Modeling and Simulation of Electric Power Systems with Large Shares of Renewable Energy

by

José Daniel Lara

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Energy and Resources

and the Designated Emphasis

in

Computational and Data Science and Engineering

in the

Graduate Division

of the

University of California, Berkeley

Spring 2022

Modeling and Simulation of Electric Power Systems with Large Shares of Renewable Energy

Abstract

Modeling and Simulation of Electric Power Systems with Large Shares of Renewable Energy

by

José Daniel Lara

Doctor of Philosophy in Energy and Resources

and the Designated Emphasis in

Computational and Data Science and Engineering

University of California, Berkeley

Professor Duncan S. Callaway, Chair

To date, electric power systems are the most important technological achievement in the energy sector. Interconnected power systems require a careful operation to maintain supply stability and maintain efficient energy usage. Amid this complexity, two critical global transitions are needed: (1) moving away from greenhouse gases which negatively contribute to climate change; and (2) moving away from fossil fuels, which currently support transportation and heating systems.

Computer simulations are the only tools available to do the necessary research and development required to accomplish these goals. However, the practices and tools used by the electric power industry were developed for operational situations that are vastly different from current challenges for energy transition. Variable Renewable Energy uncertainty, fast varying resources, and fundamental changes to the energy transformation physics require an update to the operational and simulation practices.

This work started focusing on the simulation of future systems, and by the time it was written, it had become a dissertation on the practices needed for today's system. We discuss the requirements needed to improve scientific computing practices and conduct simulations more systematically when developing new operation models. The work has designed and implemented three software tools to develop simulations: the first includes `PowerSystems.jl` — a software package that handles the data ingestion and processing required for simulations across multiple time scales. The second tool includes `PowerSimulations.jl` which resolves issues of model-limited choice when implementing operations simulations of large interconnected systems. The third tool includes `PowerSimulations.jl`, which is based on a formalization of the simulation of power system operations and provides a scalable computational platform. Third, `PowerSimulationDynamics.jl` which concentrates on the modeling of power systems dynamics with a strong focus on integrating Inverter Base Resources. `PowerSystems.jl` and `PowerSimulations.jl` both enable the development of a novel Automatic

Generation Control model that can assess reserve deployments. Both tools are used to develop a Markovian Graph approach when integrating probabilistic forecasts into operators' risk assessments.

Finally, this dissertation investigates the simulation techniques that are needed for system dynamics, and challenges the applicability long-held simulation practices. It seeks to uncover the theoretical elements of simulation needs for the future grid. Results from `PowerSimulationDynamics.jl` confirm this dissertation's hypotheses, and demonstrates the effectiveness of the simulation approach in replicating existing models. Furthermore, the findings in this thesis critically showcase the capabilities to simulate dynamics systems accurately using model transformations that reduces computational complexity without loss of accuracy.

To all of my friends, family and colleagues
who gave me support and encouragement over the years.

Contents

List of Figures

List of Tables

List of Source Code

List of Acronyms

ACE	Area Control Error
AD	Automatic Differentiation
AGC	Automatic Generation Control
AML	Algebraic Modeling Language
AS	Ancillary Services
ASAI	Average Service Availability Index
AVR	Automatic Voltage Regulator
BDF	Backwards Differentiation Formula
BDS	Band Depth Score
CIM	Common Information Model
CV@R	Conditional Value-at-Risk
DAE	Differential Algebraic Equation
DAUC	Day-ahead Unit Commitment
ED	Economic Dispatch
EMS	Energy Management System
EMT	Electro-magnetic Transient
ERCOT	Electric Reliability Council of Texas
ESS	Energy Storage System
FRR	Frequency Regulation Reserve
HAUC	Hour-ahead Unit Commitment
IBR	Inverter-based Resource
IEC	International Electrotechnical Commission
ISO	Independent System Operator
LFC	Load Frequency Control
LP	Linear Programming
MBD	Modified Band Depth Score
MC	Monte Carlo
MILP	Mixed-Integer Linear Programming
MP	Mathematical Program
NLP	Nonlinear Programming
MBDS	Modified Band Depth Score
MDP	Markov Decision Problem

NSRDB	National Solar Radiation Database
NWP	Numerical Weather Prediction
ODE	Ordinary Differential Equation
OPF	Optimal Power Flow
PCM	Production Cost Model
PF	Power Flow
PDF	Probability Density Function
PV	Photovoltaic
PWL	Piece-wise Linear
RMSD	Root Mean Square Difference
RUC	Robust Unit Commitment
PI	Proportional Integral
PLL	Phase-locked Loop
PSS	Power System Stabilizer
QSP	Quasi-Static Phasor
RMS	Root Mean Square
SACE	Smooth Area Control Error
SFA	Shifted Frequency Analysis
SO	Stochastic Optimization
SDDP	Stochastic Dual Dynamic Programming
SPT	Singular Perturbation Theory
SRF	Synchronous Reference Frame
SUC	Stochastic Unit Commitment
UC	Unit Commitment
VS	Variogram Score
VRE	Variable Renewable Energy
VSM	Virtual Synchronous Machine
WT	Wind Turbine

Preface

2.1 We make ourselves pictures of facts.
2.12 The picture is a model of reality.
2.223 In order to discover whether the picture is true or false we must compare it with reality.
2.225 There is no picture which is a priori true.
Ludwig Wittgenstein, Tractatus Logico-Philosophicus

The philosophical core in this work is to expand the old adage attributed to George E. P. Box: "*All models are wrong but some are useful*" and ask "*When is a model so wrong it leads us in the wrong direction and is no longer useful?*". Scientific experimentation is an interpretive method, and over time it refines the observations to build more accurate descriptions of reality. We make models based on understanding reality to approximate it and produce alternate realities. We make decisions and design objects that fit into our reality based on the alternates.

At the moment of writing this work, the capabilities of humanity to make models and simulations is beyond anything that any philosopher or scientist thought possible and will probably continue to increase. This new power implies that we need to learn new approaches to developing science collectively and at scales that are now beyond individual contributions. As such, modeling in computers does not consist simply of writing code, but it implies a broader "philosophical" view of reality that influences our thought process.

For over a century, power engineers have developed models of physical phenomena and gained an understanding of the systems' properties. However, the transition to renewable resources introduces changes that require understanding the information process for the control and operations of the physical system not just the physics. The expansion into the information space requires a new understanding for power engineers about what is a *model*? and what is a *simulation*?. This dissertation uses these abstract questions to develop concrete applications to power systems modeling and simulation with large shares of Variable Renewable Energy (VRE).

Wittgenstein's work focused on how language works to describe the world accurately. In answering the questions above, this dissertation had to carefully define concepts to improve the formulation of models and simulations. In retrospect, the code and the equations were the easy part, and language became a more considerable challenge. The dissertation is organized thematically between operations and dynamic simulations, given the distinct approaches and methods without diminishing the applicability of concepts around simulation.

Acknowledgments

Modern power systems research is, by definition, a multidisciplinary endeavor that requires teamwork. One of the proudest achievements of the last five years is having a long list of collaborators with whom I got to work. The work compiled in this dissertation would not be possible without multiple collaborators at UC Berkeley and other institutions.Ph.D.

I am thankful to my supervisor at UC Berkeley, Duncan Callaway, who, through thick and thin, provided me with unwavering backing to continue with the mission of making open-source simulation software a core contribution during my Ph.D. This work would not have been possible without the unlimited support from Bri-Mathias Hodge, who first allowed me to work together and accepted to play the role of co-advisor in this work from NREL. I could not have asked for a better duo to guide me through the last five years. I appreciate the guidance from the readers Philip Stark and Seth Sanders, who provided me with precious feedback during the writing process.

The software packages part of this dissertation were only possible because of the contributions of the SIIP team at NREL. Nothing in this project would have been possible without our fearless leader Clayton Barrows who, aside from keeping the team moving forward, I have to thank you for making sure that I was able to finish my Ph.D. Daniel Thom, who had the patience to teach me how to be a better code developer, Dheepak Krishnamurthy, and Sourabh Dalvi.

The work in this dissertation includes contributions developed under the SUMMER-GO project. I am thankful to my collaborators, Kate Doubleday and Jeffrey Sward, for our work to help integrate probabilistic forecasts. Thanks to Oscar Dowson for all the collaboration with `SDDP.jl` and the integration of `JuMP.jl` into the software libraries.

In the last year, my research focused on developing the dynamics simulation capabilities of the software platform. First and foremost, I have to thank Rodrigo Henriquez for the long code sprints and discussions that allowed us to push the boundaries of dynamic simulations. I could not hope for a better friend to have gone down the rabbit hole of dynamic modeling. Thanks to the group around the Advanced Grid Modeling (AGM) project Scientific Machine Learning for Power System Acceleration. Thanks to Ciaran Roberts and Matthew Bossart for pushing over many months to develop better simulation tools and using our software.

This research was supported by Siebel Foundation Scholarship Class of 2021, the National Renewable Energy Laboratory (NREL), operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. This work was supported by the Laboratory Directed Research and Development (LDRD) Program at NREL. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government.

Chapter 1

Introduction

Power systems are the largest and most complex machines ever built. They are critical to the functioning of modern society. The effects of electricity generation on the environment – particularly concerns about climate change – have since the early 2000s driven an increase in the adoption of renewable energy. From an economic standpoint, technological advances in Photovoltaic (PV) panels and Wind Turbines (WTs) have made renewable energy a viable means of powering electricity systems. The technical challenge of enabling renewable energy resources to compete with fossil fuels is, from the practical standpoint, solved. The ongoing challenge is integrating these new resources into a system not built to adapt quickly to changes.

Nevertheless, these new energy sources also come with new challenges. The contemporary electric power network was designed to operate with thermal generation, and only minor variations in the short term [173]. To date, the electric power system has operated under the overarching presumption of total controllability concerning both resources and the predictability of the load. Nevertheless, changes in the primary energy sources used to generate electricity are not the only thing changing: the very principles of energy conversion facilitate the energy transition. Large electrical machines that generate electrical power through a rotating electromagnetic field are slowly being replaced with Inverter-based Resources (IBRs) resources that employ semiconductors to perform energy conversions utilizing high-frequency switching. The techniques to model, operate, and plan a power system fueled by fossil resources and subject to mild uncertainties did not evolve much until the 2010s [122] and since the area has become one of the most exciting research fields to innovate.

The inherent variability of renewable energy generation impacts various timescales that range from seconds to decades. Integration must be analyzed across timescales and domains accordingly [143]. Although statements about cross-timescales analysis are common, the increased technical and conceptual complexity of developing these simulations is under-estimated. With the addition of models to account for system behavior after faster timescales, the data storage, transfer and computation requirements also increase. Handling the added complexity requires a new conceptual framework from the first principles

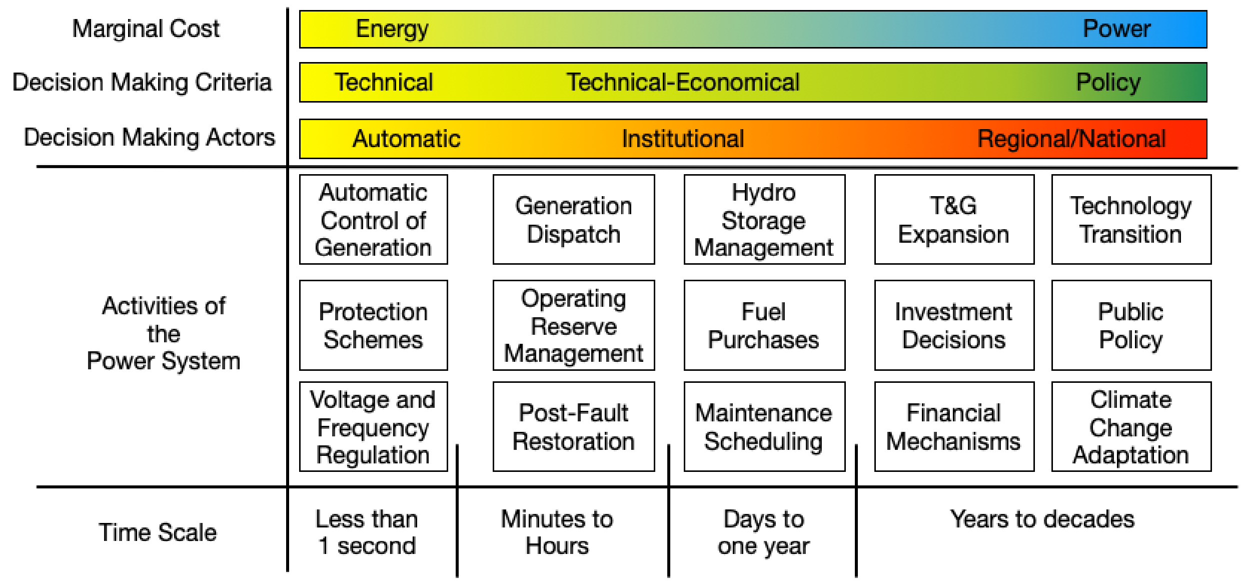


Figure 1.1: Power Systems Operations and Planning at different timescales.

point-of-view but also from the design perspective.

Figure 1.1 shows the different timescales in power systems operations and decision-making characteristics for each stage. In 2021, the U.S. National Academy of Sciences recognized the importance of understanding how the grid of the future will behave, and how operators and policy makers can ensure its continued reliability [125]. The capacity to improve the understanding of the system rests on improving the simulation capabilities necessary to build and test system integration of new devices and components demanded the energy transition. *This dissertation focuses on the development of novel simulation methodologies to assess the operations and control problems of the energy transition.*

The process of conducting a *simulation* entails acquiring knowledge about a system and employing an unambiguous, and possibly inexact representation of its characteristics. The representation of the system is known as the *model* means to enlighten experts about the behavior of the real system under certain conditions. Given a model, it is necessary to define a simulation *method* that can answer the research question through finding the solution to the model. This work identifies two major simulation categories needed to develop novel models and methodologies and critically improvements to scientific practice.

1. **Operations Simulations:** These simulations support decision-making on the scale of days to minutes and have limited representation of the system physics. The general objective of an operation simulation is to determine how the operate the system at a the minimum production cost while keeping certain reliability restrictions in place. Commonly these simulations employ discrete optimization models to find the solution of “optimal" operational decisions and have different formulation depending on the timescale requirements.

2. **Dynamic Simulations:** These simulations focus on representing the physical continuous time behavior of the system in the micro-second to minute timescales. They are generally used to study the "stability" of the system, broadly defined. These models are formulated as Ordinary Differential Equations (ODEs) or Differential Algebraic Equations (DAEs) and employ numerical integration techniques to find the trajectory of the system dynamics.

The use of simulations in power systems has evolved with the changing needs of the grid, the development of new tools, and increases in computational power. Formerly, the discipline has relied heavily on mathematical analysis of lower dimensional, deterministic systems, and presented limited computer simulation results. Due to increases in computational power, however, optimization-based models and computer simulations are standard tools used for research in systems of all scales——from bulk generation and transmission to micro-grids. The overarching contributions of this dissertation focus on the implementation of tools that facilitate using novel simulations techniques in accordance with the Principles of Scientific Computing. Each simulation category has very distinct methods, practices, and applications; hence, they need be treated differently, with a specific focus on the challenges brought by the energy transition.

Operation Simulations

Optimizing the operation of the power system is long-standing technical and research area dating to the early days of interconnected system and the works of Leon Kirchmayer [78]. Integrating Variable Renewable Energy (VRE) is a challenging problem from an operational perspective, given that the forecasting of VRE resources introduces higher levels of uncertainty where decision-making is concerned. Further, given that modern lifestyles rely heavily on electricity, failing to hedge the system against uncertainty can result in disruptive losses of load, increased costs of power curtailment, and even system collapse. In this work, the focus begins in the minutes-to-hour timescales where short-term imbalances and variations pose operational challenges. Some of these operational challenges include increases in ramping requirements that supply compensation flexibility for VRE changes; decreases in reserve quantities; and the integration of novel power generation resources like Energy Storage System (ESS).

Managing uncertainties in bulk electric power systems poses a well-recognized challenge from an operational perspective. Uncertainty is often crudely defined, which impedes the development of actionable and consistent practical approaches. This work, therefore, first defines uncertainty in a functional sense then proceeds to apply that definition to activities concerning the power system.

