Cosimulation of Intelligent Power Systems
Fundamentals, Software Architecture, Numerics, and Coupling

Palensky, Peter; Van Der Meer, Arjen A.; Lopez, Claudio David; Joseph, Arun; Pan, Kaikai

DOI
10.1109/MIE.2016.2639825

Publication date
2017

Document Version
Final published version

Published in
IEEE Industrial Electronics Magazine

Cosimulation of Intelligent Power Systems

Fundamentals, Software Architecture, Numerics, and Coupling

PETER PALENSKY,
ARJEN A. VAN DER MEER,
CLAUDIO DAVID LÓPEZ,
ARUN JOSEPH, AND KAIKAI PAN

Digital Object Identifier 10.1109/MIE.2016.2639825
Date of publication: 21 March 2017

Smart grids link various types of energy technologies—such as power electronics, machines, grids, and markets—via communication technology, which leads to a transdisciplinary, multidomain system. Simulation packages for assessing system integration of components typically cover only one subdomain, while simplifying the others. Cosimulation overcomes this by coupling subdomain 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. This article, part 1 of 2 on this subject, covers fundamental concepts. Part 2 will appear in a future issue of *IEEE Electrification Magazine* and cover applications.

Cosimulation's Value

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 the occurrence of an actual contingency (e.g., converter outage, load rejection, or line overloading) and in studying 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 widespread information and communication technology (ICT)-based infrastructure and power electronics in power systems, an important gap was bridged between technologies (e.g., ICT, smart sensing, power electronic equipment, and renewable energy sources), disciplines (e.g., consumers and producers inside the liberalized electricity market environment), and domains (e.g., electrical, thermal, telecommunication, energy storage, and 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 industrial electronics leads

1932-4529/17©2017IEEE

©ISTOCKPHOTO.COM/WUTWHANFOTO

Enhancing the power system with industrial electronics leads to cyberphysical or intelligent power systems.

to cyberphysical or intelligent power systems [1].

The increasing cyberphysical nature of the power system requires rethinking how its behavior should be addressed in planning and operation studies. Traditionally, interactions are addressed by simulating the subsystem of interest in detail—usually for a particular domain—while simplifying the remaining parts. This is the current approach, for instance, for power electronic devices. Such treatment, however, can lead to problems since the lack of detail in the characteristics of the overall system disregards underlying interactions. In intelligent power systems, therefore, phenomena need to be addressed holistically, because the reactions to events in one particular domain are now spread across multiple domains and cannot be safely simplified or disregarded.

The functionality needs of holistically analyzing intelligent power systems unfortunately clash. On the one hand, the system is heterogeneous in nature, which requires a method that can generically set up and process different models representing this heterogeneity. On the other hand, there is a need to represent each phenomenon of interest accurately (i.e., in terms of modeling detail) and efficiently (i.e., computationally and in terms of the time needed).

Heterogeneous systems can be analyzed through a number of approaches (Table 1), each with pros and cons. Experiments are often too expensive in the intelligent power systems domain, which limits the simulation runs to real time. General purpose multidomain simulation tools, however, have the capability of simulating heterogeneous components and small systems. Once the interactions in intelligent power systems become complex, they will not work correctly anymore because of 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 numerically optimized for particular purposes (e.g., load flow, dynamics, and transients). The same holds for telecommunication simulations. Unfortunately, this is true only for nonheterogeneous 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 toward simulation platforms that can handle multidomain systems with reasonable detail and speed. Coupled simulations, known as *cosimulations*, aim to fulfill the functionality needs by modeling multidomain systems using multiple simulation tools that act as a single integral simulation platform for the study [2].

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

Cosimulation [Figure 1(a)], on the other hand, works with specialized software packages that use validated model libraries and tailor-made solvers. A multiagent 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 and 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, multiagent-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 cosimulation setup. With this, the model choice is not limited to a single tool; rather, it is 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

TABLE 1 – THE ALTERNATIVES FOR ANALYZING INTELLIGENT, INTEGRATED POWER SYSTEMS.

ANALYSIS METHOD	PROS	CONS
Conduct 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 (multidomain)	Flexibility, ease of use	Bad scalability for systems
Couple a heterogeneous set of submodels	Well-suited tools and languages	Complex software coupling

information). Splitting systems into submodels also allows for distributed simulation to improve performance. Finally, the method allows multidisciplinary teams to work together and combine their knowledge.

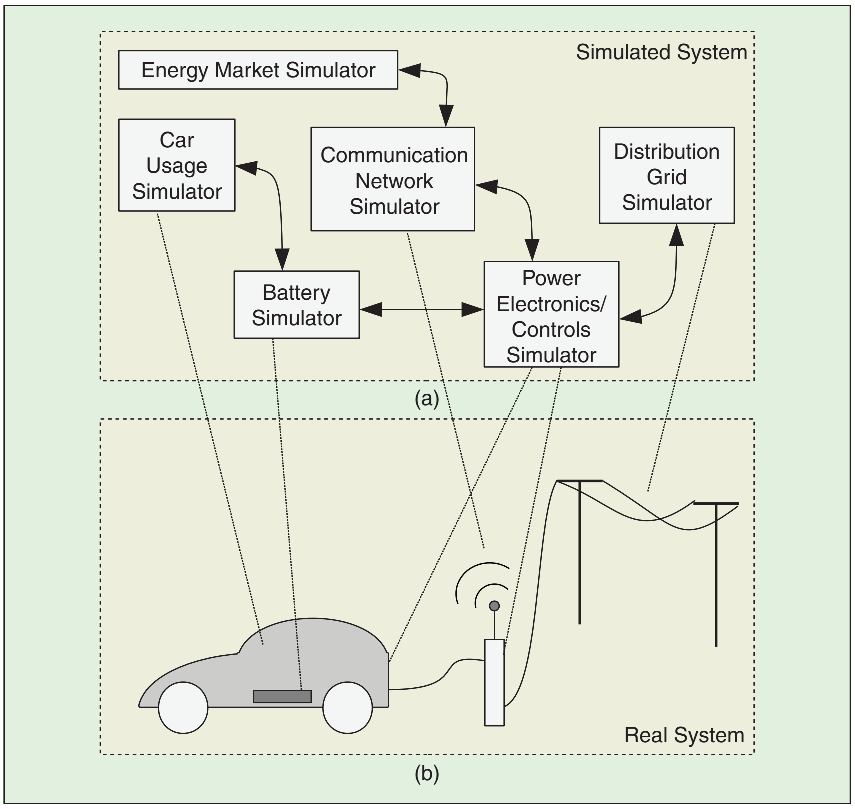

Simulated System
Energy Market Simulator
Car Usage Simulator
Communication Network Simulator
Distribution Grid Simulator
Battery Simulator
Power Electronics/ Controls Simulator
(a)
Real System
(b)

FIGURE 1 – A typical setup for cosimulating a complex system.

Basic Concepts of Cosimulation

A cosimulation 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 output of one simulator becomes the input of the other and vice versa. The variable exchange, time synchronization, and execution coordination are, in the most general case, facilitated in runtime by a so-called master algorithm, which orchestrates the entire cosimulation. The basic composition of a cosimulation is shown in Figure 2.

