Unified Modeling for Emulating Electric Energy Systems: Toward Digital Twin that Might Work

Marija D. Ilić [*], Rupamathi Jaddivada [†] and Assefaw H. Gebremedhin [‡]

August 31, 2020

Abstract

Large-scale computing, including Machine Learning and Artificial Intelligence, offer a great promise in enabling sustainability and resiliency of electric energy systems. At present, however there is no standardized framework for systematic modeling and simulation of system response over time to different continuous- and discrete-time events and/or changes in equipment status. As a result, there is generally a poor understanding of the effects of candidate technologies on the quality and cost of electric energy services. In this chapter, we discuss a unified, physically-intuitive multi-layered modeling of system components and their mutual dynamic interactions. The fundamental concept underlying this modeling is the notion of interaction variables whose definition directly lends itself to capturing modular structure needed to manage complexity. As a direct result, the same modeling approach defines an information exchange structure between different system layers, and hence can be used to establish structure for the design of a dedicated computational architecture. Such a structure would not only avoid data over-saturation and flow bottlenecks in a computational platform, but it will also enable an efficient and orderly data and control flow. We address numerical integration issues arising when simulating distributed components and their information exchange with the rest of the system. We conjecture that Algorithmic Differentiation (also known as Automatic Differentiation) can be used to overcome fundamental numerical integration issues that are a roadblock on the way to realizing high performance digital twins for electric energy systems.

1 Introduction

This chapter is motivated by a central recognition: electric energy systems could have much better performance than what is achieved today if they are enabled by rich data (appropriately collected and processed) and on-line decision-making

[*]Massachusetts Institute of Technology. email: ilic at mit.edu
[†]Massachusetts Institute of Technology. email: rjaddiva at mit.edu
[‡]Washington State University. e-mail: assefaw.gebremedhin at wsu.edu

based on learning from history and predictions. Taking this recognition a step further, we argue that novel approaches are needed for next generation modeling and computing for managing complex on-going major organizational and technological changes. Clearly, this should not be done from scratch, since there have been major investments in current information technology by entities such as the Pennsylvania-Jersey-Maryland (PJM) independent system operator (ISO), estimated at $800 million (Boston, 2020). This makes it even harder to move forward with the next generation Supervisory Control and Data Acquisition (SCADA) system (Ilić, 2010). An additional subtle challenge comes from the need to almost instantaneously balance supply and demand. This means supporting control and decision making for "guaranteed" Quality of Service (QoS), which is in sharp contrast with Internet protocols where targeting "best efforts" is sufficient. Imagine having to operate a system like Puerto Rico's during a hurricane or an earthquake, when lots of electric system equipment is damaged and disconnected. It is crucial to have a flexible digital twin for simulating such complex systems, both in support of autonomous self-adaptation to the changing conditions and for assessing impact of previously unused technologies and system disturbances, such as intermittent power. This chapter is written with such challenges and opportunities in mind.

1.1 Background and history

Computing and simulations in electric power systems enjoy a long history. Generally speaking however, the approach has been rather piece-meal software development for basic applications and computational tasks, such as state estimation (Clements et al., 1981), power flow analysis in a large grid (Stott, 1974), optimal power dispatch for minimizing generation fuel cost when supplying predicted demand (Stott et al., 2009), short circuit analysis (Zhang et al., 1995), and numerical simulation of system response to sudden large equipment failures (Pavella and Murthy, 1994). All of these applications and tasks are used in a feed-forward way with the human-in-the-loop, while relying on local embedded automated feedback to ensure stable and feasible dynamic response in near real-time.

In the early 1970's utilities worldwide begun to experience major wide-spread blackouts, and this called for more reliance on computing. These needs motivated active R&D on large-scale numerical methods, such as solving sparse matrix problems (Tinney et al., 1985; Rose, 2012), waveform relaxation for numerical integration of multi-rate differential equations (Ilic et al., 1987), small-signal stability analysis (Verghese et al., 1982) and, more generally, parallel computing for large-scale systems (Betancourt and Alvarado, 1986). In turn, these efforts resulted in computer applications used routinely by control centers for dispatching generation to supply predicted power demand (Fu and Shahidehpour, 2007; Stott, 1974; Cvijić et al., 2018). However, emulation of system dynamics has only been done in an off-line mode for select deterministic scenarios because centralized numerical integration of high-order differential-algebraic equations (DAEs) can not be done in near real-time (Pavella and Murthy, 1994; Crow and

Ilić, 1994). Only a handful of forward-looking utilities have designed a hybrid analog-digital simulators of their own power systems for the purposes of better understanding their performance, particularly during abnormal conditions, and also for simulating potential of system control and protection (Do et al., 2001; Doi et al., 1990).

1.2 Preview of main contributions

The electric energy ecosystem of our time has a combination of established as well as emerging needs, opportunities and challenges. The long-standing problems of using computing to provide uninterrupted service during extreme events and prevent wide-spread blackouts remains a key need. The root causes of these events have, however, become more complex. In addition to the usual triggers such as transmission lines touching trees and creating cascading outages (Thorp et al., 1998), the causes could be more severe and due to natural disasters causing large number of equipment failures or cyber-security breaches causing malfunctioning of the equipment. Furthermore, there has also been a major change in the type of energy resources, their locations and size (Ilic et al., 2020) and in the nature of loads. Regulatory rules encourage non-utility-owned edge-grid users to supply their own power, participate in electric power trading, etc. The grid itself could control its parameters using fast controllers (Padiyar, 2007). The confluence of all of this unravels a once monolithic system into different subsystems, tightly interacting because of the electrical connections.

Currently, there are no user-friendly computational platforms for emulating dynamics of these systems in response to these various drivers. Because of this, it is hard to understand potential problems and assess candidate solutions. While it is clear that a system would benefit from participation of distributed edge-grid users, there are no standardized modeling and computing frameworks for supporting self-adapting, distributed, and minimally-coordinated electricity services.

The ideas described in this chapter are an attempt to begin to fill this void. In Section 2 a multi-layered energy-based framework for modeling electric energy systems is briefly described for completeness [21, 22]. The fundamental concept underlying this modeling is the one of interaction variables whose definition directly lends itself to capturing the structure needed to manage complexity. In Section 3 we discuss far-reaching potential of utilizing this modeling framework as the basis for the design of computation architecture. We summarize how the use of interaction variables forms the basis of Dynamic Monitoring and Decision Systems (DyMonDS) framework which conceptualizes the problem as a structured cyber-physical systems problem (Ilić, 2010). Our research has evolved over the past two decades around DyMonDS, including the development of our home-grown computational platform and its use as the basic research tool. We describe more recent attributes of what became Scalable Electric Power System Simulator SEPSS (Ilić et al., 2018).

The idea of using this modeling framework for adaptive computing resource allocation using automata has already been recognized and it awaits further

development (Ilic and Jaddivada, 2019a). The new idea introduced in Section 4 concerns plug-and-play automated method for power systems simulations (PAMPS). This is an inherently multi-layered distributed computing approach which presents unique numerical integration challenges because of its interactive nature and the need to compute derivatives of interaction variables accurately and efficiently. While the algorithm finds its use in both distributed control and simulations, in this chapter we focus on the numerical simulations.

In Section 5 we report on our up-to-date experience with numerical integration methods in support of PAMPS. We stress that state-of-the-art in model-based distributed simulations of fast dynamics presents us with many issues, and illustrate these challenges on two typical engineering examples. In Section 6 , we establish the basis for major potential benefits to be drawn from more advanced computational methods, such as algorithmic differentiation (AD) (Griewank and Walther, 2008; Baydin et al., 2017). These methods are briefly summarized and hold the promise of overcoming some key issues in numerical integration of very fast transient processes. We close in Section 7 with the ideas for next steps and open questions on the way to establishing digital twins of electric power systems.

2 Unified multi-layered modeling in energy-power dynamical state space

In this section we briefly summarize a recently-introduced unified modeling of electric energy systems as the basis for representing the internal technology-specific local dynamics of core components and their distributed control in terms of their local state variables; and the interaction variables with the rest of the system. Figure 1 reflects the fact that electric energy systems are fundamentally social-ecological systems (SES) whose sustainability can be assessed by understanding core attributes of their resources, users, governance subsystems and their dynamical interactions; determining these core attributes is the foundation for Elinor Ostrom's Nobel prize winning generalized sustainability framework (Ostrom, 2009). Inspired by this work, and based on our modeling of power systems as complex dynamical systems, we introduce multi-layered interactive modeling. At the higher system levels, dynamic interactions between core components shown in Figure 1 are represented using their aggregate variables. Based on this, we study a multi-layered multi-granular modeling of a complex electric energy systems, as shown in Figure 2. Resources and user subsystems are modules, and they interact by means of physical grid and its data-enabled monitoring and control of interactions, and are governed by the industry rules, as shown later in Figure 1.