An effective treatment of uncertainty requires a clear distinction between inherent variability, often referred to as statistical uncertainty, and epistemic uncertainty due to limited knowledge about the process [28]. Figure 1.2 showcases which characterization of uncertainty this dissertation will adopt. On one end of the spectrum, there exist deterministic

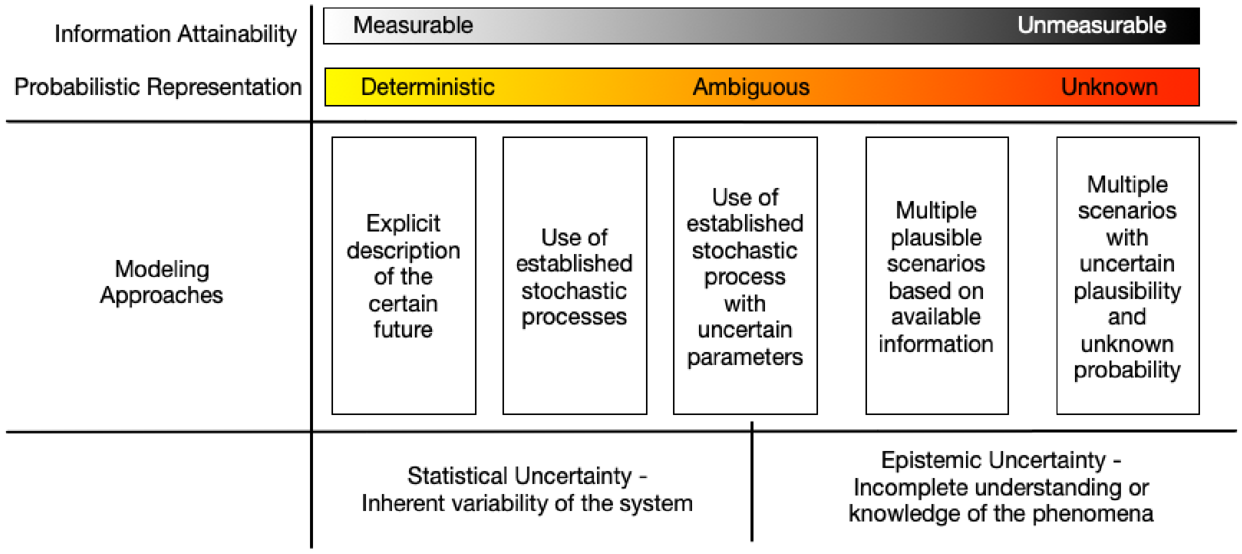

Information Attainability
Measurable
Unmeasurable
Probabilistic Representation
Deterministic
Ambiguous
Unknown
Modeling Approaches
Explicit description of the certain future
Use of established stochastic processes
Use of established stochastic process with uncertain parameters
Multiple plausible scenarios based on available information
Multiple scenarios with uncertain plausibility and unknown probability
Statistical Uncertainty - Inherent variability of the system
Epistemic Uncertainty - Incomplete understanding or knowledge of the phenomena

Figure 1.2: Characterization of uncertainty definitions.

processes that can be measured and have little or no statistical uncertainty; for example, load forecast with short look-ahead horizons. On the other hand, VRE output is more adequately characterized using probability models when available. Examples include wind power and solar average power forecasts. Extreme cases of uncertainty include the availability of long-term resources and load, or when the number of factors that determine long-term weather patterns, energy consumption, and possible technological transitions make the knowledge about the processes inherently incomplete. As a result of epistemic uncertainties, this dissertation will exclude planning problems from its discussion, since the impact of long-term unknowns reduces the usefulness of detailed structured simulation.

Dimensionality serves as one critical challenge when it comes to simulating power system operations. On the one hand, including uncertainty in operational problems helps with solving optimization problems. On the other hand, the assessment of operational models can no longer be done just for peak, mid and low load operating points. This increased dimensionality in the problem size and the number of simulation scenarios under consideration requires new simulation techniques.

This dissertation examines three distinct aspects of VRE in power systems operations and the management of the uncertainty. First, on the computational challenges, it introduces the data model that fullfills the needs for modern computational experiments to test and develop novel operation approaches. Second, on the development of a model to assess the deployment of reserves in a system with large shares of VRE. Third, a specific focus on the operational challenge of short-term risk assessment employing multi-stage stochastic optimization models to integrate probabilistic forecasts.

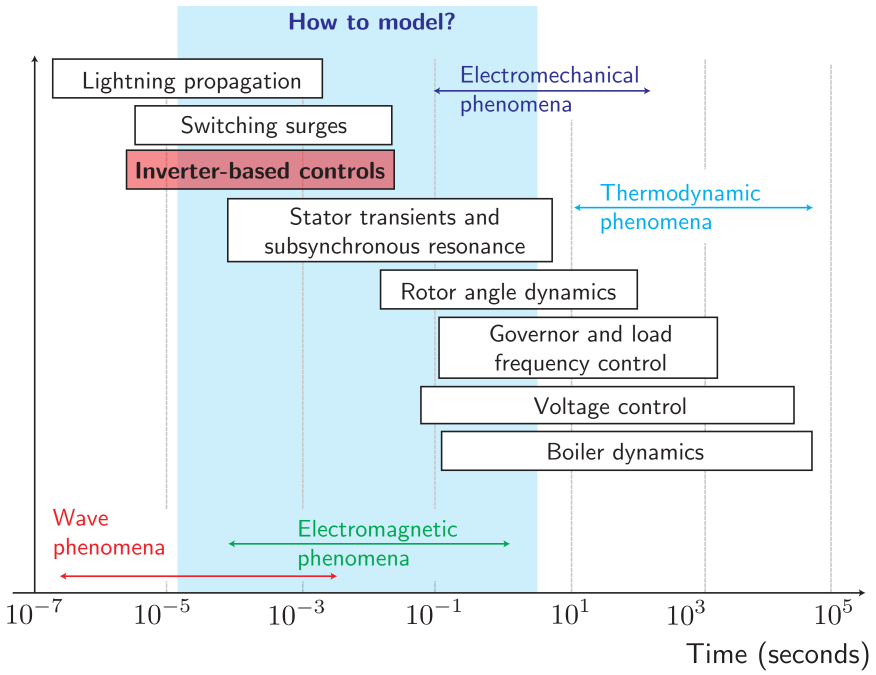


Figure 1.3: Dynamic timescales of power systems dynamics [60].

Dynamic Simulations

The increasing integration of generation sources via power electronics continues to change the underlying dynamic behavior of power systems. It is generally agreed that new dynamics in the controls of IBRs change modeling requirements for system-wide stability studies that rely on time-domain simulations [133, 60, 120]. In power systems dominated by synchronous generators, physical phenomena, such as magnetic fluxes, electro-mechanics, mechanical control reaction times, or thermo-dynamic processes, can drive dynamic behavior. The control logic associated with synchronous generators is commonly tuned to the timescale of the relevant processes, creating a *natural* separation between the dynamic behaviors attributable to physics and controls. On the other hand, IBR dynamics are dominated by their controls, including modulation, Phase-locked Loops (PLLs), voltage, current, and power controllers. Therefore, the practical requirements of control design — which often include cascading PID control – defines the relationships between timescales. In fact, interactions at higher frequencies have recently been recognized with a new stability category [60], highlighting the exigent need to revisiting our understanding of system dynamics with large shares of IBRs.

Dynamic analysis of power systems mainly include two computational numerical methods: (1) Time-domain simulations; and (2) small-signal stability analysis. For decades, there has been a focus on reducing time-domain simulations' computational complexity through model order reductions and averaging methods. Some of the modeling practices and analytical assumptions currently in use derive from singular perturbation theory from the 1980s. Seminal papers reveal that a simplified model is still a valid representation of certain system dynamics under the premise of timescale separation. With the addition of IBRs to the system, however, the validity of these simplifications have been challenged.

Figure 1.3 highlights the frequency ranges of dynamic phenomena and modeling requirements used to capture the system's behavior. IBR controls consider multiple models such as power electronics representation, phase-locked loop, voltage and current controllers, and outer-loop controllers that have wide timescale ranges and complicate the determination of which level of model complexity has to be used.

New demand for larger scales and greater details in modeling requires changing how we approach developing simulation experiments. One critical challenge in dynamic simulations is the addition of phenomena that heretofore have not been included, imposing new requirements on the solvers and resulting in computational complexity. Increasing the level of detail in dynamic simulations requires new approaches to obtaining reliable system assessments and better computational tools. The power systems sector must accelerate the adoption of improved algorithms and hardware in order to cope with the required depth of analysis when studying the integration of VRE.

This dissertation focuses on two different challenges with regard to the integration of IBR in power systems simulations. The first challenge includes the developing a cohesive set of definitions and clarifications on the scope of simulation methodologies and models commonly used in power systems. Second, on the development of simulation tools and models that incorporate dynamic behavior across multiple dynamic timescales and provide computational methods to solve these models at scale.

The rest of the dissertation is organized as follows. Chapter 2 develops a framework for simulation experiments in the power systems sector with a focus on the development of definitions for assembling the simulation model, and follows with a discussion about the software implications. The package `PowerSystems.jl` is introduced in Chapter 2 with a detailed description of the software architecture and use cases. Chapter 3 introduces the simulation package `PowerSimulations.jl`, which focuses on operation simulations; it includes a series of definitions to formalize the concept of operation simulations and cases of the package. The notable use case is in the development of an AGC model to study the deployment of reserves in systems at large scale. Chapter 4 describes the use of a multi-stage stochastic risk assessment method with a Markovian representation of renewable power and the development of an estimation method for the transition matrix of the Markov representation of uncertainty. Chapter 5 discusses the theories and models that have contributed to dynamic system simulations, and starts with a detailed review of the fundamental methodologies that formulate dynamic simulations models and a taxonomy of existing approaches. The package `PowerSimulationsDynamics.jl` is introduced in Chapter 5 which implements a solver agnostic and modular approach to simulating systems with IBRs including the software architecture and the results of the model validations. Finally, Chapter 6 provides the reader with concluding remarks and discusses next steps in the area of system simulation with large shares of VRE.

Chapter 2

Design of simulation experiments in power systems

Scientific computing has emerged as a field that studies and promotes the application of principles such as reproducibility, transparency, and accuracy to data analysis and experiments that are carried out using computer simulations. Although scientific computing has benefited from notable contributions regarding the theory and practice of reproducibility [166, 91, 5, 38], with crucial advancements in the systematic development of computational experiments for model and algorithm testing via simulations [10, 79, 155], its adoption is not widespread. The field of scientific computing includes a broad array of definitions and practices. The relative importance and specific application of any one of these varies according to discipline [138, 6]. Here, we review definitions and applications of scientific computing principles in power systems versus other disciplines in the context of the scientific process.

It stands to reason that enhancing scientific computing would carry significant implications for power systems, since computational experiments afford almost the only option for engineers to conduct research about the operation of large-scale power grids. Computing is a fundamental tool for conducting operations research [127], and, by extension, power systems operation research. Improving the definitions and practice of scientific computing principles thus also stands to benefit both of these fields.

However, there is a high setup cost associated with developing simulations for large-scale experiments. For this reason, researchers resort to using industrial tools once their systems of interest grow larger than hundreds of buses. However, these proprietary models and algorithms are not openly available; this means that using them and replicating the results obtained by other researchers who use them require money for licenses. The common practice of relying on commercial tools reduces the innovation power systems researcher. When commercial tools aren't available, researchers need to re-implement models, develop their own data sets, and handle the integration libraries. These practices mean that scientific reproducibility is often either not achieved or limited to "code sharing."

Thanks to the advent of code-sharing repositories, broad access to software development tools, and the popularity of open-source interpreted programming languages such as R, Julia, and Python developing, reproducible computational tools have now become easier. In this respect, open-source tools facilitate scientific computing for power systems research and despite challenges, several open-source efforts are successful in the community, reducing the barrier to entry for those seeking to perform reliable and high-quality research. For example, `MATPOWER` [205] is a widely used MATLAB-based[1] tool to perform steady-state analyses such as power flow, continuation power flow, and optimal power flow. `OpenDSS` is used for multiple distribution systems analysis and both the source code and the Delphi compiler are open source. In the field of transient simulations, the MATLAB-based tool `PSAT` [114] provides electro-mechanical simulations with multiple models. More recently, the Python-based hybrid symbolic–numeric simulation tool `ANDES` [29] has become available for QSP simulation. Many other tools in power systems exist with different levels of "openness" and capability to reproduce the experiments. However, the trend is towards increasing openness and easiness to reproduce the results from the publications.

However, having open-source tools is not sufficient for creating a consistent culture of reproducible computational experiments. Without a deliberate effort to develop a platform that can be re-used by researchers, we risk limiting the scope and scale of power system research or unnecessarily duplicating research efforts.

This chapter presents a power systems framework that systematically applies scientific computing principles to the development and validation of simulation models and development of a data model to enable better practices. The relevant contributions in this chapter are:

1. A template to facilitate the development of simulations and computational experiments.

2. A consistent set of scientific computing definitions, practices, and implementation details to facilitate the application of scientific computing to power systems operations research.

3. Discusses the implementation of an application-agnostic data model focused on providing a workflow consistent with scientific computing practices.

The chapter is organized as follows: Section 2.1 presents a description of scientific computing practices, definitions that indicate their relevance to power systems, and a discussion of the modeling language choice. The need to design a computational experiment is discussed in Section 2.2, which focuses on defining variables, data, models, and metrics. Section 2.3 showcases a development pipeline required to implement computational experiment in accordance with scientific computing principles. Section 2.4 shows the design and development of `PowerSystems.jl`, a library specifically developed to handle data modeling in power systems simulations. Finally, we present our conclusions in Section 2.5.

[1]The source code of Matlab-based tools is open source but the Matlab engine is a commercial product

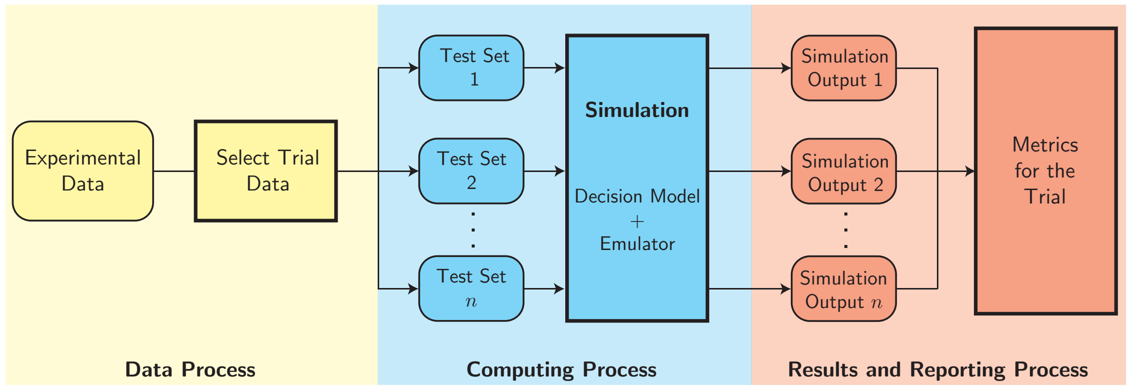


Figure 2.1: Steps required for conducting a single trial in a simulation experiment

2.1 Scientific computing practices

Establishing new scientific knowledge relies on (1) the *reproducibility* of the experiments and (2) the *validity* of conclusions derived. Although there are semantic distinctions across fields [138, 6], we define "reproducibility" as the ability to produce identical results with repeated experiments while running the same software and using the same input data and simulation settings [55, 139, 153]. "Validity" also has a variety of definitions depending on the scientific context. Here, we refer to two types: (1) internal validity, or consistency between simulation results and the experimental system, and (2) external validity, which might also be called "generalizability," or the ability to derive inferences about how specific results or relationships may persist across variations in settings and systems [138]. These definitions apply even when the experiments are carried out through simulation in computational environments.

Interest in scientific computing as a field began with a focus on the precision aspects of computational numerical methods and the conditions required for algorithms to obtain consistent results [63]. The relevance of computer simulations for science was recognized early in the 1960's when by Harlow and Fromm [59] who recognized that computer experimentation allows the investigation of phenomena "innacessible" to direct study and describe computing as the third pillar of science complementing theory and experimentation. Oberkampf *et.al* provides a detailed historical account of the evolving nature of scientific computing [130] showcasing the pivotal contributions from the fluid dynamics research community. These definitions of scientific computing are rooted in concerns about computing process precision and model calibration. However, as the use of computers for scientific research has expanded, so has the scope of scientific computing.

Later contributions focused on the *access* that researchers have to the underlying code to reproduce computational experiments. Developing open-access software for mathematical computations like `WaveLab` focused on providing open source alternatives to statistical analysis [23] is one of the notable early attempts. `WaveLab`, an open source tool for Wavelets, authors grappled with issues like code sharing[2], versioning. These challenges nowadays are resolved with widespread access to the internet and the invention of versioning control libraries like `Git`. Although the technological solutions for the broader application of reproducible scientific research has improved, the integration of good practices into the design of experiments remains limited. Donoho (one of the original authors of `WaveLab`) and Stodden provide a review of the evolution of reproducible computational research [37] up to 2015.

In the applications discussed in this chapter and through out the dissertation, the worries around numerical precision are less critical and the focus is more about the availability to replicate the experimental process. Researchers in power systems usually employ third-party libraries to solve the optimization and dynamic models, and therefore place the responsibility of numerical precision lies with the solver provider. The only exception to this "outsourcing" of the numerical precision responsibility is in areas of power systems research focused on the development of algorithms (e.g., [176]). Reproducibility concerns in power systems start with the availability of the code and the extensive use of commercial software to conduct research. Beyond code-sharing, this chapter focuses in the capability to recreate the computational experiment.

Some of the most vocal advocates in the scientific community for the principles of reproducibility and validity are from the fields of medicine [139] and computational biology [38], which strive to derive consistent, verifiable results from field or laboratory experimental data. Power systems researchers would do well to take lessons applicable to modeling and simulation, since the complexity of the power grid requires the use of numerous assumptions and simplifications [123].

Resources, time, and a certain amount of luck are needed to attempt to reproduce experiments that are not accompanied by code. However, computational reproducibility requires more than access to the code used to run computational experiments. While documentation alone helps, it is often insufficient to achieve full reproducibility. The recent focus of scientific computing on institutionalizing "best practices" has included themes directly relevant to effectively sharing code and documentation. This includes guidance on code readability, the use of version control tools, and code sharing through platforms like Github. These basic tenets have come to serve as the baseline standard for "good" code [192]. In part, these items are not practiced in power systems research because most researchers are not trained in the "why" and "how" of reproducible code [13]. In many cases, power systems researchers have no intention of re-using code following publication.

In summary, according to the scale of reproducibility proposed by [139], most contributions in this field thus fall in the category of "publication only" and can at best be regarded as "reviewable," according to [138]. Implementation details are often deemed irrelevant, even when (as in many cases) such details are crucial to understanding the final results [165]. By contrast, consider the open-source genetics software `BioConductor` [51], whose standard of reproducibility has generated more than 10,000 citations showing the value

[2]The authors describe issues with the use of CD-ROMS for results sharing

to the field of having certain common scientific practices. It has no equivalent in power systems, although researchers in this field do occasionally apply principles of scientific computing to specific areas, such as the use of Optimal Power Flow (OPF) benchmark cases for the development of new formulations [27].

The Julia programming language

Choosing a programming language for simulations in power systems might seem unimportant. However, the choice can have significant impacts on reproducibility and scalability. Modeling package developers face the challenge of choosing a programming language for large-scale applications that balances extensibility with performance. Implementations in compiled languages like C or C++ are difficult to extend, develop, maintain and are not suitable for interactive work. On the other hand, implementations in interpreted languages can be slow to parse and run for medium to large power systems. Additionally, many of the existing extensions of power systems data models use languages that are not strictly typed, which can make code development error-prone and have poor performance.

