first $t_k \to t_{k+1} = t_k + h$ and later $t_{k+1} \to t_{k+2} = t_k + 2h$. Then, a less accurate solution is obtained using only one large macro time step of size $2h$, that is, $t_k \to t_{k+2} = t_k + 2h$. In

[25] the *embedded methods approach* is applied to multirate partitioned Runge-Kutta methods. This method consists in evaluating the outputs of each subsystem using polynomial extrapolation of the inputs of two different orders, one higher than the other. In [26] a method tailored to the *predictor/corrector* co-simulation approach presented in [27] is derived. Here, the comparison is carried out between the predicted and the corrected solutions.

In all of these methods, if the estimated error is larger than a user-defined tolerance, the macro time step is repeated using a smaller macro step size, which is time consuming and rather difficult, if not impossible, to implement with most commercial simulation tools. Here lays the motivation for the method proposed in [15]. This method modifies the size of the next macro time step based on an estimation of the current error. To estimate the error, the outputs of all subsystems at the current macro time step are predicted from previous outputs using polynomial extrapolation. Once the current macro time step is executed and the co-simulation outputs become available, they are compared to the predicted outputs to derive an error estimation.

A completely different approach is taken in [28], where the error is estimated using a generalized concept of energy conservation derived from bond graph theory [29]. Here, subsystems are coupled through so called *power bonds*, which are defined by two variables, a *flow* and an *effort*, that in the case of electrical systems correspond to current and voltage. Since the product of flow and effort variables corresponds to the energy flow (power) through the power bond, any discrepancies between the energy flow calculated from the co-simulation outputs and the one calculated from energy conservation are attributed to co-simulation inaccuracy and can therefore be used to estimate the local truncation error.

A macro step size control method can be implemented once the local truncation error is estimated through traditional step size control methods for ordinary differential equations so that the local error remains withing an upper and a lower boundary. For example, [15] proposes the use of a PI controller, as presented in [30]. Since in a co-simulation a different local truncation error estimation can obtained for the outputs of each subsystem, the size of the macro time step must be chosen taking into account the most demanding of all the subsystems.

V. Coupling Power System and ICT Simulators

The integration of power systems, automated devices, and ICT gives intelligent power grids the character of a cyber-physical system [31]. On one hand, ICT-specific features such as communication network topology, protocols, communication latency, bandwidth, information security, and reliability issues intrinsically affect the behavior of the power system. On the other hand, the power system and its features also impact the corresponding ICT infrastructure.

There are four variants of how to represent ICT in a simulation setup:

- Hardware-in-the-loop (HiL): The real ICT products (tele-communication switches, controllers, etc.) are used, the setup runs in real-time.
- Emulated ICT hardware: The real binary code (of switches, etc.) is executed on an emulation platform that provides the same hardware properties (memory, speed, etc.) like the real hardware but allows for flexible reconfiguration. Again, the setup runs in real-time.
- Simulated ICT hardware: The real code is executed and the execution time of the hardware platform is estimated/imitated. This setup can run in non-real-time (ideally faster than real-time).

- Full Simulation: All ICT elements (switches, etc.) are simulated/imitated with proxy code that uses stochastic or other simplified means of representing the time-domain behavior of the system. This setup again runs in non-real-time.

This article mainly covers the last variant, full simulation, which opens up questions on how to synchronize the various parts, especially in a co-simulation setting.

Combining ICT with a physical system leads to a number of dependencies that require attention [32]. One important example of such dependency is reliability analysis. Typically, contingencies in power systems are considered as independent events, such as the loss of electric components. However, intentional cyber-attacks and vulnerabilities from the ICT domain break this assumption as ICT assets could be used to cause damage to the electric components in a coordinated manner [33].

As in any cyber-physical system, the power network and its components and the ICT infrastructure are two parts of a heterogeneous, larger system. Co-simulation is currently one of the most popular methods to analyze the behavior of intelligent power grids.

A. Modeling and Simulation Challenges

Communication networks are–as all digital systems–modeled as a sequence of discrete events (e.g., sending and receiving packets, packet buffer overflows, etc.), while power systems are typically modeled as continuous time functions using differential algebraic equations, although discrete power system events occur as well when the status of breakers, switches, and relays change. Consequently, a holistic model of a smart grid must include continuous and discrete aspects.