The Simulator

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 predicting 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 in which 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. In 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 most diverse. In the case of the power system, the models can be purely algebraic (as with steady-state simulations), purely composed of differential equations (as with electromagnetic transients and circuit simulations), or a combination of algebraic and differential equations (as with 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.

Cosimulation Master Algorithm

The task of a master algorithm is threefold:

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

The master algorithm leads all the simulators from the start throughout the simulation time. Upon starting, 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 (with model parameters, initial conditions of differential equations, and so on), the master assumes the role of stepping each simulator from one so-called communication point to the next. Individual simulators experience each time step between communication points (also

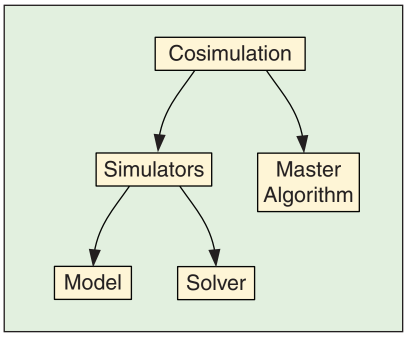


FIGURE 2 – The basic composition of a cosimulation.

known as *macro time steps*) as a sort of full simulation; that is, they receive input variables and a simulation duration in the beginning and then start simulating autonomously. At every communication point, the simulators exchange intermodel variables. Then, the master algorithm informs all the simulators about the next communication point, and each simulator proceeds until then. Once the last communication point is reached, the master algorithm ends the cosimulation. The interaction between the cosimulation master and each simulator is depicted in Figure 3.

Simulator Synchronization

Inside each macro time step, each simulator is allowed to proceed according to its own requirements. Simulators can, therefore, carry out a series of so-called micro time steps inside each macro time step, as shown in Figure 4. In most cases, the macro time step is chosen to be constant, but it is also possible to control its size at run time 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 cosimulation. If one submodel has a small micro step size and its variables are relevant for other submodels, it will slow down all the other simulators by imposing a small communication step size.

The synchronization needs for each cosimulation depend on the involved domains and their respective modeling method. The major parameters here are

- the test criteria that need to be met (e.g., qualitative behavior or 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, such as license availability and black-box modeling, that limit the choice of tools to be applied and hence restrict the freedom in synchronization alternatives.

A Software Perspective on Cosimulation

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 of the power system. The transmission network and its distribution networks are considered as a single integral part and are 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., through averaged representation of power electronic devices or through network reduction)
- choosing a faster numerical solver (e.g., adaptive time step sizes)
- providing more computational power (e.g., real-time simulation of high-voltage dc systems).

Model simplifications are acceptable as long as the user knows the price that must be paid for the simplifications. Quantitative figures on lost behavior or frequency limits are needed to decide if a simplification is acceptable for a given study. Reducing an eighth-order electric machine model to a second-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

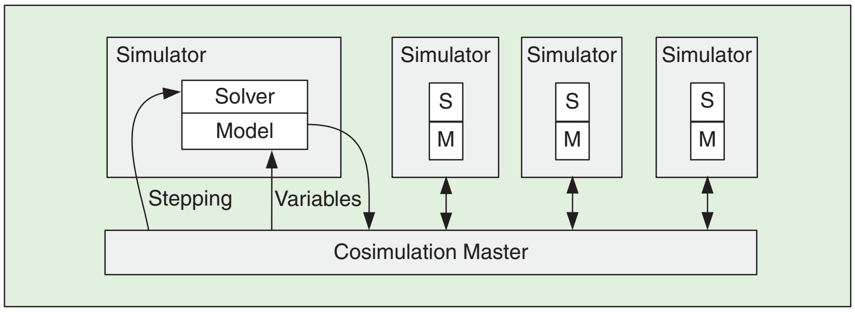


FIGURE 3 – The interaction between the cosimulator master algorithm and the simulators (S = solver; M = model).

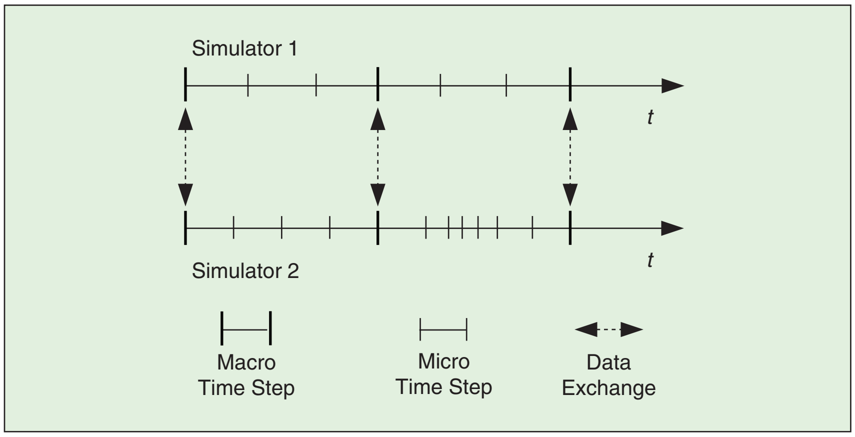


FIGURE 4 – An illustration of micro and macro time steps.

blocking—crucial information that is lost during model reduction.

Using a faster numerical solver also has 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 nonlinear parts, the choice of solvers narrows, and their speed is determined only by their implementation.

One way to increase computational power is to use faster computers, i.e., with more memory and cache, faster processing unit, and so on. The passage of time is on our side in this respect: computers grow faster every year. Another way is to separate the problem into subproblems that can be solved in parallel. This can be an easy task in the case of parallel computing: the computational problem consists of doing the same analysis many times with different parameters [5]. The analyses can easily be done on separate machines, without any communication or synchronization during the simulation runs. 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.

There is heterogeneity when the power system dynamically interacts with other systems (e.g., the heat network, markets, and so forth) whose modeling method is fundamentally different than the chosen method for 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 is the ancillary services and grid support in distributed generation, managed by a centralized controller. Even within 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 to have a detailed view of some specific components. Heterogeneity of models can be approached by using a universal modeling platform or via cosimulation (see Table 1).

The simulators that compose a cosimulation need to exchange data with each other during various stages of the simulation workflow.

Figure 5 shows the four fundamental options of solver versus model:

1) The trivial case involves one solver and one model.
2) 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 the real-time digital simulator [6], where one model is compiled onto multiple computational targets and executed in parallel on dedicated hardware.
3) Hybrid simulation involves multiple types of modeling environments and languages to make the modeling task easier. However, the models still form a monolithic unity that can be solved by a single solver. The only advantage is that the individual parts or aspects of the model receive a specialized modeling method (e.g., graphical, different languages and libraries, and so forth).
4) Cosimulation combines the advantages—but also the challenges—of the other three.

Simulator Interfacing

The simulators that compose a cosimulation need to exchange data with each other during various stages of the simulation workflow (e.g., model instantiation, initialization, runtime, and data export). The interface between them can be implemented via shared memory, if all the simulators can access one common memory, or via network communication protocols. For instance, various simulation packages offer a number of semistandard interfaces based on certain proprietary application program interfaces (APIs) [7], object linking and embedding for process control, 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: model exchange and cosimulation. Both are equally useful, depending on what needs to be achieved.

