Reducing the need for communication in remote, natural waveform co-simulations of electrical power systems

An adaptive approach

López, Claudio

DOI
10.4233/uuid:a0fbc6f9-60e0-4cba-930f-9471911d48a4

Publication date
2021

Document Version
Final published version

Citation (APA)
López, C. (2021). *Reducing the need for communication in remote, natural waveform co-simulations of electrical power systems: An adaptive approach* . [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:a0fbc6f9-60e0-4cba-930f-9471911d48a4

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Reducing the Need for Communication in Remote, Natural Waveform Co-Simulations of Electrical Power Systems

An Adaptive Approach

Reducing the Need for Communication in Remote, Natural Waveform Co-Simulations of Electrical Power Systems

An Adaptive Approach

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op donderdag 9 september 2021 om 15:00 uur

door

Claudio David LÓPEZ TORRES

Master of Science in Energy Technology,
Karlsruher Institut für Technologie, Duitsland
Uppsala universitet, Zweden

Ingeniero Civil Electrónico,
Universidad de Concepción, Chili

geboren te Concepción, Chili.

Dit proefschrift is goedgekeurd door de

promotor: Prof. dr. P. Palensky
copromotor: Dr. M. Cvetković

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. P. Palensky,	Technische Universiteit Delft, promotor
Dr. M. Cvetković,	Technische Universiteit Delft, copromotor

Onafhankelijke leden:

Prof. dr. ir. C. Vuik	Technische Universiteit Delft, Nederland
Prof. dr.-ing. K. Strunz	Technische Universität Berlin, Duitsland
Prof. dr. S. Lehnhoff	Universität Oldenburg, Duitsland
Dr. T. Strasser	Austrian Institute of Technology, Oostenrijk
Dr. B. Palmintier	National Renewable Energy Laboratory, Verenigde Staten
Prof. dr. ir. J. Rueda Torres	Technische Universiteit Delft, Nederland, reservelid

Keywords: AC systems, co-simulation, electromagnetic transient, multiscale, power systems, simulation.
Printed by: Gildeprint.
Front & Back: Major Mono Display.

ISBN 978-94-6384-243-3

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

A Orsika.

Contents

Summary

Electrical power systems are becoming more interconnected and technologically diverse to accommodate ever increasing shares of non-dispatchable generation. These changes are imposing new requirements on the simulation of electrical power systems. One of these requirements is that simulations integrate models of different subsystems, developed by different experts, from different organizations, which may not wish to disclose the information embedded in their models. This, to study the interactions between neighboring, interconnected grids, or between existing grids and new devices. Another requirement is that they reproduce phenomena in a wider range of timescales, to study the interactions between subsystems with slow and fast dynamic behavior. One way for electrical power system simulations to comply with these requirements is with remote, natural waveform co-simulation. Co-simulation is a model integration approach in which each subsystem is simulated in a different simulator. These simulators exchange interface variables, at runtime, to represent interactions between the subsystems. Since the simulators can interact remotely, over a communication network, co-simulation has the advantage that the organization that owns the model needs not disclose it. It also has the advantage that each model can be simulated with the simulator for which it was intended, so organizations that use different simulators can collaborate without having to translate their models. If such a co-simulation is performed using natural waveform models, at a high time resolution, then it is also possible to reproduce a wide range of timescales, from slow to very fast phenomena. But the fact that such a co-simulation is performed remotely, with the communication delays this entails, and that its high time resolution translates into a high communication rate, make it rather slow. Thus, it is desirable to reduce the need for inter-simulator communication. In this thesis I explore a solution to this communication challenge, based on the hypothesis that slower phenomena are easier to predict. If the co-simulated phenomena can be classified as predictable according to some criterion, it should be possible to find expressions that predict interface variables, and that each simulator can use to compute its own inputs instead of expecting inputs to be communicated. I propose a criterion for classifying phenomena as predictable or unpredictable, as well as methods for finding these expressions based on an interpolated Fourier transform and Taylor-Kalman filters. Additionally, I propose a co-simulation algorithm where the simulators compute their own inputs while the co-simulated phenomena are predictable. After applying these ideas to the co-simulation of two different test systems, I was able to reduce the need for communication up to 60%. A co-simulation framework with these characteristics is a step towards more descriptive models and better performing simulations, and a tool that increases our ability to take better advantage of existing energy infrastructure, as well as to develop it further.

Samenvatting

Elektrische energiesystemen worden steeds meer onderling verbonden en technologisch diverser om het groeiende volume aan energie afkomstig van niet-aan/uitschakelbare energiebronnen te kunnen verwerken. Deze veranderingen stellen nieuwe eisen aan de simulatie van elektrische energiesystemen. Een van deze vereisten is dat simulaties modellen van verschillende subsystemen integreren, ontwikkeld door verschillende experts, van verschillende organisaties, die de informatie die hun modellen bezit mogelijk niet willen vrijgeven. Dit om de interacties tussen naburige, onderling verbonden netwerken of bestaande netwerken en nieuwe apparaten te bestuderen. Een andere vereiste is dat ze fenomenen reproduceren in een breder scala aan tijdschalen, om zo de interacties tussen subsystemen met langzaam en snel dynamisch gedrag te bestuderen. Een manier voor elektrische energiesystemen simaluties om aan deze vereisten te voldoen, is met golfvorm co-simulatie op afstand. Co-simulatie is simulatiemethode waarbij elk subsysteem in een andere simulator wordt gesimuleerd. Tijdens het uitvoeren van de simulatie wisselen deze simulators variabelen uit via de interface, om zo interacties tussen de subsystemen weer te geven. Omdat de modellen op afstand kunnen communiceren, via een communicatienetwerk, heeft co-simulatie het voordeel dat de modeleigenaar de informatie die verwoven is in het model niet hoeft te onthullen. Dit heeft ook het voordeel dat elk model gesimuleerd kan worden met de simulator waarvoor het was bedoeld, zodat organisaties die verschillende simulators gebruiken samen kunnen werken zonder dat vertaling van hun modellen vereist is. Als een dergelijke co-simulatie wordt uitgevoerd met behulp van golfvormmodellen, met een hoge tijdsresolutie, is het ook mogelijk om een breed scala aan tijdschalen te reproduceren, van langzame tot zeer snelle verschijnselen. Maar het feit dat zulke co-simulaties op afstand worden uitgevoerd brengt communicatievertragingen met zich mee. Deze vertragingen in combinatie met de hoge tijdsresolutie, die een hoge communicatiesnelheid vereist, maken het totale systeem traag. Het is dus wenselijk om de benodigde onderlinge communicatie tussen de simulators te verminderen. In dit proefschrift verken ik een oplossing voor deze communicatie-uitdaging, gebaseerd op de hypothese dat langzamere fenomenen gemakkelijker te voorspellen zijn. Als de geco-simuleerde fenomenen volgens een bepaald criterium als voorspelbaar kunnen worden geclassificeerd, moet het mogelijk zijn om uitdrukkingen te vinden die interfacevariabelen voorspellen en die vervolgens te gebruiken voor de berekening van de inputs van elke simulator, in plaats van te verwachten dat inputs worden gecommuniceerd. Ik stel een criterium voor om fenomenen als voorspelbaar of onvoorspelbaar te classificeren, evenals methoden voor het vinden van deze uitdrukkingen op basis van een geïnterpoleerde Fouriertransformatie en Taylor-Kalman-filters. Daarnaast stel ik een co-simulatie algoritme voor waarbij de simulators hun eigen inputs berekenen terwijl de co-gesimuleerde fenomenen voorspelbaar zijn. Na deze ideeën toe te passen op de co-simulatie van twee verschillende testsystemen, kon ik de behoefte aan communicatie tot 60% verminderen.

Een co-simulatie framework met deze kenmerken is een stap in de richting van meer beschrijvende modellen en beter presterende simulaties, en een tool die het mogelijk maakt om beter te profiteren van bestaande energie-infrastructuur en deze verder te ontwikkelen.

1

Introduction

To begin, we must emphasize a statement
which I am sure you have heard before,
but which must be repeated again and again.
It is that the sciences do not try to explain,
they hardly ever try to interpret,
they mainly make models.
By a model is meant a mathematical construct which,
with the addition of some verbal interpretations,
describes observed phenomena.
The justification of such a mathematical construct is
solely and precisely
that it is expected to work – that is,
correctly to describe phenomena
from a reasonably wide area.

John von Neumann,
Method in the Physical Sciences, 1955.

Parts of this chapter have been published in [1]–[3].

1

In a field like electrical power systems engineering, that for over a century has relied on the insight provided by simulation, having sufficiently descriptive system models is fundamental. But the expectation that electrical power systems continue to accommodate larger shares of non-dispatchable generation is redefining the requirements these models must fulfill. One of these requirements is that system models take into account the interactions between more, and more diverse subsystems. Another requirement is that they represent wider timescales, so they capture the interactions between subsystems that exhibit slow and fast phenomena. These new requirements pose challenges that can only be tackled with new simulation methods.

1.1. The Need for Model Integration in Electrical Power Systems Simulation

Power grids have, and continued to become, more interconnected, interdependent and technologically heterogeneous. This, to cope with the uncertainty that characterizes non-dispatchable generation [4]. Power flows across borders and from lower to higher voltage levels, the pervasiveness of digital information and communication technology, power electronic converters, energy storage devices, as well as efforts to couple electricity grids with heat or gas grids, are manifestations thereof. For this reason, power system models must now take into account the interactions between more, and more diverse subsystems.

However, higher interconnectivity and technological heterogeneity also mean that the knowledge required to model an electrical power system is no longer available to any single party, but partially available to several. Now more than ever, modeling an electrical power system requires integrating knowledge from different sources and diverse fields. As different experts embed their knowledge of particular subsystems in models, having the capability to integrate these subsystem models is the key to sufficiently descriptive system models.

Some of the challenges that European transmission grid operators (TSOs) are currently facing represent a good example of this situation. European TSOs are interconnected, and are continuously becoming more interdependent. For this reason, they have developed a need for integrating the models of their grids to better understand how they interact. At the same time, the technological changes occurring within the girds of each TSO have created a need for integrating models of these new technologies in the existing grid models.

The relevance that standards like the Common Information Model (CIM) and the Common Grid Model Exchange Specification (CGMES) have acquired in this context, evidence the need for model integration that TSOs have developed. The CIM is a set of IEC standards for representing power system components [5], suitable for modeling power systems for static and dynamic simulations [6]. The CGMES on the other hand is an extension to the CIM promoted by the European Network of Transmission System Operators for Electricity (ENTSO-E), intended for European grid operators to provide models of their grids to a coordinating entity, so that they can be integrated into a Pan-European grid model known as the Common Grid Model (CGM) [7]. Additionally, both the CIM and the CGMES are extensible, so they are also suitable for modeling new tech-

nologies.

Although transmission system operators use the CGM for static calculations, such as adequacy forecasts or coordinated security analyses, having the capability of integrating dynamic models is just as important. For example, for calculating the stability-constrained available transfer capacity between two interconnected grids [8]. Similarly, as generation at the distribution level becomes more prevalent, there are growing concerns about the impact that distribution grids might have on transmission grids. Better transmission/distribution coordination requires integrating both static and dynamic models of these two types of grids [9].

1.2. The Need for Wider Timescales in Electrical Power Systems Simulation

Electrical power systems exhibit phenomena at a wide range of timescales, that span microseconds to years [10]. This has brought about a variety of power system simulation methods that attempt to strike a balance between the level of detail represented in the model, and the computational cost of simulating it. They do so by targeting the phenomena that occur within a specific timescale, while deemphasizing the others.

However, the technological changes that grids are experiencing have created a need for simulations that consider wider timescales. For example, power electronic devices have inherently fast dynamics, but they must be able to coexist with electromechanical devices, which have much slower dynamics. Thus, a model that considers how both of these types of devices interact, must be able to simulate the timescales where their dynamics lay [11].

There are several techniques for simulating wider timescales. One of them is to combine simulators that focus on different timescales, one for devices with fast dynamics and one for the rest of the system [12]. Another approach is multiscale simulation, where the model adapts to the phenomena that are occurring at a given point in simulation time, so slow phenomena are simulated with low time resolution and fast phenomena with high time resolution [13]. When enough computational power is available, wider timescales can also be simulated with natural waveform simulators, such as electromagnetic transient simulators, using detailed models of the grid and generators. These types of simulations are computationally expensive because they work at a high rate in order to capture even the fastest of phenomena, but from a modeling perspective they are the most straightforward solution. This is the type of simulation I will analyze in this thesis.

1.3. Co-Simulation and Remote Model Integration

There are two main approaches to model integration. One of them is integration at the data level. In this approach, the models of each subsystem are shared with a central entity that merges them into a system model, which is then simulated in one simulator. Both CIM and CGMES are examples of this approach.

The other approach is integration at the program level, also known as co-simulation. In a co-simulation each subsystem is simulated in a different simulator. The interac-

tions between subsystems are represented by exchanging model variables between simulators at runtime. These variables are referred to as interface variables. In the context of co-simulation, simulators are considered black boxes that require inputs to calculate model state and output variables, but whose inner workings are unknown [14].

Integration at the program level has several advantages over integration at the data level. One requirement for integration at the data level is that all models are expressed in the same language. If a model is not available in the language of choice, it must be translated. This is disadvantageous not only because translating a model signifies extra work, but also because the target language might not be able to express what the source language does [2]. In contrast, in a co-simulation models do not need to be translated because they can be simulated with the simulator for which they where originally intended. In addition, in a co-simulation it becomes possible to use the most appropriate simulator for each model, which opens a possibility for coupling heterogeneous subsystems without having to adapt each model to the capabilities of a given simulator. Integration at the data level also poses a privacy challenge, because sharing models can potentially reveal the private information embedded in them. Co-simulation offers a solution to this challenge because simulators can be coupled remotely over a communication network, so subsystem models do not need to be exchanged. This can be an attractive feature for organization such as TSOs, which are concerned about information privacy, and which happen to be located in different geographical regions. For the same reason, a remote co-simulation can facilitate collaboration between researchers and industry.

However, co-simulation comes with challenges of its own [2]. Simulator interfacing [15], simulator orchestration [16], [17] and numerical stability [18] are among those that have received the most attention. But particularly relevant to the case of remote co-simulation is the challenge of communication [19]. Exchanging variables between simulators takes time, so if either the exchange rate or the communication delay is high, the total execution time of the co-simulation will suffer. For example, [1] reports a total co-simulation execution time in the order of minutes, while a traditional simulation of the same system is in the order of seconds. Since the communication delay typically increases with distance, a co-simulation between organizations located in different geographical regions or even different countries, such as TSOs, will have a longer execution time than one where all simulators execute in the same computer.

The authors of [20] analyzed the challenge of communication for the case of remote, real-time, natural waveform simulators. In this case, the problem is that with high communication delays and high communication rates it becomes difficult for simulators to fulfill their real-time constraints; simulator inputs will often arrive too late. The solution that [20] proposed is to exchange phasors instead of point-on-wave values. This is equivalent to filtering out the fast phenomena at the co-simulation interface and only exchanging information for the slower phenomena, which can be done at a reduced communication rate. The underlying assumption is that the fast phenomena that occur in one simulator need not propagate to others. But if the goal is to simulate a wider timescale, this assumption might not hold true. Nevertheless, in the case of non-real time simulators there are no hard constraints regarding when inputs should arrive, so an approach that adapts the communication rate to the phenomena that occur should

be possible.

1.4. Problem Definition

In previous sections I established the need for, and interest in, co-simulation-based remote model integration and dynamic simulations that consider wider timescales. The problem I address in this thesis lays at the intersection of these two needs. I am interested in addressing the communication challenge for a remote, non-real time co-simulation, that uses natural waveform simulators at a high communication rate to reproduce fast phenomena, like electromagnetic transients, and a time horizon suitable for reproducing slow phenomena, like electromechanical transients. This, in combination with the larger communication delays associated with remote co-simulation, is a worst case scenario from the communication viewpoint.

As I mentioned at the end of the previous section, if the simulators are non-real time, one way to deal with the challenge of communication is to develop a co-simulation framework that communicates frequently while fast phenomena occur, and avoids communication as much as possible while only slow phenomena occur. My hypothesis is that the slower the phenomena, the easier to predict it becomes. So if the phenomena that occur can be classified as predictable according to some criterion, it should be possible that the co-simulation framework finds closed-form expressions that each simulator can use to predict (i.e., compute) its own inputs, instead of expecting inputs to be communicated. To test this hypothesis, I will constraint myself to black box simulators. This means that as the co-simulation executes, the only information available to the co-simulation framework are the interface variables the simulators exchange. In this context, my first research question is related to the possibility of distinguishing between predictable and unpredictable phenomena, and it reads as follows:

Is it possible to define a criterion so that a co-simulation framework can distinguish between predictable and unpredictable power system phenomena, based exclusively on interface variables?

If this is indeed possible, the next question I will address is related to the possibility of finding expressions that predict interface variables, and it reads as follows:

When the co-simulation framework detects the absence of unpredictable phenomena, can this framework identify closed-form expressions that describe the trajectories followed by the interface variables, so that each simulator computes its inputs from these expressions instead of them being supplied by other simulators?

And provided that the co-simulation framework is able to find these expressions, the final question relates to the consequences that using such a co-simulation framework might have, and it reads as follows:

How accurate are these expressions and what proportion of the co-simulation are they able to describe?

The objective of this thesis is to provide answers to these three research questions. To do so, I will follow a mostly experimental approach. I will define models that produce both fast phenomena, on the timescale of electromagnetic transients, and slow phenomena, on timescales that range from electromechanical transients to steady state. I will use these models to test the methods I propose, with the expectation that they will be able to reproduce the entire range of phenomena, with the least possible communication between simulators.

1.5. Thesis Outline

This thesis is structured as follows:

In **Chapter 2** I provide an overview of continuous-time co-simulation fundamentals, emphasizing those aspects that are relevant to this thesis.

In **Chapter 3** I propose a general criterion and method for distinguishing between predictable and unpredictable phenomena based on simulator inputs and outputs. Using this method, I describe how a co-simulation can operate in two modes, one for unpredictable phenomena where the simulators communicate frequently, and another for predictable phenomena where the simulators compute their own inputs to avoid communication as much as possible.

In **Chapter 4** I define predictable interface variables as those expected while the system is in steady state, and propose a method for finding closed-form expressions that the simulators can use to compute their inputs in the absence of unpredictable phenomena. In this chapter I analyze the case of the AC circuits used to represent electrical power systems when the mechanical aspects of the generators can be neglected. The interface variables of such systems are expected to display either fast transient behavior or to be in steady state, in which case they can be modeled as a sum of sinusoids with constant amplitude, frequency and phase.

In **Chapter 5** I extend the definition of predictable interface variables to those expected while the system is experiencing a slow (electromechanical) transient, and propose a method for finding these closed-form expressions. In this case the interface variables can be modeled as a sum of sinusoids with variable amplitude, frequency and phase.

In **Chapter 6** I introduce a co-simulation framework based on the PowerFactory simulation tool, which implements the methods from Chapters 3 and 5.

In **Chapter 7** I present the results of using the framework from Chapter 6 to co-simulate a power system composed of a transmission and a distribution grid. The system exhibits fast and slow transient behavior.

Finally, **Chapter 8** concludes the thesis.

Bibliography

[1] C. D. López, A. A. van der Meer, M. Cvetković, and P. Palensky, "A variable-rate co-simulation environment for the dynamic analysis of multi-area power systems", in *2017 IEEE Manchester PowerTech*, Jun. 2017, pp. 1–6.

[2] P. Palensky, A. A. van der Meer, C. D. López, A. Joseph, and K. Pan, "Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling", *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 34–50, Mar. 2017.

[3] C. D. López, M. Cvetković, and P. Palensky, "Speeding up AC circuit co-simulations through selective simulator decoupling of predictable states", *IEEE Access*, vol. 7, pp. 1–14, Apr. 2019.

[4] J. van der Burgt, C. Wouters, W. van der Veen, and P. van der Wijk, "Flexibility in the power system", DNV GL, Tech. Rep., Nov. 2017.

[5] "Common information model primer", Electric Power Research Institute, Tech. Rep., Jul. 2020.

[6] "Energy management system application program interface (EMS-API)–part 302: Common information model (CIM) dynamics", International Electrotechnical Comission, IEC 61970-302, Apr. 2018.

[7] "Energy management system application program interface (EMS-API)–part 600-1: Common grid model exchange specification (CGMES)–structure and rules", International Electrotechnical Comission, IEC TS 61970-600-1, Jul. 2017.

[8] O. O. Mohammed, M. W. Mustafa, D. S. S. Mohammed, and A. O. Otuoze, "Available transfer capability calculation methods: A comprehensive review", *International Transactions on Electrical Energy Systems*, vol. 29, no. 6, pp. 1–24, 2019.

[9] "Framework to analyze interactions between transmission and distribution (T&D) systems with high distributed energy resource (DER) penetrations", Power Systems Engineering Research Center, Tech. Rep., Sep. 2019.

[10] R. Thomas, "Fast calculation of electrical transients in power systems after a change of topology", PhD thesis, Delft University of Technology, Nov. 2017.

[11] A. A. der Meer, "Offshore VSC-HVDC networks–Impact on transient stability of AC transmission systems", PhD thesis, Delft University of Technology, Sep. 2017.

[12] V. Jalili-Marandi, V. Dinavahi, K. Strunz, J. A. Martinez, and A. Ramirez, "Interfacing techniques for transient stability and electromagnetic transient programs–IEEE task force on interfacing techniques for simulation tools", *IEEE Transactions on Power Delivery*, vol. 24, no. 4, pp. 2385–2395, Oct. 2009.

[13] F. Gao and K. Strunz, "Frequency-adaptive power system modeling for multi-scale simulation of transients", *IEEE Transactions on Power Systems*, vol. 24, no. 2, pp. 561–571, Apr. 2009.

[14] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: A survey", *ACM Computing Surveys*, vol. 51, no. 3, Article 49, Apr. 2018.

[15] Modelisar, "Functional mock-up interface for model exchange and co-simulation", MODELISAR, Tech. Rep., 2014.

[16] S. Schütte, S. Scherfke, and M. Tröschel, "Mosaik: A framework for modular simulation of active components in smart grids", in *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*, Oct. 2011, pp. 55–60.

[17] M. U. Awais, W. Mueller, A. Elsheikh, P. Palensky, and E. Widl, "Using the HLA for distributed continuous simulations", in *2013 8th EUROSIM Congress on Modelling and Simulation (EUROSIM)*, Sep. 2013, pp. 544–549.

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

[19] C. Gomes, "Foundations for continuous time hierarchical co-simulation", in *ACM Student Research Competition (ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems)*, Saint Malo, France: ACM New York, Oct. 2016, pp. 1–7.

[20] M. Stevic, A. Estebsari, S. Vogel, E. Pons, E. Bompard, M. Masera, and A. Monti, "Multi-site European framework for real-time co-simulation of power systems", *IET Generation, Transmission & Distribution*, vol. 11, no. 17, pp. 4126–4135, Jun. 2017.

2

Fundamentals of Continuous-Time Co-Simulation

Parts of this chapter have been published in [1], [2], and [3].

THIS CHAPTER introduces the fundamentals of continuous-time co-simulation, emphasizing those aspects relevant to this thesis. It is not meant as an exhaustive review of the topic, but as the basis for the discussions that follow. A reader looking for an in-depth review of co-simulation might instead be interested in [4].

2

2.1. Basic Definitions

Model A model is a "mathematical construct which, with the addition of some verbal interpretations, describes observed phenomena" [5]. In the specific case of this thesis, these mathematical constructs represent electrical circuits, and can be systems of algebraic, differential, or differential algebraic equations. In general, I will refer to the model of a (sub)system s as the initial value problem

$$\dot{\boldsymbol{x}}_s = \boldsymbol{f}_s(\boldsymbol{x}_s, \boldsymbol{u}_s) \;\; \boldsymbol{y}_s = \boldsymbol{g}_s(\boldsymbol{x}_s, \boldsymbol{u}_s) \;\; \boldsymbol{x}_s(t_0) = \boldsymbol{x}_{s_0}, \tag{2.1}$$

where $\boldsymbol{u}_s$ are the (sub)system inputs, $\boldsymbol{x}_s$ are the state variables, $\boldsymbol{y}_s$ are the (sub)system outputs, $\boldsymbol{x}_{s_0}$ is the initial value of $\boldsymbol{x}_s$ at t_0, and $\boldsymbol{f}_s$ and $\boldsymbol{g}_s$ are vector-valued functions.

Solver A solver implements an algorithm that calculates the behavior of the model over time, that is, the trajectories that the state and output variables follow, given certain inputs. A well known example of a solver is ode45, which is the implementation of Runge-Kutta (4,5) provided in the MATLAB environment.

Simulator A simulator is the composition of a model and a solver. A simulator uses a solver to calculate the behavior of a given model. From the point of view of co-simulation, a simulator is regarded as a black box.

Interface Variables These are the variables that simulators exchange with each other in order to co-simulate a system. They are the collection of the input and output variables of each simulator. The interface variables represent the interactions between the subsystems that compose the co-simulated system.

2.2. Composition of a Co-Simulation

A co-simulation is typically composed of a set of simulators, a co-simulation master, and co-simulation interfaces. Figure 2.1 depicts this composition. In general terms, I define the master and the interfaces as follows:

Co-Simulation Master The co-simulation master orchestrates the co-simulation. This entails keeping the simulators synchronized in simulation time, and providing the right inputs to each simulator, at the right time.