Next we summarize for completeness this unified modeling approach and its potential to characterize the input-output dynamical specifications of the core components comprising a social-ecological energy system. The approach is fundamentally based on mapping very complex internal component dynamics into

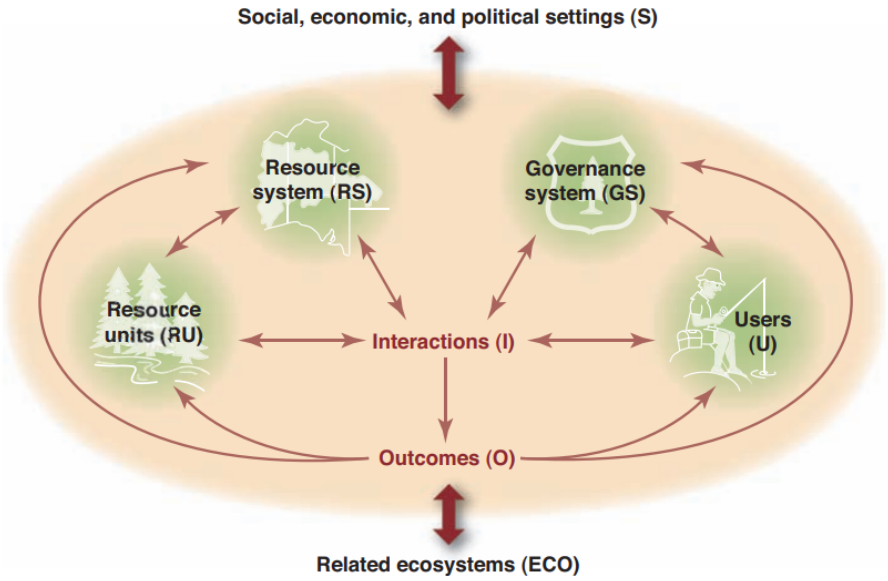


Figure 1: The core subsystems in a framework for analyzing social-ecological systems (Ostrom, 2009)

its aggregate state variables in energy-power dynamical state space model (Ilić and Jaddivada, 2018; Ilic and Jaddivada, 2020; Jaddivada, 2020). Notably, depending on the technology and its dynamic properties, its specifications in terms of limits on the ability to interact generally vary with system conditions. Because of this, it is necessary to have interactive information exchange in the online operations, instead of presently-used operating "nomograms" (Papic et al., 2007).

2.1 Basic component modeling in energy-power space

This unified modeling in energy-power space is closely related to the *bond graph* representation which has had a very rich history (Thoma, 2016). Our recent work has been specifically targeted to modular modeling of dynamics for distributed control and optimization of large-scale complex electric energy systems comprising very diverse technologies. As shown in Figure 2, the novel aspect of this modeling is that it starts by representing each component i in its standard state space form (Equations (1), (2) below), with internal states $x_i(t)$, interface variables $r_i(t)$, local primary control $u_i(t)$ and local disturbances $m_i(t)$ which are technology-specific (Ilić and Jaddivada, 2018; Ilic and Jaddivada, 2020).

$$\dot{x}_i(t) = f_{x,i}(x_i(t), u_i(t), m_i(t), r_i(t)) \tag{1}$$

$$\dot{r}_i(t) = f_{r,i}(x_i(t), \dot{P}_i(t), \dot{Q}_i(t)) \tag{2}$$

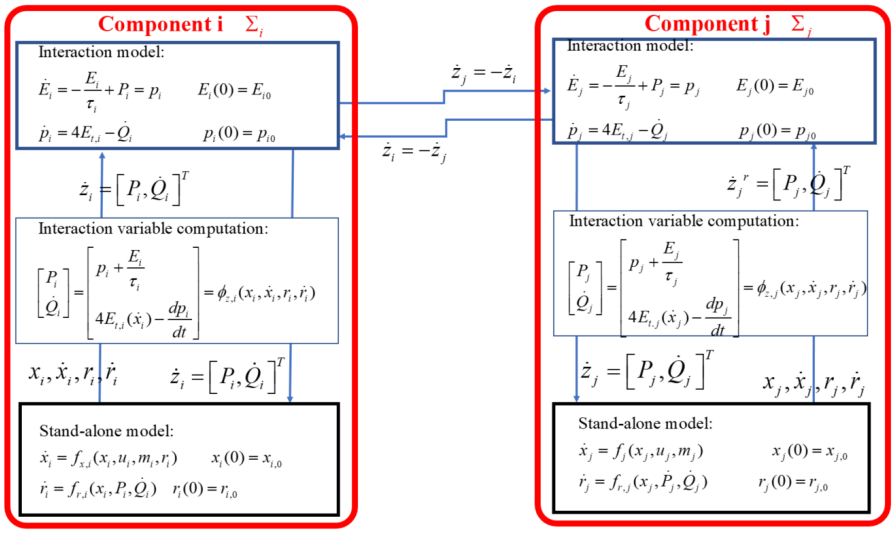


Figure 2: Sketch of a new modeling an interconnected system with heterogeneous components (Ilić and Jaddivada, 2018)

This modeling explicitly shows the general structure of an electric energy system fundamentally based on the following general definition of an interaction variable associated with any component i.

2.2 Definition and structural properties of interaction variables

The Interaction Variable $z_i(t)$ associated with component i is the integral of instantaneous power $\int_0^t P_i(s)ds$ and the instantaneous reactive power $Q_i(t)$ (Wyatt and Ilic, 1990; LaWhite and Ilic, 1997). Both of these quantities are a result of local state dynamics specific to the component and are defined as follows:

$$\int_0^t P_i(s)ds = \int_0^t \left(p_i(s) + \frac{E_i(s)}{\tau_i} \right) ds \tag{3}$$

$$Q_i(t) = \int_0^t \left(4E_{t,i}(s) - p_i(s) \right) ds \tag{4}$$

Here, $E_i, p_i, E_{t,i}$ respectively are the stored energy of the component, its first time derivative and the stored energy in tangent space. τ_i is the time constant of

the energy dynamics and can also be interpreted as the ratio of the stored energy and dissipation losses of the component. The details on the definitions of these quantities can be referred to in (Ilić and Jaddivada, 2018; Ilic and Jaddivada, 2020). Important is to note that the interaction variables depend only on local state variables, irrespective of what the connections with rest of the system are.

The key property of the interaction variable is that its derivative $\dot{z}_i(t) = [P_i(t)\ \dot{Q}_i(t)]$ is identically zero when the component is disconnected from the system, i.e. when it is in a stand-alone mode. These derivatives explicitly show dynamical effects of the component on the neighboring component when interconnected. Symmetrically, the dynamics of component i is explicitly affected by the neighboring components through the derivatives of their interaction variables. All core components, including generators, transmission lines and loads, take on the same structural form discussed next (Ilić and Jaddivada, 2018). It is this structure that sets the foundation for distributed control and computing.

Consider a two component interconnection of a source and load. For instance, consider that the source sub-system is a solar photovoltaic system, where there is a local exogenous disturbance seen because of the changes in solar radiation. These local disturbances together with the inter-component interactions can be studied by simply drawing an energy flow diagram as shown in Fig. 3. Here, solar radiation can be thought of as an exogenous disturbance characterized in terms of its interaction variable dynamics $\dot{z}_s^m(t) = [P_s^m(t)\ \dot{Q}_s^m(t)]^T$ affecting the system. The superscript m is used to differentiate the interaction variable dynamics at the interface ports defined as $\dot{z}_s(t) = [P_s(t)\ \dot{Q}_s(t)]^T$ and $\dot{z}_l(t) = [P_l(t)\ \dot{Q}_l(t)]^T$ for the source and load sub-systems respectively. The interaction variables as defined in Equations (3) and (4) for both source and load sub-systems, together with the structure-preserving interconnection relations, as will be detailed in the next section, are sufficient for a system operator to operate the interconnected system. The system operator can remain incognizant to internal details as dictated by the more-granular physical lower layer model in Eqn. (1).

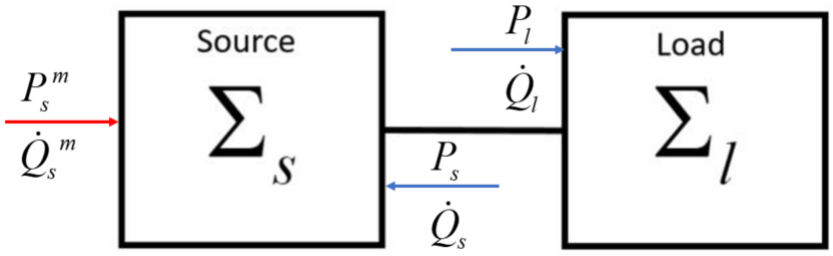


Figure 3: Input-output interfacing: Two component system comprising a source and a load

On the other hand, conventional approach of operating these systems is to

utilize the internal details of the components defined using the model in Equation 1 alone with corresponding sub-scripts of s and l, together with algebraic relations stemming from the Kirchoff's current and voltage laws. For the example under consideration, $m_s(t)$ is the local disturbance seen by source sub-system, whose local primary control is $u_s(t)$. Notably, the high penetration of solar and other intermittent power DERs which are power-electronically-controlled could lead to new electromagnetic instability phenomena already observed in Texas power grid (Adams et al., 2012). The modeling granularity of the dynamics of these components depends on specific technology. These internal physical models quickly become complex and, when interconnected, are also subject to algebraic real and reactive power balance equations, resulting in high-order nonlinear differential algebraic equations (DAE) models (Pai et al., 1995). Because of this complexity, it is very difficult to test components so that the interconnected system dynamics has desired response, say that the system stays stable for certain range of exogenous time-varying disturbances. Notably, this very serious roadblock to innovative deployments by the utilities can be overcome through *distributed computing*.

The most relevant, for purposes of introducing model-based protocols for computing and operating future electric energy systems, are the formulas shown in Figure 2. They represent a closed form mapping of the dynamics of aggregate state variables defined as energy stored in the component $E_i(t)$ and its derivative $p_i(t) = \dot{E}_i(t)$ in terms of internal extended state dynamics $\dot{x}_i(t), \dot{r}_i(t)$ and its interaction variable $\dot{z}_i(t) = [P_i(t)\ \dot{Q}_i(t)]^T$. Here $E_{t,i}(t)$ is the energy in tangent space and it has the same expression as energy, except the states are replaced by the derivatives of states. The mapping from the physical model into the aggregate states $[E_i(t)\ p_i(t)]^T$ is shown in the center panel in Figure 2. The fundamental relevance of this model is that any component which knows its own internal dynamics, can design its automation for the ranges of interaction variables it selects. Or, alternatively, the component can enhance its control and internal hardware design to implement its input-output specifications which are given solely as interaction variable specifications.