According to [34], simulation paradigms can be divided into three time-management categories:

- fixed time step-size simulation in which the simulation time is discretized in equal time steps;
- continuous simulation , which commonly apply adaptive time step-size control; and
- discrete-event simulation which advances the simulation time only when events occur.

Intelligent power grids often need multiple models, which need to fit into heterogeneous simulation paradigms. The ICT part of such a multi-domain model is normally implemented as a discrete-event simulation, while the power system part is included as a continuous or fixed time step-size simulation. As mentioned previously, hybrid simulations can be a solution for this problem, i.e., single solvers that address multiple models [35], [36]. However, such methods scale badly and can hence only be used for component analysis and simple use cases, but not for fully-fledged system studies.

As touched upon before, co-simulation of intelligent power grids, i.e., hybrid physical and discrete models with multiple solvers, comes with advantages but also challenges:

- The integration of continuous power system simulations and discrete-event communication network simulation needs sophisticated synchronization mechanisms. The next subsection will present synchronization methods to tackle this challenge.

- Error estimation and validation of co-simulation is a challenge. The interdependency of hybrid models from power system and ICT parts makes it hard to identify where the simulation error comes from. Different synchronization methods in co-simulation also impact the simulation accuracy.
- Interoperability of the various simulators requires standardized interfaces (see the HLA the FMI discussed in Section III).

B. Synchronization of Discrete and Continuous Simulators

When building a co-simulation platform for intelligent power grids, the synchronization mechanism between the subsystems under consideration is one of the performance dominating factors. It has a direct impact on the convergence and accuracy of the simulation results.

Time synchronization between continuous and discrete simulations can either happen *conservatively* or *optimistically*. Conservative synchronization guarantees strict processing of logical time by a timestamp order. The optimistic one allows a violation of the step-by-step processing, but needs additional control mechanisms that could detect and recover violations [37]. The simulators must be capable of *rolling back* the overall simulation time. Unfortunately, many power system simulators do not possess this functionality [31]. In literature, synchronization methods are mainly subdivided into three categories: point-based , event-driven and master-slave [38].

1) Point-based : While the simulation of power system dynamics uses a time stepped approach, the communication networks are typically modeled as discrete event systems. One intuitive synchronization method is to use predefined synchronization points. As shown in Figure 15, individual simulators run in parallel and stop at the synchronization points to exchange information. The synchronization points are predetermined. However, in most cases, the communication need between two simulators is created by events generated by one of the models, which, in case of ICT models, may even have a stochastic nature [39].

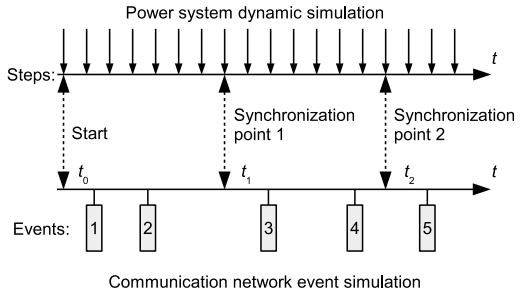

Fig. 15: Point-based synchronization method.

The point-based synchronization method may introduce inaccuracies in the co-simulation. When system output variables need to be exchanged between two synchronization points, both subsystems have to wait until the next synchronization point. This delay introduces error accumulation into the simulation, and possibly impairs the accuracy of the overall simulation results. A simple solution is to reduce the time interval between synchronization points, e.g., exchange data in each time step of power system dynamic simulation [40]. In [41], an advanced point-

based synchronization approach is proposed, in which the next synchronization point is not predefined but given as a parameter to the (continuous) power system simulator.

2) Event-driven: In [42], a global event-driven co-simulation framework is proposed. The event-driven synchronization is shown in Figure 16. It treats each iteration round of the continuous power system simulation as discrete events and mixes them with communication network events. All the discrete events form an event queue (as shown in Figure 16) in chronological order. A global event scheduler checks the event queue, and handles corresponding control for power system events and communication network events individually. Both simulators can suspend themselves and yield the control back to the scheduler when subsequent events occur.

The discrete event specification formalism could be used to model both the power system and the communication network simulation. It provides a rigorous mathematical basis for simulating hybrid system models [43], and is widely used for event-driven synchronization.