Scripted programming languages usually do not implement high-performance code natively. High-performance applications require low-level compiled languages to handle large-scale computations. For instance, there are many identical programming interfaces but distinct implementations of the NumPy library for linear algebra in Python. The same applies to many other large-scale numerical libraries such as `LSODA`, `Sundials`, or `PETSc`. These well-established high-performance numerical libraries are hard to extend. Usually, they have dependencies on other compiled libraries such as `BLAS`, `LAPACK`, or `SuiteSparse`, adding additional complexity.

Julia is a dynamically typed programming language developed by Bezanson *et al.* [16], intended to bridge the gap between scripting languages such as Python or MATLAB and high-performance languages such as C++ or Fortran. Julia uses a Just-in-Time (JIT) compiler based on a Low-Level Virtual Machine (LLVM) and incorporates some essential features from the beginning of its design, such as excellent support for parallelism and GPU programming, and it features a practical, functional programming orientation. Julia can generate the same assembly code as C/C++ with the convenience of the scripted languages.

Besides the language's computational capabilities, the Julia environment itself provides several features that facilitate the development of large-scale dynamic models following scientific computing practices. Importantly, it addresses the shortcomings of other scripted languages, such as dependency handling. These capabilities make Julia an excellent match for scientific computing challenges in the power systems community. Critical examples of Julia's good environment include:

- A comprehensive package manager that enables reproducibility among researchers by setting up the exact package and binary versions. Integrating "Artifacts" into the package manager, Julia can include information that is not related to Julia packages, including platform-specific binaries or data sets. This allows researchers to pack all the experiment's requirements in a reproducible environment that can be executed in multiple platforms. This mechanism is also used to provide reproducible binary dependencies for packages built with `BinaryBuilder.jl`.

- Julia is a flexible language that has the convenience of scripting languages without the performance penalty. As a JIT-compiled language, Julia can be as fast as C or C++ and easier to master [16]. Multiple dispatch is crucial to the effectiveness of Julia as a programming language for scientific software. Associating methods with generic functions rather than the type they operate on is a highly effective strategy for code re-use and extension. This capability promotes the implementation of generic interfaces for custom models without requiring changes to the source code. Multiple dispatch is the core concept used to allow extensibility in the modeling packages described in this dissertation.

- An expressive Algebraic Modeling Language (AML) `JuMP.jl` [45] that allows the development of large scale optimization models. For the modeling of operation models, the Julia ecosystem also offers a large variety of solver interfaces powered by `MathOptInterface` [90] which allows the developer to develop models that use both open source and commercial optimization solvers. The interface provided by `JuMP.jl` supports the formulation of Linear Programming (LP), Mixed-Integer Linear Programming (MILP), Nonlinear Programming (NLP) and conic optimization models. This flexibility enables benchmarking and testing different solvers depending on the developer needs. `JuMP.jl` was used extensively when developing the operations simulation package described in Chapter 3.

- A suite of packages for the numerical solution of differential equations. `DifferentialEquations.jl` [148] offers over 30 different numerical solvers for ODE/DAE systems including both explicit (via mass matrices) and implicit representation. Some solvers, including classic algorithms and novel ones from recent research, can outperform the "standard" C/Fortran methods. In a nutshell, `DifferentialEquations.jl` allows solving differential equations using different algorithms (even from other packages or languages) by providing a common interface. This makes it possible to exchange the algorithm layer with a single line of code. This feature enables benchmarking and testing different solution methods according to the specific differential equations problem requirements.

- Several computations from dynamics to optimization, rely on computing the Jacobian of a non-linear systems of equations. Usually, derivatives are computed via finite differences, but that is inefficient due to the high number of evaluations of the system function, and furthermore it results in approximate results due the finite small value of the step. Automatic Differentiation (AD) is a method to compute exact derivatives given only the function opposed to a symbolic approach that requires providing structural information. Julia provides a large host of methodologies to obtain Jacobians and Hessians. For instance, `ForwardDiff.jl` [150] and `ReverseDiff.jl` are the most common established packages to perform AD to compute derivatives in Julia [99]. Both of these methods generally outperform non-AD algorithms in both speed and accuracy.

2.2 Designing the experiment

In this section, we contribute a logical process for designing power systems computational experiments based on general practices from empirical research that include best practices to allow reproducible results. In a simulation context, the experiment consists of varying the inputs and executing trials of the *simulation processes* to generate a sample of outputs. These samples are later used to compute relevant performance metrics or summary statistics. Figure 2.1 shows a breakdown of the experimental process organized into three different sub-processes: data, computational modeling, and results. These steps follow the best practices in scientific computing to achieve reproducibility and validation. Each of the sub-processes presents the researchers with many design choices. However, in this section, we review exemplary practices in the literature to guide researchers in tackling common issues and following best practices. We emphasize that using this framework supports a robust evaluation of operation models and provides a clear way for reviewers and readers to assess conclusions conditional on the researchers' experimental setup.

In [155], the authors define a simulation experiment as sampling the space of input variables over trials to characterize a system in the space of output variables. Exhaustive sampling is a viable strategy when the input space is small. However, power systems operational research typically features complex and large input spaces that include time series, network configurations, demand levels, and parameters of economic and physical subsystems. The descriptions of the different sampling spaces motivates a sub-classification of the data inputs which follow the general practices from empirical methods to help inform experimental design.

In our review, we find that a formal design approach is rarely used in power systems simulation. Researchers tend to implicitly focus on reporting a few key simulation results and over-specify the case data and the experiment parameters, reducing the external validity of conclusions.

Data

The experimental data specification requires the researcher to select system parameters, a method for the selection and use of trial data, and to define the *test sets* for the experiment. A *test set* is a collection of inputs used in the simulation that produces a corresponding output. The data defining a test set are organized hierarchically with a single set of *experiment*

Table 2.1: Data requirements and best-case selection process

Input Type	Description	Selection Considerations	Examples
Experiment Parameters	All data held constant throughout the simulations, defining the scope of the experiment	Enables generalization of the experiment to other cases	Test network, cost functions, boundaries of the confounding variable space
Confounding Variables	Data varied across trials, used to test robustness of results	Unbiased sample coverage of the relevant space	Forecast and realization time series, reserve requirements
Independent Variables	Variables compared within each trial, primary objects of study	Isolation of variable of interest, including a control or null case	Operation models, forecast accuracy, renewables penetration

parameters that define the scope of the experiment, a number of sample sets of *confounding variables* that delineate a particular trial, and, for each trial, an instance of each of the *independent variables* which are the primary objects of study, as well as a hypothesized impact on the outputs. Table 2.1 shows the different types of data input along with descriptions and selection considerations.

The validity of results depends on the researchers' choices when classifying and selecting these data. The more general and representative of other cases the experimental parameters are, the higher the external validity, whereas the more robust the sampling of confounding variables is, the higher the internal validity. The simulation settings such as output interval, algorithm tolerances, and convergence criteria are also part of the experiment parameters.

Confounding variables can be sampled either randomly or deterministically: sensitivity analysis, selecting sets of representative or extreme cases, and Monte Carlo simulation are all conventional methods [155]. In power systems operational research, independent variables often include (or are directly) the operation models themselves; e.g., the alternative UC models in [185, 197] or the demand response recourse strategies in [134]. In power system operation simulations, the confounding variables are often sets of time series data, initial system state, renewable penetration, or the test system.

When selecting data, there exists a tension between using real, benchmark, and synthetic data. Real systems and real data sets are often attractive because they have better internal validity. However, using any proprietary or privileged data can undermine reproducibility. Standardized test systems are by definition more reproducible and should yield more comparable results across studies, but the prevalence of "modified IEEE systems" show that these often do not capture all relevant features and there is a need for comprehensive test data sets. Synthetic network generation algorithms address some of these shortcomings [17], but they do not eliminate the need for modification with time series and the addition of additional devices like storage or demand response. In these cases, the modification process must be transparent and unbiased, which can be achieved by providing all the details when defining the modifications necessary. The best practice is to make the modifications programatically and sharing the code used to generate the data sets. Capturing uncertainty in synthetic data sets, particularly in forecasts and realizations, is a critical component in modern operations; [146] provides an overview of modeling uncertainty across parts of the power system. Regardless of the data source or generation method, access to and interpretation of the data used is necessary to comply with standards of reproducibility and transparency. To our knowledge, only the recent update to the RTS-96 data set [7] complies with these requirements. Further, scientific principles must be followed by refraining from "cherry-picking" data to produce favorable results. Adherence to the practices discussed in this section and providing details about the data selection processes can make it harder to obscure unscientific practices.

Operation simulation models

In power systems research, computational models are both the subject of the research and the experimental apparatus for evaluating research, often leading to the conflation of different models within the experiment. In practice, most experiments are used to demonstrate the performance of a proposed *decision model* for taking an operational action. In our framework, this makes the decision model an independent variable. Following the process in Fig. 2.1, it is necessary to define at least one alternative test set (i.e., a control) in the form of a baseline decision model, often representing a heuristic or current practice. To do this, and to ensure a valid representation, it is critical to delineate a *decision model* from an *emulator model* used as a representation of the real system defined as follows:

- **Decision Model:** The model used to obtain the desired system operation behaviour; the model generates set points or policies used to drive the devices in the system.
- **Emulator Model:** A model that mimics a specific real-world behavior of the electric power system; the model produces outputs that resemble the system performance when operating under the policies resulting from the decision models.

The emulator yields the performance metrics to compare performance between decision models and should be consistent across all test sets and trials.

In some cases, where the contributions comprise a combination of operation models and solution algorithms (e.g., [124, 203]), it is important to have a clear distinction between the modeling aspects and the algorithmic ones. In such cases, details of the model and implementation of the algorithm should be clear and reproducible. For instance, update rates, constants, tolerances, and heuristics need to be clearly stated.

The results commonly reported in publications consist of the decision model output and not showing whether the decision model has the hypothesized effect on the system. The emulator model is intended to enable the evaluation of the decision model's performance in terms of the system's behavior conditional on the decision output. Not having an emulator in the simulation process is particularly problematic in operations models intended to handle uncertainty, such as SUC and Robust Unit Commitment (RUC). It also complicates matters when the decision model must make simplifying assumptions for the research to be computationally tractable. In [177], the authors explicitly point out the limitations of interpreting the results of their decision model and propose a modification to make the model more realistic. However, the standard should not be that a decision model represents reality with perfect accuracy; rather, the aim is to select a reasonably accurate emulator to capture the relevant system's performance.

An implicit model separation between decision model and emulator has been done in many highly cited works [134, 185, 177], though there are some exceptions [197]. Other examples are most readily found in receding and shrinking horizon model predictive control, often employed in microgrid Energy Management System (EMS). The stochastic EMS work in [135] uses an actual physical test system, while [168] exemplifies the computational approach using `OpenDSS` as the emulator, and [85] uses a Monte Carlo (MC) approach to test the robustness of the control.

Based on the aforementioned literature, we generalize three main characteristics that the emulator should possess to be used as the validation test-bed in simulations:

1. The emulator should be on the same or a faster time scale compared to the decision model under study, and should capture the phenomena that are significant to these time scales and the study. There needs to be a logical connection between the chosen emulator and the decision operation model.

2. The system representation should include as much detail as is necessary to test the effect of simplifying assumptions used to make operation models' tasks tractable. This often requires additional data and trials.

3. The emulator's time series realization data *must* be distinct from forecasts used in the decision model. The use of forecasts within decision models has become commonplace in power systems operations research. However, in many cases, the same time series are used to generate the decision and then re-used to test the effectiveness of the decision model. This leads to over-optimistic results since the decision models will calculate their outputs with data that is too similar to reality and won't be able to capture the effects of forecast errors.

Performance metrics

Metrics are measurements computed on outputs of the computing process in Fig 2.1. The relevant aspect to consider is that metrics should be reported and calculated in terms of the trials and the results reported in terms of probabilities. Commonly, metrics are reported only for a single test set or single trial. In a computational experiment conducted with repeated trials, the metrics should accurately demonstrate the internal and external validity of the conclusions. For instance, in [186, 108], the authors use total costs and Automatic Generation Control performance metrics (e.g., CPS2) to assess the value of improving forecasting accuracy in the system.

There are a vast number of possible metrics relevant to power systems operations that depend on the application of the decision model. However, it is possible to derive three recommended principles in the design of experimental metrics:

1. Consider the distribution of metrics across trials and analyze them statistically.
2. Disaggregate metrics rather than reporting total costs alone. Presenting the costs associated with here-and-now and recourse actions separately can yield more insight, and researchers are also encouraged to include physical metrics which provide a richer picture of phenomena left out of decision models but captured by the emulator (e.g., battery wear, insecure loading limits, etc.).
3. Include the experiment computational times for the given environment. Metrics on computational performance tend to go unreported in the publications, but they are critical for assessing the feasibility of incorporating research into real systems or further experiments, as well as indicating opportunities for computational performance improvement.

2.3 Implementing a reproducible computational experiment in power systems

In practical terms, achieving reproducibility and validation requires operationalizing the principles of scientific computing discussed in Section 2.1. In this Section, we present a template for a reproducible scientific workflow for simulation-based empirical investigations of power systems operations problems. Each step of this template weighs the two major components required for a computational experiment: the *environment* and the *workflow*.

- **Environment:** The collection of software, hardware components, and configurations used to implement a computational experiment [13]. The environment may include elements such as cloud services, third-party software, file management scripts, and external tests.
- **Workflow:** The sequence of computing tasks, from data intake to summary of results via plot and table generation, which, together, make up the scientific experiment and the analysis [33]. The development of a workflow is a requirement for validation and reproducibility.

The following template breaks down both the environment and the workflow of a power systems computational experiment into the three main stages: *data process*, *computing process*, and *results and reporting process*.

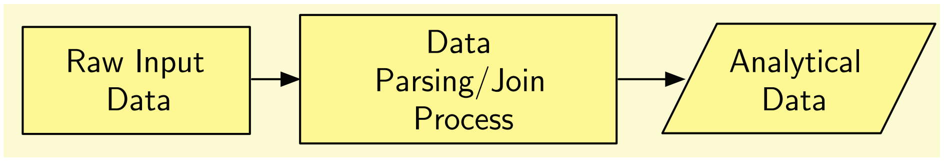


Figure 2.2: Data processing

Data process

Often, preoccupations about data in the context of scientific computing are limited to data sharing, access concerns, and best practices for transparency by using raw data files in formats humans can read [192], such as CSV for tabular data. However, the very act of processing raw data into analytical data must itself be a reproducible procedure that can be validated. The two aspects necessary for achieving this include (1) the data model and (2) the data production process.

All computational experiments need to define, document, and automate the steps followed in processing and generating data, such that these procedures can be reproduced and validated by other researchers. Figure 2.2 showcases a simple workflow for data processing and generation, distinguishing between the *raw input data* that can be obtained from many different sources (which may exist in a variety of formats) and the *analytical data* that has been parsed by the researcher to fit neatly into the organization of the data model.

Data Model

A data model is a way to organize data and standardize its internal relationships and properties, providing a structure for use in the computational experiment [190]. In power systems research, efforts to make standard data models have primarily focused on power flow data like the Common Format [196] that later evolved into the Common Information Model (CIM) developed by industry for SCADA and control center automation [178]. Power systems' operational research requires richer data models that can hold more information than a representation of the system assets' parameters. For instance, holding time series, confounding variables, and parameters is necessary for the execution of computational experiments. Forthcoming Section 2.4 discusses in detail the implementation of a data model that handles these concerns.

In power systems computation, the data model is often not considered a critical aspect of the simulation even though data transformations are a critical operation. Without a single, widely agreed-upon data model in power systems modeling for computational experimentation, researchers have to develop customized data models for individual applications. This process is time-consuming, and as a result, data models are usually underdeveloped. Researchers should first carefully evaluate whether a custom data model is necessary. If so, modern programming languages provide researchers with environments containing an extensive assortment of options to develop data models. Researchers can greatly increase

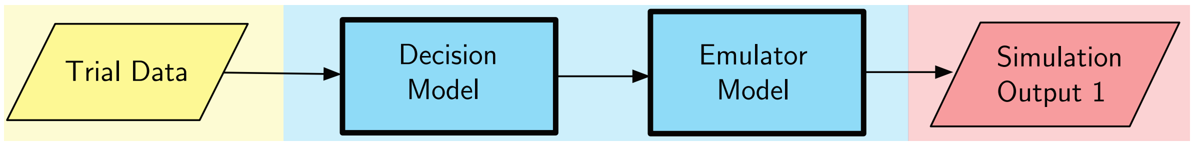


Figure 2.3: Simulation execution workflow

the value of their contribution, and their convergence to new standards by publishing their data model and implementing codes for re-use.

Data Consumption

In most cases, before the data can be arranged into the data model, some computations are necessary. Examples of such computations might include consistency checks, ensuring the anonymity of proprietary data, or adding more features to the original data. For instance, the researchers might include ramp rates to model UC in data sets that were originally developed for Power Flow (PF). Such computations are part of the data processing pipeline, and they must be recorded as part of the workflow.

In the context of the workflow depicted in Figure 2.2, synthetic data should be considered, along with "real" or observational data, to be part of the raw input for the model. Like real data, synthetic data may undergo parsing for use with a specific data model. When creating confounding data, all random number generators should be seeded and those seeds included in the data model.

Computing process

Operation Model Execution

The computing process is easily confused with the computational experiment itself. In fact, the experiment — as described earlier in this chapter — is far more nuanced than the execution of a model. A computing process is composed of a set of simulations used to test the effects on the test set in the system using the emulator model as a proxy, as shown in Fig. 2.3. Each trial of an experiment constitutes a simulation case, which is defined by the decision and emulator models' executions and interactions. For instance, simulating a standard day-ahead UC over a year results in executing 365 daily decision models, each followed by another 24 hourly ED emulator executions.