Co-Simulation Interface A co-simulation interface is an adapter that enables the interaction between possibly heterogeneous simulators and the co-simulation master.

However, there is no standardized definition of the tasks that either the master or the interfaces must carry out. In practice, some implementations favor a thin master with few responsibilities and expect more of the interfaces, while others are the opposite.

2.3. Simulator Interfacing

The first challenge one may face when setting up a co-simulation is simulator interfacing. Although simulators can be internally heterogeneous, coupling and managing simulators is much simpler when they offer homogeneous interfaces. Unfortunately, this is rarely the case. In practice, it is usually necessary to develop custom interfaces for each simulator in order to seamlessly couple and manage them. This is especially true for electrical power systems simulators, which in most cases have not been developed with co-simulation in mind. One remarkable effort to standardize simulator interfacing is the Functional Mock-Up Interface (FMI) [6]. Originally developed for the automotive industry, this standard is slowly making its way into electrical power systems co-simulation.

2.4. Orchestration

The second challenge is orchestrating execution of and interactions between simulators. The complexity of this task increases rapidly with the number of participating simulators. This is why using a dedicated co-simulation master is useful.

Co-simulations can be centrally or decentrally-orchestrated. In a centrally-orchestrated co-simulation (Figure 2.2a), the master is in complete control of the co-simulation. In this case the simulators produce outputs, and await inputs and commands from the master (e.g., initialize, step, roll back, etc.). An example of this approach is the mosaik smart grid co-simulation framework [7].

As co-simulations become more decentrally-orchestrated, some of the tasks that the master would normally perform are transfered to each co-simulation interface. This is the case of frameworks such as the High-Level Architecture (HLA) [8] and FNCS [9]. In a purely decentrally-orchestrated co-simulation, the master is absent, and it becomes the task of the interfaces to agree how to proceed (Figure 2.2b).

The main reason why a decentrally-orchestrated co-simulation can be preferred over a centrally-orchestrated one is scalability. As the number of coupled simulators increases, the master can become a bottleneck. A decentrally-orchestrated co-simulation does not have this drawback. However, decentrally-orchestrated co-simulations are more difficult to implement, and more complex algorithms are required to ensure cor-

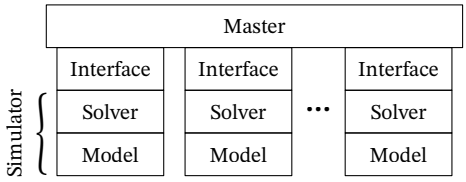
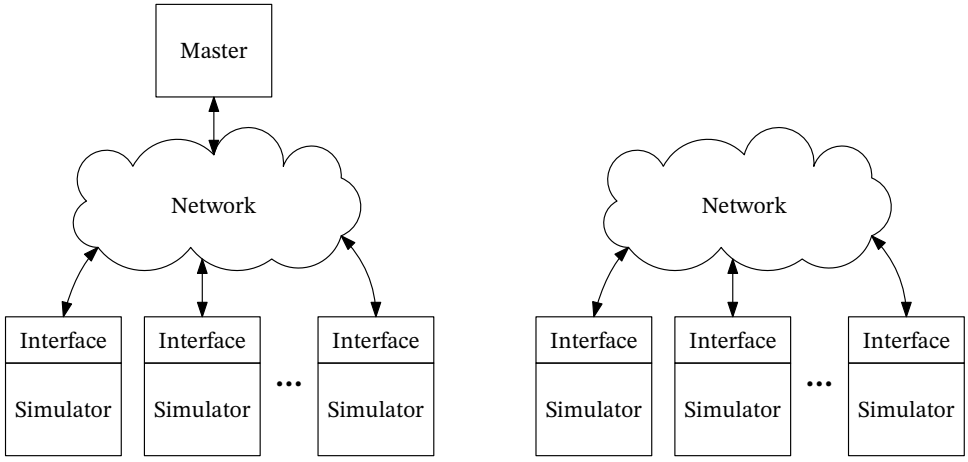


Figure 2.1: Composition of a co-simulation.



(a) Centrally orchestrated (b) Decentrally orchestrated

Figure 2.2: Co-simulation orchestration.

rect execution [2].

2.5. Management

Once the co-simulation is set up, the next challenge is to create and manage simulation scenarios. Naturally, this is more challenging in the case of a co-simulation than a monolithic simulation, since the models are distributed over several simulators and they can be heterogeneous. The mosaik framework [7], which also provides orchestration features, addresses this challenge. One more aspect to consider is simulator allocation. If the simulators are allocated in different computers, the complexity increases further.

2.6. Solving Coupled Models

Let us consider a co-simulation of two subsystems, A and B, modeled with

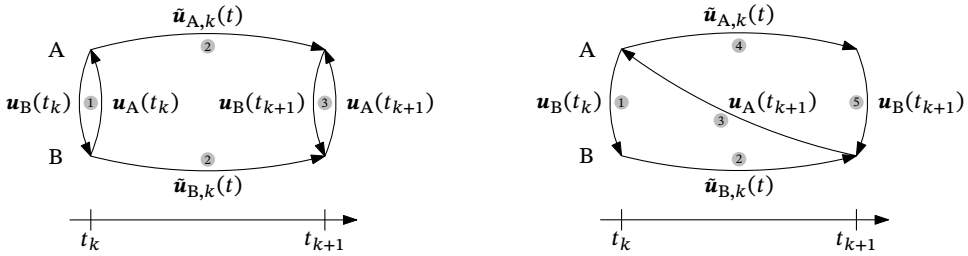
$$\dot{\boldsymbol{x}}_{\mathrm{A}} = \boldsymbol{f}_{\mathrm{A}}(\boldsymbol{x}_{\mathrm{A}}, \boldsymbol{u}_{\mathrm{A}}) \;\; \boldsymbol{y}_{\mathrm{A}} = \boldsymbol{g}_{\mathrm{A}}(\boldsymbol{x}_{\mathrm{A}}, \boldsymbol{u}_{\mathrm{A}}) \;\; \boldsymbol{x}_{\mathrm{A}}(t_0) = \boldsymbol{x}_{\mathrm{A}_0}, \tag{2.2}$$

$$\dot{\boldsymbol{x}}_{\mathrm{B}} = \boldsymbol{f}_{\mathrm{B}}(\boldsymbol{x}_{\mathrm{B}}, \boldsymbol{u}_{\mathrm{B}}) \;\; \boldsymbol{y}_{\mathrm{B}} \;\; = \boldsymbol{g}_{\mathrm{B}}(\boldsymbol{x}_{\mathrm{B}}, \boldsymbol{u}_{\mathrm{B}}) \;\; \boldsymbol{x}_{\mathrm{B}}(t_0) = \boldsymbol{x}_{\mathrm{B}_0}, \tag{2.3}$$

that are coupled so the outputs of one subsystem are the inputs of the other, that is, by enforcing the coupling equations

$$\boldsymbol{u}_{\mathrm{A}} = \boldsymbol{y}_{\mathrm{B}} \qquad\qquad \boldsymbol{u}_{\mathrm{B}} = \boldsymbol{y}_{\mathrm{A}}. \tag{2.4}$$

Since in a co-simulation the equations that model each subsystem are solved by a different solver, the coupling equations can only be enforced at every point in a discrete time grid $\boldsymbol{t} \coloneqq \{t_0, t_1 \ldots t_k \ldots t_K\}$. This time grid defines when the simulators should exchange interface variables (simulator inputs and outputs). The interval $[t_k, t_{k+1}[$ is known as



(a) Parallel (Jacobi)

(b) Serial (Gauss-Seidel)

Figure 2.3: Co-simulation sequences.

the co-simulation macro time step, and $H_k \coloneqq t_{k+1} - t_k$ as the macro time step size. Although there are accuracy and performance benefits to having a dynamically-adjusted macro time step size [10], for simplicity it is often chosen to be $H_k = H$ a constant.

Within a macro time step, each simulator can perform several micro time steps. These micro time steps need not be the same for every simulator, and may even be variable in size. To solve each micro time step in the k^{th} macro time step, the simulators approximate the inputs of each subsystem $\boldsymbol{u}_{\mathrm{A}}(t)$ and $\boldsymbol{u}_{\mathrm{B}}(t)$ with the k^{th} interpolation (or extrapolation) polynomials $\tilde{\boldsymbol{u}}_{\mathrm{A},k}(t)$ and $\tilde{\boldsymbol{u}}_{\mathrm{B},k}(t)$, which are obtained from the history of interface variables exchanged until t_k.

2.6.1. Initialization

Before starting a co-simulation, each simulator must be initialized. However, finding a set of initial conditions that is consistent for all simulators can be challenging. In some cases this can be solved with an iterative search [11], [12].

2.6.2. Execution Sequences

During a co-simulation, the participating simulators may exchange interface variables following different sequences. These sequences are determined by the order in which the simulators are executed. Figure 2.3 shows how simulators A and B proceed in simulation time using the two basic sequences, namely, parallel (also known as Jacobi) and serial (also known as Gauss-Seidel). Both of these sequences can be extended to more than two simulators. As their names suggest, in the parallel sequence simulators execute in parallel during each macro time step, whereas in the serial sequence the simulators are executed one after the other.

In the parallel sequence from Figure 2.3a, $\tilde{\boldsymbol{u}}_{\mathrm{A},k}(t)$ and $\tilde{\boldsymbol{u}}_{\mathrm{B},k}(t)$ are extrapolation polynomials because at t_k neither simulator knows the outputs of the other at t_{k+1}. In the serial sequence from Figure 2.3b on the other hand, $\tilde{\boldsymbol{u}}_{\mathrm{A},k}(t)$ is an interpolation and $\tilde{\boldsymbol{u}}_{\mathrm{B},k}(t)$ and extrapolation polynomial, because at t_k, simulator A receives the outputs that simulator B produces at t_{k+1}. Both of these sequences can be made iterative, so that each macro time step is repeated until a convergence criterion is met. The advantage of this approach is that the results can be more accurate and the co-simulation can be made numerically stable. However, this comes at a higher computational cost.

2.6.3. Accuracy and Numerical Stability

Since the coupling equations can only be enforced once per macro time step and not every micro time step, it follows that the co-simulation produces results which approximate those of a monolithic simulation of the same system. The error the co-simulation incurs in each macro time step may or may not remain constrained, depending on the characteristics of the models and on the choice of macro time step. If the error is not constrained, we talk about a numerically unstable co-simulation. In such cases, the co-simulation might be stabilized by choosing a smaller macro time step. In other cases, a different set of interface variables might need to be chosen [13]. However, the safest approach, whenever possible, is to use iterative execution sequences that guarantee stability [14].

2.7. Discrete Events in Continuous-Time Simulators

Even in a continuous-time co-simulation, where no discrete-event simulators are present, discrete events may occur. There are two types of discrete events.

External Events External events are scheduled, either by the co-simulation user or another entity, and their time of occurrence is known in advance (e.g., a short circuit).

Internal Events Internal events are a product of the co-simulation itself and their time of occurrence might be unknown (e.g., a stochastic event, a threshold crossing).

The challenge with discrete events in continuous-time co-simulation is their timely propagation to other simulators. If an event occurs in a simulator within a macro time step, its effect will not propagate to the remaining simulators until the next time interface variables are exchanged, which has a negative impact result accuracy. In the case of external events, it is possible to circumvent this problem by adjusting the discrete time grid to the event time. On the other hand, internal events can only be discovered after they occur, so it is not possible to align them with the discrete time grid beforehand. The classic solution to this problem is rolling back the co-simulation to the event time and proceed from there, this assuming that the simulators are capable of rolling back.

2.8. Communication and Execution Time

In the case of a monolithic simulation of (2.1), the total execution time is the time it takes to solve the model equations from t_0 to t_K (solver time), whereas in the co-simulation of (2.2) and (2.3) the total execution time also includes the time it takes to construct and evaluate $\tilde{\boldsymbol{u}}_{\mathrm{A},k}(t)$ and $\tilde{\boldsymbol{u}}_{\mathrm{B},k}(t)$ (interpolation time) and the time it takes to enforce (2.4) (communication time). I will refer to the time spent on interpolation and communication-related operations as *co-simulation overhead*.

To understand how co-simulation overhead affects total execution time, let us now consider a simple case in which subsystems A and B are solved with a constant micro time step of size h, the solver time per micro time step is T_h, and the overhead per

macro time step is T_O. Then, the time it takes to solve one macro time step in a parallel co-simulation is

$$T_H = T_h \frac{H}{h} + T_O, \tag{2.5}$$

and the execution time of the entire co-simulation as a function of H is

$$T_{CS}(H) = T_H \frac{t_K - t_0}{H} = (t_K - t_0)\left[\frac{T_h}{h} + \frac{T_O}{H}\right]. \tag{2.6}$$

As (2.6) indicates, T_{CS} increases rapidly as H decreases ($\lim_{H\to 0} T_{CS}(H) = \infty$). This means that co-simulations of systems that have fast dynamics and require frequent communication between subsystems can have their performance heavily penalized. This effect is especially noticeable when the overhead is large with respect to the solver time, which is the case when the models are small and/or the solvers are fast, or when the interpolation of inputs or communication between simulators is slow. Conversely, infrequent communication can make up for a large overhead. The presence of this overhead is why co-simulations can be slower than monolithic simulations of the same model, even if the subsystems are co-simulated in parallel [12].

2.9. Discussion

There are three aspects of this chapter that I would like to highlight because of their relevance to the questions I stated in Chapter 1. First is that reducing the number of times simulators exchange variables helps mitigate the overhead introduced by communication delays, but also by other co-simulation-specific operations. This should have a positive impact on the execution time of a co-simulation. Another aspect is that co-simulations do typically use closed-form expressions to compute their inputs within each macro time step, namely interpolation or extrapolation polynomials. But these expressions are meant to be valid for one macro time step only, so other types of closed-form expressions with longer validity are needed to further reduce the need for communication. Lastly, and although not entirely within the scope of this thesis, is that the operations required to find these expressions, and any other operations that might add overhead to the co-simulation, are better assigned to the interfaces to prevent the co-simulation master from becoming a bottleneck. With this in mind, the next step I will take is to define how a co-simulation framework could determine when predictable or unpredictable phenomena occur, and how it should operate to take advantage of these closed-form expressions that can predict interface variables, assuming they exist. I will address these two issues in the next chapter.

Bibliography

[1] P. Palensky, A. A. van der Meer, C. D. López, A. Joseph, and K. Pan, "Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling", *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 34–50, Mar. 2017.

[2] C. D. López, M. Cvetković, and P. Palensky, "Distributed co-simulation for collaborative analysis of power system dynamic behavior", in *Proceedings of the MED-POWER 2018 Conference*, Nov. 2018, pp. 1–5.

[3] C. D. López, M. Cvetković, and P. Palensky, "Speeding up AC circuit co-simulations through selective simulator decoupling of predictable states", *IEEE Access*, vol. 7, pp. 1–14, Apr. 2019.

[4] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: State of the art", Tech. Rep., Feb. 2017.

[5] J. von Neumann, "Method in the physical sciences", in *The Neumann Compendium*, F. Brody and T. Vámos, Eds., World Scientific, 1995, pp. 627–634.

[6] Modelisar, "Functional mock-up interface for model exchange and co-simulation", MODELISAR, Tech. Rep., 2014.

[7] S. Schütte, S. Scherfke, and M. Tröschel, "Mosaik: A framework for modular simulation of active components in smart grids", in *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*, Oct. 2011, pp. 55–60.

[8] M. U. Awais, W. Mueller, A. Elsheikh, P. Palensky, and E. Widl, "Using the HLA for distributed continuous simulations", in *2013 8th EUROSIM Congress on Modelling and Simulation (EUROSIM)*, Sep. 2013, pp. 544–549.

[9] S. Ciraci, J. Daily, J. Fuller, A. Fisher, L. Marinovici, and K. Agarwal, "FNCS: A framework for power system and communication networks co-simulation", in *Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative*, ser. DEVS '14, San Diego, CA, USA: Society for Computer Simulation International, 2014, 36:1–36:8.

[10] S. Sadjina, L. T. Kyllingstad, S. Skjong, and E. Pedersen, "Energy conservation and power bonds in co-simulations: Non-iterative adaptive step size control and error estimation", *Engineering with Computers*, vol. 33, no. 3, pp. 607–620, 2017.

[11] S. Hansen, C. Thule, and C. Gomes, "An FMI-based initialization plugin for INTO-CPS Maestro 2", in *4th Workshop on Formal Co-Simulation of Cyber-Physical Systems*, Sep. 2020, pp. 1–5.

[12] C. D. López, A. A. van der Meer, M. Cvetković, and P. Palensky, "A variable-rate co-simulation environment for the dynamic analysis of multi-area power systems", in *2017 IEEE Manchester PowerTech*, Jun. 2017, pp. 1–6.

[13] M. Busch, “Zur effizienten Kopplung von Simulationsprogrammen”, PhD thesis, Universität Kassel, Mar. 2012.

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

3

Reducing the Need for Communication with Selective Simulator Decoupling

Parts of this chapter have been published in [1].

IF EACH SIMULATOR has the capability of predicting its own inputs, at least during a portion of the co-simulation, the need for communication between simulators can be reduced. In this chapter I explore this idea, propose a method for distinguishing between predictable and unpredictable interface variables, and describe how a co-simulation could operate if the simulators were capable of predicting their inputs. I will refer to this co-simulation method as *selective simulator decoupling*, because the objective is that simulators execute autonomously (i.e., decoupled from each other) whenever possible.

3.1. Predictability of Interface Variables

Intuitively, one can argue that the predictability of the interface variables changes as the co-simulation runs. For example, when the co-simulated system is in steady state it is easier to guess what the outputs of each subsystem will be on the next macro time step. On the contrary, guessing the outputs of each simulator becomes more difficult when the system is experiencing a fast transient. If the simulators could identify, at runtime, closed-form expressions that describe the trajectories followed by the interface variables over time, each simulator could predict its own inputs. I will refer to these expressions as *trajectory models* to distinguish them from the interpolation/extrapolation polynomials used within each macro time step, as trajectory models are meant to be valid for more than one macro time step. Using the concept of trajectory models I will now introduce the more precise Definitions 3.1.1 and 3.1.2 for predictable and unpredictable interface variables.

Definition 3.1.1 (Predictable interface variables)**.** *Interface variables are considered predictable when their trajectories can be computed with sufficient accuracy from a given trajectory model or set thereof.*

Definition 3.1.2 (Unpredictable interface variables)**.** *Interface variables are considered unpredictable if they do not comply with Definition 3.1.1.*

Note that according to Definitions 3.1.1 and 3.1.2 interface variables are classified as predictable or unpredictable based on an available trajectory model or set of models, not on whether those models exist. This distinction has as consequence that the same trajectory could be classified as predictable or unpredictable depending on the method used for finding the trajectory model. Furthermore, what constitutes sufficient accuracy is entirely dependent on the requirements of the co-simulation application.

3.2. Two Modes of Co-Simulation Operation

Given that the predictability of the interface variables changes during execution, the co-simulation should be able to operate in two different modes and transition between them as needed. The first mode is the *coupled mode*. The co-simulation operates in this mode when the interface variables are considered unpredictable. In this mode, the simulators exchange interface variables at every macro time step. During the k^{th} macro time step, subsystem s is simulated using as inputs the k^{th} interpolation polynomials $\tilde{\boldsymbol{u}}_{s,k}(t)$, where $t \in [t_k, t_{k+1}[$ and $s \in \boldsymbol{s}$ the set of all subsystems. The coupled mode is the default mode of operation.

The second mode is the *decoupled mode*. The co-simulation operates in this mode when the interface variables are considered predictable. In this mode, the simulators do not exchange variables, but predict their own inputs using trajectory models. For a decoupled mode that starts on the k^{th} macro time step and lasts κ macro time steps, subsystem s is simulated using as inputs the k^{th} trajectory models $\hat{\boldsymbol{u}}_{s,k}(t)$, $t \in [t_k, t_{k+\kappa}[$ and $s \in \boldsymbol{s}$.

This bimodal co-simulation poses two main challenges. The first one is to identify appropriate trajectory models. The second one is to seamlessly transition between modes. In this chapter I will only discuss the latter challenge, as the former is application-dependent.

3.2.1. Simulator Decoupling

Any pair of coupled simulators can be decoupled if the interface variables they share follow predictable trajectories. One way of determining when an interface variable is following a predictable trajectory is to attempt to identify its trajectory model and to measure the deviation between the trajectory this model predicts and the true trajectory. If the deviation falls below a given threshold, the trajectory can be considered predictable in the sense of Definition 3.1.1. Since the inputs of a subsystem are the outputs of another, this idea can be expressed either in terms of inputs or outputs. Thus, any output $y \in \boldsymbol{y}_s$, $s \in \boldsymbol{s}$ can be considered predictable if

$$\max \left| \frac{\hat{y}(t) - y(t)}{\max \hat{y}(t) - \min \hat{y}(t)} \right| < \epsilon_{\text{p}}, t \in \boldsymbol{t}_{\text{w}}, \tag{3.1}$$

where $\boldsymbol{t}_{\text{w}} \coloneqq \{t_{k-N_{\text{s}}+1}, t_{k-N_{\text{s}}+2} \dots t_k\}$ is a discrete moving time window of length N_{s} samples and duration T_{w}, $\hat{y}$ is the trajectory model of y, and ϵ_{p} is the allowed normalized deviation. The number of samples N_{s} should be selected according to the needs of the trajectory model identification method.

Note that (3.1) measures deviation relative to the dynamic range of the trajectory model. Using a relative deviation measure simplifies the choice of a suitable ϵ_{p}. Using the dynamic range instead of $\hat{y}(t_k)$ or $y(t_k)$ prevents that (3.1) becomes indeterminate when the outputs approach zero. One caveat is that constantly recomputing $\hat{y}$ to evaluate (3.1) can be computationally expensive if H is small. This means that the *window hop size* R_{w}, that is, the number of samples $\boldsymbol{t}_{\text{w}}$ moves every time (3.1) is evaluated, might need to be adjusted. An $R_{\text{w}} = 1$ requires (3.1) to be evaluated at every macro time step, which incurs the highest computational expense. Higher values of R_{w} reduce the computational expense but might delay the transition to decoupled mode. However, this should not negatively impact the accuracy of the co-simulation, only its total execution time.

3.2.2. Simulator Recoupling

Any pair of decoupled simulators must be recoupled if one of the interface variables they share stops following a predictable trajectory, which in the sense of Definition 3.1.2 happens when the trajectory model in place is no longer representative of the interface variable. Since Definitions 3.1.1 and 3.1.2 are mutually exclusive, a pair of simulators

needs to be recoupled when

$$\max \left| \frac{\hat{y}(t) - y(t)}{\max \hat{y}(t) - \min \hat{y}(t)} \right| \geq \epsilon_{\mathrm{p}}, t \in \boldsymbol{t}_{\mathrm{w}}, \tag{3.2}$$

which is complementary to (3.1). As opposed to the case of simulator decoupling, delaying simulator recoupling would likely have a negative impact on the accuracy of the co-simulation, so an $R_{\mathrm{w}} = 1$ is recommendable.

In this case it is possible to reduce the computational expense of evaluating (3.2) at every macro time step taking into account that when a transition from predictable to unpredictable interface variables occurs, the maximum deviation between the true outputs and those calculated from the trajectory model occurs at the last sample in $\boldsymbol{t}_{\mathrm{w}}$ (provided that $R_{\mathrm{w}} = 1$). Thus (3.2) can be reduced to

$$\left| \frac{\hat{y}(t_k) - y(t_k)}{\max \hat{y}(t) - \min \hat{y}(t)} \right| \geq \epsilon_{\mathrm{p}}, t \in \boldsymbol{t}_{\mathrm{w}}, \tag{3.3}$$

which is more computationally efficient. Every time (3.3) is evaluated, $\hat{y}$ and y are evaluated only at t_k. Furthermore, $\hat{y}$ does not need to be recomputed.

The importance of expressing (3.1) and (3.3) in terms of subsystem outputs instead of inputs becomes apparent when considering that in decoupled mode the inputs are obtained from trajectory models that are not updated to reflect possible changes in the operating conditions of other subsystems. On the contrary, a change in the operating conditions of a subsystem does reflect on its outputs, causing them to deviate from their trajectory model.

A trajectory model can cease to be representative of an interface variable either due to its own limitations (e.g., limited model accuracy) or due to a change in the operating conditions of a subsystem (e.g., change of a model parameter). Changes in the operating conditions are caused by simulation events. External events are the most favorable for simulator recoupling because their occurrence is known in advance. Aside from mode transitions caused by external events, all other transitions back to the coupled mode pose an additional challenge for non-real time co-simulation.

In a non-real time environment there are no guarantees on the time it takes to execute a process. This means that as soon as the simulators decouple, they will likely progress at different rates. When a transition to the coupled mode becomes necessary, the simulator that discovers the need for recoupling can either be ahead of all the others in simulation time or behind at least one simulator. In the first case recoupling is simple; the simulator that discovers the need for recoupling informs the others and waits for them to catch up so they can all resume coupled execution from the same point in simulation time. In the second case the simulators that are ahead in simulation time must roll back before recoupling is possible. This is unfavorable not only because rolling back comes with a performance penalty, but also because in practice not all simulators support the roll-back operation. A possible solution to this problem is to slow down the execution of the faster simulators during decoupled execution so all simulators advance at the same rate, but in a non-real time environment this is difficult to implement.