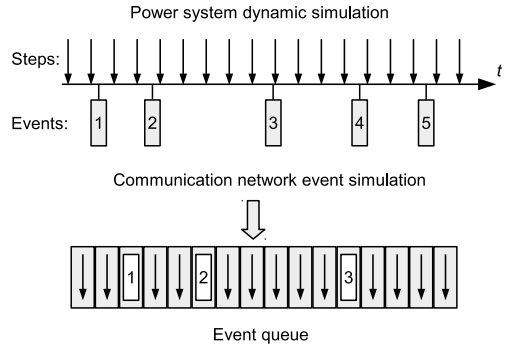

Fig. 16: Event-driven synchronization method based on [42].

Using the event-driven method, the time step-size of the power system simulation significantly impacts the overall co-simulation time. Besides, the interface between simulators can be a performance bottleneck, grinding down scalability. In [42], as the system scale grows, the simulation time also increases due to the increased number of interactions in the interface. Hence, the performance is highly dependent on the capabilities of the respective interfaces.

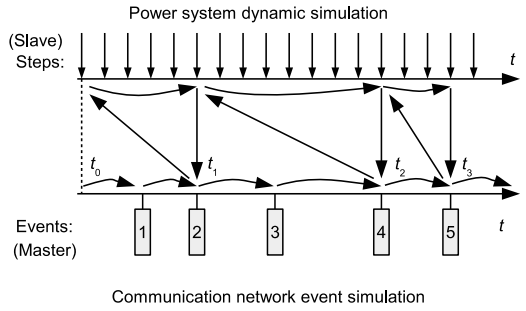

Fig. 17: Master-slave synchronization method: one simulator acts as the master.

3) Master-slave: The third type of synchronization mechanism, shown in Figure 17, is a typical master-slave configuration that allows one simulator (often the discrete-event simulator) as a master simulator to coordinate the entire co-simulation. In Figure 17, the communication network simulator (as the master) controls the power system simulator (as the slave) throughout the simulation process. The master starts the simulation at t_0. When the event at t_1 needs the information from the slave, the master coordinates the slave to simulate from t_0 to t_1 and sends data to the master. For the master-slave approach, the synchronization performance is limited by the capabilities of the master simulator. As discussed in [44], the drawbacks are:

- Events, generated in the slave cannot be communicated to the master immediately.
- Execution is typically sequential.
- Scalability issues inhibit the integration of an arbitrary number of simulators.

The later can be potentially be overcome if one dedicated master algorithm is used to orchestrate all simulators that act as slaves. All slaves have to tell the master when their next event is anticipated. The master then picks the time of the earliest event in this list and declares it to all slaves as the next synchronization point.

Figure 18 shows an example where the master tells slave 2 (power system simulator) the time t_{i1}, which is the time of the first event in slave 1 (communication network simulator). Each slave executes simulation to t_{i1}, where finally data exchange takes place (not shown in the figure for simplicity). The next events are at t_{i2} and t_{j1}. The latter wins since it is earlier, so the next synchronization point is at t_{j1}. slave 1 has to roll back the simulation time from t_{i2} to t_{j1} since it normally jumps from one event to the next.

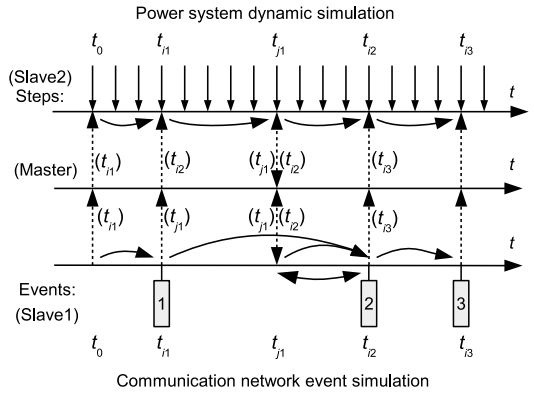

Fig. 18: Master-slave synchronization method: using a dedicated master component.

4) Practical considerations: Recent work on co-simulation of power systems and ICT infrastructure have addressed time synchronization with sophisticated methods [45]. The majority of these co-simulation platforms focus on the integration of one power system simulator with one communication network simulator. However, intelligent multi energy systems (e.g., power-to-heat settings with market-integration) need multi-physics capabilities and large scalability. The corresponding co-simulation needs to couple more than one physical systems. For this case, simple synchronization mechanisms work for coupling two simulators but would fail when coupling more. The second master-slave method with dedicated master algorithm shown in this section could be used.