2.3 Unified multi-layered dynamical model of an interconnected system

The distributed model of each component i in energy-power space described above defines the dynamics of its aggregate variable in energy-power space. Notably, it simply reflects the power conservation law and rate of reactive power conservation law written for the cutset representing boundary of the component.

$$\dot{E}_i(t) = p_i(t) \tag{5}$$

$$\dot{p}_i(t) = 4E_{t,i}(t) - \dot{Q}_i(t) \tag{6}$$

Figure 2 shows a sketch of the interconnected mathematical model of the small system. General conservation laws (power and rates of change of power) can

be written for any level of granularity, by zooming in and zooming out into the system. The dynamics of distributed components is subject to the basic power conservation laws

$$P_i(t) + \Sigma_{j \in C_i} P_j(t) = 0 \tag{7}$$

$$\dot{Q}_i(t) + \Sigma_{j \in C_i} \dot{Q}_j(t) = 0 \tag{8}$$

Since each component is characterized in terms of both instantaneous power and rate of change of instantaneous reactive power, it is critical to start by writing generalized Tellegen's Theorem for both instantaneous power and rate of change of instantaneous reactive power (Ilić and Jaddivada, 2018; Penfield et al., 1970). This model cannot be written in standard state space form since it is fundamentally interactive.

3 Model-based structure of computational platform

The unified modeling based on first principles supports DyMonDS framework and next generation Supervisory Control and Data Acquisition (SCADA) for monitoring and controlling electric energy systems so that they enable sustainable and resilient service (Ilić, 2010; Ilic and Lessard). Overlaid is the computational platform which processes the information exchange defined using this modeling. Our team has worked for quite some time on applying unified modeling for designing next generation Supervisory Control and Data Acquisition (SCADA) we named Dynamic Monitoring and Decision Systems (DyMoNDS) (Ilić, 2010). The first version of Smart Grid in a Room Simulator (SGRS) was made at Carnegie Mellon University (Wagner et al., 2015). It was used to emulate integration of adaptive load management (ALM) into electricity markets (Joo and Ilić, 2013), to demonstrate potential of transactive energy management (TEM) of small distributed resources in retail markets (Holmberg et al., 2019) as well as for the US ARPA-E project which demonstrates ability of household appliances to participate in fast power balancing (Ilic and Jaddivada, 2019b). At MIT our team has further developed the SGRS into scalable electric power system simulator (SEPSS) which is fundamentally based on the ideas of modular modeling of interactions (Ilić et al., 2018).

The SEPSS simulation platform is based on the conceptual notion of unified multi-layered modeling, which utilizes the structural property that any power system agent needs to send information in terms of derivatives of its interaction variable, and, to, symmetrically, receive information from the rest of the system in terms of derivatives of their interaction variables. This not only ensures privacy but also enhances integration of components to assess a large system and its interconnections. Based on this idea, the SGRS shell makes the first attempt to align the physics-based structure with the partitioning of computing jobs to several processes (Wagner et al., 2015). The shell facilitates allocation of computing resources, sets up the TCP/IP communication channels

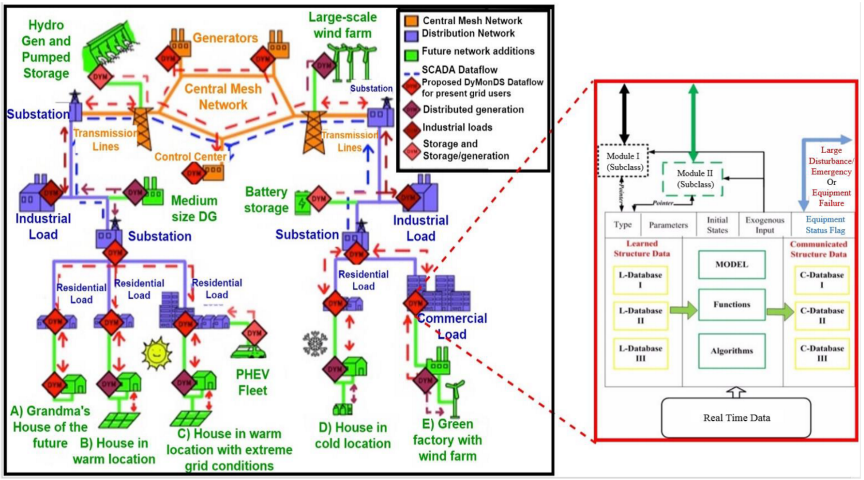


Figure 4: Dynamic Monitoring and Decision Systems (DyMonDS) framework (Ilić, 2010)

between different modules and initiates the simulation. The simulation platform is High Level Architecture (HLA)-compatible (Association et al., 2010), allowing integration with third-party software.We have implemented dynamic inter-operability between two simulation platforms: an advanced grid analysis software called NETSSWorks (Ilic and Lang, 2010) and the software platform. The sketch of the components that constitute the simulation platform and its links to external modules is shown in Figure 5 (Holmberg et al., 2019).

This figure shows interactions that may happen within the system. Here, the module named 'House' is one of the transactive agents, but industrial customers or any other component trading in an energy market can be modeled and treated as such green-colored boxes. Several of these models together constitute a community, and they are either coordinated by a community coordinator or through distributed peer-to-peer decisions. Our simulation platform does not recommend specific architectures for design and in fact is designed to be general enough for testing any given architecture in a rapid manner. Each module in the platform can be plugged in or out without having to change any of the simulation components. These concepts are similar to those proposed in the NIST ACM (Ferraiolo et al., 2001), which characterizes resources and their interfaces.

The simulation platform is still under development, with focus on making the simulations scalable to very large systems, by supporting clustering, aggregation and decomposition according to the physical and organizational structure of an electric power system that is rich in both temporal and spatial scales.

In sharp contrast with the idea of strictly data-enabled self-adaptation and self-optimization of computer architecture in response to workloads experienced (Donyanavard et al., 2020), the approach taken here is to actually define the information that needs to be exchanged between different layers using the unified

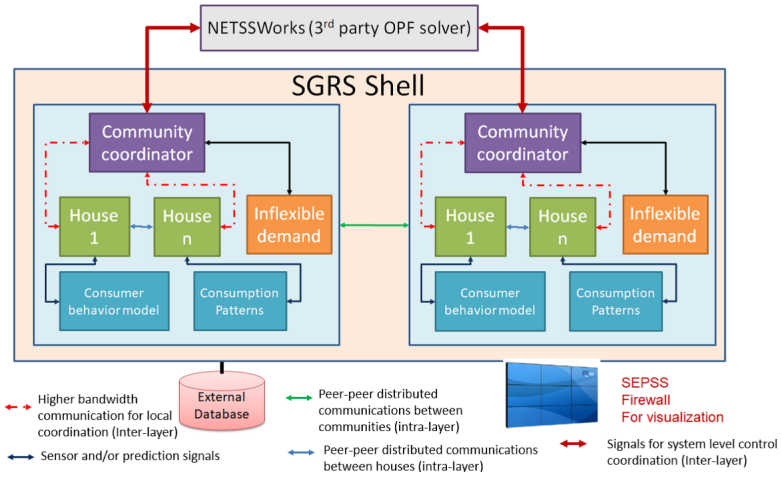


Figure 5: Interaction of software modules with SEPSS computer platform (Holmberg et al., 2019)

modeling in energy space, and set the basis for flow control in the computational architecture (Ilic and Jaddivada, 2019a). On the other hand, massive parallel data processing is mainly done by the grid users in parallel, including Machine Learning methods and Model Predictive Control (MPC) and Decisions (Ilic et al., 2019; Lauer et al., 2019; Carvalho et al., 2020). Once this is established, it becomes possible, at later stages, to design a very effective computational architecture which is self-verifying, adjusting and continuously checking its own performance, much the same way as Spiral was introduced for computing Fourier Transformation for signal processing purposes (Puschel et al., 2005). As a matter of fact, the multi-disciplinary approach to conceiving Smart Grid in a Room Simulator (SGRS) was very much motivated by Spiral since we were fortunate to have Franz Franchetti, the key contributor to Spiral, as one of our SGRS team members (Wagner et al., 2015). The unified modeling based on first principles supports Dynamic Monitoring and Decision Systems (DyMonDS) framework and next generation Supervisory Control and Data Acquisition (SCADA) for monitoring and controlling electric energy systems so that they enable sustainable and resilient service (Ilić, 2010; Ilic and Lessard). Overlaid is the computational platform which processes the information exchange defined using this modeling.

Much work remains to build on this idea of unified modeling for designing dedicated computer platforms for future emulating dynamics of electric energy systems in response to continuous disturbances and events driving changes. In this chapter only the basic idea is presented. In the remainder of this chapter we give several numerical simulation methods needed to actually implement electric energy systems using such computer architecture. As it turns out, state of the art of distributed numerical algorithms that capture fast dynamical interactions

taking place within a very complex network system are at their rudimentary stages. We describe next our progress on this problem, illustrate fundamental issues and set the basis for further work.

4 Plug-and-play automated method for power systems simulations (PAMPS)

The multi-layered model and computational platform structure described here are the first of their kind. While the interconnected system model shown in Figure 2 appears to be in the form of ordinary differential equations (ODE) in terms of energy-power variables while preserving the structure of the system, the modeling is inherently interactive in nature. This is because dynamic maps are needed to transform variables between lower layer and higher layer models at each of the components.

Furthermore, in Fig. 2, there is a need to differentiate between the $\dot{z}_i$ that is a result of local energy conversion dynamics and the one that is a function of the interconnection with rest of the system. These quantities are respectively indicated at the top and the bottom of the middle panel in Fig. 2, respectively. While these signals are physically the same in case of a feasible interconnection, in context of distributed simulations and/or control, they need to be differentiated. We thus use in what follows the superscript notation "out" and "in", respectively, to differentiate the two quantities. Further, the interaction variables can be modeled at any port of interaction. To differentiate between the local disturbance ports and the those that enter the port of interaction, superscripts m and r for these quantities are utilized to avoid confusion.