Simulations can take many different configurations depending on the research objective and the experimental design. It is therefore not possible to pre-define a generic simulation setup that fits all applications. However, it is possible to establish some general definitions to help guide development and facilitate the reproducibility of the simulation setup.

When an optimization model is executed, sub-processes relevant pieces to scientific workflow happen in the background that need to be accounted for in the computational

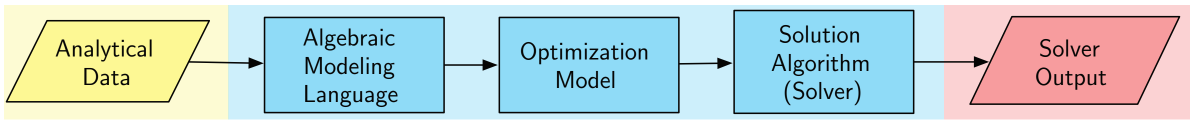


Figure 2.4: Optimization model execution pipeline

workflow. Figure 2.4 shows the pipeline and the different components that make up the environment and workflow of a model execution.

- **Algebraic Modeling Language**: The AML code describes the model's Mathematical Program (MP). The models are generally expressed in terms of *sets*, *parameters*, *variables*, *constraints*, and an *objective function*. In this stage, the optimization model is not being solved; rather, it is being processed such that it can be used in the solver.
- **Optimization Model**: This is the AML output, the optimization model in standard form. Often, AMLs make internal transformations to the MP to make it compatible with the solver. This include adding slack variables or constraint reformulations, and it can occasionally hinder reproducibility when the AML changes.
- **Solver**: The solver performs the computations necessary to obtain the solution of the MP. Modern optimization algorithms can be heavily customized, and all the details of the solver parametrization must be accounted for.

Dynamic Model Execution

Figure 2.5 shows the software stack of a dynamic modeling application: A modeling layer with the code representation of the equations, algorithms to solve the models and numerical linear algebra libraries to perform the calculations.

- **The modeling layer**: All the code representation of the system behavior in differential and/or algebraic equations (for instance, charging and discharging capacitors or PID controllers).
- **The integration algorithms:** The libraries that implement an integration scheme and are used to obtain the numerical solution of the models' differential equations (for instance, Euler, backwards differentiation formula, or Runge-Kutta, among others).
- **Linear algebra libraries:** Low-level algorithms to perform numerical linear algebra calculations for system solutions, LU decomposition, or eigenvalue calculations.

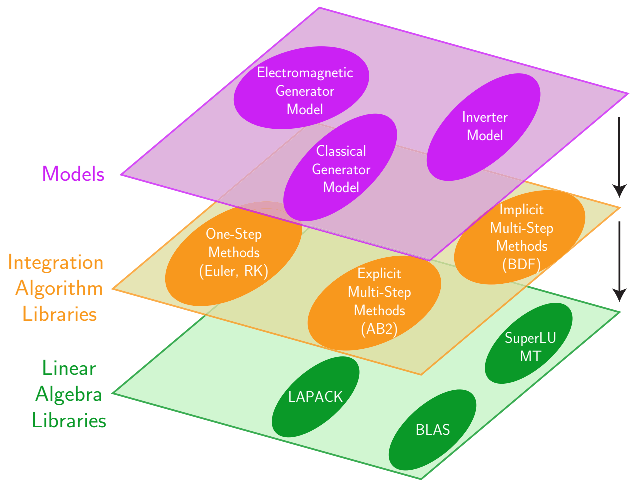

Models
Electromagnetic Generator Model
Classical Generator Model
Inverter Model
Integration Algorithm Libraries
One-Step Methods (Euler, RK)
Explicit Multi-Step Methods (AB2)
Implicit Multi-Step Methods (BDF)
Linear Algebra Libraries
LAPACK
BLAS
SuperLU MT

Figure 2.5: Description of the common software layers in a scientific research software program for dynamic simulation

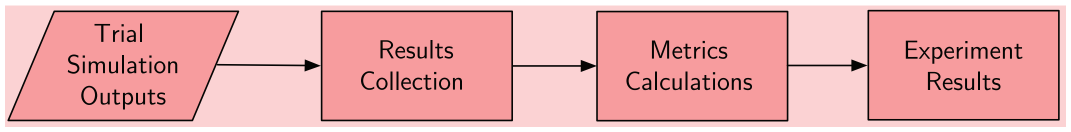


Figure 2.6: Results and reporting workflow

Results and Reporting

The output of the computing process is the collected results of the simulations, but this is not necessarily the final result of the experiment. Figure 2.6 shows the workflow to compile the outputs of the model executions into a format suitable for analysis, finishing with the calculation of the selected metrics. Often, other libraries are required in the environment to process the outputs and calculate the results. The aggregation process has the potential to confound the effects of the proposed model. Hence, the raw output of the simulations should be archived, and the code to calculate the metrics should comply with the principles of reproduciblity.

Case studies of the framework implementation

We have included two example cases for the principles described in this chapter. The cases are developed in MATLAB and Julia to showcase that the proposed framework is platform- and language-independent. The results shown here aim to exemplify the application of the principles discussed above in simplified case studies and not to derive specific conclusions about the models [3].

Case 1: Microgrid EMS with demand response

This experiment compares different predictive controllers to regulate power consumption through demand response in energy-constrained islanded microgrids.

- **Experiment Design:** The hypothesis is that each controller improves a particular performance metric relative to a baseline of no control. Here, the independent variable is the decision model used by the controller. We compared four controllers: a heuristic method that does not use forecasts, a predictive control using only a single average forecast, and two stochastic programming formulations.

  We constructed a synthetic microgrid, synthetic forecasts of solar generation based on historical data, and synthetic electricity demand forecasts based on random simulation to be used as confounding variables. The system features multiple customers and distributed solar and battery storage.

  The performance metrics are Average Service Availability Index (ASAI), total customer utility from electricity consumption, and the realization of the objective of the decision model, which is a measure we construct that captures the benefit of electricity consumption: since performance metrics include power availability and the impact of interruptions at the customer level, the emulator must capture the effect on customers.

- **Implementation:** This environment is MATLAB 2018a, and Gurobi 9.01 is the solver. We use the Gurobi MATLAB API. The controller makes a decision to send energy and power limit signals to customers every 4 hours with a receding horizon of 2 days. The emulator model includes a customer decision to ignore or respond to the signal every 4 hours, as well as a model of the physical devices that runs on a 2-minute time scale. The controller assumes that customers will reduce consumption if necessary to comply with energy and power limits, but it assumes they will reduce *exactly* to a limit. In the emulator, customers make a discrete choice of which appliances to disconnect, having some imperfect knowledge of what their consumption will be. Then, the 2-minute model simulates power sharing between generation and storage,

[3] The code for the case studies is archived in the public repository `https://github.com/Energy-MAC/PowerSystemsScientificComputing`

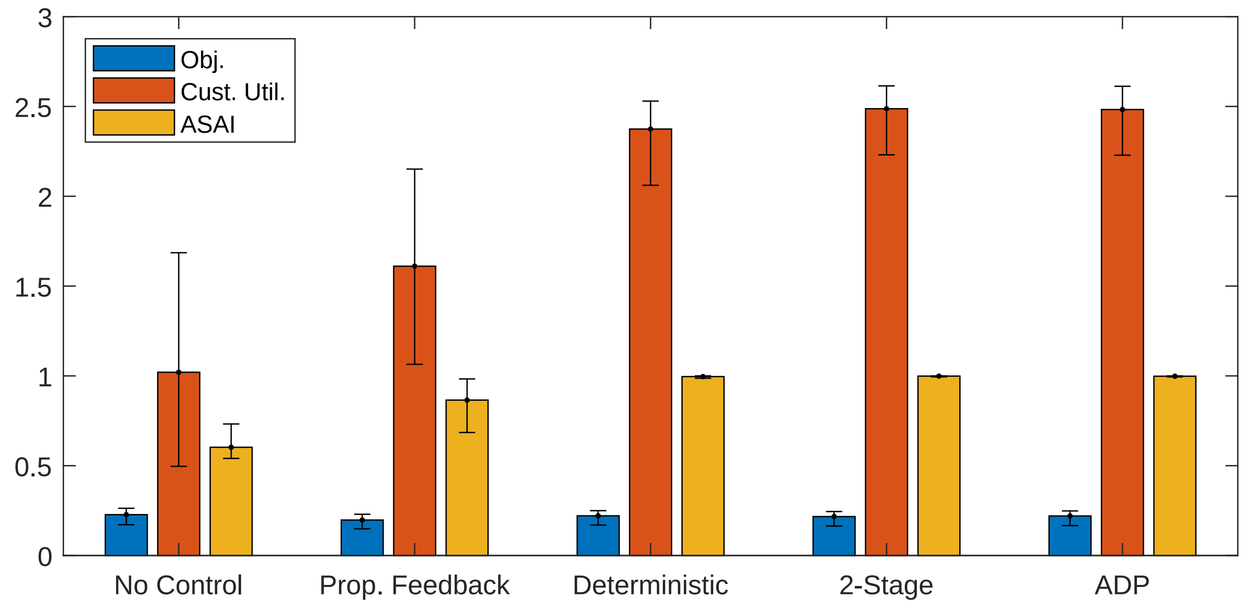


Figure 2.7: Performance metrics for different controllers for Microgrid Experiment as median values with 5th and 95th percentile error bars

the evolution of thermostatically controlled loads and battery state of charge, and interruptions when load power cannot be met or customers exceed limits.

- **Results:** Figure 2.7 shows the value of the metrics across trials for each controller relative to the baseline of no control. This illustrates the principles of displaying metrics statistically and showing the side effects on multiple concrete emulator metrics not explicitly optimized for in the decision model that has an abstract objective. The design framework emphasizes that these results are limited to the context of the experiment parameters, particularly the interruption cost function in the emulator customer model and the forecast and realization data. In fact, following the principles outlined in this chapter revealed initial poor performance with using power limits that we were able to address by instead using an energy limit, but would have been not revealed without a higher time-resolution emulator model.

Case 2: Stochastic UC and bulk power systems

The experiment is designed to compare an SUC model with a standard UC model. The detailed mathematical specifications of the models used in these experiments are located in the example repository.

- **Experiment Design:** The hypothesis is that using SUC achieves better hedging against wind power uncertainty and reduces the occurrence of load shedding. In this example, the independent variable is the day-ahead model i.e., the choice between using UC or SUC optimization models. Given that the SUC requires the use of scenarios, the complete test set is made up of the decision-model–forecast pair. The SUC uses 100 scenarios generated by sampling a truncated normal distribution with a variance of 30%. To evaluate the hypothesis, an ED model is used as the emulator with 5-minute resolution time series.

The experiment includes 10 trials for each range of 30 consecutive days in the annual data in order to capture seasonal variations through the year. For each trial, the metrics used to assess the performance of the system are total fuel cost and total energy not supplied. The models are tested in a modified 5-bus test system [93] which has been enhanced with piece-wise linear cost functions. The test system features an additional 3,600 MW of wind power generation and an increased peak load of 14,400 MW. The system has been also enhanced with yearly time series for load and wind power from the data provided in [7].

- **Implementation:**The experiment was done in Julia v1.2.0 using the `JuMP.jl` v0.20.0 as the AML with the solver Gurobi 8.11. Auxiliary `*.toml` files define the full experiment environment by fixing other libraries' versions in the experiment. The code structure in the example repository has been intentionally arranged, ensuring consistency with the definitions given in the chapter. The data model is provided using the package `PowerSystems.jl` described in the forthcoming Section 2.4, and the same is used to integrate time series data into the load flow case. The commitment decisions are synchronized with the ED model, which is executed 24 times for every commitment model execution.

- **Results:** The results showed that for most trials, the SUC reduces the amount of load shedding in the system but also has an increased fuel cost. The results are displayed as box plots in Figure 2.8 that enable further exploration of the trial results and provide more detailed insights about the model's performance. In this way, it is possible to notice that in two outlying samples, the SUC resulted in more load shedding than the UC. The result motivates an exploration of which circumstances may lead to weaker performance from the SUC than from the UC.

  From the computational point of view, running multiple trials allows us to evaluate whether the relative differences in computation speed are consistent between the models in the experiment. Each commitment model ran a total of 300 times; the SUC has an average solve time of 59.18 s and a maximum of 158.98 s, compared to the 2.87 s and a maximum of 8.2 s of the UC model — an order of magnitude faster than the SUC.

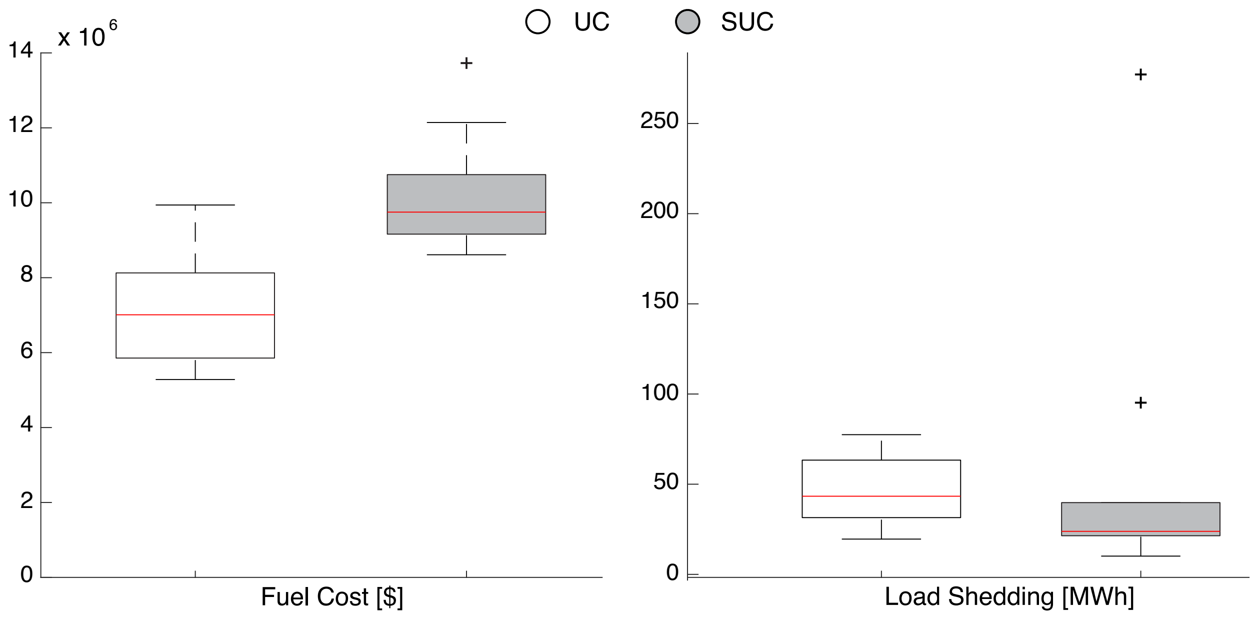


Figure 2.8: Box plots of results for Stochastic Unit Commitment (SUC) experiment

2.4 `PowerSystems.jl` data model for power systems scientific computing

Adequate access to the data sets used in scientific models leads to more transparency and reproducibility [142], improves scientific discovery, and allows third parties to verify model results [32]. Specifically, there is a common concern that computational experiments in energy systems need to define, document, and automate the processing and generation of data separately from the modeling [142, 88].

The data workflows need to distinguish between the raw input data, which exists in a variety of formats and is obtained from many different sources, and the analytical data for the model. `PowerSystems.jl` enables a consistent data model that can be populated from diverse sources of information. `PowerSystems.jl` is designed to address these challenges based on the principles laid out in [192, 193, 157]. Namely: software should always return the same outputs when given the same inputs; functionalities should be easy to use, extend, prototype, and integrate into other packages; and code should be written for people to understand, not just for efficient computer resource use.

The traditional practice in energy modeling has been to develop *ad-hoc* data structures, containers, and interfaces related to the underlying mathematical model [117]. However, by virtue of relying on custom data structures and utilities, modeling packages can discourage data sharing and create barriers to the validation, comparison, and introduction of models. As a consequence, model developers currently devote significant resources to parsing and converting between data models. In most cases, these efforts serve the specific scope of the analytical model and do not result in reusable code. For this reason, it is critical to make available an application agnostic library to support the development of models with the explicit intent of supporting data work flow reproducibility.

Applying scientific computing principles to the data processes used for energy sector models introduces several requirements:

- **Intuitive data creation scripts:** The syntax to create data sets should be easy to interpret and maintain.
- **Flexible interfaces for data intake:** Data sources for electric energy systems can be heterogeneous, requiring flexible interfaces to manipulate the data.
- **Straightforward extension for new data layouts:** With the addition of new technologies and operational modes, the software needs to be extensible in order to add new data representations.
- **Dedicated public interface for extension and integration:** User extensions and integration as a dependency should not require modifications to the source code (also known as the delegation pattern [84]).[4]
- **Optimized memory use for large data sets:** The implementation should not overwhelm system memory when handling large data sets.

Review of data models in electric energy systems

The need for a canonical data model to share data dates back to the 1950s when the first applications of a digital computer to solve the "load flow" problem appeared in the literature [64, 188]. Myriad solution methods and models had led to a problem of data and model exchange, as illustrated by the following quote:

> *With the growth in complexity of the interconnected power systems in the [1960s] came a corresponding growth in the number of load flow programs being used and in the number of study groups using those programs. This growth resulted in a need to exchange data at an increasing rate.* [196]

Historically, in electric power systems the model under study has been the source of requirements for the data model in use. Most importantly, there has been an explicit division between power systems models based on their scope. Different models require different simplifications to obtain system insights required by engineers. These engineering simplifications and assumptions have also carried over to other fields such as energy policy and economics. The most significant model that has informed data processing and sharing is the "load flow" problem with extension to the "economic load flow" problem [194].

[4]In computer science, a design pattern is a reoccurring solution that is sufficiently general to warrant its own title and description.