From the above, it can be deducted that in many cases the designers have to make a trade-off between accuracy, efficiency, and scalability. It should be noted that the accuracy and efficiency in coupling depends on the simulation tools selected, the interfaces, and the synchronization mechanisms. This also implies the level of control that the designer could have on the simulation tools (compare black box vs. open simulators, discussed in Section III).

C. Solvers for better synchronization between continuous ODE and discrete event models

The synchronization methods discussed in the previous subsection are a results of continuous simulators that are unable to produce or react to asynchronous events, that is, events that occur between the points in time at which the continuous model is solved. An alternative approach is taken with the Quantized State System (QSS) family of solvers [46]. As opposed to traditional numerical differential equation solvers that are designed to determine the value that the solution to a differential equation assumes at a given point in time (time discretization), QSS solvers are designed to determine the earliest point in time at which the solution to a differential equation assumes a given value (state variable quantization). In this sense, QSS solvers transform the continuous system described through differential equations not into a discrete time system, but into a discrete event system; every time the state variable moves from one quantization state to a neighboring one, a threshold-crossing event is generated.

In the context of co-simulations, this is an interesting property since it facilitates the integration of continuous and event based simulators. As an example, let us consider a case where a (digital) control system must react to an overvoltage in a power grid. Following the traditional time discretization approach, in order to determine the moment in time when a control action must be taken, an iterative root finding algorithm would be required to find the threshold crossing. Following the QSS approach, determining the point in time when the overvoltage occurs is straightforward if one of the quantization states is defined to match the upper voltage limit.

Despite their promising properties [46], QSS solvers are not as mature as traditional solvers, and no commercial (power system) simulators implement them yet.

VI. Conclusion

The fundamental concepts behind co-simulation of intelligent power systems are described. Software interfaces, numerical aspects, and coordinating individual simulators was discussed for both the power engineering and the ICT domain.

Co-simulation appears to be a powerful tool for dealing with complex, heterogeneous systems that can neither be investigated in analytical nor in purely experimental fashion. Co-simulation has the advantage of easier modeling, since the individual sub-domains are described within their native tools and languages.

The challenges are, however, massive. Validation is commonly done at subsystem level. System-level validation of the system under test must be achieved either with full hardware tests, or hardware-in-the-loop. Aside from the validation aspects, software interoperability is often not given or possible, numerical phenomena and problems are sometimes even unsolvable. Performance and flexibility of the models are often not satisfying, but still the method itself enables us to perform unprecedented analysis of intelligent power systems.

The second part of this article will provide a dive into practical aspects such as how to integrate hardware-in-the-loop simulators, and give an example on how complex smart grid questions can be analyzed by combining electro-mechanic with electro-magnetic transients simulations.

This work is partly supported by the European Community's Horizon 2020 Program (H2020/2014-2020) under project "ERIGrid: European Research Infrastructure supporting Smart Grid Systems Technology Development, Validation and Roll Out" (Grant Agreement No. 654113).

REFERENCES

[1] A. Vojdani, "Smart integration," *IEEE Power and Energy Magazine*, vol. 6, no. 6, pp. 71–79, Nov. 2008.

[2] S. Chatzivasileiadis, M. Bonvini, J. Matanza, R. Yin, T. S. Nouidui, E. C. Kara, R. Parmar, D. Lorenzetti, M. Wetter, and S. Kiliccote, "Cyber-pysical modeling of distributed resources for distribution system operations," *Proc. IEEE*, vol. 104, no. 4, pp. 789 – 806, Apr. 2016.

[3] E. Widl, P. Palensky, and A. Elsheikh, "Evaluation of two approaches for simulating cyber-physical energy systems," in *Proceedings of the 38th IEEE Conference on Industrial Electronics IECON 2012*, 10 2012, pp. 3582 –3587.

[4] P. Palensky, E. Widl, and A. Elsheikh, "Simulating cyber-physical energy systems: challenges, tools and methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 44, no. 3, pp. 318–326, 2013.