3.3. Algorithmic Description of the Method

Algorithm 3.1 presents a pseudocode description of the selectively-decoupled co-simulation from the point of view of a simulator. In the algorithm, the simulators start in coupled mode. Each simulator must attempt to identify a trajectory model of its inputs and outputs. When all simulators find trajectory models that predict their inputs and outputs in the sense of (3.1), they decouple and continue execution obtaining their own inputs from the trajectory models. As they progress in decoupled mode, they continuously test if their outputs have become unpredictable in the sense of (3.3). As soon as one simulator determines that its outputs are unpredictable, it notifies all other simulators so that they can recouple at a point defined by the unpredictable simulator.

Algorithm 3.1 Selectively-decoupled co-simulation (part 1)

Define: subsystem s, macro time step index k, recoupling macro time step index k_r, $t_k \in \{t_0, t_1 \dots t_k \dots t_K\}$ a discrete time grid, $\boldsymbol{t}_\mathrm{w} = \{t_{k-N_\mathrm{s}+1}, t_{k-N_\mathrm{s}+2} \dots t_k\}$ a discrete moving time widow, hop size R_w, and ϵ_p the allowed normalized deviation.

Initialize $\dot{\boldsymbol{x}}_s = \boldsymbol{f}_s(\boldsymbol{x}_s, \boldsymbol{u}_s), \boldsymbol{y}_s = \boldsymbol{g}_s(\boldsymbol{x}_s, \boldsymbol{u}_s)$
`mode` $\leftarrow$ `COUPLED`
$k \leftarrow 0$

while $k < K$ **do**
    if `mode` = `COUPLED` **then**

        if k mod $R_\mathrm{w} = 0$ **then**
            Find trajectory models $\hat{\boldsymbol{u}}_{s,k}(t)$ and $\hat{\boldsymbol{y}}_{s,k}(t)$
        end if

        if $\boldsymbol{y}_s(t), t \in \boldsymbol{t}_\mathrm{w}$ are predictable according to (3.1) **then**
            Request decoupling
        end if

        Send $\boldsymbol{y}_s(t_k)$ and receive $\boldsymbol{u}_s(t_k)$

        if Decoupling request accepted **then**
            Update trajectory models $\hat{\boldsymbol{u}}_{s,k}(t)$
            $\boldsymbol{u}_s(t) \leftarrow \hat{\boldsymbol{u}}_{s,k}(t)$
            `mode` $\leftarrow$ `DECOUPLED`
        else
            Create interpolation polynomials $\tilde{\boldsymbol{u}}_{s,k}(t)$
            $\boldsymbol{u}_s(t) \leftarrow \tilde{\boldsymbol{u}}_{s,k}(t)$
        end if

▷ Continues in Algorithm 3.2

Algorithm 3.2 Selectively-decoupled co-simulation (part 2)

```
28:     else if mode = DECOUPLED then
29:
30:         if Recoupling requested then
31:             Receive recoupling index k_r
32:             Roll back or catch up to k = k_r
33:             mode ← COUPLED
34:         else if ∃y(t_k) ∈ y_s(t_k) : y(t_k) is unpredictable according to (3.3) then
35:             k_r ← k − 1
36:             Request recoupling at k_r
37:         else if Event in next macro time step then
38:             mode ← COUPLED
39:         end if
40:
41:     end if
42:
43:     Solve ẋ_s = f_s(x_s, u_s(t)), y_s = g_s(x_s, u_s(t)) until t = t_{k+1}
44:
45:     k ← k + 1
46: end while
```

3.4. Method Speedup

To understand how selective decoupling can speed up a co-simulation, let us consider a simple case where T_S is solver time for one macro time step, T_P is the time spent on calculating and evaluating interpolation polynomials for one macro time step, T_C is the time it takes to exchange interface variables (i.e., communication time), T_D is the total time spent on operations related to simulator decoupling (i.e., calculating and evaluating trajectory models), n_H is the total number of macro time steps, n_C is the number of coupled time steps and n_D is the number of decoupled time steps, with $n_H = n_C + n_D$. Then the execution time of a traditional co-simulation is

$$T_{CS} = (T_S + T_P + T_C)n_H, \tag{3.4}$$

and the execution time of the selectively-decoupled co-simulation is

$$T_{SDCS} = T_S n_H + (T_P + T_C)n_C + T_D. \tag{3.5}$$

Thus, the speedup as a function of communication delay is

$$S(T_C) = \frac{T_{CS}}{T_{SDCS}} = \frac{(T_S + T_P + T_C)n_H}{T_S n_H + (T_P + T_C)n_C + T_D}. \tag{3.6}$$

Assuming that the solver time and the communication time are dominant, since minimizing T_P and T_D should be an implementation objective, the speedup can be approxi-

mated by

$$S(T_C) \approx \frac{(T_S + T_C)n_H}{T_S n_H + T_C n_C}. = \frac{\frac{T_S}{T_C} + 1}{\frac{T_S}{T_C} + \frac{n_C}{n_H}}. \tag{3.7}$$

In addition, the maximum theoretical speedup is

$$\lim_{T_C \to \infty} S(T_C) = \frac{n_H}{n_C}. \tag{3.8}$$

Although this is a simplified analysis, what (3.7) and (3.8) indicate is that for a given ratio between solver and communication time, and disregarding other sources of overhead, there is speedup as long as the number of coupled macro time steps is smaller than the total number of macro time steps. Also, when the communication time is large with respect to solver time, the speedup approximates the ratio between total number of macro time steps and coupled macro time steps.

3.5. Discussion

The main assumption I made in this chapter is that it is possible to find trajectory models that predict interface variables. If this is not true, then phenomena cannot be classified as predictable or unpredictable. Whether a selectively-decoupled co-simulation is possible depends on this very fact.

One aspect worth highlighting is that the method I proposed in this chapter is independent from the system it is applied to. Although the focus of this thesis is on natural waveform simulation of electrical power systems, the method could be applied to other systems if suitable trajectory models are found.

However, the fact that the method relies on rolling simulators back is a disadvantage. There are three challenges related to this operation. First, rolling a simulator back implies restarting it with the set of model states valid for that point in time. This means that each simulator must store the history of all of its states, which substantially increases the requirement for memory, especially for large systems. Second, rolling back is a time consuming operation. And finally, rollback capabilities do not seem to be a common feature of most electrical power system simulators.

Having defined how the simulators partaking in a co-simulation could selectively decouple and predict their own inputs, the next step is finding trajectory models that make this prediction possible. I will begin to address this matter in the next chapter.

Bibliography

[1] C. D. López, M. Cvetković, and P. Palensky, "Speeding up AC circuit co-simulations through selective simulator decoupling of predictable states", *IEEE Access*, vol. 7, pp. 1–14, Apr. 2019.

4

Selective Decoupling of AC Systems in Steady State

Parts of this chapter have been published in [1].

In this chapter I propose a method for finding a trajectory model that predicts simulator inputs while the co-simulated subsystems are in steady state. I focus on the case of the AC circuits used to represent power systems when the mechanical aspects of the generators can be neglected. Although this exercise is of limited applicability considering that dynamic simulations are not meant to simulate steady states, it is a useful first step because it provides insight into the feasibility of the selectively-decoupled co-simulation I proposed in Chapter 3.

4.1. A Trajectory Model for Steady State

In AC circuit co-simulation, the interface variables are typically voltage and current, although in some cases power is used as well [2]. These interface variables mainly follow sinusoidal trajectories that may contain harmonic distortion caused by non-linear devices, such as transformers and power electronic converters. Defining predictable interface variables as those that follow these sinusoidal trajectories, the trajectory model would have the form

$$\hat{u}(t) = \sum_{n=0}^{N} A_n \sin\left(2\pi f_n t + \phi_n\right), \tag{4.1}$$

where A_n, f_n and ϕ_n are the amplitude, frequency and phase of the n^{th} harmonic, and N is the total number of harmonics. This trajectory model is valid when the circuit is in steady state. Thus, finding a suitable trajectory model $\hat{u}(t)$ implies estimating the parameters A_n, f_n, ϕ_n and N in (4.1).

4.2. Fourier-Based Trajectory Model Identification

In the case of continuous trajectories, a Fourier Transform would yield the required trajectory model parameters. However, in a discrete case such as a co-simulation, neither the Discrete Fourier Transform (DFT) nor its more efficient implementation, the Fast Fourier Transform (FFT), are likely to produce accurate results due to their discrete frequency resolution. Some special considerations are required to overcome the limitations of these discrete methods.

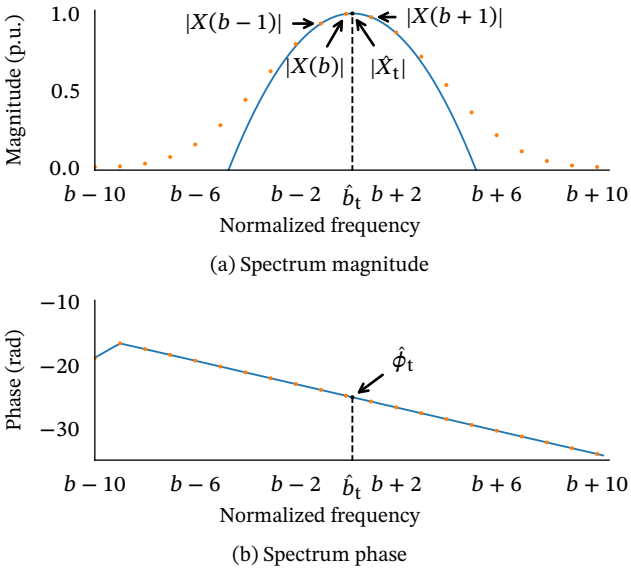
4.2.1. Accuracy of Discrete Fourier Methods

When estimating the frequency of a harmonic, the accuracy of a DFT is restricted to

$$\pm\frac{\Delta f_{\text{DFT}}}{2} = \pm\frac{f_{\text{s}}}{2N_{\text{s}}},$$

where Δf_{DFT} is the frequency resolution of the DFT, f_{s} is the sampling frequency, and N_{s} is the number of acquired samples.

As a reference, a macro time step of 0.1ms is a common choice for co-simulations of a 50Hz electrical power system AC circuit, which means that the interface variables are sampled at a frequency $f_{\text{s}} = 1/0.1\text{ms} = 10\text{kHz}$. At that sampling frequency, 25 periods need to be acquired to obtain a DFT accuracy within ±1Hz. Taking into account that a frequency deviation of 0.1Hz is significant for these systems, an accuracy of ±1Hz is



(a) Spectrum magnitude

(b) Spectrum phase

Figure 4.1: QIFFT of a discrete spectrum.

unacceptably low, especially considering how many periods need to be acquired. For applications that require more accuracy, methods that interpolate the DFT (or the FFT) to better approximate a continuous Fourier Transform exist.

4.2.2. Interpolated Fourier Methods

The Quadratically Interpolated Fast Fourier Transform (QIFFT) [3] is one of the methods that approximate a continuous Fourier Transform. The idea behind the QIFFT is to fit a parabola to the tuple $(|X(b-1)|, |X(b)|, |X(b+1)|)$, where $|X(b)|$ is a peak in the discrete spectrum and b its location in normalized frequency $f/\Delta f_{\text{DFT}}$ (bin number), as Figure 4.1a shows. In the figure neither the true peak value $|X_t|$ nor its location b_t can be directly obtained from the discrete spectrum, but the vertex of the fitted parabola $(\hat{b}_t, |\hat{X}_t|)$ provides a good approximation. To estimate the phase of each harmonic $\hat{\phi}_t$ one must find the intersection between the spectrum phase and $\hat{b}_t$ using interpolation, as in Figure 4.1b.

The eXponentially weighted QIFFT (XQIFFT) [4] differs from the QIFFT in that it weighs $|X(b-1)|$, $|X(b)|$ and $|X(b+1)|$ using an exponential function before fitting the parabola. This modification improves the accuracy of the estimates $\hat{b}_t$ and $|\hat{X}_t|$ with negligible impact on computational performance. By defining

$$\alpha = |X(b-1)| \tag{4.2}$$
$$\beta = |X(b)| \tag{4.3}$$
$$\gamma = |X(b+1)|, \tag{4.4}$$

the values of $\hat{b}_t$ and $|\hat{X}_t|$ can be obtained from

$$\hat{b}_t = b + \frac{1}{2}\frac{f(\alpha) - f(\gamma)}{f(\alpha) - 2f(\beta) + f(\gamma)} \tag{4.5}$$

$$|\hat{X}_t| = f^{-1}\left(f(\beta) - \frac{1}{8}\frac{\left[f(\alpha) - f(\gamma)\right]^2}{f(\alpha) - 2f(\beta) + f(\gamma)}\right), \tag{4.6}$$

where $f(\Theta) = \Theta^p$ and $f^{-1}(\Phi) = \Phi^{\frac{1}{p}}$ are the exponential weighing function and its inverse. According to [4], $p = 0.2308$ is a good choice for an accurate $\hat{b}_t$ and $p = 0.2318$ is a good choice for an accurate $|\hat{X}_t|$. Experimentally both of these values seemed appropriate for this application. Finally, ignoring the negative frequencies in the spectrum, equations (4.5) and (4.6) can be applied to the n^{th} peak in the discrete spectrum, and A_n, f_n and ϕ_n can be estimated as

$$\hat{A}_n = 2|\hat{X}_t| \tag{4.7}$$

$$\hat{f}_n = \hat{b}_t \Delta f_{\text{DFT}} \tag{4.8}$$

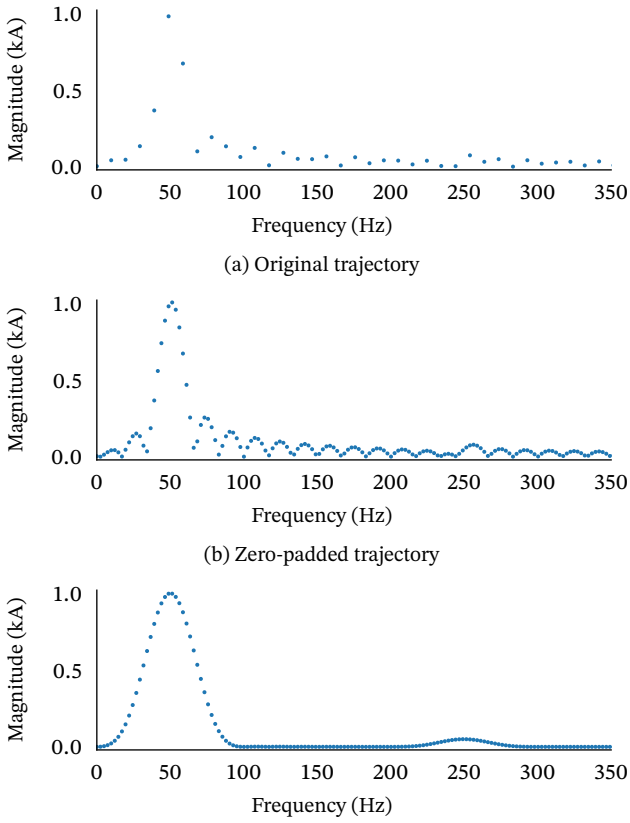
$$\hat{\phi}_n = \hat{\phi}_t. \tag{4.9}$$

4.2.3. Spectrum Preprocessing

In practice, the trajectories followed by the interface variables need to be preprocessed to maximize the accuracy of the XQIFFT. The two main challenges that need to be addressed are the possibility of an insufficient frequency resolution, which is detrimental to spectrum interpolation, and spectral leakage [5], which modifies the shape of the spectrum.

Figure 4.2a shows the spectrum magnitude of a 0.06s window of a current trajectory sampled at a 10kHz rate. The trajectory has one frequency component around 50Hz, that appears as the most prominent peak, and one around 250Hz that is almost indistinguishable. In the case of the most prominent magnitude peak, the large separation between magnitude samples would make it difficult to fit a parabola to them as precisely as in Figure 4.1. This problem can be mitigated by zero-padding the trajectory before obtaining its spectrum. This results in the smoother spectrum magnitude shown in Figure 4.2b, where the actual location of both frequency components becomes easier to estimate from the main lobes, that is, the most prominent lobes in a frequency range.

However, the resulting spectrum is affected by spectral leakage, as the presence of side lobes around each main lobe indicates. These side lobes are a challenge for peak detection because they are difficult to distinguish from the main lobes without supervision, and because they modify the amplitude of the main lobes. Spectral leakage can be mitigated by applying a windowing function to the zero-padded trajectory. Figure 4.2c shows the result of applying a Blackman window [5] to the zero-padded trajectory. The resulting spectrum has two smooth and easily distinguishable main lobes around 50Hz and 250Hz. For each main lobe it is now straightforward to identify $|X(b-1)|$, $|X(b)|$ and $|X(b+1)|$ and to apply the XQIFFT.



(a) Original trajectory

(b) Zero-padded trajectory

(c) Zero-padded and Blackman-windowed trajectory

Figure 4.2: Spectrum preprocessing of a current trajectory with one frequency compoent at 50Hz and another at 250Hz, sampled for 0.06s at a 10kHz rate.

4.2.4. Parameter Postprocessing

Experimentally I found that the ϕ_n that the XQIFFT produces are not accurate enough for this application. To remedy this, I resorted to curve fitting based on least squares optimization. I transformed the trajectory model $\hat{u}(t)$ of the true trajectory $u(t)$ into a function of time an phase, and solved

$$\{\phi_1, \dots, \phi_N\} = \underset{\varphi_1,\dots,\varphi_N}{\operatorname{argmin}} \sum_{t \in \boldsymbol{t}_w} \left[u(t) - \hat{u}(t, \varphi_1, \dots, \varphi_N)\right]^2, \tag{4.10}$$

using an iterative method. I used the ϕ_n obtained from the XQIFFT as starting point for the first iteration, and let the algorithm refine them further. To solve (4.10) I applied the Levenberg-Marquardt algorithm [6].

4

4.3. Testing

To evaluate the performance of this trajectory model identification method I first defined an AC test circuit and run a monolithic simulation, a traditional co-simulation and a selectively-decoupled co-simulation of the same circuit. I quantified accuracy by measuring the deviation (error) of the state variables computed with co-simulation from those obtained from a monolithic simulation. In every case I present the error of a given state variable in percent of the dynamic range of said state variable. I calculated speedup as the ratio between the execution time of a traditional co-simulation and a selectively-decoupled co-simulation.

4.3.1. Test Circuit

Figure 4.3 shows the test circuit. This circuit represents one phase of a simple electrical power system, composed of a generator, a transmission line and a load, and it is based on the electromagnetic transient models from [7]. The switch connected between the transmission line and the load simulates line-to-ground short circuits, and the current source connected in parallel to the load injects 3rd and 5th harmonics to simulate the presence of non-linear devices. Table 4.1 specifies the parameters of this test circuit.

For co-simulation, I split the test circuit in two subsystems as in Figure 4.4, where v_{π_1} and i_π are the interface variables. At every macro time step, subsystem A sends v_{π_1} to subsystem B, and subsystem B enforces v_{π_1} with a controlled voltage source. At the same time, subsystem B sends i_π to subsystem A, and subsystem A enforces i_π with a controlled current source.

4.3.2. Test Environment

I implemented the circuit model, the simulators and the co-simulation master in Python 3.6, aided by the numerical methods provided by SciPy [8], and by the ØMQ [9] messaging library for communication between the simulators. I ran all the (co-)simulations on a desktop computer with a 3.5GHz Intel Xeon CPU and 8GB of RAM. All processes (i.e., both simulators and the co-simulation master) run in parallel, each on a different CPU core. In this implementation, the simulators are in charge of analyzing their own inputs and outputs and of requesting mode transitions to the co-simulation master. In turn,

Table 4.1: Test Circuit Parameters

Symbol	Value	Unit
e_G	$60\sin(100\pi t)$	kV
R_{G_1}	0.1	Ω
R_{G_2}	100	Ω
L_G	0.2	mH
C_G	1	μC
ℓ_π	15	km
r_π	0.01273	Ω/km
l_π	0.9337	mH/km
c_{π_1}, c_{π_2}	6.37	μC/km
R_π	$r_\pi \ell_\pi$	Ω
L_π	$l_\pi \ell_\pi$	mH
C_{π_1}, C_{π_2}	$c_{\pi_1}\ell_\pi$ or $c_{\pi_2}\ell_\pi$	μC
R_L	20	Ω
L_L	4	mH
C_L	20	μC
i_H	$100\sin(300\pi t) + 60\sin(500\pi t)$	A

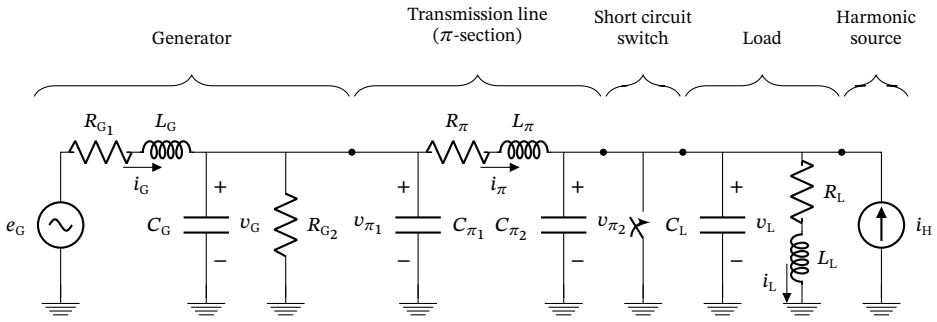


Figure 4.3: Diagram of the test circuit.

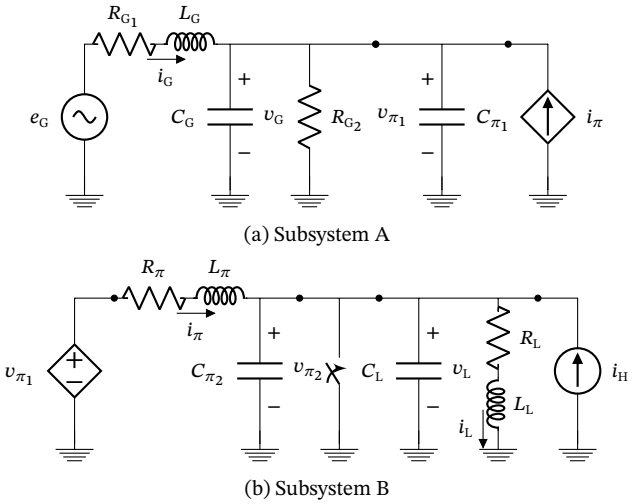


Figure 4.4: Diagram of the co-simulated test circuit split in two subsystems that exchange the interface variables v_{π_1} and i_π.

the co-simulation master has the additional task of synchronizing mode transitions at the request of the simulators.

4.3.3. Case 1: Validation

To validate this trajectory model identification method, I compared a selectively-decoupled co-simulation to a monolithic simulation and a traditional co-simulation of the test circuit. The (co-)simulated scenario includes a short circuit event at $t = 0.05$s that clears at $t = 0.15$s, and a load event at $t = 0.25$s represented as a step reduction of R_L to 5Ω. For the selectively decoupled co-simulation I considered two cases: one where these events are known in advance and another where they are unknown and must be detected. This is to study how the method reacts to external and internal events. It is important to note that in practice it is not necessary to detect all events, only internal events. This simplifies some of the mode transitions and increases result accuracy.

To solve the differential equations that model the test circuit I used the DOPRI5 solver, which is a Runge-Kutta solver of order (4,5) with step size control [10], and limited its maximum step size to the size of the macro time step. For the co-simulations I used a macro time step $H = 0.1$ms, an acquisition window size $T_w = 2/50\text{Hz} = 0.04$s, a window hop size $R_w = 1$ sample, and a predictability threshold $\epsilon_p = 0.02$p.u.

Table 4.2 presents the execution time of each method. The table shows that the co-simulation is more than four times slower than the monolithic simulation. It also shows that the selectively-decoupled co-simulation provides a speedup of about 20% with respect to the traditional co-simulation. Even though this is a substantial improvement, it is not enough to come close to the execution time of the monolithic simulation. Unexpectedly, the selectively-decoupled co-simulation with unknown event times provides a higher speedup than the one with known event times, despite the additional opera-

Table 4.2: Execution times and speedup for Case 1

Method	Execution time (s)	Speedup (p.u.)
Monolithic	3.12	–
Co-simulation	13.7	–
Selective decoupling (known event times)	11.5	1.19
Selective decoupling (unknown event times)	11.35	1.2

tions the former executes. This is because an event can only be detected after it happens, which causes the co-simulation with unknown event times to remain decoupled for slightly longer than its counterpart. Nevertheless, I do not believe this result would necessarily extend to larger models, where the penalty for rolling back a simulator is higher and could offset the gains from longer decoupled execution.

Figure 4.5 compares the trajectories of all the state variables in the test circuit, computed with each (co-)simulation method. The colored background indicates that the co-simulation is decoupled. The figure shows that all the trajectories overlap to the point where they are practically indistinguishable from each other, even when the co-simulation is decoupled. The selectively decoupled co-simulations are able to seamlessly transition between modes, and of accurately reproducing fast transients, such as the peaks in v_{L} and i_{L} at $t = 0.15$s, or the small oscillations in v_{L} at $t = 0.25$s.