The FMI for model exchange (Figure 6) exposes a compiled numerical

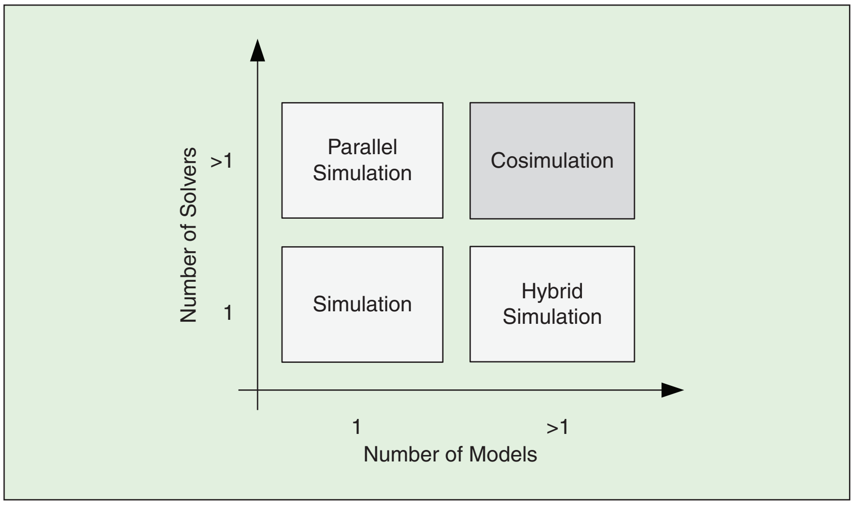


FIGURE 5 – The four types of simulation: normal (i.e., monolithic), parallel, hybrid, and cosimulation.

model to a solver/simulator with a standard interface to initialize the states, execute a time step, determine derivatives, and so forth. Wrapping a model into such an interface results in a functional mockup unit (FMU), a convenient way in which component manufacturers can provide their customers with 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 others in the intended system.

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

Two important concepts with an FMI are FMU import (where an existing FMU is plugged in and used in a supermodel) and export (where a model is compiled into an FMU to be then imported somewhere else). If the model or 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 reinitialization and restart of the tool.

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

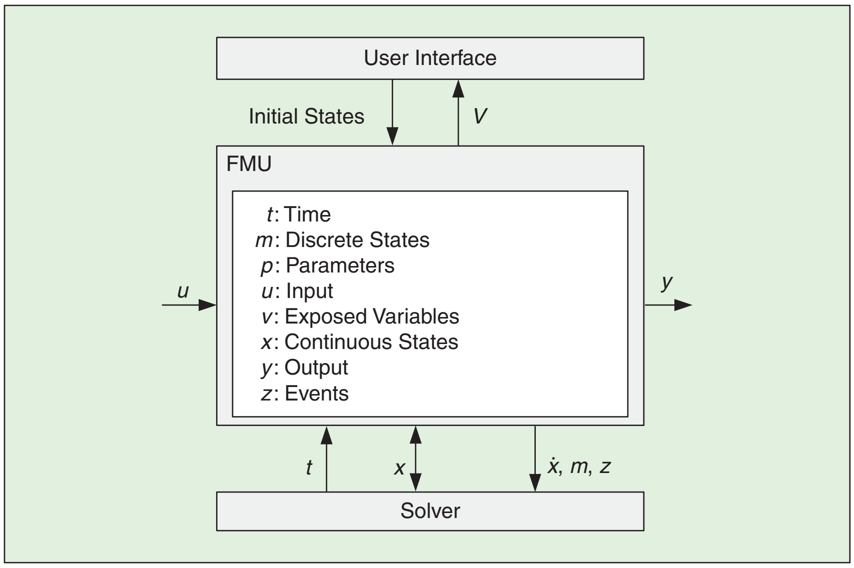


FIGURE 6 – The model exchange variant of the FMI.

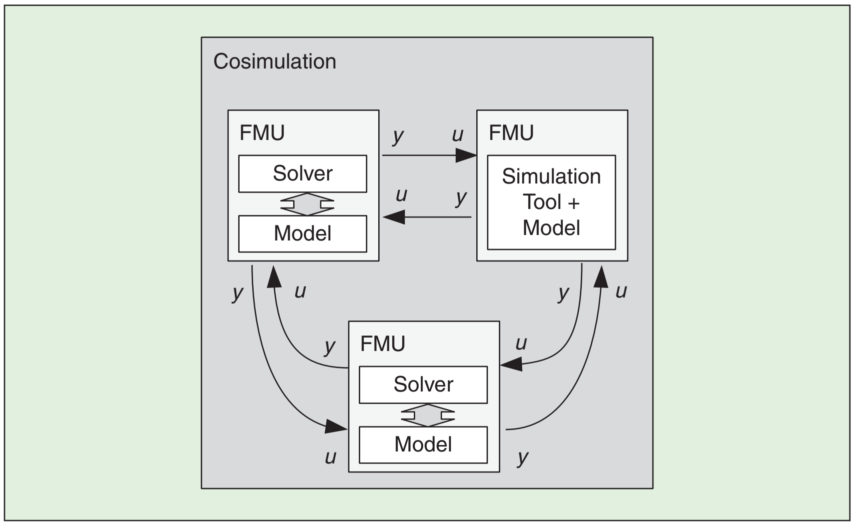


FIGURE 7 – The use of the cosimulation variant of the FMI (u = input; y = output).

Cosimulation Orchestration

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

Synchronizing the advancement in model time requires direct access to the integrators or schedulers of the individual simulation packages. Often, one of the packages acts as a 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.

The master algorithm therefore has a number of interfaces. In the case of an 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 cosimulation settings are made 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, as it were, to synchronize at the correct moment. Not many legacy simulators have this ability, 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 the same machine, using shared memory or messages, or it 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 of a complete specification of a central entity (in this case the Run-Time Infrastructure) 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, such as synchronization, service lookup, initialization, and distributed computing.

What we might call 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, such as 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).

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 becomes infeasible. A way out of this is to enhance the parameter choice during the optimization process. Design of experiments, a method dating back to the 1930s, 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 just to couple the simulators; the scripting and optimization details are of equal importance.

Once a set of simulations is defined, it is the art of cosimulation to coordinate and orchestrate their execution.

Practical Considerations

An important aspect with regard to software for cosimulation settings is their openness for interfacing. A simulator can

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

Often, software packages are available only 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. The overhead and latency of these communication protocols can dramatically slow down the system simulation. Closely coupled subsystems should therefore be hosted on 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: a standardized interface in an open-source software package.

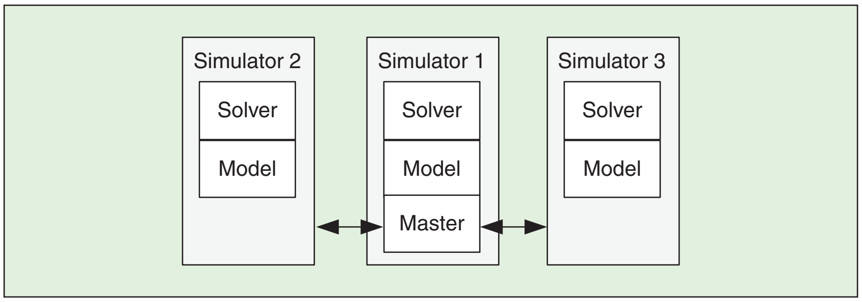


FIGURE 8 – One central simulator acts as master.