[5] G. V. Wilson, "A glossary of parallel computing terminology," *IEEE Parallel Distributed Technology: Systems Applications*, vol. 1, no. 1, pp. 52–67, Feb 1993.

[6] B. Chen, K. L. Butler-Purry, A. Goulart, and D. Kundur, "Implementing a real-time cyber-physical system test bed in RTDS and OPNET," in *Proceedings of North American Power Symposium (NAPS), 2014*, Pullman, WA, Sep. 2014.

[7] C. Dufour, S. Abourida, and J. Belanger, "Hardware-in-the-loop simulation of power drives with rt-lab," in *2005 International Conference on Power Electronics and Drives Systems*, vol. 2, Nov 2005, pp. 1646–1651.

[8] A. Elsheikh, M. U. Awais, E. Widl, and P. Palensky, "Modelica-enabled rapid prototyping of cyber-physical energy systems via the functional mockup interface," in *Proceedings of the 2013 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems*, 2013.

[9] Modelisar, "Functional mock-up interface for co-simulation," MODELISAR (ITEA 2 - 07006), Tech. Rep., 2010.

[10] J. S. Dahmann, R. M. Fujimoto, and R. M. Weatherly, "The DoD high level architecture: an update," in *Proceedings of Simulation Conference Proceedings, 1998. Winter*, vol. 1, Washington, DC, USA, Dec. 1998, pp. 797–804.

[11] W. Muller and E. Widl, "Using FMI components in discrete event systems," in *Proceedings of the 2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems*, Seattle, WA, Apr. 2015.

[12] S. Rohjans, S. Lehnhoff, S. Schuette, S. Scherfke, and S. Hussain, "Mosaik - a modular platform for the evaluation of agent-based smart grid control," in *Proceedings of Innovative Smart Grid Technologies Europe (ISGT EUROPE), 2013 4th IEEE/PES*, Lyngby, Denmark, Oct. 2013.

[13] A. Benigni, F. Ponci, and A. Monti, "Toward an uncertainty-based model level selection for the simulation of complex power systems," *IEEE Syst. J.*, vol. 6, no. 3, pp. 564–574, Sep. 2012.

[14] A. Giunta, S. Wojtkiewicz Jr., and M. Eldred, "Overview of modern design of experiments methods for computational simulations," in *proc. 41st Aerospace Sciences Meeting and Exhibit*, Reno, NV, Jan., "6–9", 2003.

[15] M. Busch, *Zur effizienten Kopplung von Simulationsprogrammen*. Kassel University Press GmbH, 2012.

[16] R. Kübler and W. Schiehlen, "Two methods of simulator coupling," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 6, no. 2, pp. 93–113, Jun. 2000.

[17] M. Benedikt, D. Watzenig, and A. Hofer, "Modelling and analysis of the non-iterative coupling process for co-simulation," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 19, no. 5, pp. 451–470, Oct. 2013.

[18] M. Busch and B. Schweizer, "Numerical stability and accuracy of different co-simulation techniques: analytical investigations based on a 2-DOF test model," in *Proceedings of The 1st Joint International Conference on Multibody System Dynamics, IMSD*, Lappeenranta, May 2010, pp. 25–27.

[19] S. Sicklinger, "Stabilized co-simulation of coupled problems including fields and signals," Ph.D. dissertation, Technische Universität München, Munich, 2014.

[20] B. Schweizer, P. Li, and D. Lu, "Explicit and implicit cosimulation methods: Stability and convergence analysis for different solver coupling approaches," *Journal of Computational and Nonlinear Dynamics*, vol. 10, no. 5, pp. 051 007–051 007, Sep. 2015.

[21] F. Casella and C. Maffezzoni, "Exploiting weak interactions in object-oriented modeling," *EUROSIM Simulation News Europe*, vol. 22, pp. 8–10, Jan. 1998.

[22] D. M. Auslander, "Distributed system simulation with bilateral delay-line models," *Journal of Basic Engineering*, vol. 90, no. 2, pp. 195–200, Jun. 1968.

[23] R. Braun and P. Krus, "An explicit method for decoupled distributed solvers in an equation-based modelling language," in *Proceedings of the 6th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, ser. EOOLT '14. New York, NY, USA: ACM, Oct. 2014, pp. 57–65.