It is not until one examines the error of each co-simulation with respect to the monolithic simulation in Figure 4.6, that the differences between the methods become clear. With the exception of a few peaks that occur at mode transitions, the errors obtained from the selectively decoupled co-simulations are well below 1%. The error plots show that all three co-simulations are similarly accurate in coupled mode, and that the error increases as soon as the simulators decouple. During the longest decoupled mode it is also possible to see that the error has a tendency to increase, which I attribute to the limited accuracy of the trajectory models. Although both selectively-decoupled co-simulations show similar accuracy, after recoupling the error is slightly higher for the co-simulation with unknown event times. The delay between event occurrence and event detection appears to cause this additional deviation.

4.3.4. Case 2: Influence of the Macro Time Step

The objective of this case is to study the influence of the macro time step H on the accuracy and execution time of a selectively decoupled co-simulation. For this purpose, I considered the same scenario and settings as in Case 1, but repeated the co-simulations for different values of H. Figure 4.7 shows the results of these (co-)simulations in terms of state variable errors and speedup with respect to H. The figure summarizes the errors using box plots that mark the 25^{th}, 50^{th}, 75^{th} and 100^{th} error percentiles.

The results show an overall tendency for both the error and the speedup to grow as H grows. It is possible to observe that most of the error of the traditional co-simulation lays

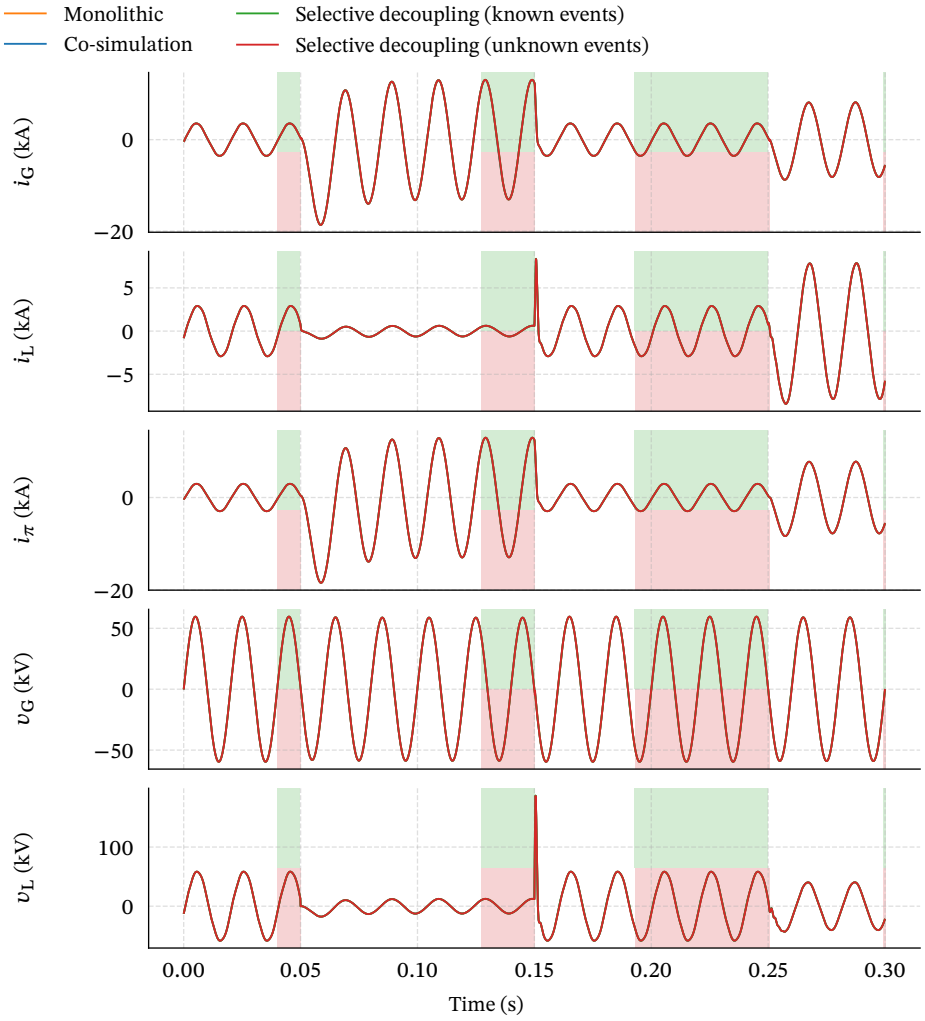


Figure 4.5: State variables computed with a monolithic simulation, a traditional co-simulation, a selectively-decoupled co-simulation with known event times, and a selectively-decoupled co-simulation with unknown event times. The colored background indicates when the co-simulation is in decoupled mode (green for known event times, red for unknown event times). Note that $v_L = v_{\pi_2}$ and $v_G = v_{\pi_1}$.

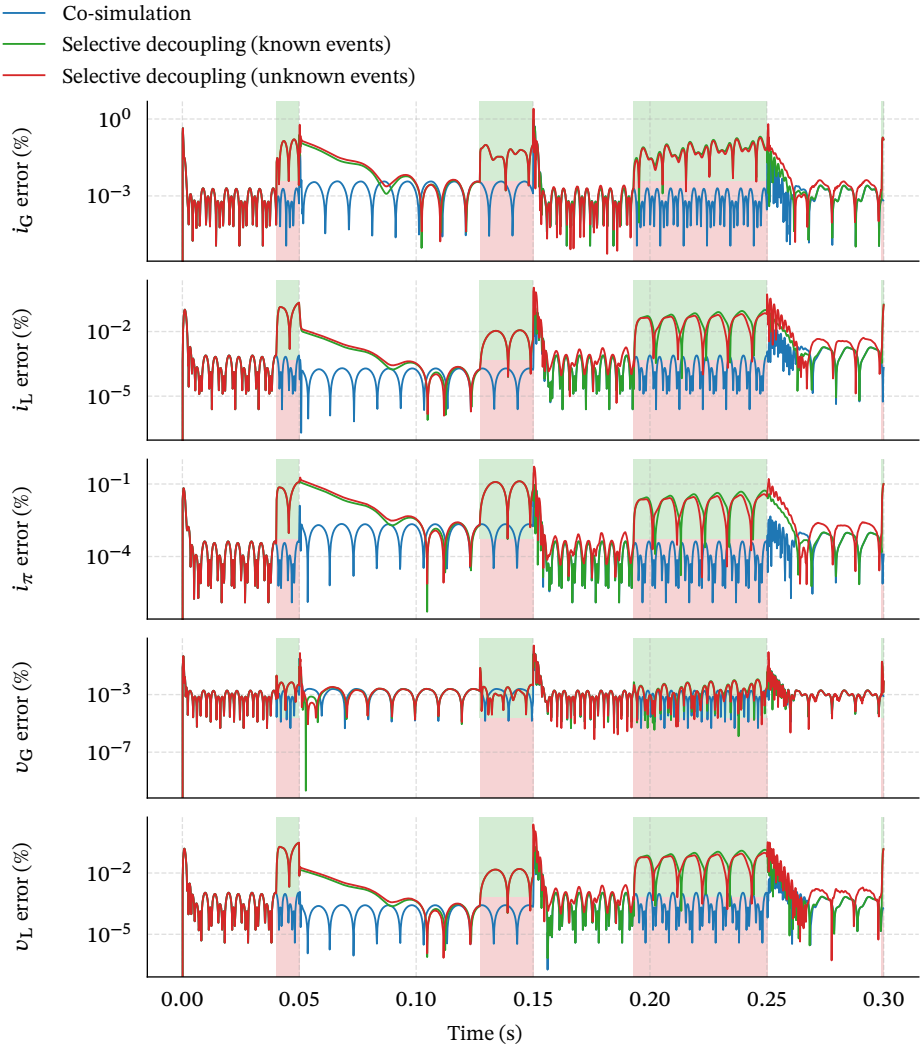


Figure 4.6: State variable errors of the traditional co-simulation, the selectively-decoupled co-simulation with known event times, and the selectively-decoupled co-simulation with unknown event times from Case 1. The errors are measured with respect to the monolithic simulation and are in percent of the dynamic range of the corresponding state variable. The colored background indicates when the co-simulation is in decoupled mode (green for known event times, red for unknown event times).

in a lower range than the error of the selectively-decoupled co-simulations, with some exceptions for large H, where the ranges are approximately the same. This tendency can be observed most clearly if by comparing the 75$^{\text{th}}$ error percentiles.

Surprisingly, there are cases where a smaller H produces a higher error range. One example of this is the error in v_{G}, which lays in a lower range for $H = 0.1 \times 2^{-2}$ms than for $H = 0.1 \times 2^{-1}$ms. In all of these cases, the 25$^{\text{th}}$, 50$^{\text{th}}$ and 75$^{\text{th}}$ do not follow this tendency, indicating that only a few error points cause the higher error range. By examining the results of each co-simulation individually, I found that the error points that cause the higher error range come from small oscillations that occur at the mode transition right after $t = 0.15$s, the amplitude of which does not show a clear tendency with respect to H.

Now, comparing both selectively decoupled co-simulations, it is possible to observe that the 75$^{\text{th}}$ error percentile is similar for every value of H, but that the 25$^{\text{th}}$ percentile drops much lower for the co-simulation with known event times as H decreases. This is because the upper error bound is mostly influenced by the accuracy of the trajectory model, whereas the lower error bound is mostly influenced by the accuracy of the coupled co-simulation (see Figure 4.6). The accuracy of the trajectory model does not significantly improve as H decreases, because the accuracy of the DFT depends on the size of the acquisition window (number of acquired periods), not the sample rate (see Section 4.2.1), provided that the minimum sample rate requirement is met. Since the co-simulation with unknown event times spends more time in decoupled mode for the reasons exposed in Case 1, a larger portion of its error lays towards the higher extreme of the error range.

Regarding the speedup, a selectively decoupled co-simulation can become slower than a traditional co-simulation if H is sufficiently low. As H decreases, (3.1) must be evaluated more often. Additionally, the trajectory model identification method has to process a larger number of samples. As a result, the overhead of detecting predictable interface variables grows to the point where the selectively decoupled co-simulation yields no benefit.

4.3.5. Case 3: Influence of the Predictability Threshold

The objective of this case is to study the influence of the predictability threshold ϵ_{p} on the accuracy and execution time of a selectively decoupled co-simulation. For this purpose, I considered the same scenario and settings as in Case 1, but repeated the (co-)simulations for different values of ϵ_{p}. Figure 4.8 shows the results of these (co-) simulations in the same style as in Case 2. Although the traditional co-simulation does not depend on ϵ_{p}, I show its error for each ϵ_{p} for ease of visual comparison.

The results in Figure 4.8 share some characteristics with those from Figure 4.7. One observes that the 75$^{\text{th}}$ error percentile of the selectively-decoupled co-simulations grows with ϵ_{p}. One can also observe that there is no clear tendency for the 100$^{\text{th}}$ error percentile, although higher error ranges do tend to appear for higher ϵ_{p}. In addition, in most cases both the 25$^{\text{th}}$ and 75$^{\text{th}}$ error percentiles are lower for the selectively decoupled co-simulation with known event times.

Even though the 75$^{\text{th}}$ error percentile increases with ϵ_{p}, it always remains below 0.5%. However, the 100$^{\text{th}}$ error percentile reaches values above 10% for high ϵ_{p}. By

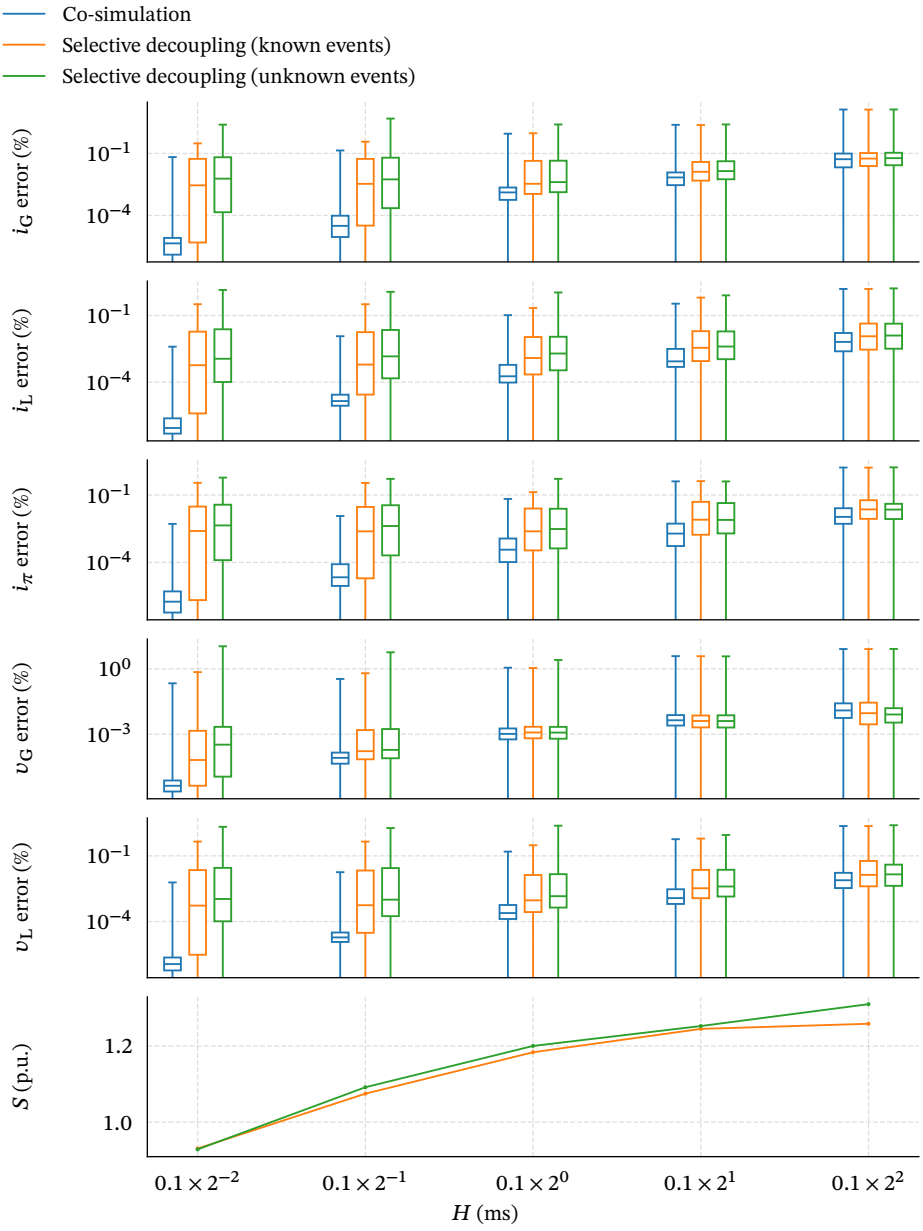


Figure 4.7: Error with respect to the monolithic simulation and speedup with respect to the traditional co-simulation for different macro time step sizes. The error is in percent of the dynamic range of the corresponding state variable. Each box plot marks the 25$^{\text{th}}$, 50$^{\text{th}}$, 75$^{\text{th}}$ and 100$^{\text{th}}$ error percentiles.

examining the results of each co-simulation individually, I found that the error points that cause such a high 100$^{\text{th}}$ percentile come, once more, from small oscillations that occur at the mode transition right after $t = 0.15$s. This indicates that variations of ϵ_{p} do not affect all mode transitions the same way, and that while some remain seamless, others do not.

Figure 4.9 shows how changes in ϵ_{p} affect when mode transitions occur. According to the figure, as ϵ_{p} increases, the transitions to the decoupled mode happen earlier, whereas the transitions to the coupled mode happen later. The figure also confirms that not all mode transitions are equally affected by changes in ϵ_{p}. For example, the 2$^{\text{nd}}$ decoupling happens around 300 macro time steps earlier for $\epsilon_{\text{p}} = 0.15$ than for $\epsilon_{\text{p}} = 0.01$, whereas all the other transitions are shifted by 20 macro time steps or fewer. This means that this transition alone produces most of the additional speedup.

How much a mode transition shifts in time as a consequence of a change in ϵ_{p} has to do with how quickly an interface variable deviates from (or converges towards) its trajectory model. Figure 4.10 shows the deviation of i_π from its trajectory model $\hat{i}_\pi$. Here one can see that the transitions to decoupled mode with the largest shift are those where the deviation decreases slowly (second and fourth), whereas the least affected transitions are those where there is virtually no deviation (first) or the deviation falls sharply (third).

4.3.6. Case 4: Influence of the Window Hop Size

The objective of this case is to study the influence of the acquisition window hop size R_{w} on the accuracy and execution time of a selectively decoupled co-simulation. Once more, I considered the same scenario and settings as in Case 1, but repeated the (co-) simulations for different values of R_{w}. Figure 4.11 shows the results of these (co-) simulations in the same style as in Cases 2 and 3. Since a change in R_{w} only affects the transitions to decoupled mode, I omit the results of the selectively decoupled co-simulation with unknown events. Although the traditional co-simulation does not depend on R_{w}, I show its error for each R_{w} for ease of visual comparison.

The results in Figure 4.11 show that the error does not change significantly for different values of R_{w}. Additionally, the maximum speedup that can be achieved by increasing this parameter is more modest than in Case 3. As opposed to previous cases, where the speedup shows a tendency to settle at a certain value, in this case I found that the speedup drops significantly for large R_{w}. This happens because as R_{w} increases, so does the probability of delaying transitions to the decoupled mode.

4.3.7. Case 5: Selecting Parameters for Additional Speedup

The objective of this case is to tune the selectively decoupled method to obtain a higher speedup than that of Case 1, guided by the results of Cases 2 to 4. Here, I considered the same scenario and settings as in Case 1 but set $\epsilon_{\text{p}} = 0.07$ and $R_{\text{w}} = 2^4$ based on the relationship between error and speedup found in Cases 3 and 4. I chose this value for ϵ_{p} because it provides higher speedup that the value from Case 1 and only a slightly higher error, and this value for R_{w} because it provides the highest speedup.

Table 4.3 shows the speedups for Case 5, which are around 10% and 20% higher than in Case 1. Observing Figure 4.12, one can see that the first and third transitions to the

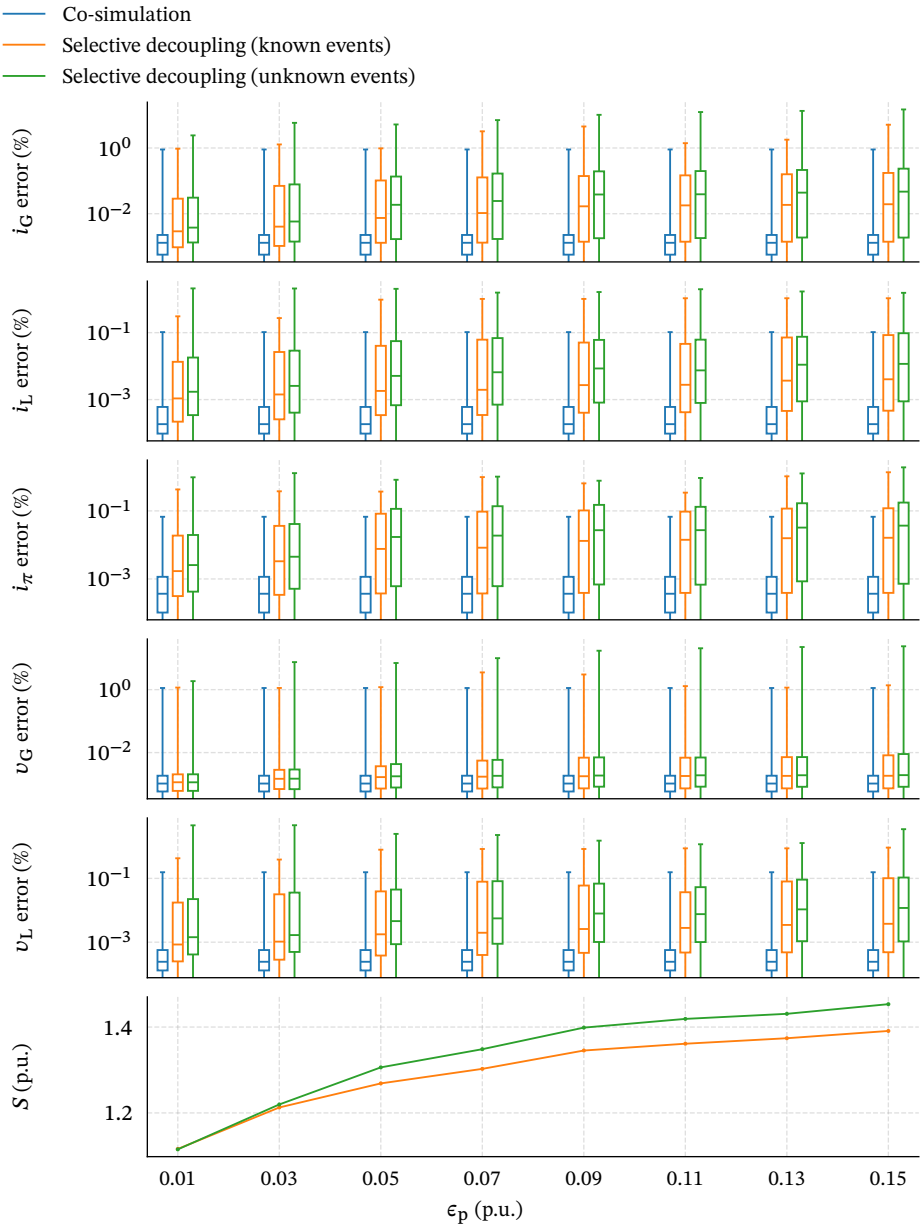


Figure 4.8: Error with respect to the monolithic simulation and speedup with respect to the traditional co-simulation for different predictability thresholds. The error is in percent of the dynamic range of the corresponding state variable. Each box plot marks the 25th, 50th, 75th and 100th error percentiles.

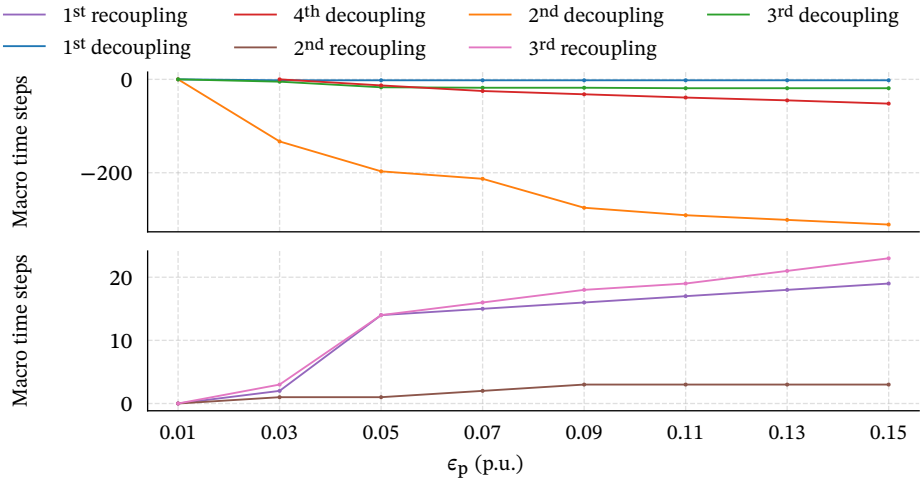


Figure 4.9: Change in the mode transition times measured in macro time steps for different predictability thresholds. As the threshold grows, the simulators decouple earlier and recouple later.

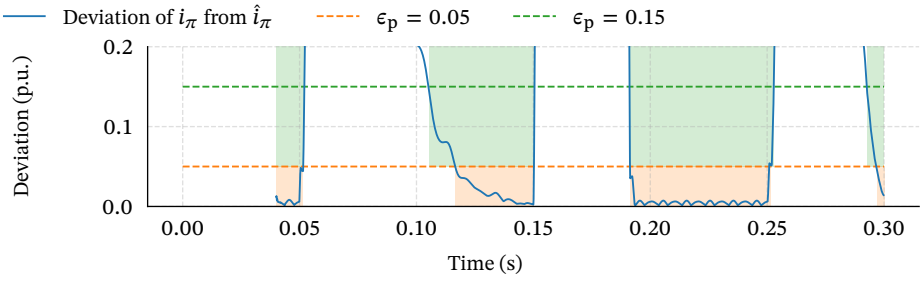


Figure 4.10: Deviation between the true trajectory of i_π and the trayectory model $\hat{i}_\pi$, and mode transitions for two predictability thresholds.