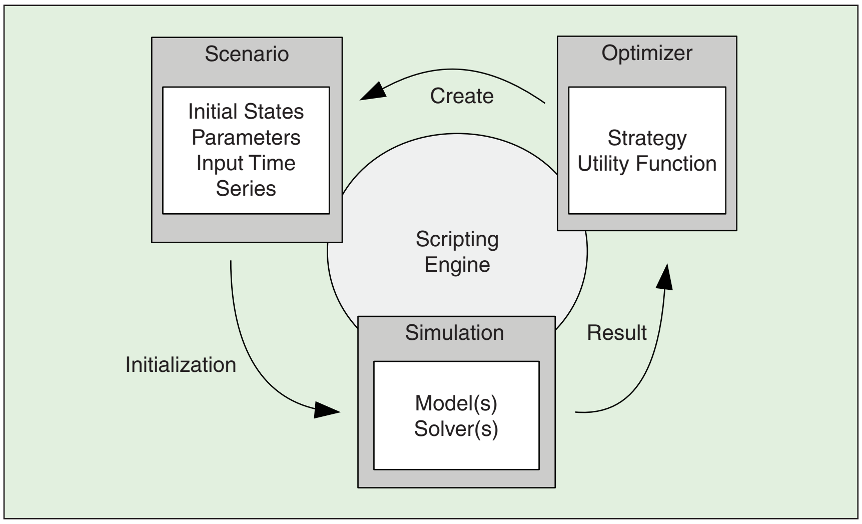


FIGURE 9 – A flowchart of simulation runs.

Another limitation of black-box, legacy simulators is that they usually do not offer a sophisticated API for simulation control. Step sizes cannot be chosen, and rollback is practically impossible. In the worst case, they can be used only in an initialize–simulate–terminate fashion; that is, 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 for interfacing other simulators.

Numerical Solution of Coupled Models

Each model in a cosimulation is solved independently using an individual 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 cosimulation 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 output, let us consider a case with two subsystems modeled through differential algebraic equations (DAEs):

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

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

where $\mathbf{u}_i$ are the input 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 output of each subsystem becomes the input of the other, i.e., that the coupling conditions

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

are fulfilled, two main approaches can be followed to simulate the coupled system. In the traditional simulation approach, (1) and (2) are composed into one system of DAEs according to (3), using the chosen simulation tool, and are then solved with only one numerical solver so that system output is 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, (1)–(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 cosimulation, (1)–(3) are weakly coupled, since each subsystem is solved independently and outputs are exchanged between subsystems only at the communication points in time defined in $\mathbf{t}$. Within each macro time step $t_k \to t_{k+1}$, 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 (3) is guaranteed at the macro time steps defined in $\mathbf{t}$, but not that of (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 nonzero integration step.

The reason for these numerical instabilities and/or inaccuracies lies in the appearance of algebraic loops between subsystems and the methods used to overcome them. In fact, zero-stability of cosimulations 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 (1)–(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 an output for t_{k+1}. However, subsystem 2 is unable to produce this output, since it also depends on the output of subsystem 1 at t_{k+1} to do so.

Several methods for overcoming algebraic loops in cosimulations 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 the dynamics in the coupling equations. However, these methods may severely affect the quality of the results and the tradeoffs between computational performance and accuracy or stability that are to be made.

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 cosimulation can run in parallel or if they must be executed one after the other. The consequences of choosing one type or the

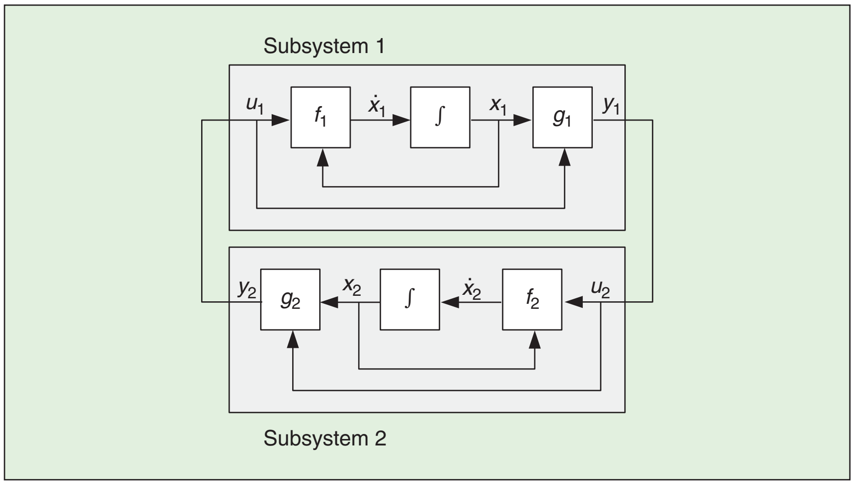


FIGURE 10 – An algebraic loop in the cosimulation of the two subsystems from (1)–(3).

other go beyond the computational performance of the cosimulation, since each communication sequence overcomes algebraic loops in a slightly different way.

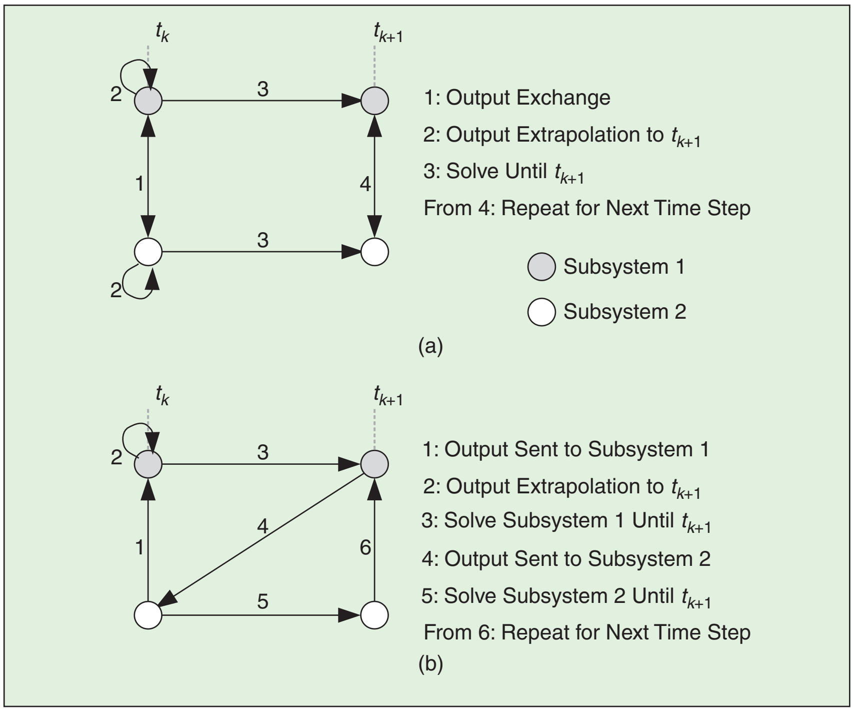
Parallel Sequence

In parallel communication sequences, also known as the *Jacobi* type, all the subsystems that compose the cosimulation 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} to advance from t_k to t_{k+1}, the output of each subsystem at t_{k+1} is extrapolated from the output at t_k. The simplest way to achieve this is through zero-order hold extrapolation. That is, the output at t_{k+1} is 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 the simulators can be executed on distributed computers, which can aid the overall computational performance of the cosimulation. The main disadvantage is that each simulator must predict the future output of the remaining simulators by means of extrapolation, making the accuracy and stability of the cosimulation completely dependent on the accuracy of this prediction [18].