As evident from Fig. 2, the outgoing interaction variables $\dot{z}_i^{\mathrm{r,out}}$ depend on both state derivatives and port input derivatives, which in turn evolve as a function of $\dot{z}_i^{\mathrm{r,in}}$. These incoming interaction variables, however are a function of interconnection i.e., they depend on $\dot{z}_j^{\mathrm{r,out}}$ $\forall j \in \mathcal{C}_i$. As a result, the models are of the form $\dot{z}_i^{\mathrm{r,out}} = f(x_i, \dot{z}_j^{\mathrm{r,out}})$, in turn resulting in $\dot{x}_{z,i} = f(x_{z,i}, \dot{x}_{z,j})$. To the best of our knowledge these distributed interactive models have not been studied in the literature. Thus, numerical methods need to be devised for distributed simulations of these interactive models, while retaining the ODE structure at the system level.

The multi-layered model proposed here lends itself to distributed computing. However, the hurdle with respect to the interactive nature of the stand-alone models needs to be overcome. By exploiting the unified higher-layer energy space models and exploiting the inherent structure of the system, we propose a novel algorithm referred to as plug-and-play automated method for power system simulations (PAMPS). While the algorithm finds its use in both distributed control and simulations, we focus on the numerical simulations first. The proposed PAMPS approach to distributed simulations would broadly entail following sequence of steps formalized in the Algorithm 1.

4.1 PAMPS-based distributed simulations

The algorithm begins with the initialization phase (Step 0 in Algorithm 1 where the assignment of the initial values of extended state variables for use by the lower layer model and an assumption on zero incoming interaction variables at initial time.

Next, by using the information of local extended state variables, outgoing interaction variables are computed that would be communicated to the neighbors (Step 1 in Algorithm 1). This dynamic map is abstracted through $\phi_{z,i}$, the expression for which has been shown in Fig. 2. Note that this computation would require the knowledge of $\dot{r}_i$ which in turn would depend on the values of the incoming interaction variables $\dot{z}_i^{r,in}$, which has been assumed to be equal to zero in the initialization phase. For the next time step though, this value is computed as a result of interconnection laws in Equations 7 and 8, also included as step 2 in the Algorithm 1.

The new values of incoming interaction variables, which are a function of neighbors' state variables are utilized in Step 3 to perform time-stepping of internal states by using an integrator. This map corresponding to the numerical integration is abstracted as $\phi_{x,i}$.

Algorithm 1 PAMPS algorithm

Time $t \leftarrow 0$ ▷ Time initialization
while $t \leq t_{end}$ **do**
  for each component $i \in \mathcal{N}$ **do** ▷ Parallel implementation
    (Step 0) Given initial extended state space variables: $x_i(0) \leftarrow x_{i,0}$; $r_i(0) \leftarrow r_{i,0}$; and incoming interaction variable dynamics: $z_i^{r,in}(t) = 0 \quad \forall t \leq 0$ ▷ Initialization
    (Step 1) Compute outgoing interaction variables as follows and send out to neighbors: $z_i^{r,out}(t) \leftarrow \phi_{z,i}\left(x_i(t), r_i(t), \dot{z}_i^{r,in}(t-\delta t)\right)$
    (Step 2) Receive outgoing interaction variables communicated by neighbors to obtain incoming interaction variables by establishing interconnection laws: $\dot{z}_i^{r,in}(t) \leftarrow -\sum_{j \in \mathcal{C}_i} \dot{z}_j^{r,out}(t)$
    (Step 3) Compute extended state trajectories: $[x_i(t+\delta t), r_i(t+\delta t)]^T \leftarrow \phi_{x,i}\left(x_i(t), r_i(t), \dot{\tilde{z}}_i^{r,in}(t)\right)$
  ▷ All components covered
  Advance time stamp $t \leftarrow t + \delta t$ ▷ Time update

In Fig. 2, the dependence of dynamics of extended state $\dot{r}_i$ has been shown to be dependent on $P_i(t)$ implicitly (Ilic and Jaddivada, 2020; Jaddivada, 2020). However, it is actually dependent on $\dot{P}_i(t)$. We thus introduce additional notation $\dot{\tilde{z}}_i$ to represent the vector $[\dot{P}_i, \dot{Q}_i]^T$ that is to be used in the time-stepping of lower layer extended state variables in Step 3.

There is now a huge disconnect between the steps 2 and 3. To carry out time-stepping of internal states in step 3, the need for estimating $\dot{P}^{r,in}$ accurately arises. This can be done by employing one of the following three alternatives

- Variant 1: By taking previous historical values of $P_i^{r,in}(t)$ for use in (Step 3)

$$\dot{P}_i^{r,in}(t) = \frac{1}{\delta t}\left(P_i^{r,in}(t) - P_i^{r,in}(t-\delta t)\right) \tag{9}$$

- Variant 2: By revising the (step 1) and (step 2) of the algorithm as follows:
  - (step 1)
    Previously step 1 of the algorithm made use of the mapping $\phi_{z,i}$ expanded out in Fig. 2. However, for computing $\dot{P}_i^{r,out}$, we propose replacing the first element of $\phi_{z,i}$ with the expression $\dot{p}_i + \frac{p_i}{\tau_i}$. Let us call this new map as $\phi_{\tilde{z},i}$. With this the algorithm's step 1 can be revised as follows:
    $\tilde{z}_i^{r,out}(t) \leftarrow \phi_{\tilde{z},i}\left(x_i(t), r_i(t), \dot{\tilde{z}}_i^{r,in}(t-\delta t)\right)$
  - (step 2)
    Since the power balancing is a result of generalized Tellegen's theorm. If the real power balances as indicated in Eqn. (7), they should also balance for the rate of change of instantaneous power, thus resulting in following step 2 replacement
    $\dot{\tilde{z}}_i^{r,in}(t) \leftarrow -\sum_{j\in\mathcal{C}_i} \dot{\tilde{z}}_j^{r,out}(t)$
- Variant 3: By assuming historical feasibility of interconnection i.e. $P_i^{r,in}(t-\delta t) = P_i^{r,out}(t-\delta t)$, leading to the following relation (Jaddivada, 2020)

$$\dot{P}_i^{r,in}(t) = \frac{1}{\delta t}\left(P_i^{r,in}(t) - P_i^{r,out}(t)\right) \tag{10}$$

Typically, the smaller is the time-step δt the closer would be the trajectories obtained by each of the three variants of derivative estimation, and thus would lead to the faithful emulation of trajectories through distributed simulations. Another strategy for ensuring the error made in estimating the derivative value to fade away with time, is to incorporate an error control strategy as follows. This is made apparent in the discrete time implementation algorithm. Furthermore, the mapping $\phi_{x,i}$, $\phi_{z,i}$ is could be explicit or an implicit map showing the dependency of variables needed for computing x_i and $z_i^{r,out}$ respectively. The actual relations will be made cleared in the discrete time implementation explained next.

4.2 Discrete time distributed implementation

In the continuous time PAMPS algorithm introduced in the previous section, we have focused on differentiating between the outgoing and incoming interaction variables, and explained the dependence of local variables on the interaction variables and vice-versa. However, the algorithm will only be more clear if it is explained in context of discrete time-implementation.

In what follows, we denote timestep T_x for lower-layer state evolution and a timestep T_z for higher-layer energy-space variables computation. Let the associated sample numbers be denoted by k_x and k_z respectively The time step selection needs to be so that $T_z << \tau_i$ and T_x must be smaller than the time constant of the fastest state variable. Note that energy variables evolve at time constant τ_i, which is an aggregate affect of all the local variables, and thus tends to be much slower than the internal state variable evolution. However, for single-state variable components, both lower layer and higher layer models may evolve at the same rates.

The discrete time implementation of PAMPS is summarized in Algorithm 2.

The steps 2.2.1, 2.2.2 or 2.2.3. can in fact be appended with a following error control scheme to reduce the effect of error in estimation of derivative of $P_i^{r,in}$.

$$\dot{z}_i^{r,in}[k_z] = -\sum_{j \in \mathcal{C}_i} \dot{z}_j^{r,out}[k_z] + \epsilon_i[k_z] \tag{11}$$

Here $\epsilon_i[k_z]$ acts as a feedback signal that controls the error made in derivative estimation as sensed by the residual of interconnection laws at the interface. One simple error control scheme is a simple proportional derivative scheme shown under

$$\epsilon_i[k_z] = -K_d \left(\dot{z}_i^{r,in}[k_z - 1] + \sum_{j \in \mathcal{C}_i} \dot{z}_j^{r,out}[k_z] \right) - K_p \left(z_i^{r,in}[k_z] + \sum_{j \in \mathcal{C}_i} z_j^{r,out}[k_z] \right) \tag{12}$$

From these expressions above, clearly $|k_d| \leq 1$ to ensure numerical stability of the updates of $z_i^{r,in}$ while the gain $0 \leq k_p \leq 1$. The right handed inequality ensures numerical stability while the left hand-sided inequality is a desired property to ensure $\left(z_i^{r,in} - z_i^{r,out} \right)$ decays over continuous time.

5 Proof-of-concept illustrations

The different variants of distributed information exchange as required for the estimation of derivative of $P_i^{r,\text{in}}$ have been explained in the last two sections in both continuous time domain and discrete time. We now take simple examples of two component interconnections to evaluate the dependence of proposed algorithm on the type of distributed information exchange and also on the type of interconnection. The benchmark considered is the response obtained by simulating the interconnected system model in terms of physical variables by eliminating the dependent state variables.

5.1 Effect of information exchange type

We first consider a simple two component interconnection shown in Fig. 6 where component 1 is a lossy inductor denoted by $R_1, L_1$with a constant voltage source u_1 and let the second component be a lossy capacitor denoted by G_2, C_2.