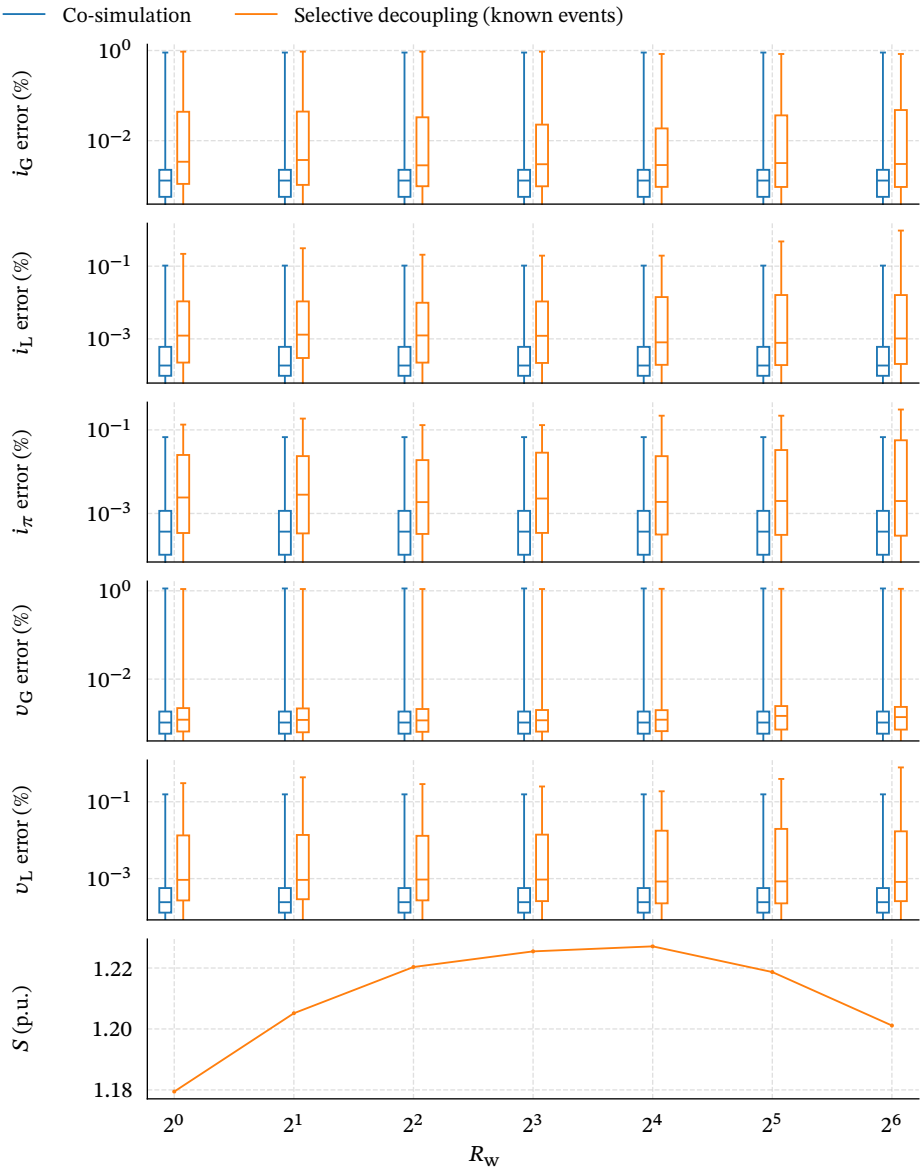


Figure 4.11: Error with respect to the monolithic simulation and speedup with respect to the traditional co-simulation for different acquisition window hop sizes. The error is in percent of the dynamic range of the corresponding state variable. Each box plot marks the 25th, 50th, 75th and 100th error percentiles.

Table 4.3: Speedup for Case 5

Method	Speedup (p.u.)
Selective decoupling (known event times)	1.31
Selective decoupling (unknown event times)	1.42

coupled mode occur much later for the co-simulation with unknown event times that for the one with known event times, which explains the speedup difference between them. In addition, by comparing Figure 4.12 to Figure 4.5 it is visible that much of the additional speedup comes from the second and fourth time the simulators decouple, which is in accordance with the results from Figure 4.9.

Regarding the accuracy of the results, it is still difficult to perceive the differences between the results from different methods in Figure 4.12. Once more, these differences become clear when observing the error in Figure 4.13. Indeed, the error is higher in this case than in Case 1, and the differences between both selectively decoupled co-simulations are also more prominent. Nevertheless, the error remains under or around 1% for all state variables, with the exception of some peaks that reach almost 10% at the second transition to the coupled mode. These errors might be acceptable if one considers the appearance of the trajectories in Figure 4.12.

4.4. Discussion

What this chapter shows is that even for the simplest case, finding a trajectory model is not trivial, but still possible. The results also show that it is possible to speed up a co-simulation by decoupling the simulators, provided that the interface variables are predictable for a large enough portion of the co-simulation. Furthermore, there are cases where detecting predictable interface variables becomes so computationally expensive that the selectively-decoupled co-simulation turns out to be slower than the traditional co-simulation. However, in these test cases both simulators run on the same computer, making the communication delay between them rather small. For a large enough communication delay, speedup will nevertheless be possible.

The selective decoupling method as presented in Chapter 3 requires that the simulators are able to roll back in time. As I already mentioned, this can be memory consuming, time consuming, and unlikely to be a feature of most electrical power system simulators. An alternative to rolling back could be that all simulators run at the same rate while in decoupled mode. Yet, this is difficult to ensure in a non-real time environment. Another alternative could be to have all simulators catch up to the simulator that has progressed the most. However, this would lead to a loss of accuracy when the simulator that has progressed the most is not the one requesting recoupling. Additionally, such approach would yield non-deterministic results because on every run, the simulators would recouple at a different point in simulation time. Perhaps the most practical approach would be for the simulators to exchange synchronization messages at a low

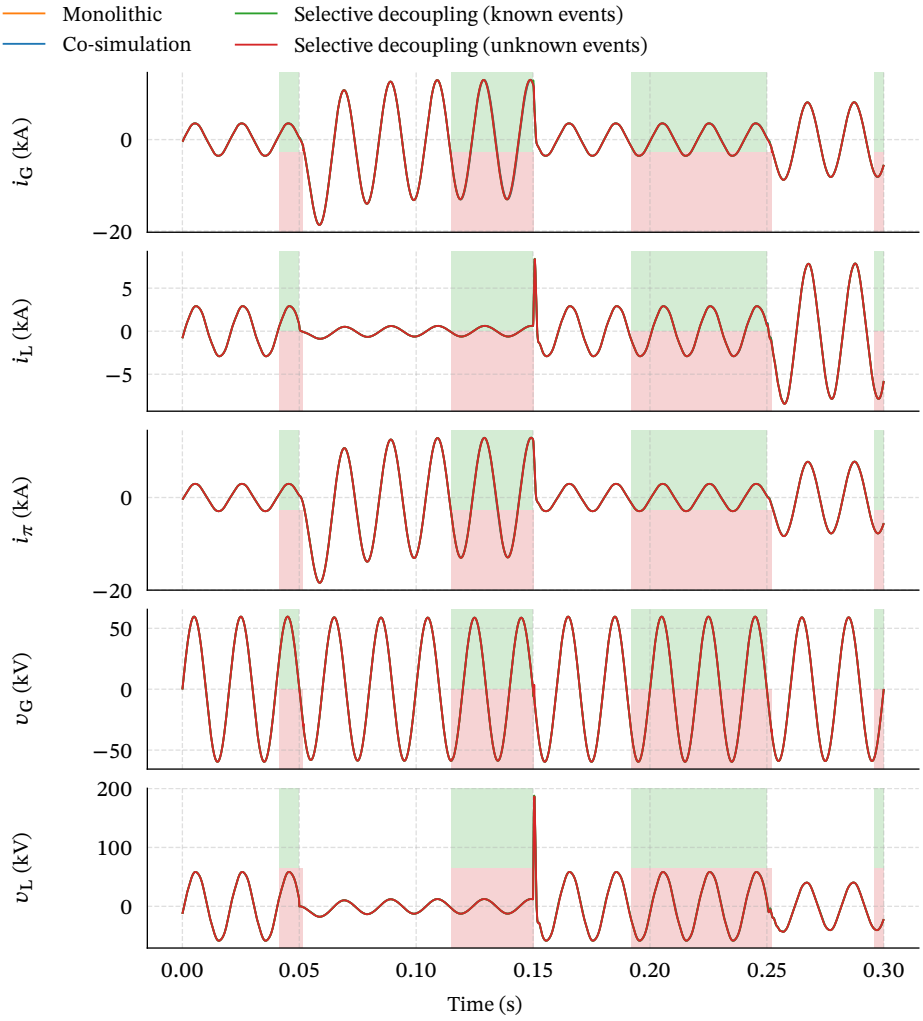


Figure 4.12: State variables computed with a monolithic simulation, a traditional co-simulation, a selectively-decoupled co-simulation with known event times, and a selectively-decoupled co-simulation with unknown event times. The colored background indicates when the co-simulation is in decoupled mode (green for known event times, red for unknown event times). Note that $v_L = v_{\pi_2}$ and $v_G = v_{\pi_1}$.

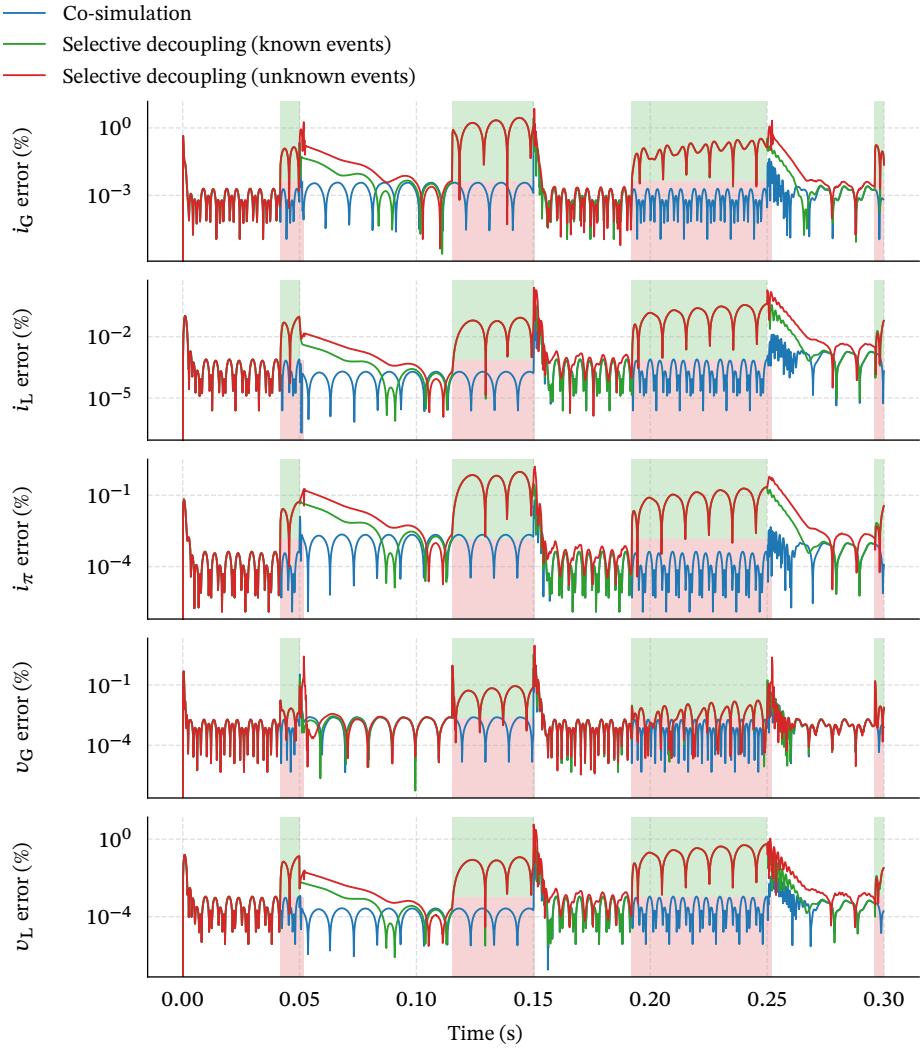


Figure 4.13: State variable errors of the traditional co-simulation, the selectively-decoupled co-simulation with known event times, and the selectively-decoupled co-simulation with unknown event times from Case 5. The errors are measured with respect to the monolithic simulation and are in percent of the dynamic range of the corresponding state variable. The colored background indicates when the co-simulation is in decoupled mode (green for known event times, red for unknown event times).

rate, to prevent any one of them from advancing too far ahead from the rest.

Finally, I must once more acknowledge how this test is limited by the simplicity of the chosen trajectory model. The circuit model that I used for these tests is meant for representing only fast electromagnetic phenomena, so once an electromagnetic transient fades away, the circuit returns quickly to steady state. This is not the case with power system models that consider electromechanical generators. In their case, after an electromagnetic transient fades, an electromechanical transient might still be occurring. Under such circumstances, the steady state trajectory model will undoubtedly fall short. Thus, it is important to find a trajectory model that can describe interface variables in a wider range of conditions. I will address this issue, as well as the rollback issue, in the next chapter.

Bibliography

[1] C. D. López, M. Cvetković, and P. Palensky, "Speeding up AC circuit co-simulations through selective simulator decoupling of predictable states", *IEEE Access*, vol. 7, pp. 1–14, Apr. 2019.

[2] A. A. der Meer, "Offshore VSC-HVDC networks–Impact on transient stability of AC transmission systems", PhD thesis, Delft University of Technology, Sep. 2017.

[3] J. O. Smith III and X. Serra, "PARSHL: A program for the analysis/synthesis of inharmonic sounds based on a sinusoidal representation", in *International Computer Music Conference*, Champaign-Urbana, Illinois, 1987.

[4] K. J. Werner, "The XQIFFT: Increasing the accuracy of quadratic interpolation of spectral peaks via exponential magnitude spectrum weighting", in *International Computer Music Conference (ICMC)*, Denton, Texas, Sep. 2015.

[5] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform", *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, Jan. 1978.

[6] K. Levenberg, "A method for the solution of certain non-linear problems in least squares", *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[7] R. Thomas, "Fast calculation of electrical transients in power systems after a change of topology", PhD thesis, Delft University of Technology, Nov. 2017.

[8] E. Jones, T. Oliphant, P. Peterson, *et al.* (2001–Present). SciPy: Open source scientific tools for Python, [Online]. Available: `www.scipy.org`.

[9] iMatix. (2007–Present). ØMQ, [Online]. Available: `zeromq.org`.

[10] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae", *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

5

Modeling Interface Variables in AC Systems Experiencing Slow Transients

Parts of this chapter have been published in [1].

THE PRACTICALITY of the selective decoupling method depends on the availability of a trajectory model that is valid under conditions beyond steady state. In this chapter I propose a trajectory model that describes interface variables while the system is experiencing a slow, electromechanical transient. I also propose a criterion for preventive recoupling to avoid rollbacks, and describe how the co-simulation must operate to take advantage of these ideas.

5.1. A Trajectory Model for Slow Transients

For a power system subject to electromechanical transients, the interface variables are of the form

$$\hat{u}(t) = \sum_{n=0}^{N} A_n(t) \cos\left(2\pi f_n t + \phi_n(t)\right), \tag{5.1}$$

where $A_n(t)$, f_n and $\phi_n(t)$ are the amplitude, frequency and phase of the n^{th} harmonic, and N is the total number of harmonics. In contrast to the trajectory model for steady states described by (4.1), which is defined by a set of parameters, the trajectory model for electromechanical transients described by (5.1) is defined by two functions of time. This makes it much more challenging to identify. One way to overcome this challenge is to transform this function identification problem into a parameter identification problem.

Let us consider the trajectory model for the n^{th} harmonic

$$\begin{aligned}
\hat{u}_n(t) &= A_n(t) \cos\left(\omega_n t + \phi_n(t)\right) && (5.2)\\
&= \frac{1}{2}\left(p_n(t) e^{j\omega_n t} + \overline{p}_n(t) e^{-j\omega_n t}\right) && (5.3)\\
&= \Re\left(p_n(t) e^{j\omega_n t}\right), && (5.4)
\end{aligned}$$

where $\omega_n = 2\pi f_n$ and $p_n(t) = A_n(t) e^{j\phi_n(t)}$ is a dynamic phasor. Approximating $p_n(t)$ with an M^{th} order Taylor expansion about an arbitrary $t_k \in \boldsymbol{t}$ yields

$$p_{n,M}(t) = \sum_{m=0}^{M} p_n^{(m)}(t_k) \frac{(t - t_k)^m}{m!}, \tag{5.5}$$

where $p_n^{(m)}(t_k)$ is the m^{th} derivative of $p_n(t)$ at $t = t_k$, so the trajectory model can be approximated as

$$\begin{aligned}
\hat{u}(t) &= \sum_{n=0}^{N} \hat{u}_n(t) && (5.6)\\
&\approx \sum_{n=0}^{N} \Re\left(p_{n,M}(t) e^{j\omega_n t}\right) && (5.7)\\
&= \sum_{n=0}^{N} \sum_{m=0}^{M} \Re\left(p_n^{(m)}(t_k) e^{j\omega_n t}\right) \frac{(t - t_k)^m}{m!}. && (5.8)
\end{aligned}$$

Using this approximation, identifying the trajectory model $\hat{u}(t)$ is equivalent to identifying the set of parameters $p_n^{(m)}(t_k)$, $n \in \{0, 1 \dots N\}$, $m \in \{0, 1 \dots M\}$, which can be accomplished with a dynamic phasor estimation method.

5.2. Dynamic Phasor Estimation with Taylor-Kalman Filters

The Taylor-Kalman filter, originally proposed in [2], [3], and later improved in [4], [5], is an application of the discrete Kalman filter to the problem of estimating a dynamic phasor from a measurement of its associated waveform. Out of the existing dynamic phasor estimation methods, the Taylor-Kalman filter is particularly interesting because it can estimate a dynamic phasor and M of its derivatives, without delay [6]. Since the Taylor-Kalman filter is an application of the discrete Kalman filter, at this point it becomes helpful to briefly discuss the discrete Kalman filter algorithm and how to apply it.

5.2.1. The Discrete Kalman Filter Algorithm

A discrete Kalman filter is a recursive filter that estimates the states $\boldsymbol{x}$ of a system affected by Gaussian noise, using a discrete, linear, stochastic model of said system, and a noisy measurement of its outputs $\boldsymbol{z}$. The model takes the form

$$\boldsymbol{x}[k+1] = \boldsymbol{\mathcal{F}}\,\boldsymbol{x}[k] + \boldsymbol{\mathcal{G}}\,\boldsymbol{u}[k] + \boldsymbol{w}[k] \tag{5.9}$$

$$\boldsymbol{y}[k+1] = \boldsymbol{\mathcal{H}}\,\boldsymbol{x}[k+1] + \boldsymbol{v}[k+1], \tag{5.10}$$

where $\boldsymbol{u}$ are the model inputs, $\boldsymbol{y}$ are the model outputs, $\boldsymbol{w}$ represents process noise, $\boldsymbol{v}$ represents output measurement noise, and k is the discrete index. Note that according to this description, $\boldsymbol{y}$ is an estimation of the system outputs based on a model of the real system, whereas $\boldsymbol{z}$ is a measurement of the real outputs.

If both a model of the system in the form of (5.9) and (5.10), and the system outputs are available, the states can be estimated by applying the discrete Kalman algorithm described in Algorithm 5.1.

5.2.2. Derivation of the Taylor-Kalman Filter

To derive the Taylor-Kalman filter, a state transition equation in the form of (5.9) and an output equation in the form of (5.10) must be found. The equations must describe a process whose output is a waveform (i.e., an interface variable), and whose states are related to the dynamic phasor that produces the waveform. For this, it is possible to start from the Taylor expansion of the n^{th} harmonic described by (5.5), this time about a time origin t_k. By defining $\tau \coloneqq t - t_k$, it follows that the dynamic phasor and its first

Algorithm 5.1 Discrete Kalman filter

Define: discrete index $\Bbbk$, number of output measurements $\mathcal{K}$, state error covariance matrix $\boldsymbol{\mathcal{P}}$, process noise covariance matrix $\mathbb{Q}$, measurement noise covariance matrix $\boldsymbol{\mathcal{R}}$, Kalman gain $\boldsymbol{\mathcal{K}}$, estimated states $\hat{\boldsymbol{x}}$, $\square^-$ prediction of any variable $\square$, and $\square^{\mathsf{H}}$ Hermitian of $\square$.

$\hat{\boldsymbol{x}}[0] \leftarrow \hat{\boldsymbol{x}}_0$
$\boldsymbol{\mathcal{P}}[0] \leftarrow \boldsymbol{\mathcal{P}}_0$
$\Bbbk \leftarrow 0$

while $\Bbbk < \mathcal{K}$ **do**

▷ time update (prediction)
$\hat{\boldsymbol{x}}^-[\Bbbk+1] = \boldsymbol{\mathcal{F}}\hat{\boldsymbol{x}}[\Bbbk] + \boldsymbol{\mathcal{G}}\boldsymbol{u}[\Bbbk]$ ▷ state prediction
$\boldsymbol{\mathcal{P}}^-[\Bbbk+1] = \boldsymbol{\mathcal{F}}\boldsymbol{\mathcal{P}}[\Bbbk]\boldsymbol{\mathcal{F}}^{\mathsf{H}} + \mathbb{Q}$ ▷ covariance prediction

▷ measurement update (correction)
$\boldsymbol{\mathcal{K}}[\Bbbk+1] = \boldsymbol{\mathcal{P}}^-[\Bbbk+1]\boldsymbol{\mathcal{H}}^{\mathsf{T}}\left(\boldsymbol{\mathcal{H}}\boldsymbol{\mathcal{P}}^-[\Bbbk+1]\boldsymbol{\mathcal{H}}^{\mathsf{T}} + \boldsymbol{\mathcal{R}}\right)^{-1}$ ▷ gain update
$\hat{\boldsymbol{x}}[\Bbbk+1] = \hat{\boldsymbol{x}}^-[\Bbbk+1] + \boldsymbol{\mathcal{K}}[\Bbbk+1]\left(\boldsymbol{z}[\Bbbk+1] - \boldsymbol{\mathcal{H}}\hat{\boldsymbol{x}}^-[\Bbbk+1]\right)$ ▷ state update
$\boldsymbol{\mathcal{P}}[\Bbbk+1] = \left(\boldsymbol{I} - \boldsymbol{\mathcal{K}}[\Bbbk+1]\boldsymbol{\mathcal{H}}\right)\boldsymbol{\mathcal{P}}^-[\Bbbk+1]$ ▷ covariance update

$\Bbbk \leftarrow \Bbbk + 1$

end while

M derivatives can be approximated as

$$\begin{aligned} p_{n,M}^{(0)}(t) &= p_{n,M}^{(0)}(t_\Bbbk) + p_{n,M}^{(1)}(t_\Bbbk)\tau + \cdots + p_{n,M}^{(M)}(t_\Bbbk)\frac{\tau^M}{M!} \\ p_{n,M}^{(1)}(t) &= p_{n,M}^{(1)}(t_\Bbbk) + p_{n,M}^{(2)}(t_\Bbbk)\tau + \cdots + p_{n,M}^{(M)}(t_\Bbbk)\frac{\tau^{M-1}}{(M-1)!} \\ &\vdots \\ p_{n,M}^{(M)}(t) &= p_{n,M}^{(M)}(t_\Bbbk). \end{aligned} \tag{5.11}$$

Defining

$$\boldsymbol{p}_{n,M}(t) \coloneqq \begin{bmatrix} p_{n,M}^{(0)}(t) \\ p_{n,M}^{(1)}(t) \\ \vdots \\ p_{n,M}^{(M)}(t) \end{bmatrix} \tag{5.12}$$

and

$$\boldsymbol{\Phi}_M(\tau) \coloneqq \begin{bmatrix} 1 & \tau & \dots & \frac{\tau^M}{M!} \\ & 1 & \dots & \frac{\tau^{M-1}}{(M-1)!} \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix}, \tag{5.13}$$

equation (5.11) can be expressed in matrix form as

$$\boldsymbol{p}_{n,M}(t) = \boldsymbol{\Phi}_M(\tau)\boldsymbol{p}_{n,M}(t_{\hbar}). \tag{5.14}$$

Finally, a state transition equation in the form of (5.9) can be obtained by substituting $t_{\hbar} = \hbar\tau$ and $t = (\hbar+1)\tau$ in (5.14), and adding noise to the discrete equation that results, which produces

$$\boldsymbol{p}_{n,M}[\hbar+1] = \boldsymbol{\Phi}_M(\tau)\boldsymbol{p}_{n,M}[\hbar+1] + \boldsymbol{w}[\hbar]. \tag{5.15}$$

Comparing (5.15) to the state transiton equation stated in (5.9) it becomes apparent that $\boldsymbol{x} = \boldsymbol{p}_{n,M}$, $\boldsymbol{\mathcal{F}} = \boldsymbol{\Phi}_M$ and $\boldsymbol{\mathcal{G}} = \boldsymbol{0}$.