Serial Sequence

In serial communication sequences, also known as the *Gauss–Seidel* type, subsystems are solved one after the other. 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} to advance from t_k to t_{k+1}, the output of subsystem 2 at t_{k+1} is extrapolated from its output at t_k. Once subsystem 1 has advanced to t_{k+1}, its output becomes 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 the simulators are executed one after the other, a greater number of coupled simulators results in longer execution times [15].



t_k
t_{k+1}
2
3
1
4
3
2
1: Output Exchange
2: Output Extrapolation to t_{k+1}
3: Solve Until t_{k+1}
From 4: Repeat for Next Time Step
Subsystem 1
Subsystem 2
(a)
t_k
t_{k+1}
2
3
1
4
6
5
1: Output Sent to Subsystem 1
2: Output Extrapolation to t_{k+1}
3: Solve Subsystem 1 Until t_{k+1}
4: Output Sent to Subsystem 2
5: Solve Subsystem 2 Until t_{k+1}
From 6: Repeat for Next Time Step
(b)

FIGURE 11 – The communication sequences for cosimulation (weakly coupled) for one macro time step $t_k \rightarrow t_{k+1}$: (a) parallel (Jacobi) type and (b) serial (Gauss–Seidel) type.

Iterative Sequences

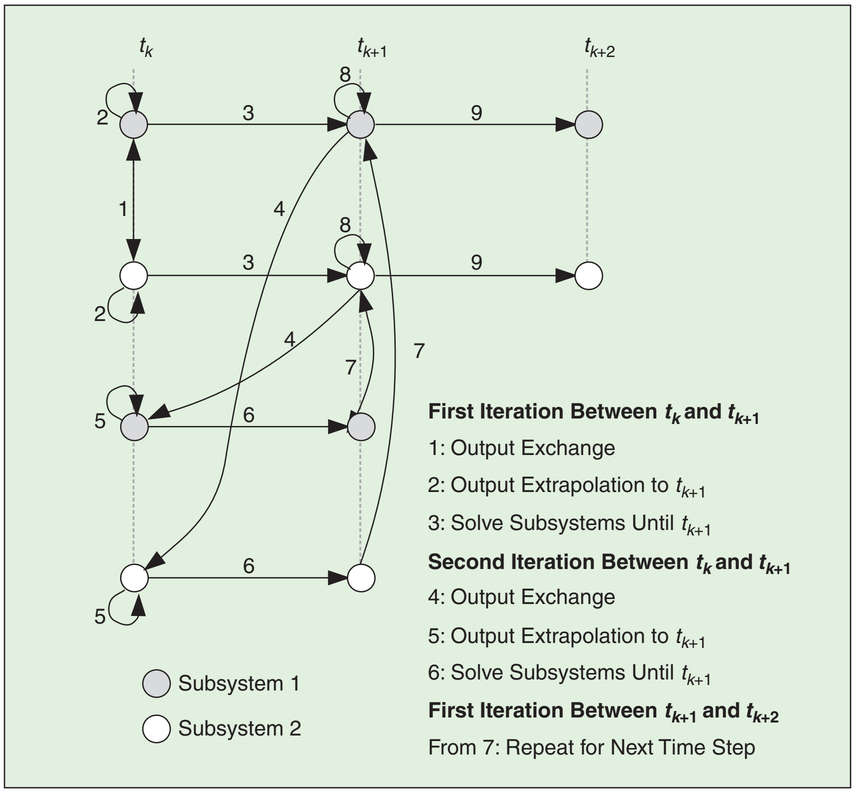
Iterative sequences can be derived from both parallel and serial sequences. Figures 12 and 13 show diagrams of the iterative version of each sequence type. Although only two iterations are displayed in the figures, these sequences can be extended to any number of iterations. In practice, iterations are 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 noniterative 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 noniterative counterparts, a much higher computational effort is required to run the cosimulation.

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 requirement that all the simulation tools involved in the cosimulation 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.

Coupling Variables and Models

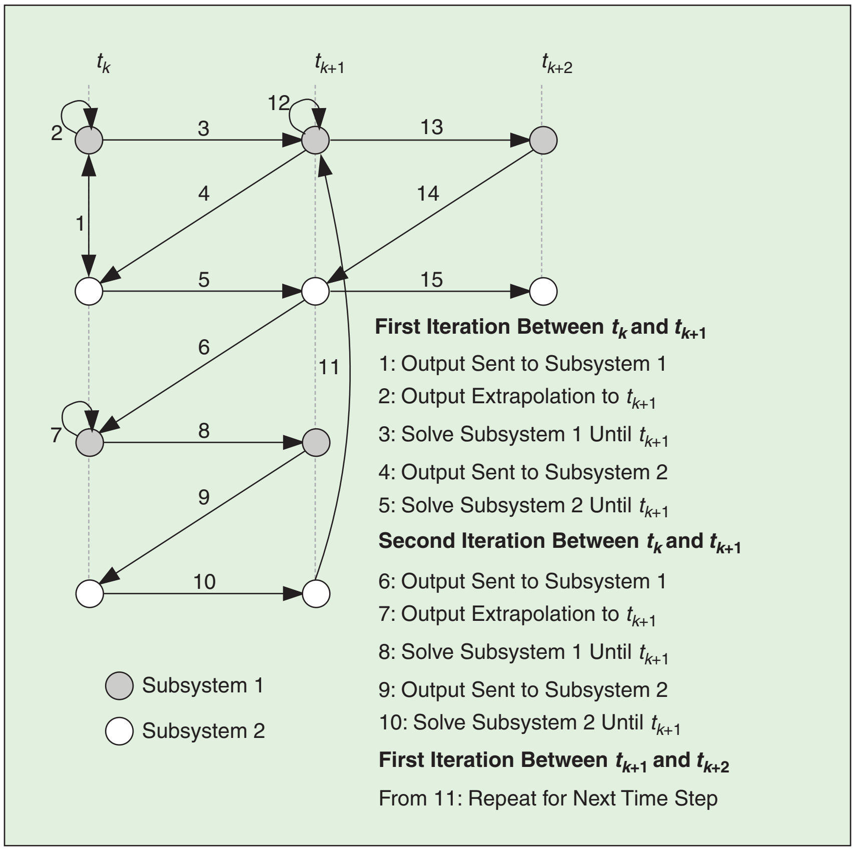
The choice of coupling variables (i.e., the variables that are exchanged between models) or, equivalently, the way a system is partitioned into subsystems influences the accuracy and stability of the cosimulation. For example, in the case of mechanical systems, the choice of different combinations of 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 (1)–(3), extrapolation



t_k
t_{k+1}
t_{k+2}
1
2
3
4
5
6
7
8
9
First Iteration Between t_k and t_{k+1}
1: Output Exchange
2: Output Extrapolation to t_{k+1}
3: Solve Subsystems Until t_{k+1}
Second Iteration Between t_k and t_{k+1}
4: Output Exchange
5: Output Extrapolation to t_{k+1}
6: Solve Subsystems Until t_{k+1}
First Iteration Between t_{k+1} and t_{k+2}
From 7: Repeat for Next Time Step
Subsystem 1
Subsystem 2