Algorithm 2 Multi-rate discrete time distributed simulations

$k_z \leftarrow 0, k_x \leftarrow 0$ ▷ Time initialization

while $k_z T_z \leq t_{end}$ **do**

  for each component $i \in \mathcal{N}$ **do** ▷ Parallel implementation

    (Step 0.0)Initialization at T_x rate: $x_i[k_x] \leftarrow x_{i,0}; r_i[k_x] \leftarrow r_{i,0}$;

    (Step 0.1) Initialization at T_z rate $\dot{z}_i^{r,in}[k_z] \leftarrow 0 \forall k \leq 0$

    (Step 1.1) Compute $\dot{Q}_i^{r,out}[k_z]$ and send it out to neighbors

    (Step 2.1) Receive from neighbors and apply dynamics of reactive power balance relation $\dot{Q}_i^{r,in}[k_z] \leftarrow -\sum_{j \in \mathcal{C}_i} \dot{Q}_i^{r,out}[k_z]$

    Choose one of the three alternatives ▷ For estimation of $\dot{P}_i^{r,in}[k_z]$

- Variant 1
  - (Step 1.2.1) Compute $P_i^{r,out}[k_z]$ and send it out to neighbors
  - (Step 2.2.1) Receive from neighbors and apply instantaneous power balance equations $P_i^{\mathrm{r,in}}[k_z] \leftarrow -\sum_{j \in \mathcal{C}_\rangle} P_j^{\mathrm{r,out}}[k_z]$
  - Estimate derivative of instantaneous power using historical values $\dot{P}_i^{\mathrm{r,in}}[k_z] = \frac{1}{T_z}\left(P_i^{\mathrm{r,in}}[k_z] - P_i^{\mathrm{r,in}}[k_z - 1]\right)$
- Variant 2
  - (Step 1.2.2) Compute $\dot{P}_i^{r,out}[k_z]$ and send it out to neighbors
  - (Step 2.2.2) Receive from neighbors and apply instantaneous power dynamics balance equation $\dot{P}_i^{\mathrm{r,in}}[k_z] \leftarrow -\sum_{j \in \mathcal{C}_\rangle} \dot{P}_j^{\mathrm{r,out}}[k_z]$
- Variant 3
  - (Step 1.2.3) Compute $P_i^{r,out}[k_z]$ and send it out to neighbors
  - (Step 2.2.3) Receive from neighbors and apply instantaneous power balance equations $P_i^{\mathrm{r,in}}[k_z] \leftarrow -\sum_{j \in \mathcal{C}_\rangle} P_j^{\mathrm{r,out}}[k_z]$
  - Estimate derivative of instantaneous power assuming feasibility of interconnection at previous time-step $\dot{P}_i^{\mathrm{r,in}}[k_z] = \frac{1}{T_z}\left(P_i^{\mathrm{r,in}}[k_z] - P_i^{\mathrm{r,out}}[k_z - 1]\right)$

    while $k_x T_x \leq (k_z + 1) T_z$ **do**

      (Step 3.1.) $(r_i[k_x + 1] \leftarrow f_r'(r_i[k_x], \dot{z}_i^{\mathrm{r,in}}[k_z])$

      (Step 3.2.) $x_i[k_x + 1] \leftarrow f_x'(x_i[k_x], r_i[k_x + 1])$

      Advance $k_x \leftarrow k_x + 1$

    $x_i[k_z + 1] \leftarrow x_i[k_x]$; $r_i[k_z + 1] \leftarrow r_i[k_x]$ ▷ Multi-rate synchronism ▷ All components considered

  Advance $k_z \leftarrow k_z + 1$

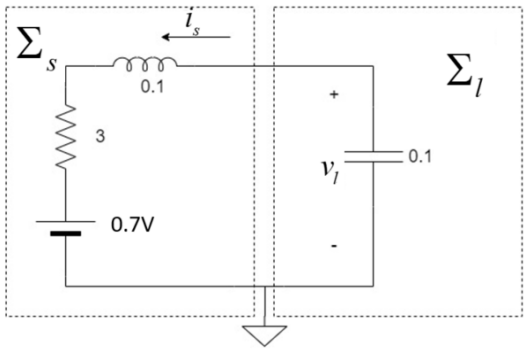


Figure 6: Two component - LC-type interconnection

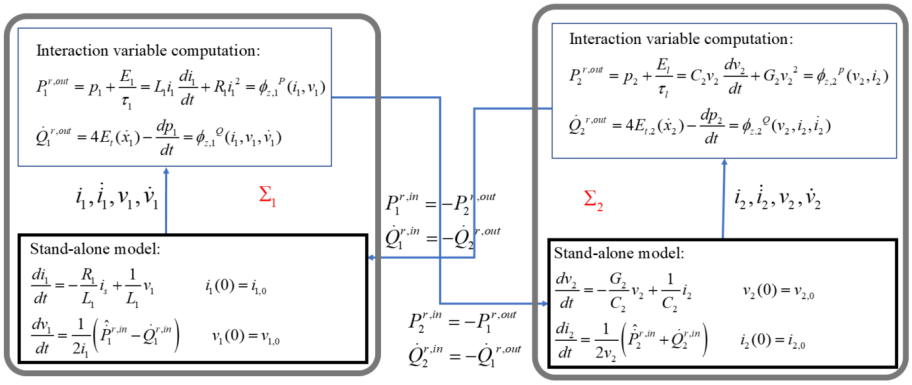


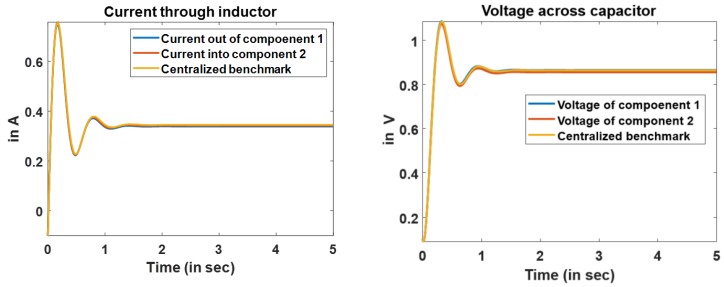
Figure 7: Multi-layered interactive model and information exchange - LC-type interconnection

The multi-layered interactive model for this specific example is shown in Fig. 7. We have implemented Algorithm 2 by choosing a different variant each time to compare the qualitative responses obtained and to understand the propagation of errors in power balancing over a period of time.

For both the variants 1 and 2, similar current and voltage trajectories are obtained as shown in Fig. 8. There is however a significant difference between the two approaches as evident from the error accumulation over time, as seen in Fig. 9. While the shapes of the two plots appear to be the same, the scale can be checked to see that the variant 2 incurs thousand times as large errors as variant 1.

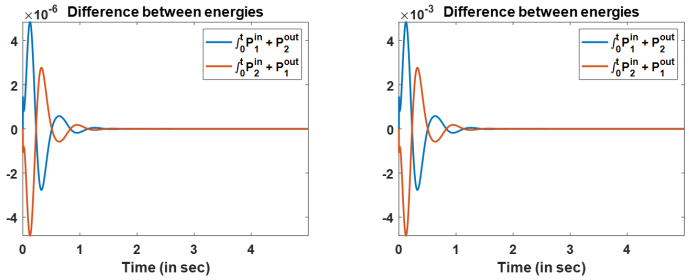
The variant 3 results in a numerically sensitive performance, as evident through high frequency oscillations observed in the error plots in Fig. 10b. Furthermore, the error control gains had to be carefully chosen for this case in contrast to the other two variants.

For an LC interconnection, it is known that the symplectic schemes where



(a) Current (b) Voltage

Figure 8: Variation 1-based implementation of PAMPS algorithm on LC interconnection



(a) Variant 1 (b) Variant 2

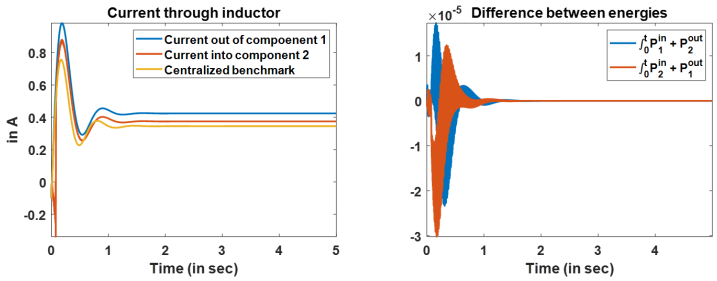
Figure 9: Error accumulation over time for variants 1 and 2 of PAMPS implementation on LC interconnection

present values of current and future values of voltage when utilized by component 1, results in energy-preserving numerical scheme (Hairer et al., 2006). Variant 1 is analogous to the symplectic scheme, since the future voltage of component 2 can be computed by component 1, with the knowledge of $\dot{Q}_1^{r,in}$ that is communicated by component 2. The present value of current is available through the value of $P_1^{r,in}$ sent out by the component 2.

5.2 Effect of the interconnection type

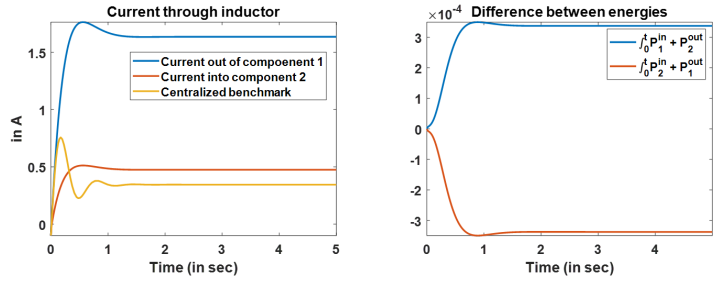
It has further been noticed that the LC-type interconnection considered in the previous example is bound to work only for when $K_p > 0$. The performance is best when $K_p = 1$. However, when $K_p = 0$, i.e. when the proportional component is absent, we see that the exact simulation responses could not be reproduced as expected. Shown in the Fig. 11 are the corresponding plots.

In contrast, consider another type of interconnection when the component 1 is a lossy inductor with a source voltage serving an ideal power sink. Its



(a) Current (b) Error accumulation

Figure 10: Variation3-based implementation of PAMPS algorithm on LC interconnection



(a) Current (b) Error accumulation

Figure 11: Variation1-based implementation of PAMPS algorithm on LC interconnection without error control

distributed modular model is shown in Fig. 12.

For this example, it has been observed that variant 1 of PAMPS implementation with or without error control results in the same response shown in Fig. 13.