Having found a suitable state transition equation, the following step is to find an output equation in the form of (5.10), whose states are $\boldsymbol{x} = \boldsymbol{p}_{n,M}$. To put this derivation in context, let us consider that the output waveform is the n^{th} harmonic y_n of a co-simulation interface variable y. Thus, an M^{th} order approximation of y_n is

$$y_{n,M}(t) = \Re\left(\boldsymbol{h}^{\mathsf{T}}\boldsymbol{p}_{n,M}(t)e^{j\omega_n t}\right) \tag{5.16}$$

where $\boldsymbol{h}^{\mathsf{T}} = [1\,0\,\dots\,0] \in \mathbb{N}^{1\times(M+1)}$. However, even if (5.16) were discretized, it would not be possible to identify a $\boldsymbol{\mathcal{H}}$ for two reasons: the presence of the $\Re$ operator and the $e^{j\omega_n t}$ factor, which makes the equation time-variant. The first problem can be solved by taking advantage of the property $\Re(c) = 1/2(c+\bar{c})$, $c \in \mathbb{C}$, and including the conjugate of the dynamic phasor and derivatives in the state vector. The second problem can be solved by not using the dynamic phasor as a state variable, but instead a rotating dyamic phasor defined as

$$r_{n,M}(t) = p_{n,M}(t)e^{j\omega_n t}, \tag{5.17}$$

as to absorb the time-variant factor $e^{j\omega_n t}$ into the state variables. With these two changes, the state vector and all the matrices that define (5.9), (5.10), must be derived again. Following the same logic as before, the derivatives of $r_{n,M}(t)$ are

$$\begin{aligned} r^{(0)}_{n,M}(t) &= p^{(0)}_{n,M}(t)e^{j\omega_n t} \\ r^{(1)}_{n,M}(t) &= j\omega_n p^{(0)}_{n,M}(t)e^{j\omega_n t} + p^{(1)}_{n,M}(t)e^{j\omega_n t} \\ r^{(2)}_{n,M}(t) &= (j\omega_n)^2 p^{(0)}_{n,M}(t)e^{j\omega_n t} + 2j\omega_n p^{(1)}_{n,M}(t)e^{j\omega_n t} + p^{(2)}_{n,M}(t)e^{j\omega_n t} \\ &\vdots \end{aligned} \tag{5.18}$$

which rearranged in matrix form becomes

$$\boldsymbol{r}_{n,M}(t) = \begin{bmatrix} r_{n,M}^{(0)}(t) \\ r_{n,M}^{(1)}(t) \\ \vdots \\ r_{n,M}^{(M)}(t) \end{bmatrix} = e^{j\omega_n t}\boldsymbol{\mathcal{M}}_{n,M}\boldsymbol{p}_{n,M}(t), \tag{5.19}$$

where

$$\boldsymbol{\mathcal{M}}_{n,M} = \begin{bmatrix} 1 & & & \\ j\omega_n & 1 & & \\ \vdots & \vdots & \ddots & \\ (j\omega_n)^M & M(j\omega_n)^{M-1} & \dots & 1 \end{bmatrix} \in \mathbb{C}^{(M+1)\times(M+1)} \tag{5.20}$$

is a non-singular matrix whose element in row p and column q is

$$m_{pq} = \begin{cases} \binom{p-1}{q-1}(j\omega_n)^{p-q}, & p \geq q \\ 0, & p < q \end{cases}. \tag{5.21}$$

The next step is to establish a relationship between $\boldsymbol{r}_{n,M}(t)$ and $\boldsymbol{r}_{n,M}(t_k)$. Evaluating $\boldsymbol{r}_{n,M}(t)$ at t_k produces

$$\boldsymbol{r}_{n,M}(t_k) = e^{j\omega_n t_k}\boldsymbol{\mathcal{M}}_{n,M}\boldsymbol{p}_{n,M}(t_k), \tag{5.22}$$

and solving for $\boldsymbol{p}_{n,M}(t_k)$ yields

$$\boldsymbol{p}_{n,M}(t_k) = e^{-j\omega_n t_k}\boldsymbol{\mathcal{M}}_{n,M}^{-1}\boldsymbol{r}_{n,M}(t_k). \tag{5.23}$$

Substituting (5.14) in (5.19)

$$\boldsymbol{r}_{n,M}(t) = e^{j\omega_n \tau}\boldsymbol{\mathcal{M}}_{n,M}\boldsymbol{\Phi}_M(\tau)\boldsymbol{p}_{n,M}(t_k), \tag{5.24}$$

and substituting (5.23) in (5.24)

$$\boldsymbol{r}_{n,M}(t) = e^{j\omega_n t}e^{-j\omega_n t_k}\boldsymbol{\mathcal{M}}_{n,M}\boldsymbol{\Phi}_M(\tau)\boldsymbol{\mathcal{M}}_{n,M}^{-1}\boldsymbol{r}_{n,M}(t_k) \tag{5.25}$$

$$= e^{j\omega_n \tau}\boldsymbol{\Psi}_{n,M}(\tau)\boldsymbol{r}_{n,M}(t_k), \tag{5.26}$$

with $\boldsymbol{\Psi}_{n,M}(\tau) = \boldsymbol{\mathcal{M}}_{n,M}\boldsymbol{\Phi}_M(\tau)\boldsymbol{\mathcal{M}}_{n,M}^{-1}$. Now, (5.26) can be discretized by substituting $t_k = k\tau$ and $t = (k+1)\tau$ to produce

$$\boldsymbol{r}_{n,M}[k+1] = e^{j\omega_n \tau}\boldsymbol{\Psi}_{\boldsymbol{n,M}}(\tau)\boldsymbol{r}_{n,M}[k]. \tag{5.27}$$

To build a state transition equation in the form of (5.9), it is necessary to include the conjugate of $\boldsymbol{r}_{n,M}$ in the state vector for the reasons explained above, which gives

$$\begin{bmatrix} \boldsymbol{r}_{n,M}[\hbar+1] \\ \overline{\boldsymbol{r}}_{n,M}[\hbar+1] \end{bmatrix} = \begin{bmatrix} e^{j\omega_n\tau}\boldsymbol{\Psi}_{\boldsymbol{n,M}}(\tau) & \\ & e^{-j\omega_n\tau}\overline{\boldsymbol{\Psi}}_{m,M}(\tau) \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_{n,M}[\hbar] \\ \overline{\boldsymbol{r}}_{n,M}[\hbar] \end{bmatrix} + \boldsymbol{w}[\hbar]. \tag{5.28}$$

Comparing (5.28) to (5.9) it becomes apparent that

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r}_{n,M}[\hbar+1] \\ \overline{\boldsymbol{r}}_{n,M}[\hbar+1] \end{bmatrix}, \tag{5.29}$$

$$\boldsymbol{\mathcal{F}} = \begin{bmatrix} e^{j\omega_n\tau}\boldsymbol{\Psi}_{n,M}(\tau) & \\ & e^{-j\omega_n\tau}\overline{\boldsymbol{\Psi}}_{n,M}(\tau) \end{bmatrix}, \text{and} \tag{5.30}$$

$$\boldsymbol{\mathcal{G}} = \boldsymbol{0}. \tag{5.31}$$

Having defined the state vector as $\boldsymbol{x} = [\boldsymbol{r}_{n,M}\ \overline{\boldsymbol{r}}_{n,M}]^{\mathsf{T}}$, an M^{th} order approximation of y_n is

$$y_{n,M}(t) = \Re\left(\boldsymbol{h}^{\mathsf{T}}\boldsymbol{r}_{n,M}(t)\right) = \frac{1}{2}\begin{bmatrix}\boldsymbol{h}^{\mathsf{T}} & \boldsymbol{h}^{\mathsf{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{r}_{n,M}(t) \\ \overline{\boldsymbol{r}}_{n,M}(t)\end{bmatrix}, \tag{5.32}$$

where $\boldsymbol{h}^{\mathsf{T}} = [1\,0\,\ldots\,0] \in \mathbb{N}^{1\times(M+1)}$. Discretizing (5.32) and adding Gausian noise produces

$$y_{n,M}[\hbar+1] = \frac{1}{2}\begin{bmatrix}\boldsymbol{h}^{\mathsf{T}} & \boldsymbol{h}^{\mathsf{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{r}_{n,M}[\hbar+1] \\ \overline{\boldsymbol{r}}_{n,M}[\hbar+1]\end{bmatrix} + \boldsymbol{v}[\hbar+1], \tag{5.33}$$

which when compared to (5.10) reveals that $\boldsymbol{\mathcal{H}} = {}^1\!/_2[\boldsymbol{h}^{\mathsf{T}}\ \boldsymbol{h}^{\mathsf{T}}]$.

The last step needed before applying Algorithm 5.1 to the dynamic phasor estimation problem is defining the noise covariance matrices, which can be set to $\mathbb{Q} = \boldsymbol{I}\sigma_v^2$ and $\mathbb{Q} = \sigma_w^2$, where $\boldsymbol{I}$ is an identity matrix of size $2M$, and σ_v^2 and σ_w^2 are the variances of the process and output measurement noise.

Finally, the estimated dynamic phasor is

$$\hat{\boldsymbol{p}}_{n,M}[\hbar+1] = e^{-j\omega_n(\hbar+1)\tau}\boldsymbol{\mathcal{M}}_{n,M}^{-1}\hat{\boldsymbol{r}}_{n,M}[\hbar+1]. \tag{5.34}$$

Each element in the $\hat{\boldsymbol{p}}_{n,M}$ vector is an estimation of the dynamic phasor or one of its derivatives, which define the trajectory model from (5.8).

5.3. Extension to Harmonics

To extend the Taylor-Kalman filter to waveforms with harmonic infiltration, it is necessary to define a state vector that considers all harmonics of interest. Defining

$$\boldsymbol{r}_M = \begin{bmatrix} \boldsymbol{r}_{1,M} \\ \bar{\boldsymbol{r}}_{1,M} \\ \vdots \\ \boldsymbol{r}_{N,M} \\ \bar{\boldsymbol{r}}_{N,M} \end{bmatrix}, \tag{5.35}$$

it is possible to build an output equation as

$$y_M[k+1] = \frac{1}{2}\begin{bmatrix} \boldsymbol{h}^\mathsf{T} & \boldsymbol{h}^\mathsf{T} & \dots & \boldsymbol{h}^\mathsf{T} & \boldsymbol{h}^\mathsf{T} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_{1,M}[k+1] \\ \bar{\boldsymbol{r}}_{1,M}[k+1] \\ \vdots \\ \boldsymbol{r}_{N,M}[k+1] \\ \bar{\boldsymbol{r}}_{N,M}[k+1] \end{bmatrix} + \boldsymbol{v}[k+1], \tag{5.36}$$

so it follows that $\boldsymbol{x} = \boldsymbol{r}_M$ and $\mathscr{H} = 1/2[\boldsymbol{h}^\mathsf{T}\ \boldsymbol{h}^\mathsf{T} \dots \boldsymbol{h}^\mathsf{T}\ \boldsymbol{h}^\mathsf{T}]$, and that the state transition matrix should be

$$\mathscr{F} = \begin{bmatrix} \mathscr{F}_1 & & & & & \\ & \mathscr{F}_2 & & & & \\ & & \ddots & & & \\ & & & \mathscr{F}_n & & \\ & & & & \ddots & \\ & & & & & \mathscr{F}_N \end{bmatrix}, \tag{5.37}$$

with

$$\mathscr{F}_n = \begin{bmatrix} e^{j\omega_n\tau}\boldsymbol{\Psi}_{n,M}(\tau) & \\ & e^{-j\omega_n\tau}\overline{\boldsymbol{\Psi}}_{n,M}(\tau) \end{bmatrix}. \tag{5.38}$$

5.4. Rollback Prevention

Rollbacks are undesirable and better avoided. One of the reasons why simulators need to recouple, and thus roll back, is limitations of the trajectory model. To prevent this type of rollback, it would be useful to determine when a trajectory model will deviate from the true trajectory beyond ϵ_p, so that the simulators can determine in advance when to recouple. Since the estimation of the dynamic phasor is based on a Taylor expansion, one way to do this is to rely on an upper error bound of the Taylor series.

Considering an M^{th} order expansion, it is possible to define the upper error bounds for the real and imaginary parts of $p_n(t)$ as

$$R_{M_{\Re}}(t) = P_{n_{\Re}}^{(M+1)} \frac{(t-t_k)^{M+1}}{(M+1)!} \tag{5.39}$$

$$R_{M_{\Im}}(t) = P_{n_{\Im}}^{(M+1)} \frac{(t-t_k)^{M+1}}{(M+1)!}, \tag{5.40}$$

where $P_{n_{\Re}}^{(M+1)}$ is the upper bound of $\Re(p_n^{(M+1)})$, and $P_{n_{\Im}}^{(M+1)}$ is the upper bound of $\Im(p_n^{(M+1)})$, in the interval between t and t_k. For simplicity I will assume that $p_n^{(M+1)}$ is constant in the interval of interest, so $P_{n_{\Re}}^{(M+1)} = \Re(p_n^{(M+1)})$ and $P_{n_{\Im}}^{(M+1)} = \Im(p_n^{(M+1)})$. This means that the deviation of the trajectory model from the true trajectory fulfills

$$|\hat{y}_n(t) - y_n(t)| \leq |\Re\left((R_{M_{\Re}}(t) + jR_{M_{\Im}}(t))e^{j\omega_n t}\right)|, \tag{5.41}$$

which can be related to ϵ_{p} as

$$\epsilon_{\text{p}} = \left|\frac{\hat{y}_n(t) - y_n(t)}{\max \hat{y}(t) - \min \hat{y}(t)}\right| \leq \left|\frac{\Re\left((R_{M_{\Re}}(t) + jR_{M_{\Im}}(t))e^{j\omega_n t}\right)}{\max \hat{y}(t) - \min \hat{y}(t)}\right|, \tag{5.42}$$

so the trajectory model should not deviate from the true trajectory more than ϵ_{p} before t_{R}, which is the largest $t \in \{t_k, t_{k+1} \dots t_K\}$ that fulfills (5.42), that is

$$t_{\text{R}} = \max t \in \{t_k, t_{k+1} \dots t_K\} : \epsilon_{\text{p}} \leq \left|\frac{\Re\left((R_{M_{\Re}}(t) + jR_{M_{\Im}}(t))e^{j\omega_n t}\right)}{\max \hat{y}(t) - \min \hat{y}(t)}\right|. \tag{5.43}$$

This is a transcendental inequality, but it can be easily solved by trial and error since the solution lays in $\{t_k, t_{k+1} \dots t_K\}$, which is a discrete and finite set.

To implement this idea, a Taylor-Kalman filter of order $M+1$ is required. Instead of using two filters, one of order M to find the trajectory model and one of order $M+1$ to determine its validity, I will use only one filter of order $M+1$. I will use the first M derivatives to build the trajectory model, and the $(M+1)^{\text{th}}$ derivative to determine t_{R}.

5.5. Trajectory Model Exchange and Preventive Recoupling

Strictly speaking, it is not necessary that each simulator models both its inputs and outputs, as in Algorithm 3.1. Since the outputs of a subsystem become the inputs of another, it is sufficient that each simulator models only its own outputs, and that it exchanges these trajectory models with the rest, whenever needed. The advantage of this approach is that fewer Taylor-Kalman filters are needed. Combining this idea with the preventive recoupling criterion I proposed in the previous section produces Algorithm 5.2.

5.6. Discussion

In this chapter I introduced main points. The first one is the trajectory model for slow transients, and the method for estimating its parameters. The second one is the criterion for recoupling simulators preventively and reducing the need for rollbacks. And

Algorithm 5.2 Selectively-decoupled co-simulation with predictive recoupling (part 1)

Define: subsystem s, macro time step index k, recoupling macro time step index k_r, $t_k \in \{t_0, t_1 \dots t_k \dots t_K\}$ a discrete time grid, $\boldsymbol{t}_w = \{t_{k-N_s+1}, t_{k-N_s+2} \dots t_k\}$ a discrete moving time widow, hop size R_w, and ϵ_p the allowed normalized deviation.

Initialize $\dot{\boldsymbol{x}}_s = \boldsymbol{f}_s(\boldsymbol{x}_s, \boldsymbol{u}_s), \boldsymbol{y}_s = \boldsymbol{g}_s(\boldsymbol{x}_s, \boldsymbol{u}_s)$
mode $\leftarrow$ COUPLED
$k \leftarrow 0$

while $k < K$ **do**
  if mode = COUPLED **then**

    if k mod $R_w = 0$ **then**
      Find trajectory models $\hat{\boldsymbol{y}}_{s,k}(t)$ and their validities $\hat{\boldsymbol{v}}_{y,s,k}$ using (5.43)
    end if

    if $\boldsymbol{y}_s(t), t \in \boldsymbol{t}_w$ are predictable according to (3.1) **then**
      Request decoupling
      Send trajectory models $\hat{\boldsymbol{y}}_{s,k}(t)$ and and their validities $\hat{\boldsymbol{v}}_{y,s,k}$
    end if

    Send $\boldsymbol{y}_s(t_k)$ and receive $\boldsymbol{u}_s(t_k)$

    if Decoupling request accepted **then**
      Receive trajectory models $\hat{\boldsymbol{u}}_{s,k}(t)$ and their validities $\hat{\boldsymbol{v}}_{u,s,k}$
      $\boldsymbol{u}_s(t) \leftarrow \hat{\boldsymbol{u}}_{s,k}(t)$
      Create recoupling event at $t = \min\{\hat{\boldsymbol{v}}_{u,s,k}, \hat{\boldsymbol{v}}_{y,s,k}\}$
      mode $\leftarrow$ DECOUPLED
    else
      Create interpolation polynomials $\tilde{\boldsymbol{u}}_{s,k}(t)$
      $\boldsymbol{u}_s(t) \leftarrow \tilde{\boldsymbol{u}}_{s,k}(t)$
    end if

▷ Continues in Algorithm 5.3

the third one is that the simulators can exchange trajectory models before decoupling, to reduce the number of Taylor-Kalman filters required for finding the parameters of trajectory models. Although the co-simulation framework I used in Chapter 4 could be extended to test these ideas, including models of AC systems that exhibit slow transients would be challenging. A co-simulation framework that uses specialized power system simulators would be more suitable. I will introduce this framework in the next chapter.

Algorithm 5.3 Selectively-decoupled co-simulation with predictive recoupling (part 2)

30: **if** Recoupling requested **then**
31: Receive recoupling index k_r
32: Roll back or catch up to $k = k_\mathrm{r}$
33: `mode` $\leftarrow$ `COUPLED`
34: **else if** $\exists y(t_k) \in \boldsymbol{y}_s(t_k) : y(t_k)$ is unpredictable according to (3.3) **then**
35: $k_\mathrm{r} \leftarrow k - 1$
36: Request recoupling at k_r
37: **else if** Event in next macro time step **then**
38: `mode` $\leftarrow$ `COUPLED`
39: **end if**
40:
41: **end if**
42:
43: Solve $\dot{\boldsymbol{x}}_s = \boldsymbol{f}_s(\boldsymbol{x}_s, \boldsymbol{u}_s(t)), \boldsymbol{y}_s = \boldsymbol{g}_s(\boldsymbol{x}_s, \boldsymbol{u}_s(t))$ until $t = t_{k+1}$
44:
45: $k \leftarrow k + 1$
46: **end while**

Bibliography

[1] C. D. López, A. A. van der Meer, M. Cvetković, and P. Palensky, "Reducing the need for communication in natural waveform co-simulations of electrical power systems using adaptive interfaces", Submitted to IEEE Access.

[2] J. A. de la O Serna and J. Rodriguez-Maldonado, "Instantaneous oscillating phasor estimates with Taylor^K-Kalman filters", *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 2336–2344, Jun. 2011.

[3] J. A. de la O Serna and J. Rodríguez-Maldonado, "Taylor-Kalman-Fourier filters for instantaneous oscillating phasor and harmonic estimates", *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 4, pp. 941–951, Jan. 2012.

[4] J. Liu, F. Ni, J. Tang, F. Ponci, and A. Monti, "A modified Taylor-Kalman filter for instantaneous dynamic phasor estimation", in *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, Oct. 2012, pp. 1–7.

[5] J. Liu, F. Ni, P. A. Pegoraro, F. Ponci, A. Monti, and C. Muscas, "Fundamental and harmonic synchrophasors estimation using modified taylor-kalman filter", in *2012 IEEE International Workshop on Applied Measurements for Power Systems (AMPS) Proceedings*, 2012, pp. 1–6.

[6] J. Khodaparast and M. Khederzadeh, "Least square and Kalman based methods for dynamic phasor estimation: A review", *Protection and Control of Modern Power Systems*, vol. 2, no. 1, pp. 1–18, Jan. 2017.

6

Co-Simulation Framework for Fast and Slow Transients

Parts of this chapter have been published in [1]–[4].

TESTING THE METHODS I proposed in Chapter 5 requires a co-simulation framework capable of simulating power system models that represent fast and slow phenomena. Specialized electrical power system simulators, such as DIgSILENT PowerFactory, can easily do so. But creating a framework that operates as I described in Chapter 5, based on a closed-source simulator, poses several implementation challenges. After experimenting with different approaches, such as those from [2], [3], I settled on the design I describe in this chapter. Some readers might find useful information in the descriptions of the interfaces, which can be used for purposes other than co-simulation [4], the limitations of this solution, and the workarounds I found to overcome them.

6.1. Requirements

The co-simulation framework has to comply with the following requirements to be suitable for testing the methods I proposed in Chapter 5:

R1 It must be able to reproduce electromagnetic (fast) and electromechanic (slow) transients.

R2 It must be flexible enough for implementing complex co-simulation interfaces, with non-standard functionality.

R3 The simulators must run with a constant macro time step for the Taylor-Kalman filters to work.

R4 The co-simulation must be able to roll back in time.

DIgSILENT PowerFactory complies with **R1**. With this simulator it is possible to run natural waveform simulations using full generator models, so that both electromagnetic and electromechanical phenomena are reproduced. Its unique model enhancement facilities can also be used to create complex co-simulation interfaces, which makes it compliant with **R2**. However, PowerFactory does not comply with **R3** and **R4**. I discuss workarounds to overcome these limitations in Section 6.4.

6.2. Design and Implementation

Figure 6.1 shows the structure of the co-simulation framework. Each subsystem is (co-) simulated in an different instance of PowerFactory 2019, running on a different Windows Server 2012 virtual server.

6.2.1. Communication

The simulators communicate with each other via a message bus, using a publisher/subscriber pattern implemented with the ØMQ messaging library [5]. In this pattern, each simulator publishes output messages, and subscribes to relevant input messages. The messages are encoded in JavaScript Object Notation (JSON).

6.2.2. Orchestration

Unlike in previous implementations of this framework [2], [3], in this case I opted for a design without a co-simulation master. As I explained in Chapter 2, this has the ad-

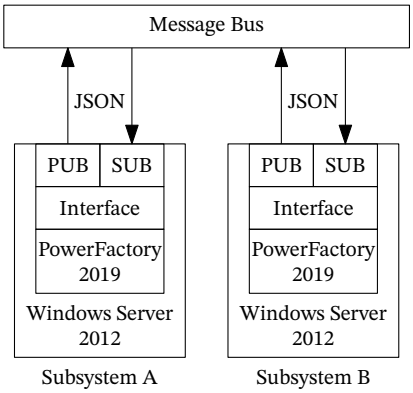


Figure 6.1: Co-simulation framework. The framework is based on the PowerFactory simulator. Each simulator runs on an independent Windows virtual server. They communicate via publisher/subscriber sockets. There is no co-simulation master, so the co-simulation interfaces are also in charge of orchestration.

vantage of eliminating a possible bottleneck. It also has the advantage that there is one less program to maintain. With a co-simulation master, a change in the interface might require a change in the master as well. The disadvantage is that the interfaces must take care of all orchestration tasks, which increases their complexity.

The orchestration method works as follows. In coupled mode, each simulator publishes its outputs and waits until all necessary inputs have arrived before it executes the next macro time step. When a simulator is ready to transition to decoupled mode, it publishes this information. When all simulators are ready to transition, the transition happens. To transition back to coupled mode, it is sufficient that only one simulator publishes this information for all simulators to recouple.

6.2.3. Interfacing

Interfacing requires a mechanism for setting model variables based on simulator inputs, and getting the model variables that become simulator outputs. Figure 6.2 shows the structure of a co-simulation interface, displaying the flow of interface variables right before simulator decoupling, when the interface sends and receives trajectory models. Since the interface variables are voltage and current, controlled voltage and current sources can be used for setting variables. PowerFactory provides both of them in the standard model library. This appears at the bottom of Figure 6.2 for the case of an interface based on a current source.

The DIgSILENT Simulation Language (DSL) provides mechanisms for controlling these sources. DSL is a language intended for modeling dynamic systems, mainly aimed at modeling control structures. This means that with DSL it is possible to provide set points to voltage and current sources (i.e., setting inputs), and to measure model variables (i.e., getting outputs). Figure 6.2 also shows this interaction between a controlled source and a DSL block. In addition, DSL provides a library of commonly used macros, and allows the user to define new macros in C++. This means that all the functionality needed for creating trajectory models, evaluating them, and communicating with other

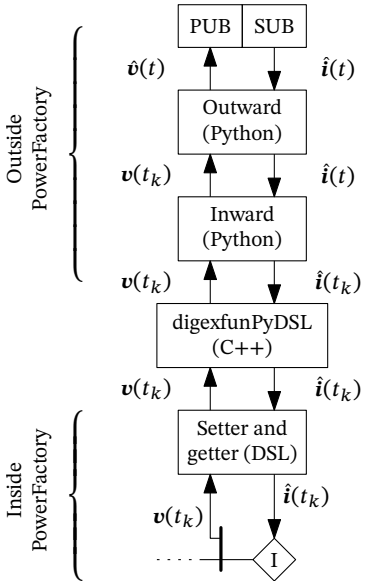


Figure 6.2: Structure of a co-simulation interface based on a controlled current source, displaying the flow of interface variables right before simulator decoupling. At this point the interface sends and receives trajectory models.

simulators can be embedded in a DSL macro.

Initially, I set out to implement all the required interface functionality in C++ [2], [3]. However, it soon became apparent that the development effort required was excessive. For this reason I developed the digexfunPyDSL library, which makes it possible to execute arbitrary Python code directly from DSL (see Section 6.3). Using this library I was able to implement the remaining interface functionality directly in Python. Aside from the fact that it is usually quicker to develop an interface in Python than in C++, this library also has the advantage that during development, there is not need for recompiling and restarting PowerFactory every time a change is introduced in the interface code, which speeds up the development process.