FIGURE 12 – The iterative (strongly coupled) communication sequences for the cosimulation—parallel (Jacobi) type—for one macro time step $t_k \to t_{k+1}$.



t_k
t_{k+1}
t_{k+2}
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
First Iteration Between t_k and t_{k+1}
1: Output Sent to Subsystem 1
2: Output Extrapolation to t_{k+1}
3: Solve Subsystem 1 Until t_{k+1}
4: Output Sent to Subsystem 2
5: Solve Subsystem 2 Until t_{k+1}
Second Iteration Between t_k and t_{k+1}
6: Output Sent to Subsystem 1
7: Output Extrapolation to t_{k+1}
8: Solve Subsystem 1 Until t_{k+1}
9: Output Sent to Subsystem 2
10: Solve Subsystem 2 Until t_{k+1}
First Iteration Between t_{k+1} and t_{k+2}
From 11: Repeat for Next Time Step
Subsystem 1
Subsystem 2

FIGURE 13 – The iterative (strongly coupled) communication sequences for cosimulation—serial (Gauss–Seidel) type—for one macro time step $t_k \to t_{k+1}$.

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 cosimulation 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 cosimulated 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 (both 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.

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 the two equations in (3) become

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

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 classic 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].

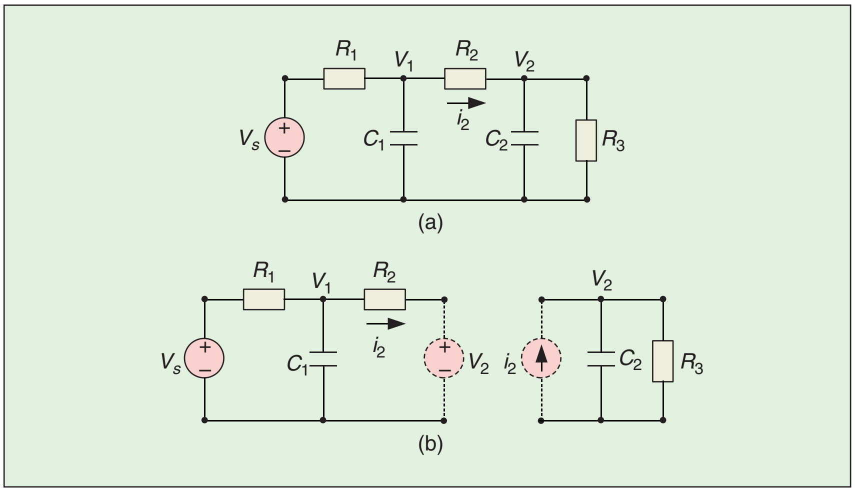


FIGURE 14 – The subsystem coupling at a point of weak dynamic interaction [21]: (a) the monolithic model and (b) cosimulated subsystems.

Error Estimation and Step-Size Control

The ability to estimate the local truncation error of a cosimulation 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 cosimulation 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 output 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 cosimulation 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 lies 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 the 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 cosimulation 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, called 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 cosimulation outputs and the one calculated from energy conservation are attributed to cosimulation 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 (ODEs) so that the local error remains within an upper and a lower boundary. For example, [15] proposes the use of a proportional integral controller, as presented in [30]. Since in a cosimulation a different local truncation error estimation can obtained for the output of each subsystem, the size of the macro time step must be chosen taking into account the most demanding of all the subsystems.

Coupling Power System and ICT Simulators

The integration of power systems, automated devices, and ICT gives intelligent power grids the character of a cyberphysical system (CPS) [31]. On the 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:* The real ICT products (telecommunication switches, controllers, and so forth) are used, and the setup runs in real time.
- *Emulated ICT hardware:* The real binary code (e.g., of switches) is

The integration of power systems, automated devices, and ICT gives intelligent power grids the character of a cyberphysical system.

executed on an emulation platform that provides the same hardware properties, such as memory and speed, as 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 or imitated. This setup can run in nonreal time (ideally faster than real time).
- *Full simulation*: All ICT elements (e.g., switches) are simulated or 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 nonreal time.

This article mainly covers the last variant, full simulation, which opens up questions on how to synchronize the various parts, especially in a cosimulation 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 cyberattacks and vulnerabilities from the ICT domain challenge this assumption, as ICT assets could be used to cause damage to the electric components in a coordinated manner [33].

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

Modeling and Simulation Challenges

As with all digital systems, communication networks are modeled as a sequence of discrete events (e.g., sending and receiving packets, packet buffer overflows, and so on), while power systems are typically modeled as continuous-time functions using DAEs, 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 applies adaptive time-step-size control
- 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 multidomain 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 hence can be used only for component analysis and simple-use cases, not for fully fledged system studies.

As touched upon before, cosimulation 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 cosimulation are a challenge. The interdependency of hybrid models from the power system and ICT parts makes it hard to identify where the simulation error comes from. Different synchronization methods in cosimulation also impact the simulation accuracy.
- The interoperability of the various simulators requires standardized interfaces (see the HLA and FMI discussion in the section "A Software Perspective on Cosimulation").

Synchronization of Discrete and Continuous Simulators

When building a cosimulation 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 happen either conservatively or optimistically. Conservative synchronization guarantees strict processing of logical time by a time stamp order. The optimistic alternative 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 the literature, synchronization methods are mainly subdivided into three categories: point based, event driven, and master–slave [38].

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 the case of ICT models, may even have a stochastic nature [39].

The point-based synchronization method may introduce inaccuracies in the cosimulation. 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., to 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.

Event Driven

In [42], a global event-driven cosimulation 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 individually handles corresponding control for power system events and communication network events. 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.

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

One way to increase computational power is to use faster computers, i.e., with more memory and cache, faster processing unit, and so on.

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 cosimulation. 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 as follows:

- Events generated in the slave cannot be communicated to the master immediately.