The constant power load-type interconnection has one of the power variables that is $P_2^{r,in}$ is fixed, while $\dot{Q}_2^{r,in}$ is a free variables and has no dependence on any other variables, since the load component does not have its own dynamics. As a result, any value of $\dot{Q}_2^{r,in}$ sent out by the component 1 is acceptable from the load's perspective and thus there is no need of any error control to reach a consensus as long as the value of $P_1^{r,in} = P_L^*$ is within a range that does not affect dynamic stability of component 1.

5.3 Feasibility assessment

Distributed models and simulations render computation of ranges of outgoing interaction variables for given incoming interaction variables. This range is to be a subset of the possible variation of interaction variables for feasibility of

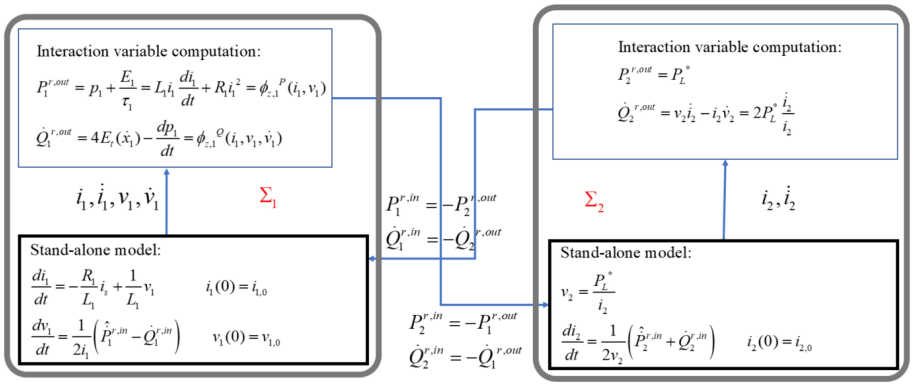
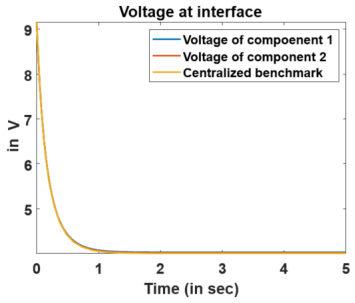
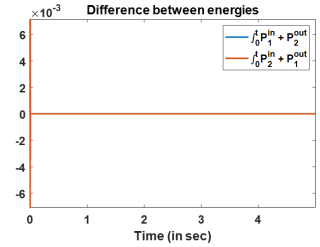


Figure 12: Multi-layered interactive model and information exchange - CPL-type interconnection



(a) Current



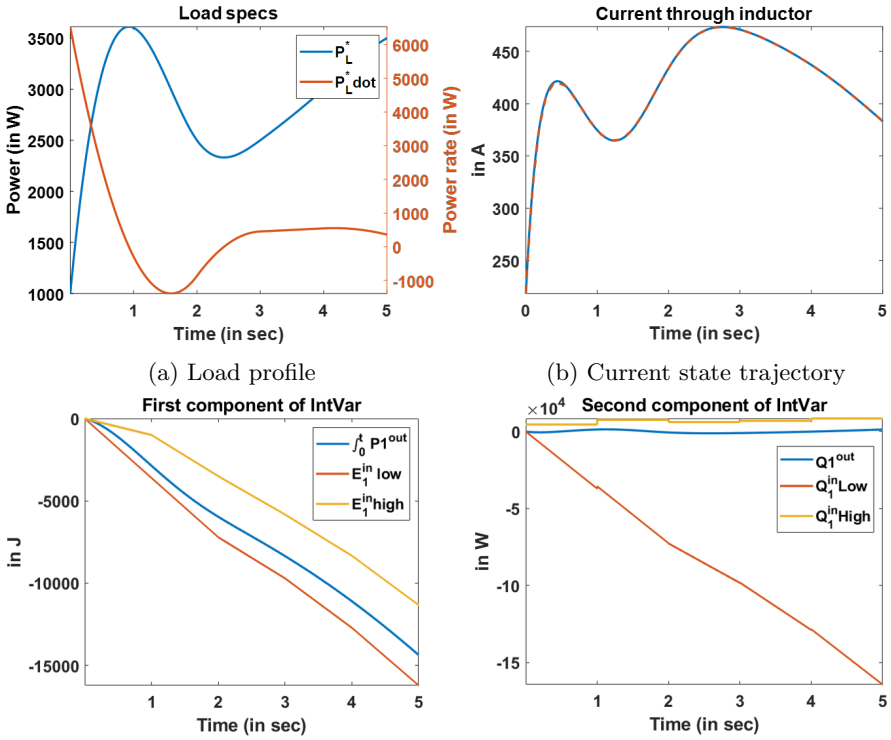
(b) Error accumulation

Figure 13: Variation1-based implementation of PAMPS algorithm on CPL-type interconnection

the interconnection (Ilic and Jaddivada, 2020; Jaddivada, 2020). For the two component interconnection serving a constant power load varying in time, the limits of the changes in $z_i^{r,in}$ can be estimated ahead of time, given the ranges of P_L^* and $\dot{P}_L^*$.

These bounds are plotted in Figures 14c and 14d (red and yellow plots) for the load profile in Fig. 14a where ranges of P_L^* and $\dot{P}_L^*$ are communicated every 1 second to the component 1. The actual outgoing interaction variables are indicated by blue. Notice that at all times, the outgoing interaction variables lie within the predicted ranges. Thus, we see also through the plots of internal states in Fig. 14b, that there has been no dynamic instability.

Shown in Fig. 15 is the validation of the case when the system is subject to steep changes in power shown in Fig. 15a making the interconnection infeasible after time $t = 0.8$ seconds. While this is not immediately evident through the bounds on first component of interaction variables in Figure. 15c, the dynamic



(a) Load profile

(b) Current state trajectory

(c) First component

(d) Second component

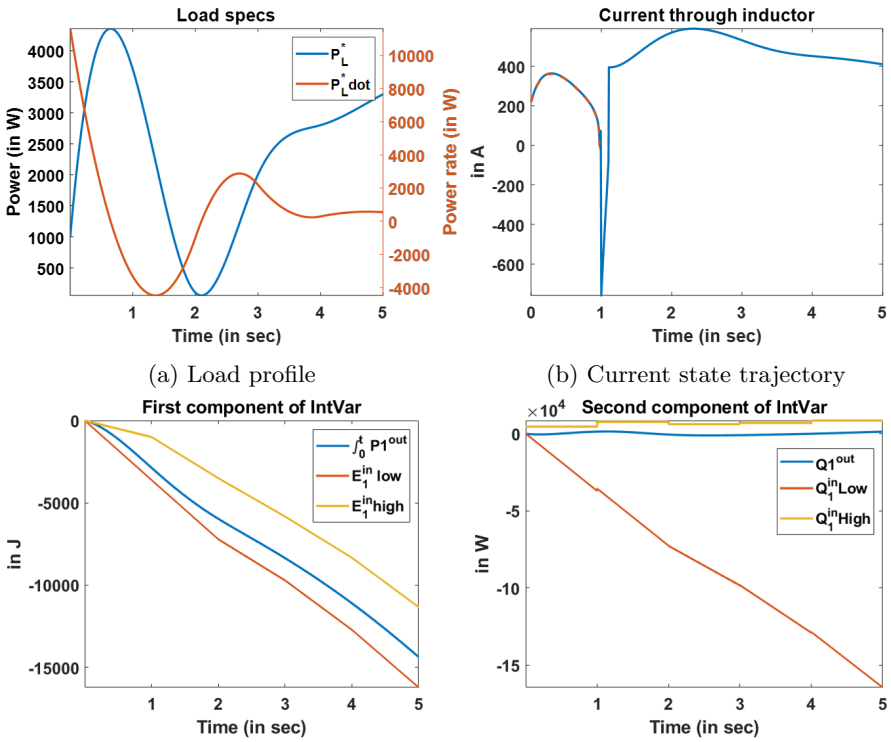
Figure 14: Case of a feasible interconnection at all times: Proof-of-concept illustration of validation of conditions on ranges of interaction variables over a period of time

instability can predicted by looking at the the ranges on second component being violated in Figure 15d. Consequently, the internal states show unstable behavior as well as shown in Figure 15b.

In this exercise, we have only validated the energy-based bounds for feasible interconnection. However, distributed computing can also be put to use for ahead-of-time predicting the ranges of incoming interaction variables so that the component takes local preventive or corrective actions to avoid operating closer to the limits.

6 The challenge of multi-layered interactive computing: Accurate and efficient derivatives

As introduced in this chapter, the distributed interactive algorithms based on the unified modeling approach lend themselves to parallel distributed computing based on local data processing and information exchange in terms of derivatives



(a) Load profile

(b) Current state trajectory

(c) First component

(d) Second component

Figure 15: Case of a infeasible interconnection: Proof-of-concept illustration of validation of conditions on ranges of interaction variables over a period of time

of interaction variables information exchange with the rest of the system. This sets the foundations for computational platforms with lots of structure using hybrid computing. Essentially, much data processing specific to technologies and sub-objectives of the distributed subsystems is internalized locally. The information flow in between modules is mainly in terms of derivatives of interaction variables, and this makes it possible to design hybrid computational platforms with well-defined design specifications.

However, as illustrated in the first proof-of-concept simulations of such distributed algorithms, their performance is fundamentally dependent on the accuracy and efficiency of computing derivatives of interaction variables exchanged in between the subsystems. The collaboration of the co-authors of this chapter has opened a new avenue for overcoming this sensitivity. In particular, algorithmic Differentiation (AD), in sharp contrast to finite differencing, provides *exact* derivative information about a function F given in a high-level programming language and it does so with time and space complexity that can be bounded by the complexity of evaluating the function itself. The key idea in algorithmic differentiation is the systematic application of the chain rule of calculus. To

that end, the computation of the function F is decomposed into a (typically long) sequence of simple evaluations, e.g., additions, multiplications, and calls to elementary functions such as $\sin(v)$ or $\cos(v)$. The derivatives with respect to the arguments of these simple operations can be easily calculated. A systematic application of the chain rule then yields the derivatives of the entire sequence with respect to the input variables $x \in \mathbb{R}^n$.