Leon Kirchmayer, in his seminal work [78], provides details about the early use of computers for the economic optimization of power systems. These rudimentary computational systems were limited to punch cards as the data loading medium. As a result, the first data models were merely column indexes of physical quantities used in the model.

Punch cards evolved to become fixed-position and fixed-order file data models. The first generally accepted data model for power systems computational analysis, the IEEE Common Format [196], was published in 1973. The common format data file had lines of up to 128 characters, the lines are grouped into sections with section headers, and data items are entered in specific columns. It provided a standard format to store and exchange data based on the original punch card specification, emulating the physical storage medium that had preceded it.

Although there has been a significant increase in computational power, algorithm development, and novel applications of computers to the analysis of electrical power systems since 1973, tabular data models still dominate. All major data formats and models for commercial and academic power systems software have employed tables with custom specifications to store and exchange system data. In the context of open-source modeling, `MATPOWER` data format [205] is the standard for encoding system data sets due to the popularity of MATLAB among power system researchers.

The need to share information evolved in the early 1990s with the advent of automation, and it was spurred by increasingly complex data needs for power systems operations. The industry required standardized models to exchange more extensive information, resorting to an object-oriented data model. The CIM was developed and was later made a standard maintained by the International Electrotechnical Commission (IEC) Technical Committee 57 Working Group 13. The aim was to provide a standard definition for power system components geared towards automated EMSs, SCADA systems, and asset-management databases. Automation-oriented modeling makes CIM challenging to implement for modeling purposes, and it is not widely used in any modeling software available today. It is available in only a few commercial power system software programs, and the only open-source parsing implementation is the `iTesla` library [181].

One of the key characteristics of electric power systems modeling is the rigid separation between steady-state and dynamic modeling practices. Modeling tools have kept separate data models between the two classes of models, and a few commercial providers dominate the market for dynamic modeling. As a result, a dynamic data model is dependent on the software available for the researcher. Such artificial separation hinders cross-domain research and further limits the development of newer models. Some efforts to develop open data models geared towards dynamic modeling, such as `PSAT` [114], have been limited to teaching and are no longer maintained. The data model implemented in [113] is described partially in [117] but has had little uptake.

With the advent of new algorithms, models, and programming languages, as well as broad access to computers, new software tools and data formats have proliferated. Milano provides a detailed taxonomy of 17 commercial and open-source tools up to 2010 [117].

Recently, new static modeling tools such as `Pandapower` [175], `PyPSA` [22], `PSST` [81], and `PowerModels` [27] have used data models largely based on the original `MATPOWER` schema. In the dynamic modeling domain, a Python implementation of a data model using symbolic libraries has been proposed [29], and an `OpenModelica` library with the capabilities to parse PSS®E and CIM [181] is available. However, developing extensions requires some source code modification, and the intermediate data structures cannot be integrated with steady-state models.

The review so far highlights the progress coming from the power systems community, given a more widespread adoption of certain "standard" practices. Several commercial software applications dominate in other modeling communities, and each relies on its proprietary data format. Such is the case for production cost modeling, which requires a rich data model to handle large amounts of time series data. Significant efforts have been made towards developing and to process XML proprietary data formats into open data sets [9] but have not resulted in a more systematic approach since for each new study the process needs to be refactored.

When augmentations are required, `MATPOWER` [205] provides a certain amount of flexibility to augment the data though its "extensions." This is the most commonly used approach. Extending `MATPOWER`'s data requires the creation of makeshift relationships between the user-added arrays and the arrays already in the model. Fixed location and length representations are not inherently designed to store data with mixed data representations and hierarchical structures. Tables are difficult to extend beyond their original design. For instance, adding a new feature implies adding a new column for the totality of the category. To the authors' knowledge, the production cost modeling community does not have an effort similar to the power systems community, and in most cases, data models used in cost production modeling are extensions of power systems data models. Moreover, the growing importance of data provenance and reproducibility demands solutions that minimize the need the develop *ad-hoc* data models.

In recent years, there has been increasing multi-sector modeling of energy systems. Initiatives such as OpenGenome [161], Spine [104], and the Open Energy Platform [131] have focused on integrating power systems data into broad energy infrastructure models. These initiatives exploit modern computing concepts and architectures like REST API, portable databases, and version control to provide users with a more straightforward pathway to integrate decision models with data. Importantly, these initiatives seek to contribute curated data sets as part of their repositories. Commonly, multi-sector projects focus on long-term planning and strategic decision making, which require economic data on top of technical device-level data. These techno-economic modeling communities make outstanding contributions by exploiting modern concepts in data management for large systems. For instance, the Open Energy Platform implements advanced table format data sets to facilitate the inspection of data sets.

As a consequence of these data representations' explosions, model developers devote significant resources to parsing and data model conversion. In most cases, these efforts are developed to serve within the analytical model's scope. Creating a standard data model and dedicated tools for data management across domains is critical to improving electric energy systems' modeling practices.

To the authors' knowledge, `PowerSystems.jl` is the only tool designed to provide model-agnostic data structures and model development capabilities as an independent library. Although other authors have advocated for a canonical data model in XML format [112], `PowerSystems.jl` also provides generic tools and interfaces required for data processing, verification, and extending the library of models. The principal use case for `PowerSystems.jl` is to provide efficient intake and utilization of power systems model input data.

`PowerSystems.jl` Software description

The interfaces in `PowerSystems.jl` are designed with three types of users, and specific uses of the data modeling package, in mind.

- **Modeler:** Develop standard data sets, share data, or generate reproducible computational experiments.
- **Model Developer:** Develop custom components, data sets, and models; use `PowerSystems.jl` as a dependency on a modeling package.
- **Code Developer:** Contribute source code to `PowerSystems.jl` to implement new features.

This chapter focuses on the first two categories: modelers and model developers. Readers are encouraged to look through the tutorial sections in the documentation as a starting point and to consult the developer section when looking into the requirements for integrating custom data structures.

`PowerSystems.jl` is developed in Julia [16] due to the inherent composability of the language and the extensive interoperability capabilities with other programming languages. `PowerSystems.jl` exploits the type system and multiple dispatch of the Julia programming language to promote the open development of energy data sets across domains. This capabilities are used extensively in the forthcoming Chapter 3 and 5 to develop modeling libraries for operations and dynamic simulations.

The main features of `PowerSystems.jl` include:

- Comprehensive and extensible library of data structures for modeling electrical systems.
- Large-scale data set development tools that use PSS®E `.raw` and `.dyr`, and `MATPOWER .m` common text-based data formats as base as well as a configurable tabular data (e.g., CSV) parsing capabilities.
- An optimized container for component and a time series store for data that supports serialization to portable file formats and configurable validation routines.

In `PowerSystems.jl`, a device is defined using a Julia structure embedded in a type hierarchy. Each device is discussed in detail in Section 3.1. This implementation enables the researcher to categorize the devices by their abstract operational characteristics. In principle, the generalization of each component is done at the categorical level, preventing the shortcomings of prescriptive data models. Representing all potential devices in energy modeling is not possible; neither is it desirable, as new technologies become available and make parts of the library obsolete. Thus, it is necessary to provide an extensible data model with simple rules such that different users can store custom data in an organized way and still use other core functionalities of the package.

The implementation of the data structures is *method forward*, which implies that the information stored in each object is accessible through the implementation of methods (e.g., `get_parameter_value(::DataStruct)`) and not by accessing specific fields (e.g., `datastruct.parameter_value`). Similarly, the requirements for extensions are described as interface implementations, not data fields, providing modelers with more flexibility. Utilizing a method forwards design prevents introducing the known fragilities from the classic implementations of class inheritance [84].

Implementing data structures through interfaces is especially valuable for long-term code maintenance and for designing reproducible experiments. The use of *accessor functions* enables the modeler to manipulate the parameters without concerns about the underlying implementation. On the other hand, if the model accesses the data by field, subsequent implementation changes can generate unsustainable maintenance costs. The accessor interface also reduces the cost of integrating `PowerSystems.jl` into modeling applications. The model developer can re-use the data management methods already implemented and thereby minimize the development of custom code to handle data input.

In data sets for energy systems simulation, the largest demand on memory often comes from time series data. Given the size of this data, it can overwhelm system memory and must remain on persistent media (e.g., disks). However, it is also critical to maintain low read/write latency. `PowerSystems.jl` implementation of the external data container solves the aforementioned issues by leveraging HDF5 storage to execute fast data loads into memory on demand.

The implementation of the underlying data for time series data utilizes different formats and data types depending on the modeling needs and originating process. For instance, data storage is optimized for forecast data dependent on the structure. `PowerSystems.jl` supports forecast data formatted as overlapping time windows (e.g., 4-hr-ahead forecasts created every 15 minutes) and contiguous time series of observations. The flexible representation of time series data allows for multiple uses including day-ahead market models or slice selections to model limited operation periods. Also, `PowerSystems.jl` provides a mechanism to share time series data across components; this can greatly reduce primary storage requirements.

Software architecture

`PowerSystems.jl` is structured to meet the requirements discussed in Section 2.4 through the implementation of the following features:

1. Abstract type hierarchy,
2. Optimized read/write data container (named `System`), and
3. Utilities to facilitate modeling, extensions, and integration.

The optimized container and generic extension interfaces in `PowerSystems.jl` are implemented through the utility library `InfrastructureSystems.jl` [174]. `PowerSystems.jl` contains the code, methods, and utilities specific to the electricity sector's physical representation and relationships. `InfrastructureSystems.jl` provides generic methods to handle components and manage data. This design follows from the recognition that several of the general features and requirements in `PowerSystems.jl` apply to any networked infrastructure data handling software; in the future, an extension to water, gas, and similar infrastructure systems is possible.

Type Hierarchy

Type trees (or taxononomies) to organize data classes or types are commonly used in libraries that implement an object-oriented approach. For instance, the CIM [179] uses a taxonomy to represents all the major components in electric utility operations. However, the CIM is meant to facilitate the integration controls developed independently by different vendors and is not suitable for modeling. Also, initiatives such as the Open Energy Ontology [53], which is currently under development, focus on organizing terms and relationships within energy system modeling.

The abstract hierarchy implemented in `PowerSystems.jl` enables categorization of the devices by their operational characteristics and modeling requirements. Figure 2.9 shows the abstract hierarchy of components. For instance, generation is classified by the distinctive data requirements for modeling in three categories: thermal, renewable, and hydropower. As a result of this design, developers can define model logic entirely based on abstract types and create generic code to support modeling technologies that are not yet implemented in the package. `PowerSystems.jl` has a category of topological components (e.g., bus, arc) separate from the physical components [117]. The hierarchy also includes components absent in standard data models, such as services. The services category includes reserves, transfers, and AGC. The power of `PowerSystems.jl` lies in its ability to provide abstraction without an implicit mathematical representation of the component in question.

Other abstractions provide flexible specification of parameters with distinct representations. An example is the representation of costs. `PowerSystems.jl` implements several `DeviceParameters` to support composition patterns.

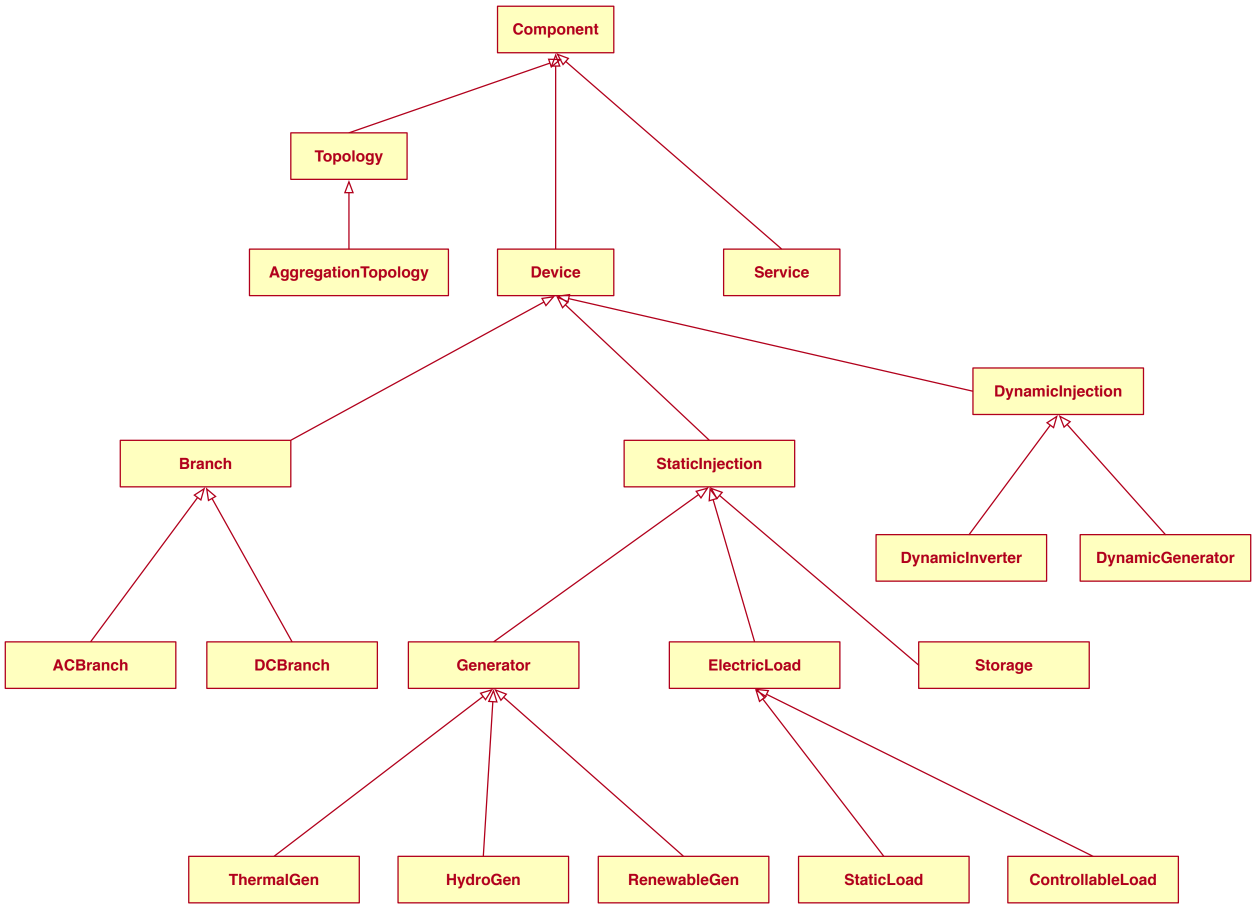


Figure 2.9: Abstract tree hierarchy

A comprehensive example of the applications of composition and methods to access struct field is the implementation of the `ThermalStandard`. An implementation of a thermal generator using common data fields such as power limits, fuel and ramping is shown in Figure 2.10. `ThermalStandard` implements several methods to retrieve device parameters such as the active power limits. It is composed of the bus, operational cost, services, and the possibility of including a `DynamicInjection` to represent dynamic models for the same device.

Classically, data models have been separated into dynamics and quasi-static analyses in different data sets. However, composition enables a joint representation of the components and eliminates the requirement to maintain two discrete databases.

Data Container

The `System` is the main container of components and the time series data references. `PowerSystems.jl` uses a hybrid approach to data storage, as shown in Figure 2.11, where meta-

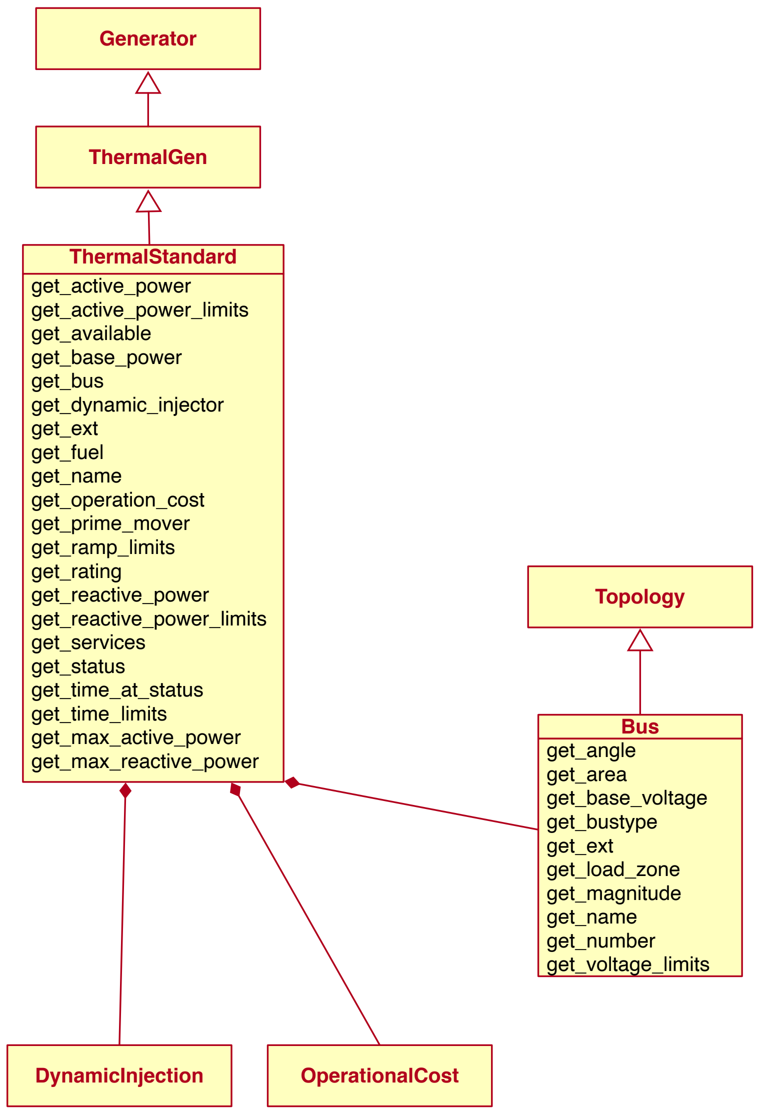


Figure 2.10: Implementation of ThermalStandard

data (for describing package version compatibility, unique identifiers, and other system-level meta-information), component data, and time series references are stored in volatile memory, while the actual time series data are stored in an HDF5 file. This design loads into memory the portions of the data that are relevant at time of the query, and so avoids overwhelming the memory resources.