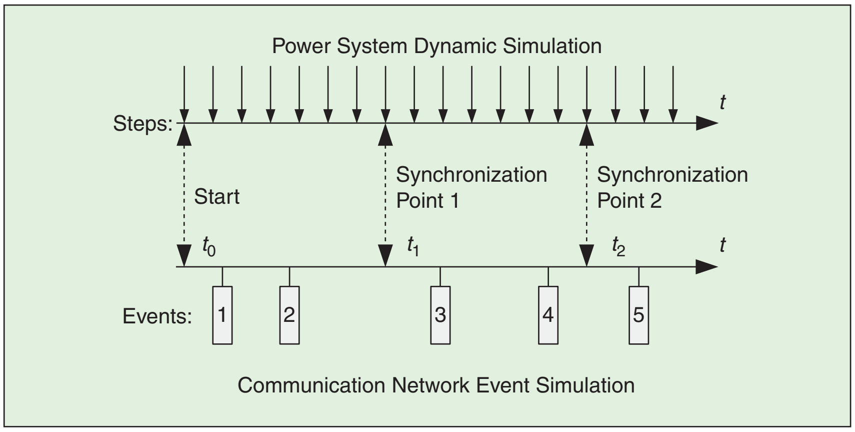
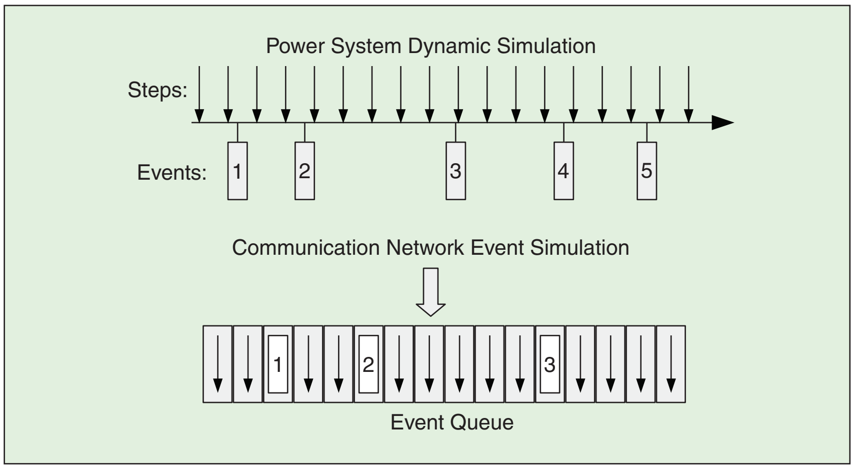


FIGURE 15 – The point-based synchronization method.



Power System Dynamic Simulation
Steps:
Events:
1
2
3
4
5
Communication Network Event Simulation
1
2
3
Event Queue

FIGURE 16 – An event-driven synchronization method based on [42].

The accuracy and efficiency in coupling depend on the simulation tools selected, the interfaces, and the synchronization mechanisms.

- Execution is typically sequential.
- Scalability issues inhibit the integration of an arbitrary number of simulators.

The latter can potentially be overcome if one dedicated master algorithm is used to orchestrate all the simulators that act as slaves. All the 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 the slaves as the next synchronization point.

Figure 18 shows an example where the master tells slave 2 (the power system simulator) the time t_{i1}, which is the time of the first event in slave 1 (the communication network simulator). Each slave executes a 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.

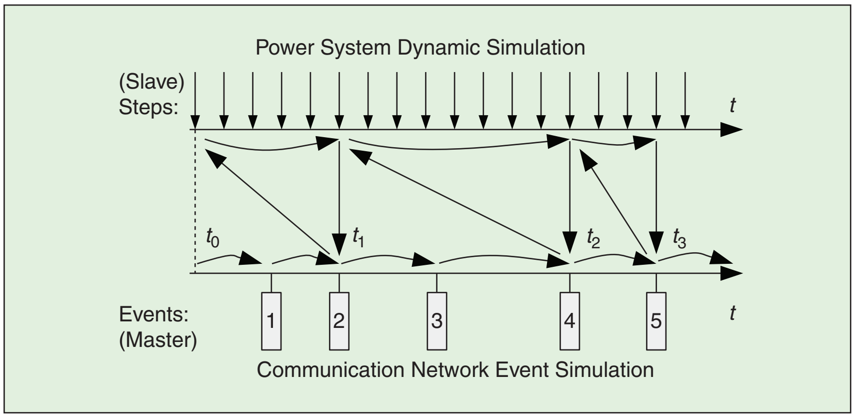

Power System Dynamic Simulation
(Slave)
Steps:
t
t_0
t_1
t_2
t_3
Events:
(Master)
1
2
3
4
5
t
Communication Network Event Simulation

FIGURE 17 – A master–slave synchronization method: one simulator acts as the master.

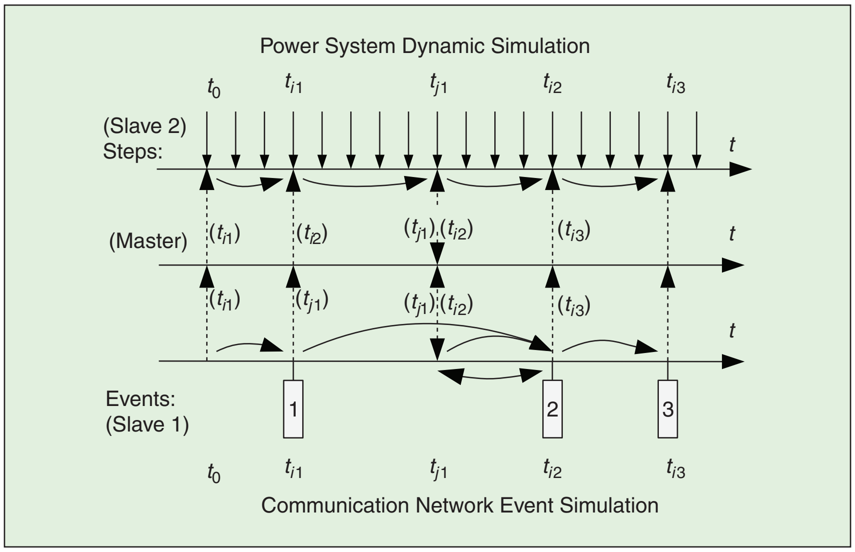


FIGURE 18 – A master–slave synchronization method: using a dedicated master component.

Practical Considerations

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

From the preceding, it can be deduced that in many cases the designers have to make a tradeoff between accuracy, efficiency, and scalability. It should be noted that the accuracy and efficiency in coupling depend on the simulation tools selected, the interfaces, and the synchronization mechanisms. This also implies the level of control the designer could have on the simulation tools (compare the black-box versus open simulators, discussed in the section “A Software Perspective on Cosimulation”).

Solvers for Better Synchronization Between Continuous ODEs and Discrete Event Models

The synchronization methods discussed in the previous subsection are a result 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 cosimulations, 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, 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.

Conclusions

In this article, the fundamental concepts behind cosimulation of intelligent power systems were described. Software interfaces, numerical aspects, and coordinating individual simulators were discussed for both the power engineering and the ICT domain.

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

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

Part 2 on this subject will provide a dive into practical aspects of cosimulation, such as how to integrate hardware-in-the-loop simulators, and will give an example of how complex smart grid questions can be analyzed by combining electromechanical with electromagnetic transients simulations.

Cosimulation has the advantage of easier modeling, since the individual subdomains are described within their native tools and languages.

Acknowledgment

This work is partly supported by the European Community's Horizon 2020 Program through the project ERIGrid: European Research Infrastructure supporting Smart Grid Systems Technology Development, Validation and Roll Out (grant 654113).

Biographies

Peter Palensky (palensky@ieee.org) is a full professor of intelligent electric power grids at Delft University of Technology, The Netherlands. Before that, he was principal scientist at the Austrian Institute of Technology; an associate professor in the Department of Electrical, Electronic, and Computer Engineering, University of Pretoria, South Africa; a university assistant at the Vienna University of Technology, Austria; and a researcher at the Lawrence Berkeley National Laboratory, California. He is active in international organizations such as the IEEE. His main areas of research are energy automation networks and modeling intelligent energy systems. He is a Senior Member of the IEEE.