Modes of AD: AD has two basic modes of operation, called *forward mode* and *reverse mode*. In the forward mode, derivatives are propagated together with the evaluation of the function during the execution of a given code segment. From a mathematical point of view, this can be interpreted as the computation of tangent information in the following way. Assume for a moment that the argument vector x represents the value of a time-dependent path $x(t)$ at $t = 0$. Then this time-dependent path defines a tangent $\dot{x} = \frac{\partial x(t)}{\partial t}|_{t=0}$. Using the forward mode of AD, one computes the resulting tangent $\dot{y}$ obtained for the time-dependent path $F(x(t))$ that is well defined if the function F is sufficiently smooth, i.e.

$$\dot{y} = \frac{\partial}{\partial t} F(x(t))\Big|_{t=0} = F'(x(t)) \frac{\partial x(t)}{\partial t}\Big|_{t=0} = F'(x)\dot{x} \equiv \dot{F}(x, \dot{x}).$$

This interpretation also explains the alternative names sometimes used for the forward mode of AD, namely tangent mode or tangent-linear mode.

Instead of being propagated together with the function evaluation, the derivatives can also be computed backwards starting from the function value y towards the independent variables x after a complete function evaluation. This is called the reverse mode of AD. Much like the mathematical interpretation given earlier for forward mode AD, one can give a similar interpretation for the reverse mode as well. Consider for a given vector $\bar{y} \in \mathbb{R}^m$ and a given value $c \in \mathbb{R}$ the hyperplane $\{y \in \mathbb{R}^m \,|\, \bar{y}^\top y = c\}$ in the range of F. The hyperplane has the inverse image $\{x \in \mathbb{R}^n | \bar{y}^\top F(x) = c\}$. By the implicit function theorem, this set is a smooth hypersurface with the normal $\bar{x}^\top = \bar{y}^\top F'(x) \equiv \bar{F}(x, \bar{y})$ at x, provided $\bar{x}^\top$ does not vanish. Geometrically, $\bar{y}$ and $\bar{x}$ can be seen as normals or cotangents.

Over the last several decades, extensive research activities have led to a thorough understanding of the complexities (time as well as memory) of these two basic modes of AD. The forward mode of AD provides one column of the Jacobian $F'(x)$ with a time complexity equal to no more than three function evaluations. The time needed to compute one row of the Jacobian $F'(x)$, e.g., the gradient of a scalar-valued component function of F, is less than four function evaluations using the reverse mode of AD in its basic form. Importantly, this bound for the reverse mode is completely independent of the number of input variables. This is highly attractive in contexts where there are many inputs—e.g., design parameters, neural network weights and the like—and a scalar-valued function such as a scalar loss function to be minimized.

AD Tools: Along with the progress on theoretical foundation, numerous AD software tools have also been developed over the years. Many of these tools have matured to a state where they can be used to provide derivatives for

Table 1: Sample list of stand-alone AD tools.

Language	**Tool**	**Paradigm**	**Mode**
C/C++	ADOL-C	OOL	FM, RM, Taylor arithmetic
	ADIC	ST	FM
	CasADi	OOL	FM, RM
	OpenAD	ST	FM, RM
	Sacado	ST	FM, RM
	TAPENADE	ST	FM, RM
Fortran	ADIFOR	ST	FM
	OpenAD	ST	FM, RM
	TAF	ST	FM, RM
	TAPENADE	ST	FM, RM
Python	ADOL-C	OOL	FM, RM, Taylor arithmetic
	CasADi	OOL	FM, RM
Julia	JuliaDiff	OOL	FM, RM, Taylor arithmetic
MATLAB	AdiMat	ST+OOL	FM, RM
	CasADi	OOL	FM, RM
R	ADOL-C	OOL	FM, RM, Taylor arithmetic
	Madness	OOL	FM, Taylor arithmetic

OOL: operator overloading, ST: source transformation, FM: forward mode, RM: reverse mode.

large and unstructured codes. The implementation paradigms underlying most existing AD tools are *source transformation* and *operator overloading*.

When using source transformation to implement AD, a source code that evaluates the function to be differentiated is given to the AD tool together with information as to which variables are the independents and which variables are the dependents. The AD tool then generates a new source code to compute the function as well as the required derivative information. To achieve this, one needs in essence a complete compiler. As a first step, this compiler parses the program to be differentiated to perform analysis, including syntactic and semantic analysis. In a second step, transformation is performed to include the statements for the derivative calculations. In a third step, optimization of the resulting code is carried out to increase performance. Finally, in a fourth step, the extended program is written out in the target programming language.

The technique of operator overloading exploits the capability of programming languages like C++, Fortran 95 and the like to define a new class of variables and to *overload* operators like $+, -, *$ and functions like sin(). Using operator overloading, it is rather straightforward to implement a tool for the forward mode AD.

Table 1 gives some representative examples of currently publicly available stand-alone AD tools. The website `www.autodiff.org` has a more comprehensive list of available AD tools. There is also a burgeoning research activity in

AD for domain-specific and emerging languages. Examples include *autodiff* for Halide (Ragan-Kelley et al., 2013), *JuliaDiff* for Julia (Bezanson et al., 2017), and *Clad* as AD tool using Clang and LLVM (Vassilev et al., 2015). *Tangent* (van Merriënboer et al., 2017) can be used for the algorithmic differentiation of code generated with the scripting language Python.

Exploiting Sparsity in Derivative Computation: For large-scale problems involving vector functions, the underlying Jacobians and Hessians are often very sparse. That is to say, they contain numerous zero entries. Instead of computing all the zero entries with the one of the modes of AD as described earlier, one can explicitly exploit the sparsity to make the derivative calculation much more efficient. A well-established way of achieving this efficiency (saving in both storage and time) is using a compress-then-recover strategy. This strategy consists of four steps: (1) determine the sparsity structure of the desired derivative matrix A (Jacobian or Hessian); (2) using graph coloring on an appropriate graph representation of the matrix A, obtain a seed matrix S of much fewer columns (p) than those f A; (3) compute the numerical values of the entries of the compressed matrix $B \equiv AS$; and (4) recover the entries of A from B. This framework has been proven to be an effective technology for computing sparse Jacobian as well as sparse Hessian matrices in a variety of applications (Coleman and Moré, 1983; Coleman and Moré, 1984; Coleman and Cai, 1986; Gebremedhin et al., 2008, 2009). A comprehensive review and synthesis of the use of graph coloring in derivative computation is available in (Gebremedhin et al., 2005). Efficient star and acyclic coloring algorithms for Hessian computation have been developed in (Gebremedhin et al., 2007), and Hessian recovery algorithms that take advantage of the structures of star and acyclic coloring have been presented in (Gebremedhin et al., 2009). A suite of implementations of various coloring algorithms, recovery methods and graph construction routines (needed in Steps 2 and 4 of the procedure outlined above) is provided in the software package ColPack (Gebremedhin et al., 2013). ColPack is integrated with ADOL-C and has been in distribution since ADOL-C version 2.1.0.

7 Conclusions

In closing, this chapter makes an attempt to establish structural unified modeling for computational platforms needed to emulate dynamics of complex electric energy systems. Such platforms do not exist at present, instead computer applications in support of scheduling resources to supply predictable system demand in this field have been mainly feed-forward off-line and with human-in-the-loop. System dynamics created by such changes in demand and resource scheduling are not simulated on-line because of the overwhelming complexity of high-order ODE models subject to many algebraic constraints. Because of this, the actual utilization of resources is based on the worst-case approach of ensuring acceptable operations for a handful of deterministic scenarios. As new technologies are integrated to meet decarbonization objectives, the renewable resources are highly stochastic. Also, as a direct consequence of climate change, the world

is experiencing many more natural disasters than in the past. Providing clean and resilient electricity services over broad ranges of highly-varying and uncertain conditions, requires change of today's operating paradigm from top-down deterministic to a minimally-coordinated distributed self adaptation of many resources and users, and of the electric power grid itself.

In this chapter we propose that it is indeed possible to evolve today's paradigm into a more flexible and distributed paradigm, and still ensure high QoS. To do so, a more systematic unified modeling is needed for setting information exchange protocols and for data-enabled computing and self-adaptation. Important for this book is the fact that the same unified modeling can and should be used to begin to design computational platforms capable of emulating complex interactions. This is needed to both assess potential effects of different technologies, hardware and software, on system performance, as well as for enabling more flexible operations. A digital twin is needed to emulate evolution of system dynamics at different temporal and spatial granularity. The same computational platform can be used to monitor and indicate feasibility and stability of the system as conditions vary, ultimately enabling its autonomous provision of electricity services. We suggest that while the basic modeling and its use for defining information flows are understood, the self-adaptation and self-organization will require use of Machine Learning and Artificial Intelligence tools for exact and efficient computing of derivatives. In this chapter we have only illustrated this challenge when attempting to emulate in a distributed way very fast dynamics accurately. Similar challenge examples of the need for accurate computation of various sensitivities of cost functions with respect to active constraints exist in computational platforms relevant for electricity markets. It would be of tremendous value to pursue multi-disciplinary work which furthers the ideas in this chapter to the point of having robust hybrid computational platforms as the basis for scalable digital twins of complex electric energy systems. This chapter reports on the first steps of such collaboration by the co-authors.

References

J. Adams, C. Carter, and S.-H. Huang. Ercot experience with sub-synchronous control interaction and proposed remediation. In *PES T&D 2012*, pages 1–5. IEEE, 2012.

I. S. Association et al. Ieee standard for modeling and simulation (m&s) high level architecture (hla)—framework and rules. *Institute of Electrical and Electronics Engineers, New York. IEEE Standard*, (1516-2010):10–1109, 2010.

A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *The Journal of Machine Learning Research*, 18(1):5595–5637, 2017.