`PowerSystems.jl` implements a wide variety of methods to search for components to aid in the development of models. Code listing 2.1 shows an example of retrieving components through the type hierarchy with the `get_components` function and exploiting

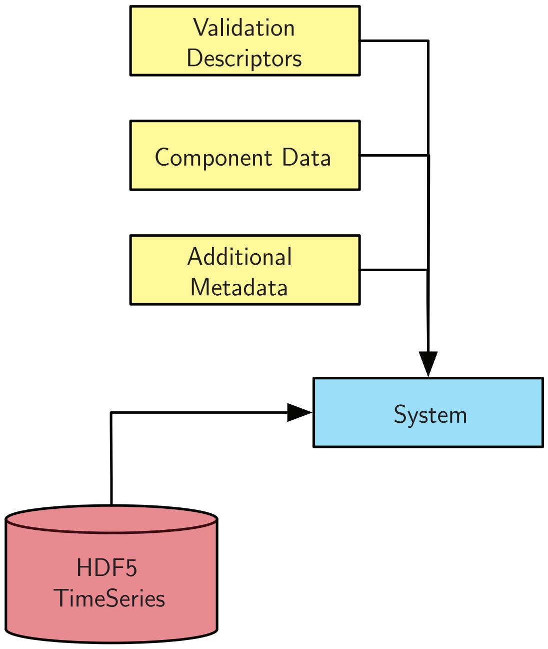


Figure 2.11: `System` data container structure

the type hierarchy for modeling purposes.[5] The default implementation of the function `get_components` takes the desired device type (concrete or abstract) and the system and also accepts filter functions for a more refined search. The most common filtering style is by component name, and for this case the method `get_component` returns a single component taking the device type, system, and name as arguments. The container is optimized for iteration over abstract or concrete component types as described by the type hierarchy. Given the potential size of the return, `PowerSystems.jl` returns Julia iterators in order to avoid unnecessary memory allocations.

An essential workflow in energy systems modeling is developing data sets that combine existing data sources with new components and time series data. Code listing 2.2 shows the code process to execute the following workflow:

1. Load component data from a power flow file.
2. Add a new component. The example creates and adds a single wind power plant. The component constructors use keyword arguments making data entries very explicit.
3. Load time series data from a pointers file and add to the system components.
4. Load time series data specific to newly added component and attach to a single component.

[5] See Julia's punctuation to facilitate reading the code listing: https://docs.julialang.org/en/v1/base/punctuation/.

```julia
1  using PowerSystems
2
3  # Load the system from a power flow case
4  system_data = System("case_ACTIVSg70k.m")
5
6  function installed_capacity(system::System; technology::Type{T} = Generator) where T
   ↪ <: Generator
7      installed_capacity = 0.0
8      for g in get_components(T, system)
9          installed_capacity += get_max_active_power(g)
10     end
11     return installed_capacity
12 end
13
14 installed_capacity(system_data)
15 installed_capacity(system_data; technology = RenewableGen)
16 installed_capacity(system_data; technology = ThermalStandard)
```

Code 2.1: `PowerSystems.jl` code example applied to modeling

5. Serialize system for future use.

Code listing 2.1 shows the implementation of a function where the modeler can choose the technology by its type and use the different implementations of `get_max_active_power`. Note that in line 15 the function takes an abstract type and in line 16 it takes a concrete type. This code listing exemplifies the flexibility of the container interface to facilitate the development of models consistent with the ontology defined in Fig. 2.9.

The data is serialized to a JSON file and the time series data to an HDF5 file. This feature has been developed to enable reproducible data workflows. Modelers can develop data sets as a separate exercise from the modeling and later use `PowerSystems.jl` to load the final data in a fraction of the time it takes to re-create, transform and load into memory the raw data. This feature has proven to be particularly useful when the raw data have been generated from free-form tables stored as CSV files [198, 9].

Examples of data cases already available in the `PowerSystems.jl` format include the data from the RTS-96 data set [9], and we have made available some simpler data sets as examples in a separate repository[6] that allows users to automatically build systems from a standard specification.

Handling Time Series

The bulk of the data sets in many power system simulations models is the time series data. Whether the data represents forecasts used in day-ahead operations of realization scenarios

[6] `https://github.com/NREL-SIIP/PowerSystemCaseBuilder.jl`

```julia
using PowerSystems, CSV, TimeSeries, Dates

# (1) Load the system from a power flow case
system = System("src/case5.m")

# Define a new device
new_renewable = RenewableDispatch(
        name = "WindBusA",
        available = true,
        bus = get_component(Bus, system, "3"),
        active_power = 2.0,
        reactive_power = 1.0,
        rating = 1.2,
        prime_mover = PrimeMovers.WT,
        reactive_power_limits = (min = 0.0, max = 0.0),
        base_power = 100.0,
        operation_cost = TwoPartCost(22.0, 0.0),
        power_factor = 1.0
    )

# (2) Add that device as a new component
add_component!(system, new_renewable)

# Manually create a TimeSeriesData
time_series_data_raw = TimeArray(CSV.read("wind_data.csv"), timestamp=:timestamp)
ts_data = SingleTimeSeries(label = "active_power", data = time_series_data_raw)

# (3) Add the time series to the system and component
add_time_series!(system, new_renewable, ts_data)

# (3) Load time series from pointers file
add_time_series!(system, "timeseries_pointers_load.json")

to_json(system, "serialized_system.json")
```

Code 2.2: Example of data set composition workflow

of renewable energy they data that container the component level details is usually small compared to time series. To organize the time series data given the potential inherent complexity, `PowerSystems.jl` has a set of definitions to enable consistent modeling

- **Resolution**: The period of time between each discrete value in the data; all resolutions are represented using `Dates.Period` types. For instance, a day-ahead market data set usually has a resolution of 1 hour, and a real-time market data set usually has a resolution of 5 minutes

- **Static data:** A single column of time series values for a component field (such as active power) where each time period is represented by a single value. These data are commonly obtained from historical information or the realization of a time-varying quantity. Static timeseries data in `PowerSystems.jl` comes in the following format:

  Table 2.2: Static time series data format

DateTime	**Value**
2020-09-01T00:00:00	100.0
2020-09-01T01:00:00	101.0
2020-09-01T02:00:00	99.0

  where a column (or several columns) represents the time stamp associated with the value, and a column stores the values of interest.

- **Forecasts**: Predicted values of a time-varying quantity that commonly features a look-ahead and can have multiple data values representing each time period. This type of data is used in simulations with receding horizons or data generated from forecasting algorithms.

  Forecast data in `PowerSystems.jl` comes in the following format:

  Table 2.3: Forecast time series data format

DateTime	**0**	**1**	**2**	**3**	**4**	**5**	**6**
2020-09-01T00:00:00	100.0	101.0	101.3	90.0	98.0	87.2	88.0
2020-09-01T01:00:00	101.0	101.3	99.0	98.0	88.9	88.3	67.1
2020-09-01T02:00:00	99.0	67.0	89.0	99.0	100.0	101.0	112.0

  where a column (or several columns) represents the time stamp associated with the initial time of the forecast, and the columns represent the forecasted values.

- **Interval**: The period of time between forecasts' initial times. In `PowerSystems.jl`, all intervals are represented using `Dates.Period` types. For instance, in a day-ahead market simulation, the interval of the time series is usually 24 hours; in the example above, the interval is 1 hour.

- **Horizon**: The count of discrete forecasted values; all horizons in `PowerSystems.jl` are represented with `Int`. For instance, many day-ahead markets will have a forecast with a horizon of 24.

- **Forecast window**: Represents the forecasted value starting at a particular initial time.

A common workflow in developing models involves transforming data generated from a realization and stored in a single column into deterministic forecasts to account for the effects of the look-ahead. Usually, this workflow leads to large data duplications in the overlapping windows between forecasts.

In order to reduce data duplication, `PowerSystems.jl` provides a method to transform static time series stored as `SingleTimeSeries` data into `Deterministic` forecasts without duplicating any data. The resulting object behaves exactly like a `Deterministic`. Instead of storing windows at each initial time, it provides a view into the existing data at incrementing offsets. Currently, `PowerSystems.jl` does not support `Forecasts` or `SingleTimeSeries` with dissimilar intervals or resolution. This imposes a requirement that all the time series data in a system be consistently spaced.

Code listing 2.3 shows a minimal example of `PowerSystems.jl` used to develop an ED model. The code listing shows the stages explicitly:

1. Make the data set from power flow and time series data,
2. Serialize the data, and
3. Pass the data and algorithm to the model.

One of the main advantages of `PowerSystems.jl` is not having to re-run lines 4–5 for every model execution. The model code shows an example of populating the constraints and cost functions using accessor functions inside the model function (lines 9–37). The example concludes by reading the data created earlier in line 40 and passing the algorithm with the data in line 41.

`PowerSystems.jl` Impact

`PowerSystems.jl` is designed to account for the common workflows that analysts and engineers use to manipulate data sets, and to develop new models and packages for the changing landscape of energy systems. The two main contributions are as follows: (1) Decouple data processing from the computations in the models; (2) provide the field with an inherently extensible data modeling framework to develop new models and software packages that can be shared across a variety of modeling objectives.

Existing power systems software packages have addressed a subset of the principles established in Section 2.1 but with the specific focus of providing a data model for the analytic objective of the underlying mathematical model. As a result, developers of new models are faced with the choice of either adapting their data model needs to the existing structures or developing custom ones. Often, the underlying data are formatted in `MATPOWER` [205] developed in MATLAB and industrial tools such as PSS®E. Both formats are based on fixed-order, fixed-position text files that require code development for data input. `PowerSystems.jl` goes beyond the implementation of the canonical data model and also provides manipulation methods and utilities for its integration into other packages.

```julia
using PowerSystems, JuMP, Ipopt

########## Data Process ###########
system = System("src/5bus_ts/case5_re.m")
add_forecasts!(system, "timeseries_pointers.json")
to_json(system, "system_data.json")

######## Model Process #########
function ed_model(system::System, optimizer)
  m = Model(optimizer())
  time_periods = get_time_series_horizon(system)
  thermal_gens_names = get_name.(get_components(ThermalStandard, system))
  @variable(m, pg[g in thermal_gens_names, t in time_periods] >= 0)

  for g in get_components(ThermalStandard, system), t in time_periods
      name = get_name(g)
      @constraint(m, pg[name, t] >= get_active_power_limits(g).min)
      @constraint(m, pg[name, t] <= get_active_power_limits(g).max)
  end

  net_load = zeros(time_periods)
  for g in get_components(RenewableGen, system)
      net_load -= get_time_series_values(SingleTimeSeries, g, "max_active_power")
  end

  for g in get_components(StaticLoad, system)
      net_load += get_time_series_values(SingleTimeSeries, g, "max_active_power")
  end

  for t in time_periods
      @constraint(m, sum(pg[g, t] for g in thermal_gens_names) == net_load[t])
  end

  @objective(m, Min, sum(pg[get_name(g), t]^2 *
  ↪   get_cost(get_variable(get_operation_cost(g)))[1] + pg[get_name(g), t] *
  ↪   get_cost(get_variable(get_operation_cost(g)))[2] for g in
  ↪   get_components(ThermalGen, system), t in time_periods))

  return optimize!(m)
end

#### Execution ####
system_data = System("system_data.json")
results = ed_model(system_data, Ipopt.Optimizer)
```

Code 2.3: Example usage of `PowerSystems.jl` to the development of a multi-time step Economic Dispatch (ED) model

`PowerSystems.jl` data handling improves the scientific integrity of power systems research and analysis and enables the implementation of scientific computing principles for several research communities that rely on electric power systems data. One recent example of its application in the analysis of the Cambodian grid [8], where the authors separated data manipulation from modeling, making both processes transparent individually.

High-impact research in energy systems requires the analysis of large amounts of data, and `PowerSystems.jl` has several key features for modelers and analysts to handle and extend large data sets. `PowerSystems.jl` enables easy data creation with parsers for standard file formats, and it also optimizes in-memory data access by creating methods for efficient parameter and time series access and iteration. The fast data access and efficient model instantiation features of `PowerSystems.jl` have been leveraged to enumerate large contingency sets in numerous power system test cases [25, 24].

For model developers, `PowerSystems.jl` provides a generic, reusable, and customizable data model applicable to multiple modeling objectives. Not only does it provide the computational improvement to handle data at large scales, but it also provides the extension capabilities by design that make it easier to integrate into modeling packages.

`PowerSystems.jl` implementation in the Julia programming language allows for fast development and prototyping, as well as fast compilation and runtime performance [16]. In contrast to other object-oriented-programming, where inheritance is the only way to develop extensions or custom methods, it provides user flexibility and extensibility by providing type abstractions on which to implement methods using multiple dispatch.

The type-based and method-forward paradigm in `PowerSystems.jl` incentivizes the adoption of best practices in scientific computing [192] by creating accessible interfaces to enable code re-use in modeling and to create modular and reproducible scientific computing applications. Code listing 2.3 summarizes the breakdown of the modeling process between data, model, and algorithm following the definitions in Section 2.3.

Recent publications [87] have used `PowerSystems.jl` for the development of AGC simulation models following scientific computing practices to develop the experiment. Two recent contributions in low-inertia power systems [151] [65] exploited the flexibility of the dynamic model specification in `PowerSystems.jl` to implement the data manipulation code in the simulation experiments. `PowerSystems.jl` serves as the data handling platform for the rest of the software contributions in this dissertation for the study of power system operations and dynamics.

2.5 Conclusions

- The power systems community can enjoy many potential benefits from the adoption of scientific computing practices. Current computational experimentation practices are not as well developed when compared to other fields. Issues with data access, experiment reproducibility and access to code have been discussed in this chapter.

- Developing novel operation models requires validation through computational experiments. Appropriate experimental design is critical in demonstrating models' capability to handle the large-scale penetration of renewable energy.

- The software developed for power systems operational research must be able to reproduce the experimental results, which implies access to the input data, code access, and consistent results when the experiment is replicated. Beyond code sharing, there needs to be intent on the side of the researcher to enable the experiment to be reproducible.

- This chapter introduced `PowerSystems.jl`, the first tool dedicated exclusively to providing data management tools for electricity system modeling across research domains. The focus of `PowerSystems.jl` is to provide a structured data scheme, efficient in-memory data handling, and parsing capabilities from popular file formats. The primary motivation is to provide model-agnostic data structures that incentivize separation between the data processing code and the modeling code.

- The software implementation of `PowerSystems.jl` specifically addressed several practical challenges in the development of reproducible simulations. The use of consistent interfaces, data checking and model agnostic implementation allows researchers to develop their own applications without embedding assumptions.

Chapter 3

Simulation of Power Systems Operations

With the integration of large amounts of VRE in electric grids, there is a need to advance the flexibility and robustness of industry practices. Optimization-based models have become an essential tool in power systems operations to achieve increased integration in a reliable fashion [82]. Assessing the effects of VRE expansion and demonstrating the value of novel approaches over conventional operational strategies relies on computer simulations as a means of testing. The outcomes of such simulations are of great use to diverse stakeholders – including planners, analysts, and regulators – who depend on operational model simulations for analyses that range from the cost of supply estimation to expansion plan feasibility. In many cases, the outcomes of these analyses are relevant to the public, the scientific community, or both. Given this breadth of applications for power systems simulations, transparency and reproducibility [142] are imperative as they allow the verification of results [31], which can be accomplished with rigorous scientific computing practices. Scientific computing practices promote the application of principles such as reproducibility, transparency, and accuracy to experiments carried out using computer simulations [88]. There is large and growing relevance of scientific computing for energy simulations that should provide adequate access to the models and the data used.

One of the key innovations arising from systems operations modeling has been the inclusion of sub-hourly effects. Recent contributions show the relevance of intra-period coordination is operations assessment [128, 186]. The main driver to assess the intra-temporality research questions is software that simulates system operations with sequentially coupled optimization problems. Handling large scale and long range simulations, however, has largely been limited to commercial software tools like `PLEXOS`, Hitachi-`PROMOD`, or GE-`MAPS`. Given the complex, data-intensive nature of an operations simulation and the general lack of access to source code and data, many of the underlying assumptions remain hidden to external observers [31].

Models used to simulate power system operations usually feature significant simplifications of the system's physics to formulate optimization problems that focus on power system decision-making such as commitment of units, dispatch, or reserve allocation. Hence, a transparent framework requires that the simulation experiment properly account for all the assumptions that go into the models' simplifications, particularly when using the simulations to produce scientific or policy statements. However, commercial simulation software models are intellectual property not available to the public and their cost can be a significant barrier to reproducibility. When commercial simulators are not available power system operations researchers have to develop ad-hoc heuristics and solutions to combine optimization and modeling software from multiple vendors. The resulting lack of coherence across platforms and reliance on bespoke frameworks to execute experiments makes simulation evaluation, adoption, modification, and expansion costly, and in many cases it is impossible to reproduce [88].

In recent years, open-source operations simulation software has become available in recent years for research purposes. Some examples of software that to date are actively maintained include the Python library `PyPSA` [21] which focuses on expansion planning models and enables users to assess the plans that result from simulating operations. Another is `FESTIV` [46], a primarily Matlab-based software that relies on a mixture of `GAMS` and Matlab for the study of the impact of VRE in system operations at high levels of granularity. Although the code for `FESTIV` is publicly available, its implementation requires commercial software, while model modification requires changing the source code. Finally, the Julia package `PowerModels.jl` [27] focuses on the formulation and solution of OPF problems. Although `PowerModels.jl` is capable of handling realistically-sized systems and is openly sourced, it cannot inherently be used to perform long-range simulations, i.e., representing months to years of operation, or solve sequences of operational problems.