The portion of this interface I implemented in Python is composed of two objects, one facing inwards and one facing outwards, both show in Figure 6.2 as well. Both of these objects have an interface method that, when called, triggers the tasks each object is in charge of. The DSL macros I defined in digexfunPyDSL call the interface method from the inward-facing object once every micro time step. At the same time, this method calls the interface method from the outward-facing object once per macro time step. The inward-facing object is in charge of evaluating the input interpolation polynomials or the trajectory models it gets from the outward-facing object, and passing the calculated inputs to the DSL block that controls the current and voltage sources. It is also in charge of passing the outputs the DSL model measures to the outward-facing object. The responsibilities of the outward-facing object are to create trajectory models of the output variables, and interpolation/extrapolation polynomials of the input variables, as well as

orchestration and handling communication with other simulators.

6.2.4. Initialization

Initializing an unbalanced co-simulation in PowerFactory poses a challenge because three phase current sources cannot impose unbalanced initial conditions. To overcome this challenge, I implemented an interface composed of four current sources. The first one is a three phase current source and is used during the co-simulation. The other three are single phase current sources used only for initialization. These four sources must be connected to the grid node where the models are interfaced. At initialization, the three single phase sources are connected to the grid to impose initial conditions at the interface node, and as soon as the co-simulation begins, they disconnect from the grid and the three phase source connects instead. This happens automatically.

For finding the initial conditions it is possible to follow an iterative approach as the one I implemented in [2]. However, in this case I did not implement an initialization routine. Instead, I use the initial conditions from a monolithic simulation of the same system, which I have available as a benchmark for result accuracy.

6.2.5. Management

To manage the simulators, I reused the Python scripts I developed for [1]. These scripts take advantage of the PowerFactory Python API, and provide functionality to set up and run a simulator, and then retrieve its results. It is possible to execute these scripts remotely via a terminal or directly in each virtual server.

6.3. digexfunPyDSL

The digexfunPyDSL library [4] makes it possible to call arbitrary code from a PowerFactory DSL model. For this, it relies on the fact that it is possible to define new DSL macros in C++, and that Python is implemented in C, which for practical purposes is a subset of C++. The only requirement this library imposes on the Python code is that it must be callable from a Python function or method that takes an arbitrary number of floating point arguments, and returns an arbitrary number of floating point values.

The library provides four DSL macros to interact with such Python functions or methods. The first macro is `LoadPyFun`, which loads a Python function or method into PowerFactory. DSL macros do not support a variable number of arguments, so the arguments must be set one by one with sequential calls to the `SetPyFunArg` macro. Once all the arguments are set, the `CallPyFun` macro can be used to call the Python function or method. Finally, sequential calls to the `GetPyFunRetVal` macro can be used to retrieve the returned values one by one.

6.4. Simulator Limitations and Workarounds

As previously mentioned, PowerFactory complies neither with **R3** about constant time step nor **R4** about rollback. For this reason, I developed two workarounds to test the methods from Chapter 5. I describe these workarounds in the following sections.

6.4.1. Constant Time Step Limitation

PowerFactory does not provide a way to enforce a constant time step at runtime. Even if the option for adjustable time step is not selected, PowerFactory may still reduce its time step size. I identified two cases where this happens, both related to the occurrence of fast phenomena. One is when an event causes a discontinuity (e.g., a short circuit), and the other is when there are power electronic devices connected to the grid. Although it is possible to equip the co-simulation interface with output interpolation capabilities to make it seem, from the outside, as if the simulator had a constant time step size, this approach added numerical artifacts that interfere with the objectives of this thesis. For this reason, I decided to abstain from using the event and power electronic modeling capabilities that PowerFactory provides, and to develop an alternative short circuit model and PV system model, both of which do not interfere with the time step size.

Short Circuit Model The first model is meant to recreate a short circuit event. This short circuit model is based on three variable resistors, one per phase, connected between a node and ground. When a short circuit happens at said node, the resistors drop from a large R_{open} that represents an open circuit between the node and ground, to a small R_{closed} that represents a highly-conductive connection between the node and ground. For a short circuit that occurs at t_{SC} and clears at t_{CSC} the resistors are described by

$$R_{\text{SC}}(t) = \begin{cases} R_{\text{open}}, & t < t_{\text{SC}} \\ R_{\text{closed}}, & t_{\text{SC}} \leq t \leq t_{\text{CSC}} \\ (R_{\text{closed}} - R_{\text{open}})(1 - e^{(t_{\text{CSC}} - t)/\tau_{\text{CSC}}}) + R_{\text{open}}, & t > t_{\text{CSC}} \end{cases} \tag{6.1}$$

were τ_{CSC} is the time it takes to transition between R_{closed} and R_{open} while clearing the short circuit. Both that R_{closed} is not zero and that the short circuit is cleared with a smooth transition from R_{closed} to R_{open} are to avoid numerical divergence. One limitation of this approach is that the short circuit is not necessarily cleared when the current is zero. However, I consider this to be irrelevant for this thesis, since the objective of the short circuit model is to cause a disturbance and not to mimic the behavior of protection devices. For this model I chose $R_{\text{closed}} = 1\Omega$, $R_{\text{open}} = 100\text{k}\Omega$ and $\tau_{\text{CSC}} = 10\text{ms}$.

PV System Model The second model is meant to approximate a PV system that injects current harmonics. The PV systems is modeled as a current source that for each phase p injects a current

$$i_{\text{PV}_p}(t) = \mathcal{b}(t) \sum_{n \in \mathbf{N}} I_n(t) \cos\left(2\pi f_n t + \phi_{\text{PV}_p}\right), \tag{6.2}$$

where $\mathbf{N}$ is the set of harmonics and $\mathcal{b}(t)$ is a boolean function that represents the disconnection behavior of the PV system during voltage sags. When the RMS voltage at the point of common coupling drops below the disconnection threshold, $\mathcal{b}(t)$ changes its value from 1 to 0, and when the voltage recovers and crosses the reconnection threshold, $\mathcal{b}(t)$ changes from 0 to 1 after a preset delay. Since both disconnection and reconnection depend on threshold crossings, they are internal events.

6.4.2. Rollback Limitation

It was clear from the onset that PowerFactory was inappropriate to implement time rollback. In fact, I am not aware of any off-the-shelf power systems simulation tool that supports this operation. To roll a simulator back it must be possible to restart the simulator at an arbitrary point in time, with an arbitrary set of model states valid for that point in time. But PowerFactory does not provide direct access to model states. The states can only be set automatically by a power flow calculation, which means the simulator assumes a steady state as a starting point. Nevertheless, if a co-simulation method requires rollback, it is still possible to replicate its results with the following steps.

1. Run the co-simulation until the need for rolling back arises.
2. Record this time as an external event.
3. Stop the co-simulation and go back to point 1.

Once all the rollback events are identified and recorded, it is possible to run the co-simulation until the end and obtain the same results that would be obtained if rollback were possible. The drawback is that following this method the execution time of the co-simulation is no longer a meaningful result.

Bibliography

[1] C. D. López and J. L. Rueda-Torres, "Python scripting for DIgSILENT PowerFactory: Leveraging the Python API for scenario manipulation and analysis of large datasets", in *Advanced Smart Grid Functionalities Based on PowerFactory*, F. M. Gonzalez-Longatt and and J. L. Rueda-Torres, Eds., Springer, 2018, pp. 19–48.

[2] C. D. López, A. A. van der Meer, M. Cvetković, and P. Palensky, "A variable-rate co-simulation environment for the dynamic analysis of multi-area power systems", in *2017 IEEE Manchester PowerTech*, Jun. 2017, pp. 1–6.

[3] C. D. López, M. Cvetković, and P. Palensky, "Distributed co-simulation for collaborative analysis of power system dynamic behavior", in *Proceedings of the MEDPOWER 2018 Conference*, Nov. 2018, pp. 1–5.

[4] ——, "Enhancing PowerFactory dynamic models with Python for rapid prototyping", in *Proceedings of the 28th IEEE International Symposium on Industrial Electronics*, Jun. 2019, pp. 1–7.

[5] iMatix. (2007–Present). ØMQ, [Online]. Available: zeromq.org.

7

Selective Decoupling of AC Systems during Slow Transients using Preventive Recoupling and Trajectory Model Exchange

Parts of this chapter have been published in [1].

In this chapter I test the methods I proposed in Chapter 5, using the co-simulation framework I described in Chapter 6. For this, I use a test system composed of a transmission and a distribution grid. This test system considers the mechanical aspects of the generators, and includes distributed generation at the distribution level.

7.1. Testing Approach

To test the methods from Chapter 5 I will use the same approach to measure accuracy that I used in Chapter 4. However, in contrast to Chapter 4, and in line with my research questions, I will not measure execution time. Instead, I will only focus on the percentage the co-simulation remains decoupled. There are two main reasons for this. First, and most importantly, the co-simulation framework I described in Chapter 6 does not support rollback, so the execution time will not reflect the penalty for rolling back a simulator. Second, execution time is implementation dependent. In the implementation I described in Chapter 6, I did not pay particular attention to, for example, optimizing the Taylor-Kalman filters, I preferred the Python language over C++, and did not implement filter parallelization. For these reasons I consider that the percentage the co-simulation remains decoupled is a better measure of the potential benefit a selectively-decoupled co-simulation can provide.

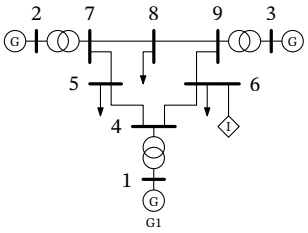
7.2. Test System

A good test system must be suitable for testing the trajectory model, so it must produce interface variables that are

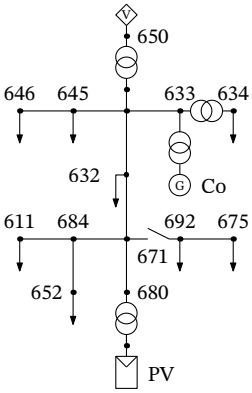
- three phase,
- unbalanced,
- have variable amplitude and phase, and
- contain harmonics.

The system must also be suitable for testing how the co-simulation transitions between coupled and decoupled modes, so it should experience both internal and external events. With these requirements in mind I chose the test system from Figure 7.1, which is composed of two subsystems, one transmission and one distribution grid. These grids are slightly modified versions of the grids from [2]. The interface between both subsystems is located at node 650 in the distribution grid and at node 6 in the transmission grid. During co-simulation, the controlled voltage source in Figure 7.1b represents the transmission grid in the distribution grid, while the controlled current source in Figure 7.1a represents the distribution grid in the transmission grid.

The test system has a nominal frequency of 60Hz. At the distribution level, it has a 3.5MW synchronous generator that represents a co-generation unit, whose role is to cause oscillations during disturbances. It also includes a 3.6MW PV system, based on the model from Section 6.4.1, whose role is to inject current harmonics and to produce internal events. The generators at the transmission level cause oscillations as well, but of smaller magnitude in comparison to the co-generation unit, given their much higher inertia.



(a) 9 bus transmission system



(b) 13 bus distribution feeder

Figure 7.1: One-line diagrams of the two subsystems that compose the test system. The subsystems exchange the interface variables v_a, v_b, v_c, which are the phase voltages at node 6 in the transmission system, and i_a, i_b, i_c, which are the phase currents at node 650 in the distribution feeder.

7.3. Case 6: No Harmonic Infiltration

In this case I compare a selectively-decoupled co-simulation to a monolithic simulation and a traditional co-simulation of the test system. For the selectively-decoupled co-simulation I use Algorithm 3.1. This algorithm relies exclusively on simulator rollback for recoupling, so I use the workaround from Section 6.4.2.

The (co-)simulated scenario includes a short circuit event at $t = 0.1$s that clears at $t = 0.2$s. These are external events. When the short circuit causes the voltage at the point of common coupling to drop bellow 0.8p.u., the PV system disconnect from the grid. Once the voltage recovers above 0.6p.u., the PV system reconnects with a delay of 0.5s. Both disconnection and reconnection are internal events because the times when the voltage at the point of common coupling crosses the given thresholds is unknown.

For the co-simulations I used a macro time step $H = 0.1$ms and a micro time step $h = 10\mu$s, an acquisition window size $T_w = 1/60\text{Hz} = 16.7$ms, a window hop size $R_w = 10$ samples, and a predictability threshold $\epsilon_p = 0.05$p.u. Based on the values proposed in [3], I chose $\sigma_v^2 = 1 \times 10^{-6}$, $\sigma_w^2 = 1 \times 10^{-6}$, $M = 2$, and $\tau = 1$ms as Taylor-Kalman filter parameters.

Figure 7.4 compares a subset of the results obtained with a selectively decoupled co-simulation to those obtained with a monolithic simulation of the test system. The results are one phase of the interface voltage and current, and the speeds of generator G1 and the co-generation unit. In this case, the co-simulation remains decoupled 34% of the time. Although this is far from negligible, the co-simulation never remains decoupled for long. In comparison to the steady state case, the trajectory models for slow transients have a much shorter lifespan. This means that several rollbacks are necessary, which is highly undesirable.

Nevertheless, the results appear to be accurate. Figure 7.3 compares the error of the variables from Figure 7.4 to the errors obtained from a traditional co-simulation. The error in the selectively decoupled co-simulation is higher, albeit not considerably,

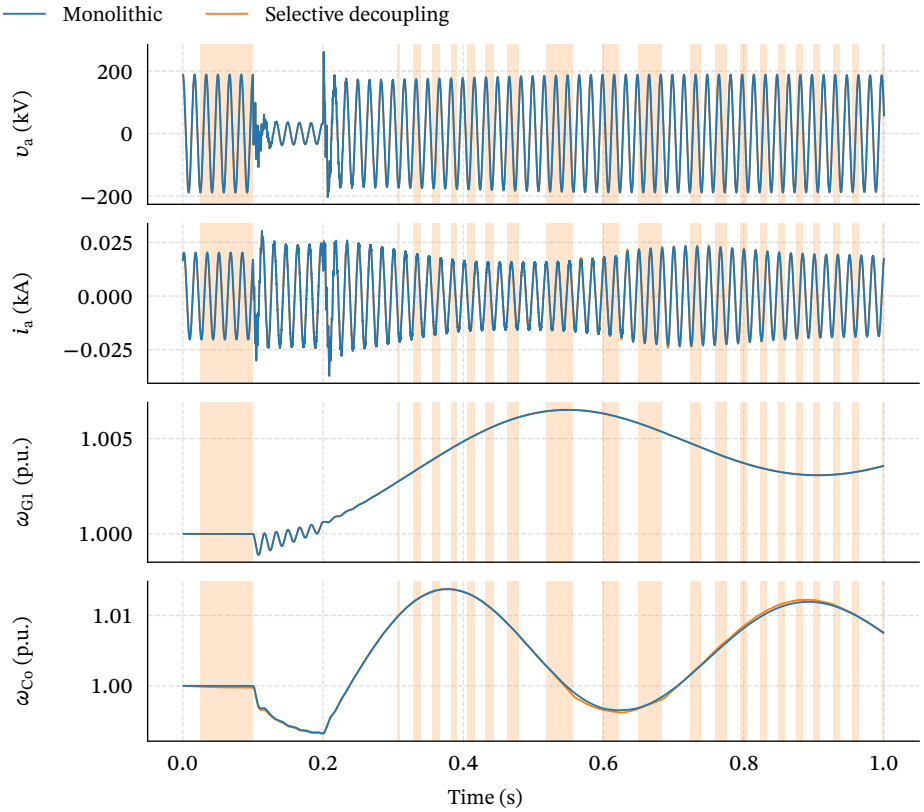


Figure 7.2: Phase a of the interface voltage and current, speed of generator G1 (transmission) and the co-generation unit (distribution), computed with a monolithic simulation and the selectively-decoupled co-simulation from Case 6. The colored background indicates when the co-simulation is in decoupled mode. The selectively-decoupled co-simulation remains decoupled 34% of the time.

and both high and low frequency phenomena are reproduced successfully. Additionally, the selectively decoupled co-simulation is able to transition between modes rather seamlessly. The PV disconnection event happens while the co-simulation is in coupled mode, so it does not need to be detected, but the reconnection event happens while the co-simulation is decoupled, and is successfully detected.

7.4. Case 7: Harmonic Infiltration

In this case I modified the (co-)simulated scenario from Case 6 so the PV system injects a 5^{th} current harmonic with a constant amplitude 5% of the fundamental, and constant phase. Figure 7.4 shows that using a $\boldsymbol{t}_{\text{w}} = 1/60\text{Hz}$ (one fundamental period) limits the possibilities to transition to decoupled mode; the trajectory model is not able to describe both the fundamental and the 5^{th} harmonic for such a long time window, which results in a co-simulation that remains decoupled only 15% of the time.

One way to understand why this is, is to consider that the trajectory model is in

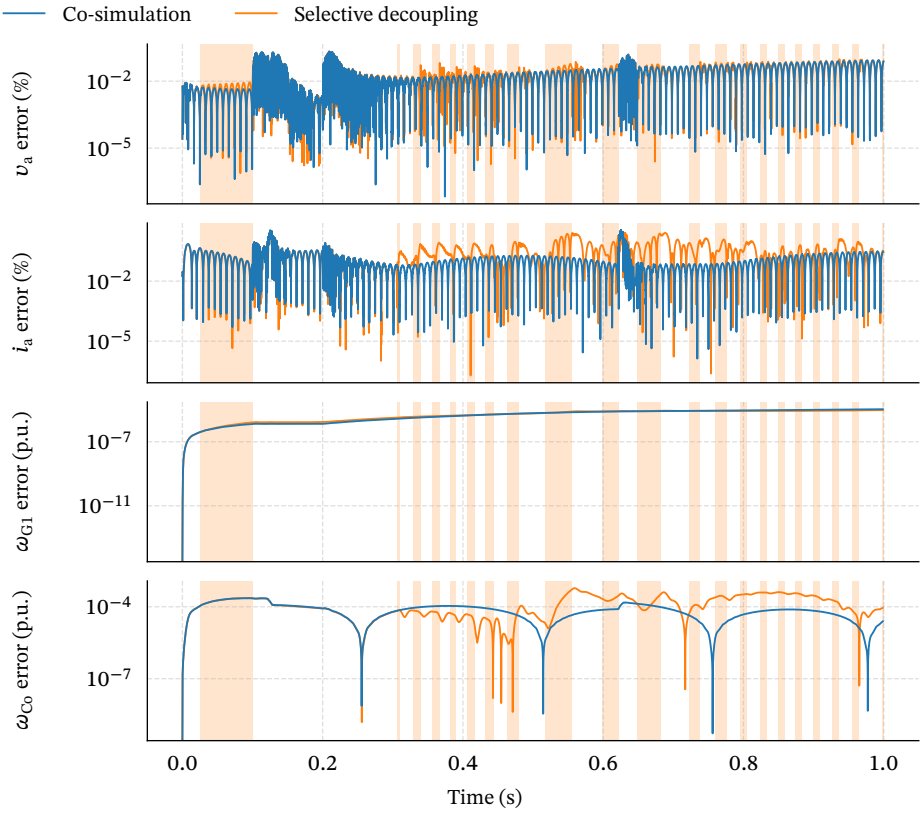


Figure 7.3: Errors of phase a of the interface voltage and current, speed of generator G1 (transmission) and the co-generation unit (distribution), computed with the selectively-decoupled co-simulation from Case 6. The errors are measured with respect to the monolithic simulation and are in percent of the dynamic range of the corresponding state variable. The colored background indicates when the co-simulation is in decoupled mode.

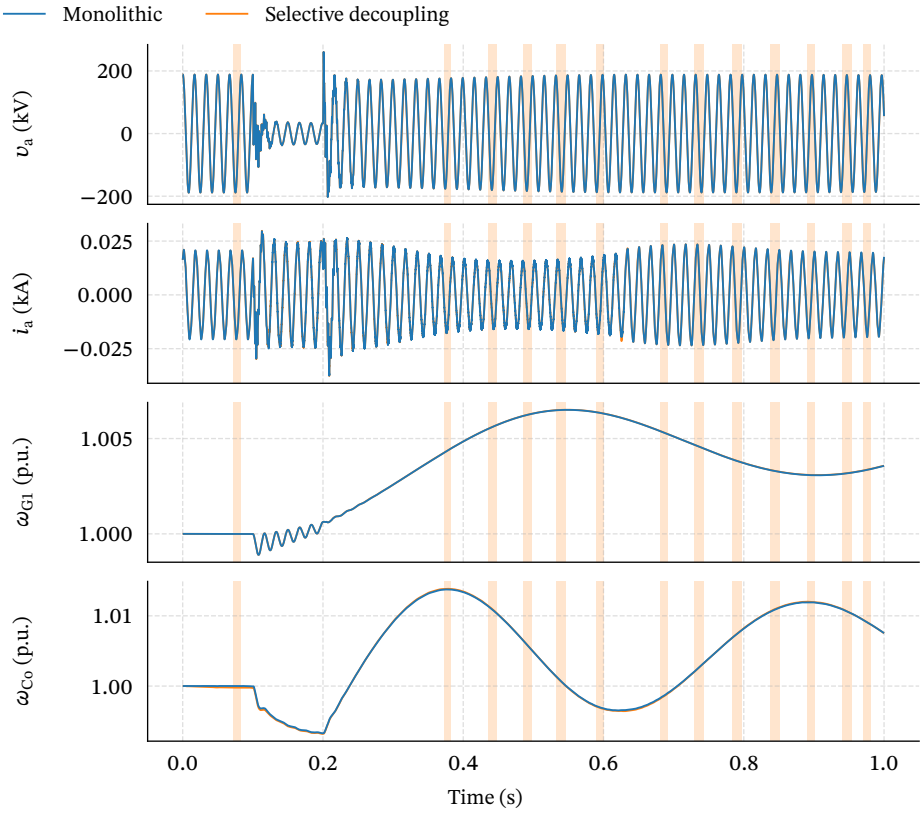


Figure 7.4: Phase a of the interface voltage and current, speed of generator G1 (transmission) and the co-generation unit (distribution), computed with a monolithic simulation and the selectively-decoupled co-simulation from Case 7, with $t_w = 1/60\text{Hz}$. The PV system injects 5% 5th current harmonic. The colored background indicates when the co-simulation is in decoupled mode. The selectively-decoupled co-simulation remains decoupled 15% of the time.

fact a superposition of two trajectory models, one for the fundamental and one for the 5th harmonic. Since both of these are based on a Taylor series, each can only remain accurate for a given number of periods before diverging. However, one period of the 5th harmonic is much smaller than that of the fundamental, so the trajectory model for this harmonic diverges much sooner.

By reducing the acquisition window size to $t_w = 1/(300\text{Hz})$ (one 5th harmonic period), the situation improves, as Figure 7.5 shows. In this case the co-simulation stays decoupled 29% of the time, but it never stays decoupled for long, and rollbacks become extremely frequent. Adding higher frequency harmonics would naturally worsen the situation.

One important difference with the steady state case from Chapter 4 is that the Taylor-Kalman filter does not detect which harmonics are present in the interface variable. However, this could be automated using the same method from Chapter 4 as a pre-modeling stage, although at a higher computational cost.

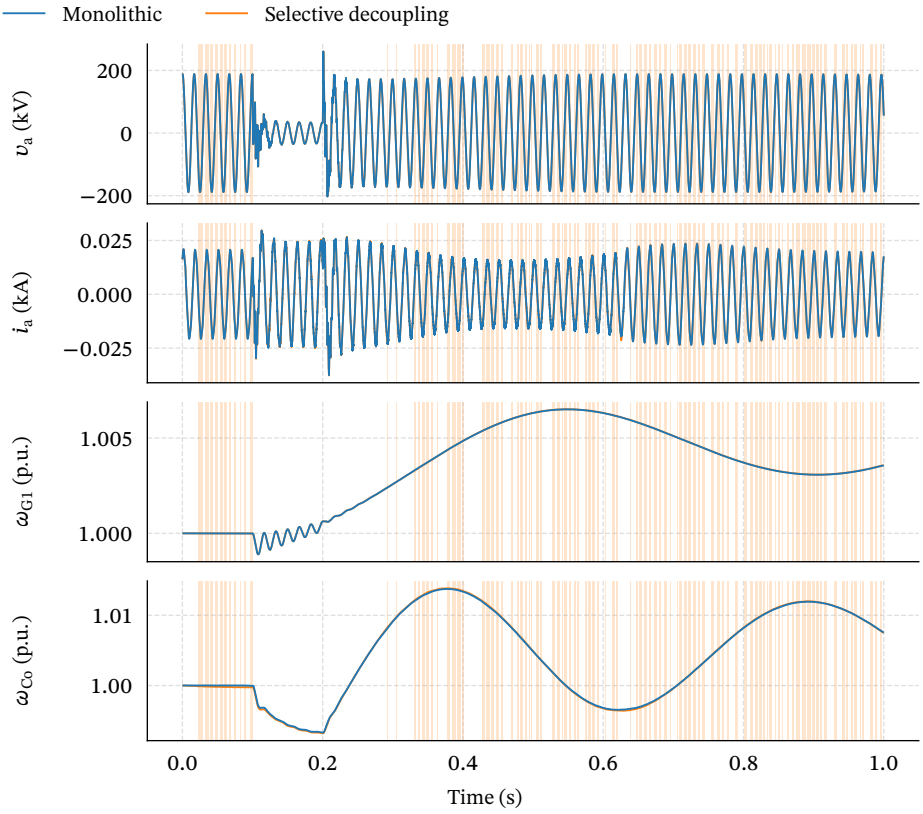


Figure 7.5: Phase a of the interface voltage and current, speed of generator G1 (transmission) and the co-generation unit (distribution), computed with a monolithic simulation and the selectively-decoupled co-simulation from Case 7, with $t_w = 1/300\text{Hz}$. The PV system injects 5% 5th current harmonic. The colored background indicates when the co-simulation is in decoupled mode. The selectively-decoupled co-simulation remains decoupled 29% of the time.