R. Betancourt and F. L. Alvarado. Parallel inversion of sparse matrices. *IEEE transactions on power systems*, 1(1):74–81, 1986.

J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

T. Boston. Private correspondence with Terry Boston, former PJM CEO, CIRCA, 2020.

P. M. Carvalho, J. D. Peres, L. A. Ferreira, M. D. Ilic, M. Lauer, and R. Jaddivada. Incentive-based load shifting dynamics and aggregators response predictability. *Electric Power Systems Research*, 189:106744, 2020.

K. Clements, G. Krumpholz, and P. Davis. Power system state estimation residual analysis: An algorithm using network topology. *IEEE Transactions on Power Apparatus and Systems*, (4):1779–1787, 1981.

T. Coleman and J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20(1):187–209, 1983. doi: 10.1137/0720013. URL `https://doi.org/10.1137/0720013`.

T. F. Coleman and J.-Y. Cai. The cyclic coloring problem and estimation of sparse Hessian matrices. *SIAM Journal on Algebraic Discrete Methods*, 7(2): 221–235, 1986.

T. F. Coleman and J. J. Moré. Estimation of sparse Hessian matrices and graph coloring problems. *Mathematical Programming*, 28(3):243–270, 1984.

M. L. Crow and M. Ilić. The waveform relaxation method for systems of differential/algebraic equations. *Mathematical and computer modelling*, 19(12): 67–84, 1994.

S. Cvijić, M. Ilić, E. Allen, and J. Lang. Using extended ac optimal power flow for effective decision making. In *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE, 2018.

V.-Q. Do, D. McCallum, P. Giroux, and B. De Kelper. A backward-forward interpolation technique for a precise modeling of power electronics in hypersim. In *Proc. Int. Conf. Power Systems Transients*, pages 337–342, 2001.

H. Doi, M. Goto, T. Kawai, S. Yokokawa, and T. Suzuki. Advanced power system analogue simulator. *IEEE Transactions on Power Systems*, 5(3):962–968, 1990.

B. Donyanavard, A. M. Rahmani, A. Jantsch, O. Mutlu, and N. Dutt. Intelligent management of mobile systems through computational self-awareness. *arXiv preprint arXiv:2008.00095*, 2020.

D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

Y. Fu and M. Shahidehpour. Fast scuc for large-scale power systems. *IEEE Transactions on Power Systems*, 22(4):2144–2151, 2007.

A. Gebremedhin, F. Manne, and A. Pothen. What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Rev.*, 47(4):629–705, 2005.

A. Gebremedhin, A. Tarafdar, F. Manne, and A. Pothen. New acyclic and star coloring algorithms with application to computing Hessians. *SIAM J. Sci. Comput.*, 29:1042–1072, 2007.

A. Gebremedhin, A. Pothen, and A. Walther. Exploiting sparsity in Jacobian computation via coloring and automatic differentiation: a case study in a Simulated Moving Bed process. In C. Bischof, M. Bücker, P. Hovland, U. Naumann, and J. Utke, editors, *Advances in Automatic Differentiation*, pages 339–349. Springer, 2008.

A. Gebremedhin, A. Pothen, A. Tarafdar, and A. Walther. Efficient computation of sparse Hessians using coloring and automatic differentiation. *INFORMS Journal on Computing*, 21(2):209–223, 2009.

A. Gebremedhin, D. Nguyen, M. Patwary, and A. Pothen. ColPack: Software for graph coloring and related problems in scientific computing. *ACM Transactions on Mathematical Software*, 40(1):1–31, 2013.

A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2nd edition, 2008. URL `http://www.ec-securehost.com/SIAM/OT105.html`.

E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

D. Holmberg, M. Burns, S. Bushby, A. Gopstein, T. McDermott, Y. Tang, Q. Huang, A. Pratt, M. Ruth, F. Ding, et al. Nist transactive energy modeling and simulation challenge phase ii final report. *NIST Special Publication*, 1900: 603, 2019.

M. Ilic and R. Jaddivada. Introducing dymonds-as-a-service (dymaas) for internet of things. In *HPEC*, pages 1–9, 2019a.

M. Ilic and R. Jaddivada. Toward technically feasible and economically efficient integration of distributed energy resources. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 796–803. IEEE, 2019b.

M. Ilic and R. Jaddivada. Unified value-based feedback, optimization and risk management in complex electric energy systems. *Optimization and Engineering*, pages 1–57, 2020.

M. Ilic and J. Lang. Netssworks software: An extended ac optimal power flow (ac xopf) for managing available system resources. In *AD10-12 Staff Technical Conference on Enhanced Power Flow Models Federal Energy Regulatory Commission*, 2010.

M. Ilic, M. Crow, M. Pai, et al. Transient stability simulation by waveform relaxation methods. *IEEE Transactions on Power Systems*, 2(4):943–949, 1987.

M. Ilic, R. Jaddivada, X. Miao, and N. Popli. Toward multi-layered mpc for complex electric energy systems. In *Handbook of Model Predictive Control*, pages 625–663. Springer, 2019.

M. D. Ilić. Dynamic monitoring and decision systems for enabling sustainable energy services. *Proceedings of the IEEE*, 99(1):58–79, 2010.

M. D. Ilić and R. Jaddivada. Multi-layered interactive energy space modeling for near-optimal electrification of terrestrial, shipboard and aircraft systems. *Annual Reviews in Control*, 45:52–75, 2018.

M. D. Ilic and D. R. Lessard. Distributed coordinated architecture of electrical energy systems for sustainability. Under submission to Nature Energy.

M. D. Ilić, R. Jaddivada, and X. Miao. Scalable electric power system simulator. In *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE, 2018.

M. D. Ilic, R. Jaddivada, and M. Korpas. Interactive protocols for distributed energy resource management systems (derms). *IET Generation, Transmission & Distribution*, 14(11):2065–2081, 2020.

R. Jaddivada. *A unified modeling for control of reactive power dynamics in electrical energy systems*. PhD thesis, Massachusetts Institute of Technology, 2020.

J.-Y. Joo and M. D. Ilić. Multi-layered optimization of demand resources using lagrange dual decomposition. *IEEE Transactions on Smart Grid*, 4(4):2081–2088, 2013.

M. Lauer, R. Jaddivada, and M. Ilić. Secure blockchain-enabled dymonds design. In *Proceedings of the International Conference on Omni-Layer Intelligent Systems*, pages 191–198, 2019.

N. LaWhite and M. D. Ilic. Vector space decomposition of reactive power for periodic nonsinusoidal signals. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 44(4):338–346, 1997.

E. Ostrom. A general framework for analyzing sustainability of social-ecological systems. *Science*, 325(5939):419–422, 2009.

K. Padiyar. *FACTS controllers in power transmission and distribution*. New Age International, 2007.

M. Pai, P. W. Sauer, and B. C. Lesieutre. Static and dynamic nonlinear loads and structural stability in power systems. *Proceedings of the IEEE*, 83(11): 1562–1572, 1995.

M. Papic, M. Vaiman, M. Vaiman, and M. Povolotskiy. A new approach to constructing seasonal nomograms in planning and operations environments at idaho power co. In *2007 IEEE Lausanne Power Tech*, pages 1320–1325. IEEE, 2007.

M. Pavella and P. Murthy. Transient stability of power systems: theory and practice. 1994.

P. Penfield, R. Spence, and S. Duinker. A generalized form of tellegen's theorem. *IEEE Transactions on Circuit Theory*, 17(3):302–305, 1970.

M. Puschel, J. M. Moura, J. R. Johnson, D. Padua, M. M. Veloso, B. W. Singer, J. Xiong, F. Franchetti, A. Gacic, Y. Voronenko, et al. Spiral: Code generation for dsp transforms. *Proceedings of the IEEE*, 93(2):232–275, 2005.

J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. *Acm Sigplan Notices*, 48 (6):519–530, 2013.

D. Rose. *Sparse Matrices and their Applications: Proceedings of a Symposium on Sparse Matrices and Their Applications, held September 9 10, 1971, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, the National Science Foundation, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department.* Springer Science & Business Media, 2012.

B. Stott. Review of load-flow calculation methods. *Proceedings of the IEEE*, 62 (7):916–929, 1974.

B. Stott, J. Jardim, and O. Alsaç. Dc power flow revisited. *IEEE Transactions on Power Systems*, 24(3):1290–1300, 2009.

J. U. Thoma. *Introduction to bond graphs and their applications*. Elsevier, 2016.

J. S. Thorp, C. E. Seyler, and A. G. Phadke. Electromechanical wave propagation in large electric power systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 45(6):614–622, 1998.

W. Tinney, V. Brandwajn, and S. Chan. Sparse vector methods. *IEEE transactions on power apparatus and systems*, (2):295–301, 1985.

B. van Merriënboer, A. Wiltschko, and D. Moldovan. Tangent: Automatic differentiation using sourcecode transformation in Python. In *31st Conference on Neural Information Processing System*, 2017.

V. Vassilev, M. Vassilev, A. Penev, L. Moneta, and V. Ilieva. Clad – Automatic differentiation using Clang and LLVM. *Journal of Physics: Conference Series*, 608(1):012055, 2015.

G. C. Verghese, I. Perez-Arriaga, and F. C. Schweppe. Selective modal analysis with applications to electric power systems, part ii: The dynamic stability problem. *IEEE Transactions on Power Apparatus and Systems*, (9):3126–3134, 1982.

M. R. Wagner, K. Bachovchin, and M. Ilic. Computer architecture and multi time-scale implementations for smart grid in a room simulator. *IFAC-PapersOnLine*, 48(30):233–238, 2015.

J. Wyatt and M. Ilic. Time-domain reactive power concepts for nonlinear, nonsinusoidal or nonperiodic networks. In *IEEE International Symposium on Circuits and Systems*, pages 387–390. IEEE, 1990.

X. Zhang, F. Soudi, D. Shirmohammadi, and C. S. Cheng. A distribution short circuit analysis approach using hybrid compensation method. *IEEE Transactions on Power Systems*, 10(4):2053–2059, 1995.