Arjen A. Van Der Meer (a.a.vandermeer@tudelft.nl) received his B.Sc. degree in electrical engineering at Noordelijke Hogeschool Leeuwarden University of Applied Sciences, Leeuwarden, The Netherlands, in 2006. In 2008, he received his M.Sc. degree (honors) in electrical engineering from Delft University of Technology, The Netherlands. Currently, he is working at Delft University of Technology toward his Ph.D. degree on the grid integration of offshore voltage sourced converter–high-voltage dc grids. 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. He is a Member of the IEEE.

Claudio David López (c.d.lopez@tudelft.nl) received his M.Sc. degree in energy technologies from the Karlsruhe Institute of Technology, Germany, and Uppsala University, Sweden, in 2015 and an engineer's degree in electronics from the University of Concepción, Chile, in 2009. He is a doctoral researcher with the Intelligent Electrical Power Grids group at Delft University of Technology, The Netherlands. 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 cosimulation of complex and large-scale power systems. He is a Member of the IEEE.

Arun Joseph (arun.joseph@tudelft.nl) received his B.Tech. degree in electrical engineering from the University of Calicut, Malappuram, India, in 2009 and his M.Tech. degree in control systems from the Indian Institute of Technology, Kharagpur, in 2012. He has worked as a research assistant in the Aerospace Department of the Indian Institute of Science, Bangalore, and as senior research fellow in the Power System Division of the Central Power Research Institute, Bangalore, India. Currently, he is a doctoral researcher with the Intelligent Electrical Power Grids group at the Delft University of Technology, The Netherlands. His research areas include real-time model validation of power systems using cosimulation techniques and hardware-in-the-loop methods. He is a Member of the IEEE.

Kaikai Pan (k.pan@tudelft.nl) received his B.Eng. and M.Eng. degrees in measuring and control from Beihang University, Beijing, China, in 2012 and 2015, respectively. Currently, he is working toward his Ph.D. degree in the Intelligent Electrical Power Grids group at Delft University of Technology, The Netherlands. His research interests include cyberphysical energy systems, cybersecurity of intelligent power grids, risk assessment for data attacks, and cosimulation techniques. He is a Member of the IEEE.

References

[1] A. Vojdani, "Smart integration," *IEEE Power Energy Mag.*, 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-physical 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 *Proc. 38th IEEE Conf. Industrial Electronics (IECON 2012)*, 2012, pp. 3582–3587.

[4] P. Palensky, E. Widl, and A. Elsheikh, "Simulating cyber-physical energy systems: Challenges, tools and methods," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 3, pp. 318–326, 2013.

[5] G. V. Wilson, "A glossary of parallel computing terminology," *IEEE Concurrency*, 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 *Proc. North American Power Symp. (NAPS)*, Pullman, WA, Sept. 2014.

[7] C. Dufour, S. Abourida, and J. Belanger, "Hardware-in-the-loop simulation of power drives with rt-lab," in *Proc. 2005 Int. Conf. Power Electronics, Drives, Energy 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 *Proc. Workshop Modeling and Simulation Cyber-Physics Energy Systems*, 2013.

[9] "Functional mock-up interface for co-simulation," MODELISAR, Information Technology for European Advancement, Eindhoven, The Netherlands, Tech. Rep., ITEA 2-07006, 2010.

[10] J. S. Dahmann, R. M. Fujimoto, and R. M. Weatherly, "The DoD high level architecture: An update," in *Proc. Simulation Conf. Proc.*, vol. 1, Washington, D.C., Dec. 1998, pp. 797–804.

[11] W. Muller and E. Widl, "Using FMI components in discrete event systems," in *Proc. Workshop Modeling and Simulation Cyber-Physics 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 *Proc. Fourth IEEE/PES Innovative Smart Grid Technologies Europe (ISGT EUROPE)*, 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, Sept. 2012.

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

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

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

[17] M. Benedikt, D. Watzenig, and A. Hofer, "Modelling and analysis of the non-iterative coupling process for co-simulation," *Math. Computer Modelling of Dynamical Syst.*, 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 *Proc. First Joint Int. Conf. Multibody System Dynamics (IMSD)*, Lappeenranta, Finland, 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, Germany, 2014.

[20] B. Schweizer, P. Li, and D. Lu, "Explicit and implicit cosimulation methods: Stability and convergence analysis for different solver coupling approaches," *J. Computational and Nonlinear Dynamics*, vol. 10, no. 5, pp. 051007-1–051007–12, Sept. 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," *J. Basic Eng.*, vol. 90, no. 2, pp. 195–200, June 1968.

[23] R. Braun and P. Krus, "An explicit method for decoupled distributed solvers in an equation-based modelling language," in *Proc. Sixth Int. Workshop Equation-Based Object-Oriented Modeling Languages and Tools*, 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 *Proc. Ninth Int. Modelica Conf.*, Munich, Germany, Sept. 2012.

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

[26] T. Meyer and B. Schweizer, "Error estimation approach for controlling the comunication step size for semi-implicit co-simulation methods," *Proc. Appl. Math. and Mechanics*, 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 - J. Appl. Math. and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 95, no. 9, pp. 911–938, Sept. 2015.

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

[29] W. Borutzky, "Bond graph based physical systems modelling," in *Bond Graph Methodology*. London, U.K.: Springer-Verlag, 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 Math.*, vol. 28, no. 2, pp. 270–287, June 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 Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2364–2373, June 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*, S. K. Khaitan, J. D. McCalley, and C. C. Liu, Eds. Berlin, Germany: Springer-Verlag, 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 Trans. Smart Grid*, vol. 6, no. 5, pp. 2464–2475, Sept. 2015.

[34] H. L. Vangheluwe, "DEVS as a common denominator for multi-formalism hybrid systems modelling" in *Proc. IEEE Int. Symp. Computer-Aided Control System Design (CACSD 2000)*, Anchorage, Alaska, Sept. 2000, pp. 129–134.

[35] M. S. Branicky, V. Liberatore, and S. M. Phillips, "Networked control system co-simulation for co-design," in *Proc. American Control Conf.*, vol. 4, Denver, Colorado, June 2003, pp. 3341–3346.

[36] D. Henriksson, A. Cervin, and K.E. Årzén, "Truetime: Simulation of control loops under shared computer resources," in *Proc. 15th Int. Federation Automat. Control World Congr. Automatic Control*, Barcelona, Spain, July 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 *Proc. IEEE Int. Conf. Smart Grid Communications (SmartGridComm)*, 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 Elect. Eng.*, vol. 40, no. 1, pp. 270–288, Jan. 2014.

[39] V. Liberatore and A. Al-Hammouri, "Smart grid communication and co-simulation," in *Proc. 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 *Proc. IEEE EUROCON 2013*, Zagreb, Croatia, July 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 *Proc. 2014 Sixth Int. Conf. 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 Trans. 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 *Proc. IEEE Power Engineering Society General Meeting*, Tampa, FL, June 2007.

[44] K. Mets, J. A. Ojea, and C. Develder, "Combining power and communication network simulation for cost-effective smart grid analysis," *IEEE Commun. Surveys Tuts.*, 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 Trans. Smart Grid,* vol. PP, no. 99, 2016.

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