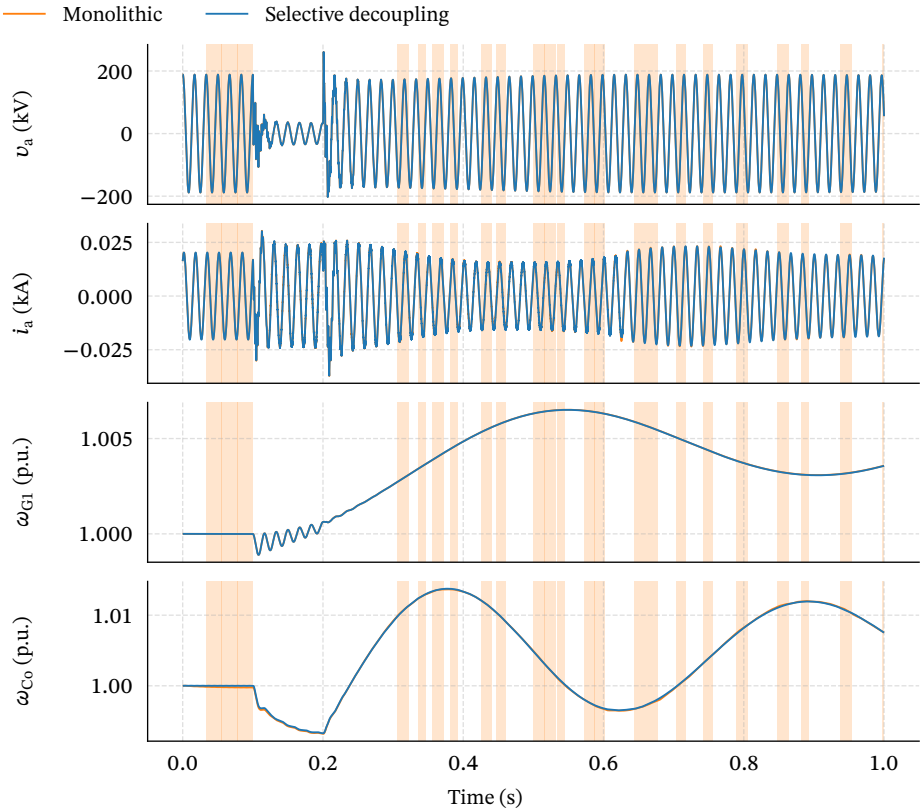


Figure 7.6: Phase a of the interface voltage and current, speed of generator G1 (transmission) and the co-generation unit (distribution), computed with a monolithic simulation and the selectively-decoupled co-simulation from Case 8. The colored background indicates when the co-simulation is in decoupled mode. The selectively-decoupled co-simulation remains decoupled 35% of the time.

7.5. Case 8: Preventive Recoupling and Trajectory Model Exchange

In this case I used the (co-)simulated scenario from Case 6, but co-simulated it with Algorithm 5.2. The only change required here is increasing the order of the Taylor-Kalman filter to $M = 3$. Figure 7.6 shows the results obtained with this algorithm. In this case the co-simulation remains decoupled 35% of the time and the error remains low, as Figure 7.7 shows. This is in line with the results from Case 6, which indicates that the recoupling criterion is appropriate. In this co-simulation, only one rollback is necessary, and it is caused by the internal event which cannot be predicted by (5.43).

In an attempt to increase the percentage the co-simulation remains decoupled, I halved the difference between the decoupling time and the predicted recoupling time, so that the trajectory models deviate less from the true trajectories and the co-simulation does not need to stay coupled for long before being able to decouple again. Figure 7.8 shows these results. Based on the co-simulation mode shown in the background the fig-

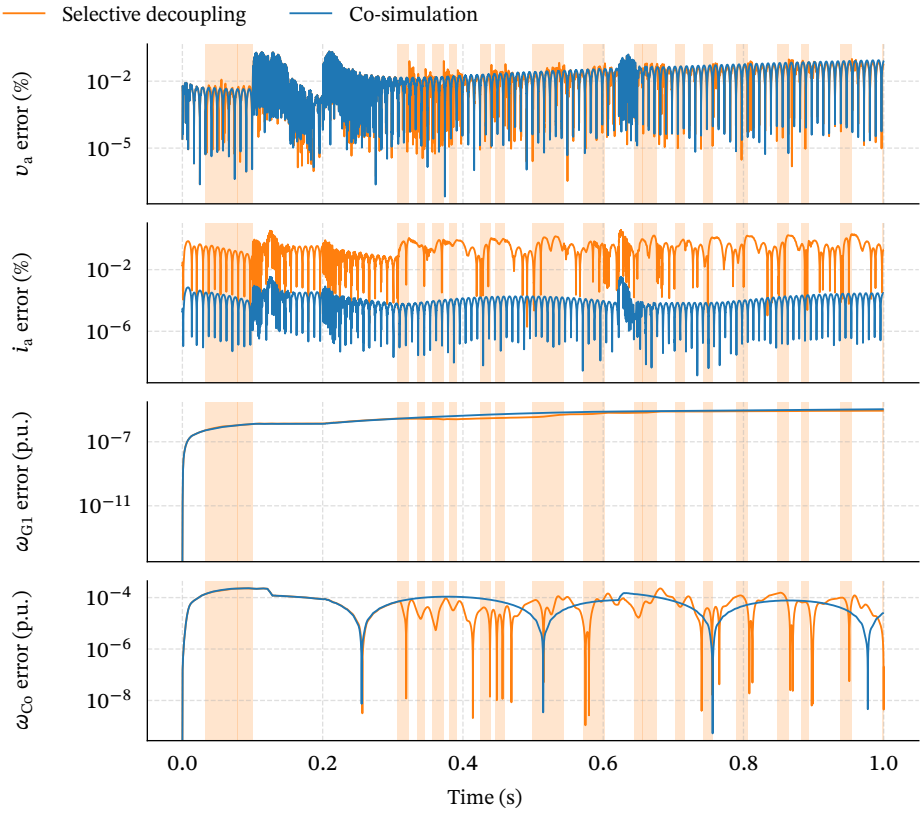


Figure 7.7: Errors of phase a of the interface voltage and current, speed of generator G1 (transmission) and the co-generation unit (distribution), computed with the selectively-decoupled co-simulation from Case 8. The errors are measured with respect to the monolithic simulation and are in percent of the dynamic range of the corresponding state variable. The colored background indicates when the co-simulation is in decoupled mode.

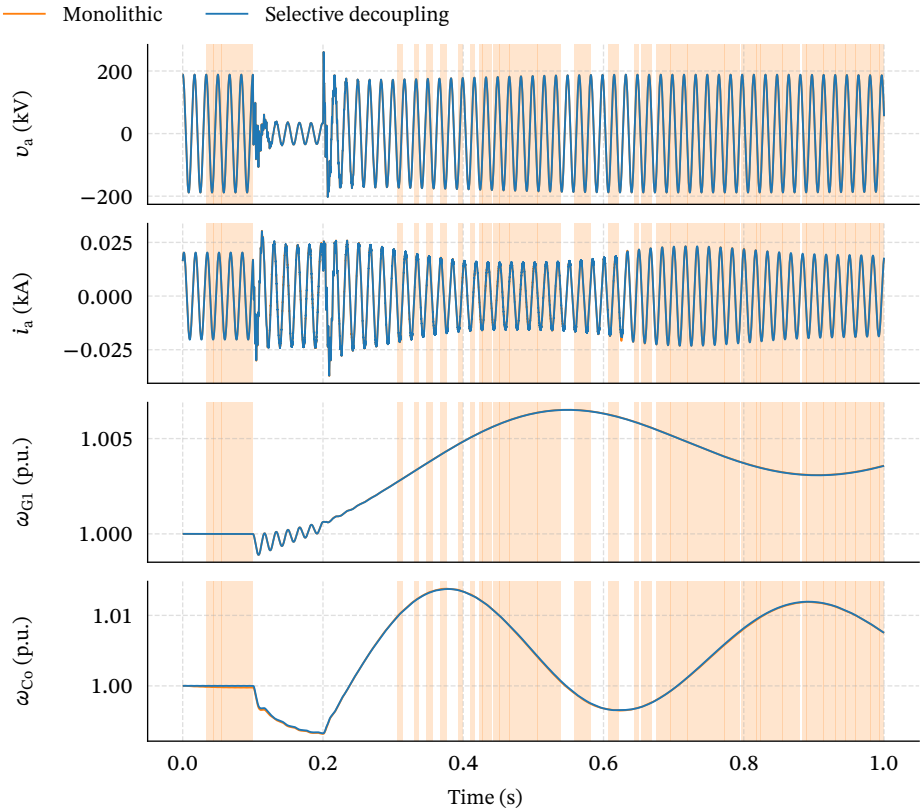


Figure 7.8: Phase a of the interface voltage and current, speed of generator G1 (transmission) and the co-generation unit (distribution), computed with a monolithic simulation and the selectively-decoupled co-simulation from Case 8. The colored background indicates when the co-simulation is in decoupled mode. The selectively-decoupled co-simulation remains decoupled 60% of the time.

ure, it would appear that the trajectory model is able to describe the interface variables for longer periods of time than in previous cases. However, what is actually happening is that the simulators are exchanging trajectory models and immediately returning to decoupled mode. In this case the proportion the co-simulation remains decoupled increases to 60%. In addition, Figure 7.9 shows that the error remains comparable to previous cases.

7.6. Discussion

With the trajectory model and identification method I proposed in Chapter 5, the results are accurate, even in the presence of harmonics. However, the co-simulation must roll back rather frequently, and even more so when harmonics are present. This is a highly undesirable situation. Nevertheless, applying the criterion for preventive recoupling, the co-simulation required only one rollback, caused by an internal event. With sufficient knowledge about the co-simulated system, it could be possible to roughly estimate

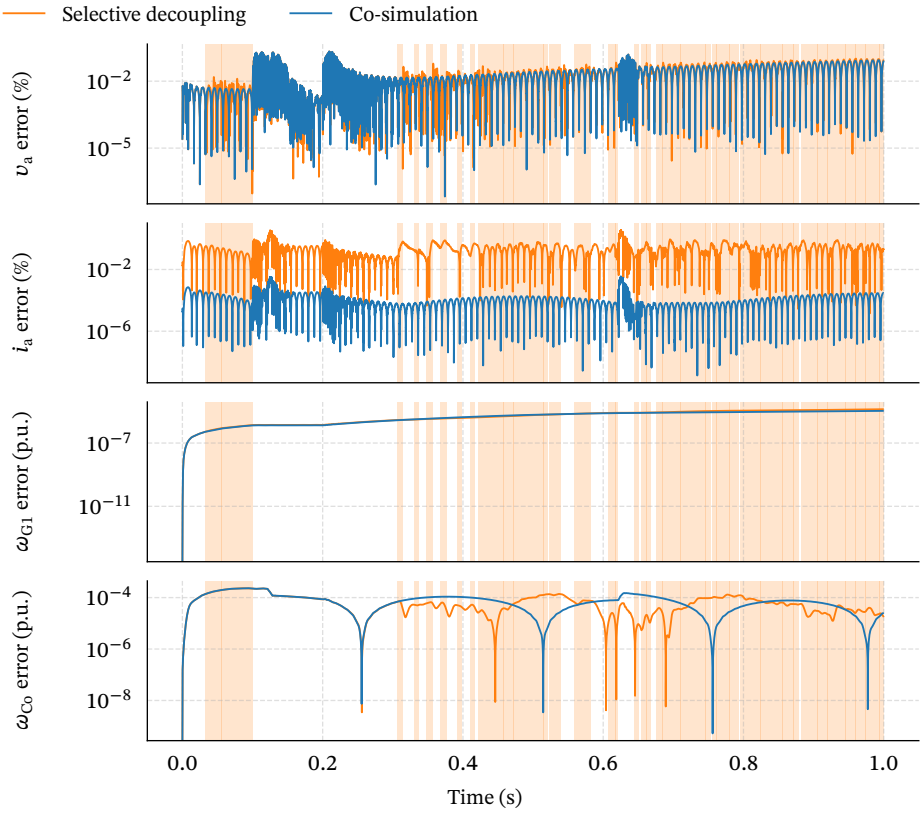


Figure 7.9: Errors of phase a of the interface voltage and current, speed of generator G1 (transmission) and the co-generation unit (distribution), computed with the selectively-decoupled co-simulation from Case 8. The errors are measured with respect to the monolithic simulation and are in percent of the dynamic range of the corresponding state variable. The colored background indicates when the co-simulation is in decoupled mode.

when an internal event might occur, and completely eliminate the need for rollbacks. Nevertheless, this would not be a general solution. Although the criterion for preventive recoupling produced good results, it is not optimal. The results suggest that finding a better criterion is worthwhile, as this could substantially reduce the need for communication. Finally, and at least for this test system, there are no apparent disadvantages to exchanging trajectory models instead of interface variable scalars.

Bibliography

[1] C. D. López, A. A. van der Meer, M. Cvetković, and P. Palensky, "Reducing the need for communication in natural waveform co-simulations of electrical power systems using adaptive interfaces", Submitted to IEEE Access.

[2] C. D. López, M. Cvetković, and P. Palensky, "Distributed co-simulation for collaborative analysis of power system dynamic behavior", in *Proceedings of the MED-POWER 2018 Conference*, Nov. 2018, pp. 1–5.

[3] J. Khodaparast and M. Khederzadeh, "Least square and Kalman based methods for dynamic phasor estimation: A review", *Protection and Control of Modern Power Systems*, vol. 2, no. 1, pp. 1–18, Jan. 2017.

8

Conclusion

At the beginning of this thesis I set out to determine whether it is possible to reduce the need for communication in a remote electrical power system co-simulation between natural waveform simulators. This, by implementing a co-simulation framework that is able to distinguish between predictable and unpredictable phenomena, and that can find closed-form expressions that describe the trajectories of the interface variables while phenomena are predictable. The idea being that the simulators compute their own inputs from these expressions instead of expecting them to be communicated.

To achieve this goal, I proposed methods for classifying phenomena as predictable or unpredictable, for finding these closed-form expressions, and described how a co-simulation can take advantage of these methods. Additionally, I designed and implemented a co-simulation framework for experimentation. With this framework, a co-simulation operates in two modes, one for unpredictable interface variables, which behaves as a traditional co-simulation, and one for predictable interface variables. In the mode for predictable interface variables, the simulators exchange expressions that describe their outputs over time, then decouple from each other, and proceed by computing their own inputs from these expressions until the interface variables become unpredictable again. Having run a set of experiments with the framework, I can now answer the research questions that motivated this thesis, describe the limitations of my findings, their possible applications and implications, and suggest directions for further research.

8.1. Answers to Research Questions

In Chapter 1 I stated three research questions that I will now answer.

Is it possible to define a criterion so that a co-simulation framework can distinguish between predictable and unpredictable power system phenomena, based exclusively on interface variables?

This is possible with a precise definition of what constitutes predictable interface variables. With sufficient knowledge about the co-simulated system, this definition can be stated in terms of a trajectory model that can be fitted to the true trajectories the interface variables exhibit during co-simulation. Such a trajectory model expresses an expectation of what the interface variables must look like while the co-simulated phenomena are predictable, so the co-simulation framework can test whether the interface variables match this expectation to distinguish predictable from unpredictable interface variables. Naturally, depending on the chosen definition, the same phenomena could be classified as predictable or unpredictable.

When the co-simulation framework detects the absence of unpredictable phenomena, can this framework identify closed-form expressions that describe the trajectories followed by the interface variables, so that each simulator computes its inputs from these expressions instead of them being supplied by other simulators?

The answer to the previous question hints that this must be possible as well. I

worked with two definitions of predictable interface variables. In Chapter 4, I defined them as variables that can be described by a finite sum of sinusoids with constant amplitude, frequency and phase, which is the behavior expected from a system in steady state. Thus, fitting this trajectory model to the true trajectories means identifying three parameters: the amplitude, frequency and phase of each sinusoid. For this, an interpolated Fourier transform was effective.

In Chapter 5, I defined them as variables that can be described as a finite sum of sinusoids with variable amplitude, frequency and phase, which is the behavior expected from a system experiencing a slow (electromechanical) transient. The challenge here is that fitting such a trajectory model means identifying three functions of time, namely, the amplitude, frequency and phase. To circumvent this issue, I approximated the sum of sinusoids with a Taylor series, which transforms the function identification problem into a parameter identification problem. A Taylor-Kalman filters proved effective for fitting this trajectory model.

Implicit in the research question is that the co-simulation framework must be able to operate in two modes, one where the simulators exchange interface variables at every macro time step, and another where they compute their own inputs. I referred to these modes as coupled and decoupled. Transitioning from coupled to decoupled is the simplest of the two transitions; when the trajectory models accurately describe the true trajectories of all interface variables, the transition is allowed. However, the inverse transition is complicated by the fact that, when decoupled, each participating simulator runs at a different pace in a non-real time environment, so recoupling might require simulator rollback. This is undesirable because rolling a simulator back is time consuming, memory consuming, and because most off-the-shelf power system simulators do not support this feature. I was able to avoid rollbacks by preemptively recoupling simulators. However, this approach does not solve the problem when recoupling is needed because an internal event has occurred.

How accurate are these expressions and what proportion of the co-simulation are they able to describe?

Regarding the accuracy, it is difficult to distinguish the difference between the results obtained with these expressions and those obtained with a traditional co-simulation. Differences do exist, but considering that comparable differences usually appear when simulating a model with two different simulators, or when comparing simulated results to real phenomena, I deem the results to be accurate enough for electrical power system simulation. Having said that, the results I obtained in this thesis cannot guarantee that the methods are accurate for every system. Co-simulations do pose numerical challenges, and having simulators find expressions they can use to compute their inputs should have consequences I was not able to observe in my experiments.

Regarding the proportion of the co-simulation the expressions are able to describe, this depends on the specific simulated scenario. I studied cases where a short circuit produced fast and slow transients, and was able to reduce the need for communication up to 60%. However, if the cases had had more events, the proportion would have been lower.

8.2. Applications

Although this thesis focused on AC systems, the main idea about a co-simulation that decouples the simulators while the interface variables are predictable can be applied to other systems, as long as suitable trajectory models can be identified. This means that there is a possibility for these methods to also be applicable to the case of DC systems. It is also possible that it can be applied in other industries, for example automotive and aerospace.

The application I had in mind at the beginning of this work was remote collaborative co-simulation between different organizations. The idea being that, in remote co-simulation, each organization can continue using its simulator of choice and avoid exchanging models, thus circumventing data privacy limitations. Organizations such as neighboring grid operators could find this approach beneficial. Additionally, this could also open up more possibilities for collaboration between researchers and industry.

8.3. Implications

The motivation for reducing the need for communication in a remote co-simulation is that the delay associated with communication over long distances could add significant overhead to the co-simulation, thus slowing it down. Whether the methods I proposed lead to speedup actually depends on the magnitude of the method overhead and the communication delay. If the method overhead is small (i.e., the overhead related to identifying and evaluating trajectory models) and the communication delay is large, speedup should be possible. An advantage of the approach I proposed is that a part of the co-simulation overhead is transfered from the communication channel, which is not under the control of the co-simulation developer, to the co-simulation interfaces, which are. So making appropriate implementation decisions, the co-simulation developer could minimize the method overhead and maximize the speedup.

Our ability to take full advantage of the existing energy infrastructure, and to develop it further economically, reliably and sustainably, is in part constrained by the availability of sufficiently descriptive models and high-performing computations. A remote co-simulation that is less affected by the communication channel is a step forward in this direction. With such a tool, different organizations can collaboratively create better models, and gather deeper insight into the systems they wish to improve.

8.4. Suggestions for Further Research

The work I presented has several limitations, some of which I consider more pressing to address. I consider that the biggest disadvantage of the method I proposed is that it relies on simulator rollback. Although the criterion for preventive recoupling does produce good results, it is not optimal. Furthermore, it does not address the case of rollbacks caused by internal events. Another aspect that deserves more attention is the consequences of using these trajectory models to compute inputs. This approach will likely have a bigger impact on some systems that others, and it would be beneficial to quantify it. Finally, there is the application to HVDC systems. These systems are becoming ubiquitous, so for this method to be truly useful to transmission grid operators, it is imperative to find expressions that predict the trajectories of DC variables as well.

Acknowledgements

Perusing a doctoral education has been a privilege for which I am most grateful. But as rewarding as it has been, it has been challenging. For helping me reach this milestone I owe my gratitude to many people. To Peter for this very opportunity, for his trust, for giving me the freedom to explore my own ideas, and for supporting me beyond what is expected of a promotor. To Milos for his honesty, patience and time. To José for his advice. To my colleagues Arjen, Ilya, Matija, Mario, Swasti, Lian, Hossein, Romain, Arun, Shahab, Arcadio, Kaikai, Rishabh, and Digvijay for the discussions, collaborations, encouragement and friendship. To Sharmila, Ellen and Carla for removing obstacles. To my father for fostering my curiosity. To my mother for explaining to me that C pointers are, in fact, simple. To Mario and Gisela for their guidance and help. To Orsika for her support, faith and seemingly endless patience. To all of you a owe a debt of gratitude the extent of which cannot be conveyed on this page.

Munich, March 2021

List of Publications

Journal Papers

[A] C. D. López, A. A. van der Meer, M. Cvetković, and P. Palensky, "Reducing the need for communication in natural waveform co-simulations of electrical power systems using adaptive interfaces", Submitted to IEEE Access.

[B] C. D. López, M. Cvetković, and P. Palensky, "Speeding up AC circuit co-simulations through selective simulator decoupling of predictable states", *IEEE Access*, vol. 7, pp. 1–14, Apr. 2019.

[C] P. Palensky, A. A. van der Meer, C. D. López, A. Joseph, and K. Pan, "Applied cosimulation of intelligent power systems: Implementing hybrid simulators for complex power systems", *IEEE Industrial Electronics Magazine*, vol. 11, no. 2, pp. 6–21, Jun. 2017.

[D] ——, "Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling", *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 34–50, Mar. 2017.

Book Chapters

[E] A. Shetgaonkar, A. Perilla, I. Tyuryukanov, D. Gusain, C. D. López, and J. L. Rueda-Torres, "Applications of PowerFactory for the study of basic notions of power system dynamics in graduate courses", in *Modelling and Simulation of Power Electronic Converter Dominated Power Systems in PowerFactory*, F. M. Gonzalez-Longatt and and J. L. Rueda-Torres, Eds., Springer, 2020, pp. 337–377.

[F] C. D. López, M. Cvetković, A. A. van der Meer, and P. Palensky, "Co-simulation of intelligent power systems", in *Intelligent Integrated Energy Systems: The PowerWeb Program at TU Delft*, P. Palensky, M. Cvetković, and T. Keviczky, Eds., Springer, 2019, pp. 99–119.

[G] C. D. López and J. L. Rueda-Torres, "Python scripting for DIgSILENT PowerFactory: Leveraging the Python API for scenario manipulation and analysis of large datasets", in *Advanced Smart Grid Functionalities Based on PowerFactory*, F. M. Gonzalez-Longatt and and J. L. Rueda-Torres, Eds., Springer, 2018, pp. 19–48.

Conference Papers

[H] C. D. López, M. Cvetković, and P. Palensky, "Enhancing PowerFactory dynamic models with Python for rapid prototyping", in *Proceedings of the 28^{th} IEEE International Symposium on Industrial Electronics*, Jun. 2019, pp. 1–7.

[I] ——, “Distributed co-simulation for collaborative analysis of power system dynamic behavior”, in *Proceedings of the MEDPOWER 2018 Conference*, Nov. 2018, pp. 1–5.

[J] K. Pan, A. Teixeira, C. D. López, and P. Palensky, “Co-simulation for cyber security analysis: Data attacks against energy management system”, in *Proceedings of the 8th IEEE SmartGridComm*, Oct. 2017, pp. 1–6.

[K] M. Cvetković, K. Pan, C. D. López, R. Bhandia, and P. Palensky, “Co-simulation aspects for energy systems with high penetration of distributed energy resources”, in *Proceedings of the AEIT International Annual Conference*, Sep. 2017, pp. 1–6.

[L] M. Cvetković, H. Krishnappa, C. D. López, R. Bhandia, J. L. Rueda-Torres, and P. Palensky, “Co-simulation and dynamic model exchange with consideration for wind projects”, in *Proceedings of the 16^{th} Wind Integration Workshop*, Sep. 2017, pp. 1–6.

[M] C. D. López, A. A. van der Meer, M. Cvetković, and P. Palensky, “A variable-rate co-simulation environment for the dynamic analysis of multi-area power systems”, in *2017 IEEE Manchester PowerTech*, Jun. 2017, pp. 1–6.