Motivations

Power systems research has benefited from having open source analysis and simulations tools. However, existing operations simulations tools have focused on single-period problems (e.g. load flow and contingency analysis) or multi-period problems like UC or OPF, thus avoiding the complexities of inter-problem information flow. Despite the widespread usage of Production Cost Model (PCM) in research, and industry decision-making, the definitions and algorithmic structure that enables PCM simulations is poorly represented in literature. The lack of availability of operations simulation software drives the lack of reproducible research in this area.

Furthermore, given that the alternatives to developing a large scale operations simulation results in complex software projects or depend on commercial tools, the underlying optimization models used in the simulation tend to be rigidly defined. A model's rigidity can constrain the questions that researchers and analysts examine, resulting in *modeling-limited choice* [39, 137]. The lack of flexibility characterizing the models and operational sequences influences the type of simulations that can be conducted. Model-limited choices can stem from two factors: (1) Structural exclusion of certain forms of simulation and analysis; and (2) formulation limitations due to restrictions in underlying models or whether data is available. The chapter introduces the software package `PowerSimulations.jl` (`PSI.jl`) that enables scalibility and flexibility of *operations simulations*.

This chapter begins by defining and *operations simulation*, along with other terminology required to formulate a *operations simulation model*. Throughout the chapter we introduce definitions to properties of an operations simulation commonly used but rarely discussed in the simulation literature. The objective of `PowerSimulations.jl` is twofold: (1) Enable a scientific approach to the simulation of large-scale power system operations; and (2) reduce model-limited choice when framing operations simulation experiments. `PSI.jl` also aims to provide the necessary utilities to develop simulations at a scale and scope on par with commercial tools.

This chapter's contributions include:

- Formal definitions to structure an operations simulation that employs sequential solves of multiple optimization problems
- A systematic method to formulate operation optimization models for the purpose of simulation incorporating devices, network and services that span multiple time resolutions.
- The description of methods and software implementation requirements to support long range simulations.
- A framework that enables developers and researchers to scientifically reproduce an operations simulation.
- The implementation of an experimental set-up with decision model and emulator model for the study of reserve deployment using a detailed quasi-steady-state AGC simulation model that considers steady-state frequency deviations and device limits.
- A realistic mechanism to assess generator saturation effects and account for re-deployment in reserve deployment.
- An optimization-based implementation of an AGC model that can be solved efficiently using LP solvers.

The remainder of the chapter includes a section on 3.1, which provides readers with definitions for operations simulation and provides a robust description of its atomic components. Section 3.2 showcases the software implementation of an operations simulation package, with consideration to two main components: namely, model building and execution. Section 3.3 examines the validation and example uses of `PSI.jl` to publicly available datasets, while Section 3.4 presents in detail the use of the concepts to the simulation of a system considering the AGC. Finally, 3.6 discusses conclusions and future work.

Notation

Lower-case letters x denote one-dimensional real variables, parameters, and functions; upper-case letters in the form F and $F()$ are used for matrices and functions respectively; arrows (as in $\vec{x}$) represent vectors of variables or parameters and calligraphic symbols (as in $\mathcal{X}$) is used to denote sets.

3.1 Defining an operations simulation

Simulation vs Optimization

Power systems operations is a rich field for the study and development of *optimization models* to find the solution of *operation problems*. Given its importance, and the complexity and scale of the electric grid, there exists an abundance of literature on approaches that efficiently formulate and solve *optimization models* with applications to power systems [132]. For instance, UC is an operation problem typically formulated as a MILP optimization model and has been the focus of countless formulations and algorithmic improvements. Also, the OPF problem has received significant contributions with the development of relaxed formulations that exploit recent advances in convex optimization algorithms.

On the other hand, the objective of an *operations simulation* is to find solutions to a *sequence* of *operation problems* in a way that resembles the procedures followed by operators. For instance, an *operations simulation* seeks to replicate the day-ahead UC, hour-ahead UC and real-time ED market clearing processes. The results of an operations simulation could be used to assess the generation fleet fuel costs during a year. The evolution from “load-duration" curves to assess systems costs to the use of optimization has enabled increased insight into the effects VRE in the system operation [128]. However, as the complexity of the analysis grows to incorporate new technologies or bring a more detailed representation of the system’s operation into the simulation, the scope of PCM has grown beyond its original application. Hence, a simulation requires the specification of one or more *operation problems* and *optimization models* used to represent and solve each problem. Each *optimization model* is an *atomic* component of a simulation, while the formulation of an *operation problem* can differ depending on the application, jurisdiction, or phenomena of interest.

Operation Problem Definitions

We begin by defining a *decision* problem and an *emulation* problem, two types of operation problems used in the course of an operations simulation. These definitions follow from the framework for experimental design from Chapter 2. Decision Problems and Emulation Problems are discrete models with a specific *resolution* Δt depending on the requirements and available input data. The time keeping deifinitions of these problems follows from the time-series structures defined in Section 2.4.

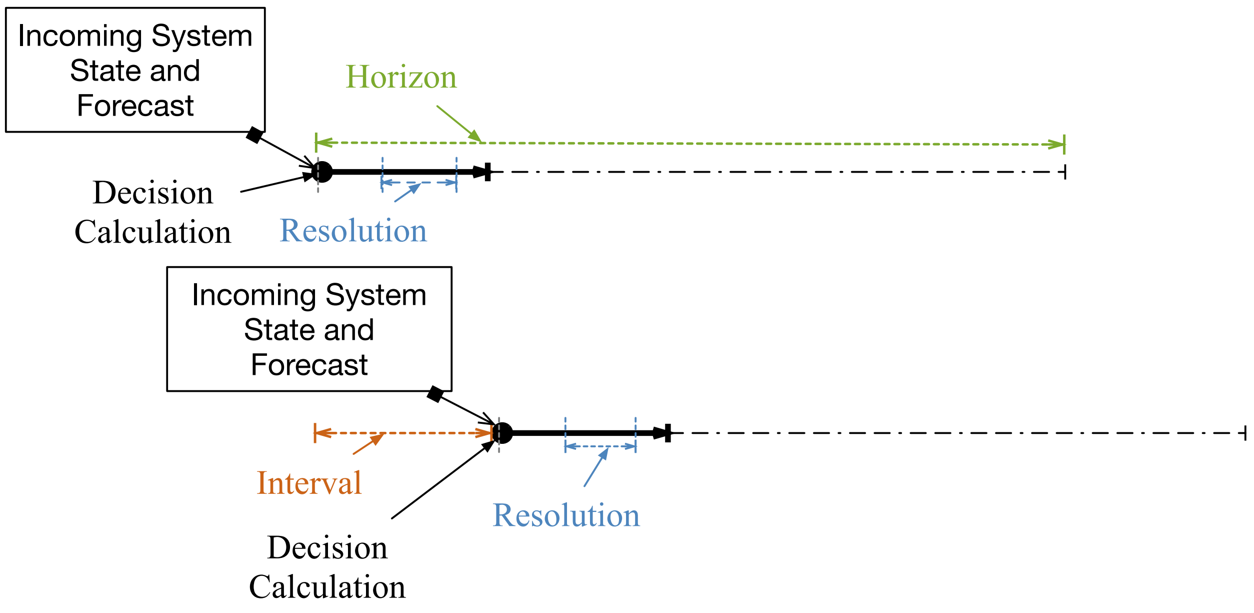


Figure 3.1: Sequential properties of a decision model.

- **Decision Problem:** A decision problem calculates the desired system operation based on forecasts of uncertain inputs and information about the state of the system. The output of a decision problem represents the policies used to drive the set-points of the system's devices, like generators or switches, and depend on the purpose of the problem. The decision problems employ the forecasts to make decisions for the discrete time-steps over a *horizon*. The model of a decision problem updates the decision values at an *interval* which is larger than its *resolution*, Fig. 3.1 depicts these definitions. Each calculation step is taken at a particular interval and employs information about the state of the system and a forecast to estimate future estates. For instance in day-ahead operations the common practice is to use a UC problem with a 24-hour *interval*, 1-hour *resolution* and over a 48-hour *horizon*. The decisions calculated in the portion of the horizon after the end of the interval (dash line in Fig. 3.1) are not implemented and are used as a way to handle inter-temporal effects of the uncertain quantities[1]. Decisions in power systems are calculated in a *sequential* or *staged* manner since decisions are continually refined as information about the operating period updates.

- **Emulation Problem:** An emulation problem is used to mimic the system's behavior subject to an incoming decision and the realization of a forecasted inputs. The solution of the emulator produces outputs representative of the system performance when operating subject the policies resulting from the decision models. The emulator

[1]Some operators call that segment of the look-ahead the "study period".

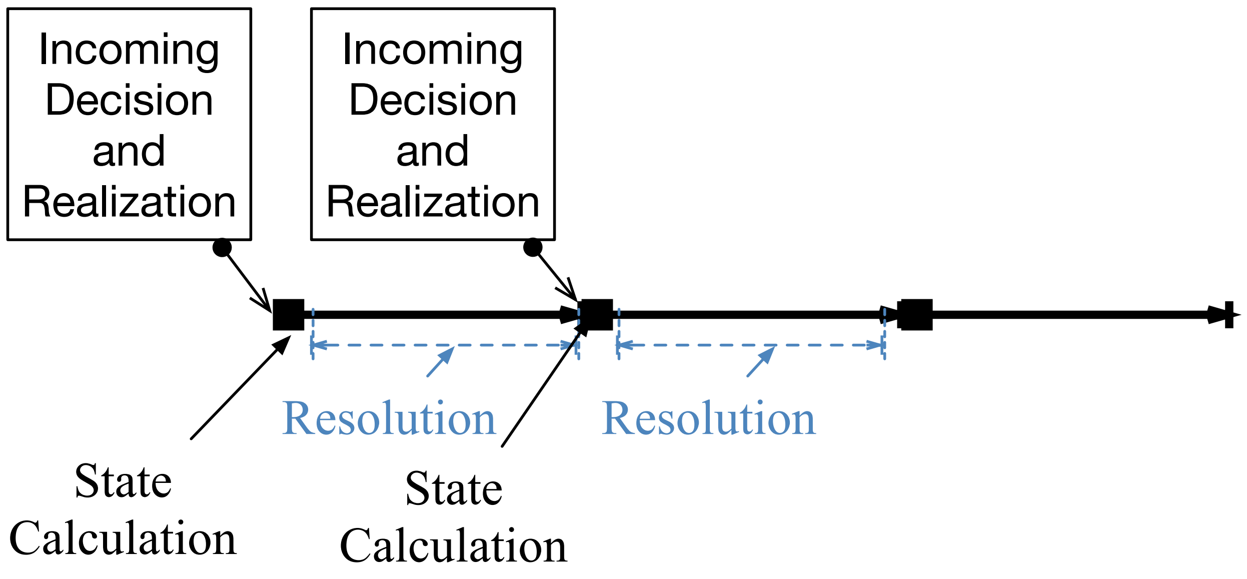

Incoming Decision and Realization
Incoming Decision and Realization
Resolution
Resolution
State Calculation
State Calculation

Figure 3.2: Sequential properties of an emulation model.

model is "myopic" and executed along a single timeline, as shown in Fig. 3.2, since its results are not affected by future system states. At each calculation step, the model uses the incoming decisions and the realization of the uncertain quantities such that the state values at a time t are independent of future system states. For instance, an AGC model can be used as an emulator problem to evaluate the deployment of reserves. Alternatively, an AC power flow can be used to assess the resulting node voltages after a dispatch decision. The emulator also plays a critical role in simulations with inter-temporal constraints since it can better represent the initial conditions in models that require initial condition values like those featuring ramp constraints or storage devices.

Conceptually, the relationship between the decision problem and the emulator problem is akin to the control-plant model commonly used in the field of automatic control. The representation of the system state through an emulation problem is not common practice in PCM. This is due in part to the fact that most commercial PCM tools only support a one- or two-problem simulation framework. Extending an operations simulation to use multiple problems is significantly more challenging. An example of these relationships is shown in Fig. 3.3, which shows a simulation set-up that uses two decision model stages and a AGC emulation problem. In this examples, the first level problem finds optimal decisions with hourly time scales that are later refined at a five-minute resolution and finally the AGC models the system status.

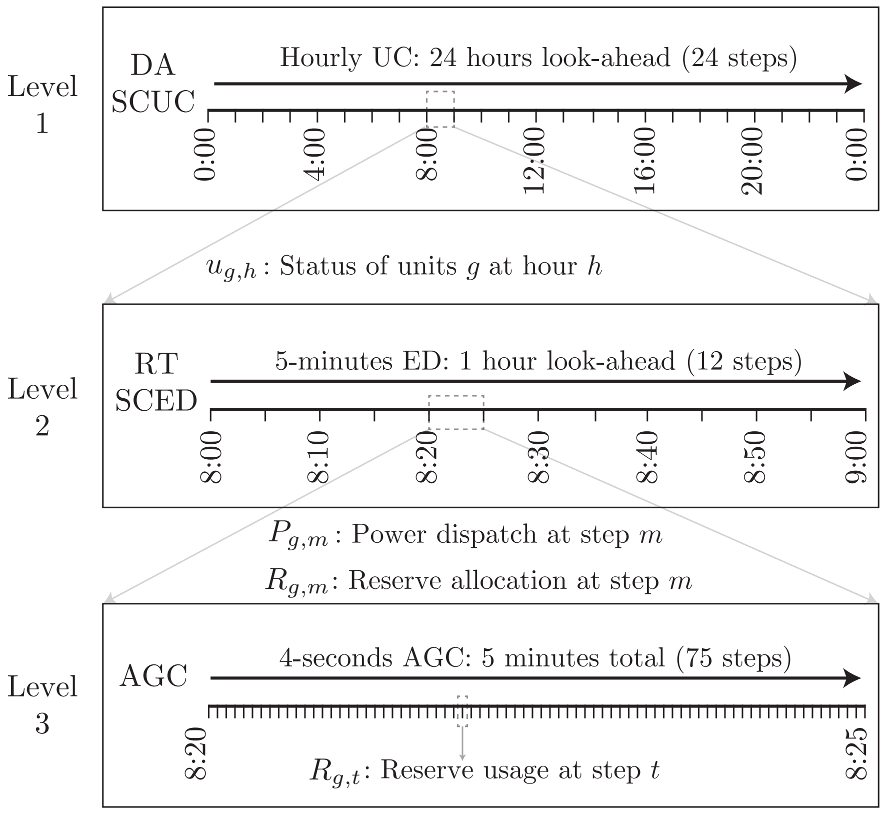

Level 1
DA SCUC
Hourly UC: 24 hours look-ahead (24 steps)
0:00
4:00
8:00
12:00
16:00
20:00
0:00
$u_{g,h}$: Status of units g at hour h
Level 2
RT SCED
5-minutes ED: 1 hour look-ahead (12 steps)
8:00
8:10
8:20
8:30
8:40
8:50
9:00
$P_{g,m}$: Power dispatch at step m
$R_{g,m}$: Reserve allocation at step m
Level 3
AGC
4-seconds AGC: 5 minutes total (75 steps)
8:20
8:25
$R_{g,t}$: Reserve usage at step t

Figure 3.3: Characterization of the multiple timeframes considered in the simulations.

Model of an operations simulation

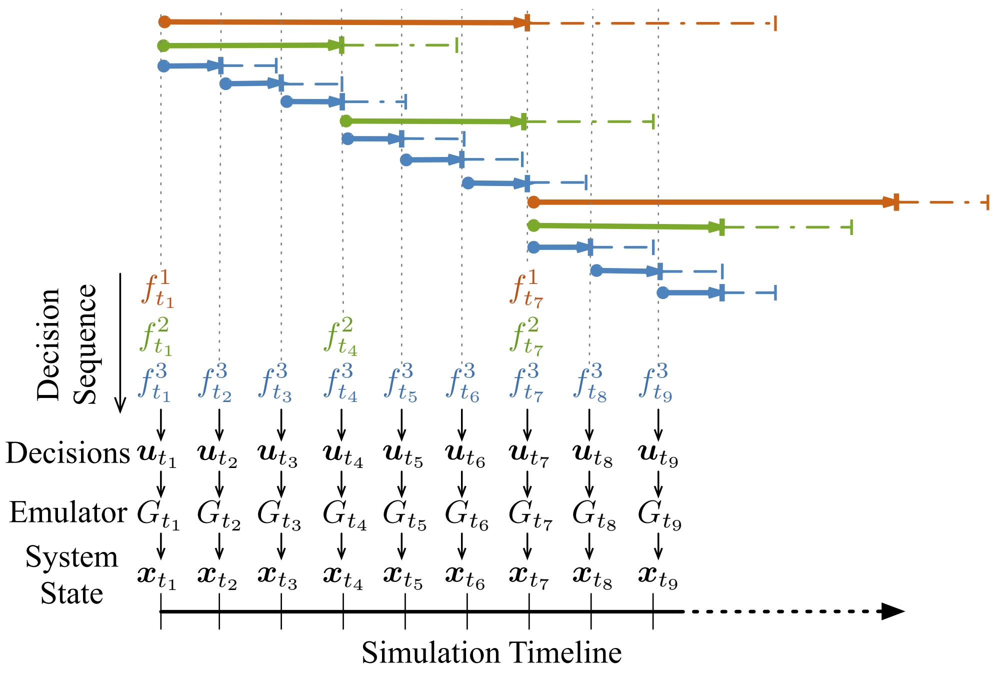
Based on the previous definitions of decision and emulation problems, it is possible to formulate an operations simulation using a discrete time model as follows:

$$\vec{u}_t = F_t(\vec{x}_{t-1}, \vec{u}_{t-1}, \vec{\rho}_t, \Phi|t), \quad \vec{u}_{t_0} = \vec{u}_0 \tag{3.1}$$

$$G_t(\vec{x}_t, \vec{x}_{t-1}, \vec{u}_t, \vec{\psi}_t) = 0, \quad \vec{x}_{t_0} = \vec{x}_0 \tag{3.2}$$

where $\vec{u}_t$ represents the operation decisions conditional on a forecast issued for time t, $\Phi|t$ over a horizon $\mathcal{H}$ and using parameters $\vec{\rho}$. The decision state $\vec{u}_t$ is calculated over a horizon $\mathcal{H}$ such that $\vec{u}_t = \left\{u_{h|t}|\ h \in \mathcal{H}\right\}$ where $u_{t|h}$ represents a decision taken at time t for time-step h. The decision variables are updated by function F_t that represents the sequential solution of decision problems. Function G_t is the emulator problem that updates the system state $\vec{x}_t$ given the decision $\vec{u}_t$ and the realized inputs $\vec{\psi}_t$.