[24] T. Schierz, M. Arnold, and C. Clauß, "Co-simulation with communication step size control in an FMI compatible master algorithm," in *Proceedings of the 9th International Modelica Conference*, Munich, Sep. 2012.

[25] M. Günther, A. Kværnø, and P. Rentrop, "Multirate partitioned Runge-Kutta methods," *BIT Numerical Mathematics*, vol. 41, no. 3, pp. 504–514, Jun. 2001.

[26] T. Meyer and B. Schweizer, "Error estimation approach for controlling the comunication step size for semi-implicit co-simulation methods," *PAMM*, vol. 15, no. 1, pp. 63–64, Oct. 2015.

[27] B. Schweizer and D. Lu, "Predictor/corrector co-simulation approaches for solver coupling with algebraic constraints," *ZAMM - Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 95, no. 9, pp. 911–938, Sep. 2015.

[28] S. Sadjina, L. T. Kyllingstad, E. Pedersen, and S. Skjong, "Energy conservation and power bonds in co-simulations: Non-iterative adaptive step size control and error estimation," *ArXiv e-prints*, 2016. [Online]. Available: http://adsabs.harvard.edu/abs/2016arXiv160206434S

[29] W. Borutzky, "Bond graph based physical systems modelling," in *Bond Graph Methodology*. Springer London, 2010, pp. 17–88.

[30] K. Gustafsson, M. Lundh, and G. Söderlind, "A PI stepsize control for the numerical solution of ordinary differential equations," *BIT Numerical Mathematics*, vol. 28, no. 2, pp. 270–287, Jun. 1988.

[31] H. Georg, S. Muller, C. Rehtanz, and C. Wietfeld, "Analyzing cyber-physical energy systems: The INSPIRE cosimulation of power and ICT systems using HLA," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2364–2373, Jun. 2014.

[32] C. B. Vellaithurai, S. S. Biswas, R. Liu, and A. Srivastava, "Real time modeling and simulation of cyber-power system," in *Cyber Physical Systems Approach to Smart Electric Power Grid*. Springer-Verlag Berlin Heidelberg, 2015, pp. 43–74.

[33] K. R. Davis, C. M. Davis, S. A. Zonouz, R. B. Bobba, R. Berthier, L. Garcia, and P. W. Sauer, "A cyber-physical modeling and assessment framework for power grid infrastructures," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2464–2475, Sep 2015.

[34] H. L. Vangheluwe, "DEVS as a common denominator for multi-formalism hybrid systems modelling," in *Proceedings of Computer-Aided Control System Design, 2000. CACSD 2000. IEEE International Symposium on*, Anchorage, Alaska, USA, Sep. 2000, pp. 129–134.

[35] M. S. Branicky, V. Liberatore, and S. M. Phillips, "Networked control system co-simulation for co-design," in *Proceedings of American Control Conference, 2003. Proceedings of the 2003*, vol. 4, Denver, Colorado, USA, Jun. 2003, pp. 3341–3346.

[36] D. Henriksson, A. Cervin, and K.-E. Årzén, "Truetime: Simulation of control loops under shared computer resources," in *Proceedings of Proceedings of the 15th IFAC World Congress on Automatic Control*, Barcelona, Spain, Jul. 2002.

[37] H. Georg, S. C. Muller, N. Dorsch, C. Rehtanz, and C. Wietfeld, "INSPIRE: Integrated co-simulation of power and ICT systems for real-time evaluation," in *Proceedings of Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, Vancouver, Canada, Oct. 2013, pp. 576–581.

[38] W. Li and X. Zhang, "Simulation of the smart grid communications: Challenges, techniques, and future trends," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 270–288, Jan. 2014.

[39] V. Liberatore and A. Al-Hammouri, "Smart grid communication and co-simulation," in *Proceedings of 2011 IEEE Energytech*, Cleveland, OH, May 2011.

[40] R. Bottura, D. Babazadeh, K. Zhu, A. Borghetti, L. Nordstrom, and C. A. Nucci, "SITL and HLA co-simulation platforms: Tools for analysis of the integrated ICT and electric power system," in *Proceedings of IEEE EUROCON 2013*, Zagreb, Croatia, Jul. 2013, pp. 918–925.

[41] D. Bhor, K. Angappan, and K. M. Sivalingam, "A co-simulation framework for smart grid wide-area monitoring networks," in *Proceedings of 2014 Sixth International Conference on Communication Systems and Networks (COMSNETS)*, Bangalore, India, Jan. 2014.

[42] H. Lin, S. Veda, S. Shukla, L. Mili, and J. Thorp, "GECO: Global event-driven co-simulation framework for interconnected power system and communication network," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1444–1456, May 2012.

[43] J. Nutaro, P. T. Kuruganti, L. Miller, S. Mullen, and M. Shankar, "Integrated hybrid-simulation of electric power and communications systems," in *Proceedings of 2007 IEEE Power Engineering Society General Meeting*, Tampa, FL, Jun. 2007.

[44] K. Mets, J. A. Ojea, and C. Develder, "Combining power and communication network simulation for cost-effective smart grid analysis," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1771–1796, Mar. 2014.

[45] S. C. Mueller, Y. Deng, P. Palensky, M. Stifter, C. Dufour, X. Wang, V. Dinavahi, A. Davoudi, M. O. Faruque, A. Monti, M. Ni, and A. Mehrizi-Sani, "Interfacing power system and ict simulators: Challenges, state-of-the-art, and case studies," *IEEE Transactions on Smart Grids*, 2016.

[46] F. Cellier, E. Kofman, G. Migoni, and M. Bortolotto, "Quantized state system simulation," *Proceedings of the GCMS08, Grand Challenges in Modeling and Simulation*, pp. 504–510, 2008.

PLACE
PHOTO
HERE

Peter Palensky (M'03-SM'05) is full Professor for intelligent electric power grids at TU Delft. Before that he was Principal Scientist at the Austrian Institute of Technology (AIT), associate Professor at the University of Pretoria, South Africa, Department of Electrical, Electronic and Computer Engineering, University Assistant at the Vienna University of Technology, Austria, and researcher at the Lawrence Berkeley National Laboratory, California. He is active in international committees like ISO, IEEE and CEN. His main research fields are energy automation networks, and modeling intelligent energy systems.

PLACE
PHOTO
HERE

Arjen van der Meer (M'08) Arjen van der Meer - obtained the B.Sc. degree in electrical engineering at NHL University of applied sciences, Leeuwarden, the Netherlands, in 2006. In 2008, he received the M.Sc. degree (Hons.) in electrical engineering from Delft University of Technology, the Netherlands. Currently, he is working towards the Ph.D. degree on the grid integration of offshore VSC-HVDC grids at Delft University of Technology. His main research topic is the interconnection of large scale wind power to transnational offshore grids. His research interests include power system computation, the modeling and simulation of smart grids, renewable energy sources, power electronic devices, and protection systems.

PLACE
PHOTO
HERE

Claudio David López (M'16) is a doctoral researcher in the Intelligent Electrical Power Grids group at the Delft University of Technology. He obtained an M.Sc. in Energy Technologies from the Karlsruhe Institute of Technology and Uppsala University in 2015 and an engineer's degree in electronics from the University of Concepción in 2009. He has worked as a research assistant in the Fraunhofer Institute for Wind Energy and Energy System Technology and as a consulting engineer on energy-related projects in the public and private sectors. His research interests are related to co-simulation of complex and large-scale power systems.

PLACE
PHOTO
HERE

Arun Joseph obtained his bachelor degree (B.Tech) in electrical engineering from Calicut University, India in 2009 and master degree (M.Tech) in control system from Indian Institute of Technology, Kharagpur in 2012. He has worked as a research assistant in aerospace department of Indian Institute of Science, Bangalore and senior research fellow in power system division of Central Power Research Institute, Bangalore. Currently he is a doctoral researcher in the Intelligent Electrical Power Grids group at the Delft University of Technology and research areas include real-time model validation of power system using co-simulation techniques, hardware-in-the-loop methods.

PLACE
PHOTO
HERE

Kaikai Pan (M'16) received the B.Eng. and M. Eng. Degrees in measuring and control from Beihang University, Beijing, China, in 2012 and 2015, respectively. Currently he is working towards the Ph.D. degree in the Intelligent Electrical Power Grids group from Delft University of Technology, Delft, The Netherlands. His research interests include cyber-physical energy systems, cyber security of intelligent power grids, risk assessment for data attacks, and co-simulation techniques.

LIST OF FIGURES

LIST OF TABLES