Given (3.1)-(3.2), a simulation can be set up as follows: given an initial condition for a decision and system state, advance in time t from one point to the next considering a discrete timeline $\{t_0, t_1, \ldots, t_n, \ldots, T\}$. A simulation requires a stepping procedure that finds the solution in time t_{n+1} provided the values of the variables at $\{t|t_0 \leq t < t_{n+1}\}$.



Decision Sequence
$f^1_{t_1}$ $f^2_{t_1}$ $f^3_{t_1}$ $f^3_{t_2}$ $f^3_{t_3}$ $f^2_{t_4}$ $f^3_{t_4}$ $f^3_{t_5}$ $f^3_{t_6}$ $f^1_{t_7}$ $f^2_{t_7}$ $f^3_{t_7}$ $f^3_{t_8}$ $f^3_{t_9}$
Decisions $\boldsymbol{u}_{t_1}$ … $\boldsymbol{u}_{t_9}$
Emulator G_{t_1} … G_{t_9}
System State $\boldsymbol{x}_{t_1}$ … $\boldsymbol{x}_{t_9}$
Simulation Timeline

Figure 3.4: operations simulation chronological sequence of calculations.

At each time-step t the sequential solution of decision problems F_t is formulated as a composition of functions over a set $\mathcal{K}_t$:

$$F_t := f^k \circ f^{k-1} \circ \cdots \circ f^1(\vec{x}_{t-1}, \vec{u}_{t-1}, \vec{\rho}_t, \Phi|t) \tag{3.3}$$

each f^k corresponds to solving an optimization model to update the decision states based on the previous decisions $\vec{u}_{t-1}$, the system state $\vec{x}_{t-1}$ and the available forecast $\Phi^k|t$. Figure 3.4 shows an example of the stepping process in a simulation, where the sequence F_t can have between 1 and 3 function evaluations to arrive to the decision value $\vec{u}_t$. Before the next set of decisions are updated the emulation problem is evaluated.

Each function f^k is an optimization model of a sub-set of the decisions $\vec{u}^k_t$ using a sub-set of the forecast data $\Phi^k|t$ as follows:

$$f^k(\cdot) = \min_{\vec{u}^k_t} \quad C_{f_k}(\vec{u}^k_t) \tag{3.4a}$$

$$\text{s.t.} \quad H^D_{f_k}\left(\vec{u}_t, \vec{u}_{t-1}, \vec{x}_{t-1}, \vec{\rho}_t, \Phi^k|t\right) \leq 0 \tag{3.4b}$$

$$H^B_{f_k}\left(\vec{u}_t, \vec{u}_{t-1}, \vec{x}_{t-1}, \vec{\rho}_t, \Phi^k|t\right) \leq 0 \tag{3.4c}$$

$$H^N_{f_k}\left(\vec{u}_t, \vec{u}_{t-1}, \vec{x}_{t-1}, \vec{\rho}_t, \Phi^k|t\right) = 0 \tag{3.4d}$$

$$H^S_{f_k}\left(\vec{u}_t, \vec{u}_{t-1}, \vec{x}_{t-1}, \vec{\rho}_t, \Phi^k|t\right) \leq 0 \tag{3.4e}$$

$$H^F_{f_k}\left(\vec{u}_t, \vec{u}_{t-1}, \vec{x}_{t-1}, \vec{\rho}_t, \Phi^k|t\right) \leq 0 \tag{3.4f}$$

C_{f_k} is the objective function of the model. The constraint function $H^D_{f_k}$ corresponds to the models for the injection devices like generation, loads, and storage. Constraints $H^B_{f_k}$ correspond to the models of the branches like Lines and HVDC, while $H^N_{f_k}$ corresponds to the model of the network like CopperPlate, DC powerflow, or AC powerflow. $H^S_{f_k}$ correspond to the models of the services like reserves or transfers. Finally, $H^F_{f_k}$ are introduced as the *feedforward* constraints.

Feedforwards constraints are used to represent the relationships that the decision variables $\vec{u}_t$ have between the models. For instance, a *feedforward* is commonly used to pass the values of the on/off decision variables from a UC problem into an ED problem. Other examples of *feedforward* constraints include upper-bounds, lower-bounds, and end-of-horizon targets among others. These constraints are not commonly formalized in commercial or open source simulation models despite the fact that their implementation can affect the model's results.

3.2 Software Structure and Description

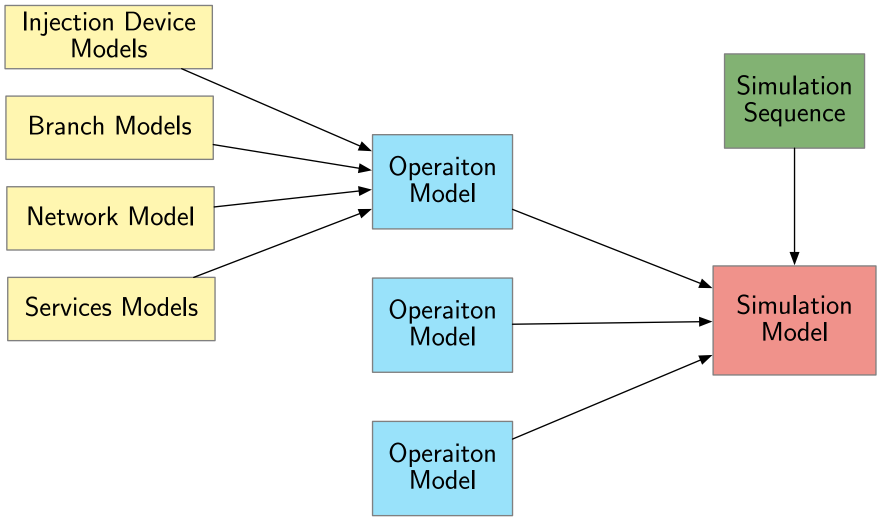


Figure 3.5: Model relationship graph of a simulation.

An implementation of an operations simulation software needs to consider two large workflows: (1) building the optimization model;and (2) handling the sequencing and bookkeeping of the optimization model solutions.

From a software perspective, the challenge in implementing a scalable and flexible simulation framework lies in handling the required data structures. As described in [88], building and solving optimization model within a simulation requires several data translations. Traditionally, PCM simulators employ disk writes to handle these data transactions; at each model solve the software writes a text file to disk that is later solved by the optimizer.

```julia
using PowerSystems
using PowerSimulations

# create default template
tmplt = ProblemTemplate(NetworkModel(DCPPowerModel))

# set generator models
set_device_model!(
 tmplt, ThermalStandard, ThermalStandardUnitCommitment
)
set_device_model!(
 tmplt, RenewableDispatch, RenewableFullDispatch
)

# set load models
set_device_model!(tmplt, PowerLoad, StaticPowerLoad)

# set branch models
set_device_model!(tmplt, Line, StaticBranchUnbounded)
```

Code 3.1: Definition of a problem template. High level specification of H_{f_k} functions

In an optimization context, the limitations of this workflow are further discussed in detail by [44], where the authors detail the implementation of the AML `JuMP.jl` and the comparison with text-interpreted alternatives like GAMS or AMPL. From a simulation perspective, this workflow is inefficient since it implies that the solver garbage collects the incumbent solution that can be used as an initial guess and for models that employ LP or MILP solvers it implies losing matrix factorizations and other intermediate values that can accelerate the solution of the model.

`PSI.jl` addresses the limitations of existing simulation workflows by exploiting the in-memory representation of the optimization model provided by `JuMP.jl` [44] and the underlying abstract solver interface `MathOptInterface.jl` [90].

Optimization Models Build

To minimize issues of model-limited choice, the optimization model building implementation follows two principles:

- **Flexibility:** a change to a component model does not affect the model of an unrelated component
- **Modularity:** separate the individual models of the system components

One critical consideration when building (3.4) programmatically is providing sufficient flexibility for the exploration of a hypothesis while maintaining a certain level of structure such that a software solution can be implemented. In `PSI.jl`, we divide the optimization model in the sub-models described by (3.4) as shown in Figure 3.5 where each simulation is comprised of optimization formulated from component level models and feedforwards.

The specifications of a problem are contained in a *template* that determines formulations for each component. An example of building an operation's problem template is shown in Listing 3.1. In this example the template is based on a DC Powerflow, with the specific models for each component explicitly set. For example, `ThermalStandard` devices are modeled using the formulation `ThermalStandardUnitCommitment`. `PSI.jl` provides a comprehensive and fully documented library of formulations for diverse components in the documentation. To tackle the requirements discussed in [31, 142] and provide users with access to the model details and data, the optimization model objects are stored in the optimization container. This container is serialized to disk in order to guarantee that the simulation can be reproduced at a later time.

The optimization model build process is shown in Figure 3.6. First, if a feasible initial point is not provided, `PSI.jl` builds and solves a relaxed optimization model with a shortened horizon to obtain valid values for $\vec{u}_0$ and $\vec{x}_0$ based on the template. This feature reduces the possibility of having infeasible problems at the start of the simulation due to mismatches in the incoming data. The mathematical description for a device is implemented using Julia's multiple dispatch. The construction methods can be defined based on abstract data structures to enable code re-use and interfacing with other models [16] which enables greater flexibility in modeling choice. `PSI.jl` builds the optimization model using the structure hierarchy defined in `PowerSystems.jl` and the data handling features [89]. For each component type and formulation, `PSI.jl` adds the specified arguments (i.e., variables, parameters and expressions).

`PSI.jl` implements *parameters* that enable the in-memory-modification of the model instead of the common practice of using "dummy" variables. *Parameters* are implemented by keeping in memory the constraint index of the parameter and request precise model modifications without rebuilds. The limitation of this approach that the parameters can only be used to update the right-hand side of linear constraints and linear objective function coefficients. However, for most applications the inputs that change over the course of a simulation such as forecasts, feedforwards, or costs can be readily implemented within this limitation.

After the arguments are built, the network formulation adds the constraints required to model the power flow through the network into the optimization model. The modularity features in `PSI.jl` design allows an seamless integration with `PowerModels.jl` [27] which implements the network model. As a result, users can explore distinct network formulations that have been validated and tested. Next, the device's models are constructed (i.e., adding the constraints into the model) and the objective function is set. After the process concludes, we perform a series of checks to the problem to determine if there are scaling issues or invalid values in the constraints. If the process succeeds, the model is serialized to disk for

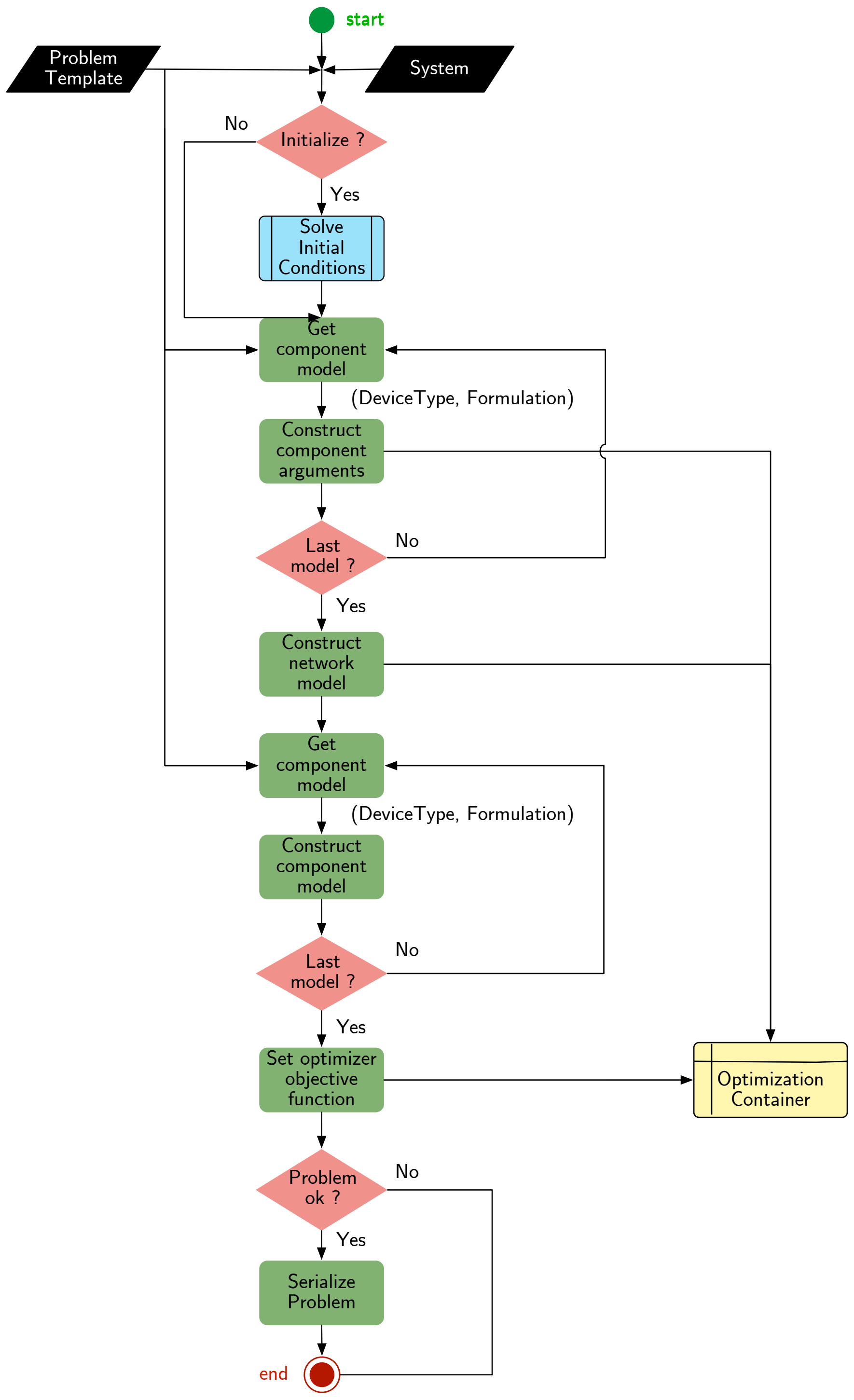


Figure 3.6: Modular Optimization Build Flow Chart.

```julia
using GLPK
day_ahead = DecisionModel(
    UnitCommitmentProblem,
    tmplt,
    System(day_ahead_system_file.json);
    optimizer=GLPK.Optimizer,
    name="Day Ahead",
)

dispatch = EmulationModel(
    template_emulator, #definition not shown
    System(realization_system_file.json),
    name="Dispatch",
    optimizer=optimizer_with_attributes(GLPK.Optimizer),
),
```

Code 3.2: Definition of a UC decision problem based on the template from Listing 3.1 and an emulation problem

record keeping purposes.

Listing 3.2 shows the specification of a decision and emulation models using the template. The design allows modelers to mix and match formulations for the devices according to their needs.

Simulation Sequence

A simulation also requires setting the cadence of the problems' sequential solves and specification of how information regarding $\vec{u}_t$ is exchanged between the decision and the emulator models. In `PSI.jl` the *SimulationSequence* plays three roles in the specification of a simulation: (1) It performs series data validations at all times to guarantee that the intervals, resolution, and horizons in the models have the consistency required to execute a simulation; (2) it is used to set the models' *feedforwards*; and (3) defined the models' initial conditions data sharing (i.e., the *initial conditions chronology*).

Listing 3.3 shows the example of specifying the sequence for a simulation that uses the models defined in Listing 3.2. The example sets a *SemiContinousFeedforward* between the UC problem and the emulator. This feedforward introduces the constraint:

$$\mathrm{ON}_{g,t} P_g^{lb} \leq p_{g,t} \leq \mathrm{ON}_{g,t} P_g^{lb} \quad \forall g \in \mathcal{G}_{th}\ t \in \mathcal{T} \tag{3.5}$$

where $\mathrm{on}_{g,t}$ is a parameter in the model to input the values of $\vec{u}_t$. $p_{g,t}$ are the active power output variables of the set of thermal generators $\mathcal{G}_{th}$ at time t.

Listing 3.3 also shows the explicit handling of the initial conditions. We implement two *chronologies* commonly used in simulations. The *InterProblemChronology* which implies

```julia
models = SimulationModels(
    decision_models=[
        day_ahead
    ],
    emulation_model=dispatch
    ),
)

sequence = SimulationSequence(
    models=models,
    Feedforwards=Dict(
        "Dispatch" => [
            SemiContinuousFeedforward(
                component_type=ThermalStandard,
                source=OnVariable,
                affected_values=[ActivePowerVariable],
            ),
        ],
    ),
    ini_cond_chronology=InterProblemChronology(),
)
```

Code 3.3: Definition of a `SimulationSequence`

that the initial conditions are updated based on the system state and assumes that at the moment to update the decisions the model has knowledge of the state of the system.[2] The other alternative is the *IntraProblemChronology* which uses the results of the decision problems as initial conditions, this chronology's assumption is that the initial conditions in the decision models are estimates resulting from previous solutions of the decision variables and taken without information regarding the system's state.

Simulation

The simulation is then fully specified by the models and previously defined sequence. By allowing the user to arrive to the simulation by instantiating intermediate objects, it is possible to modularize the process and facilitate debugging. Once the model inputs and sequence have been validated, the simulation can be built and executed as shown in Listing 3.4.

The build process of the simulation is shown in Fig. 3.7. Once the models and sequence are validated, the optimization model corresponding to each problem is built following the process in Fig. 3.6. Note that given the in-memory problem modification design, the simulation build process is the only time that the models are built. Once the models are

[2]*InterProblemChronology* is similar to the "interleaved" simulations used in some commercial simulators

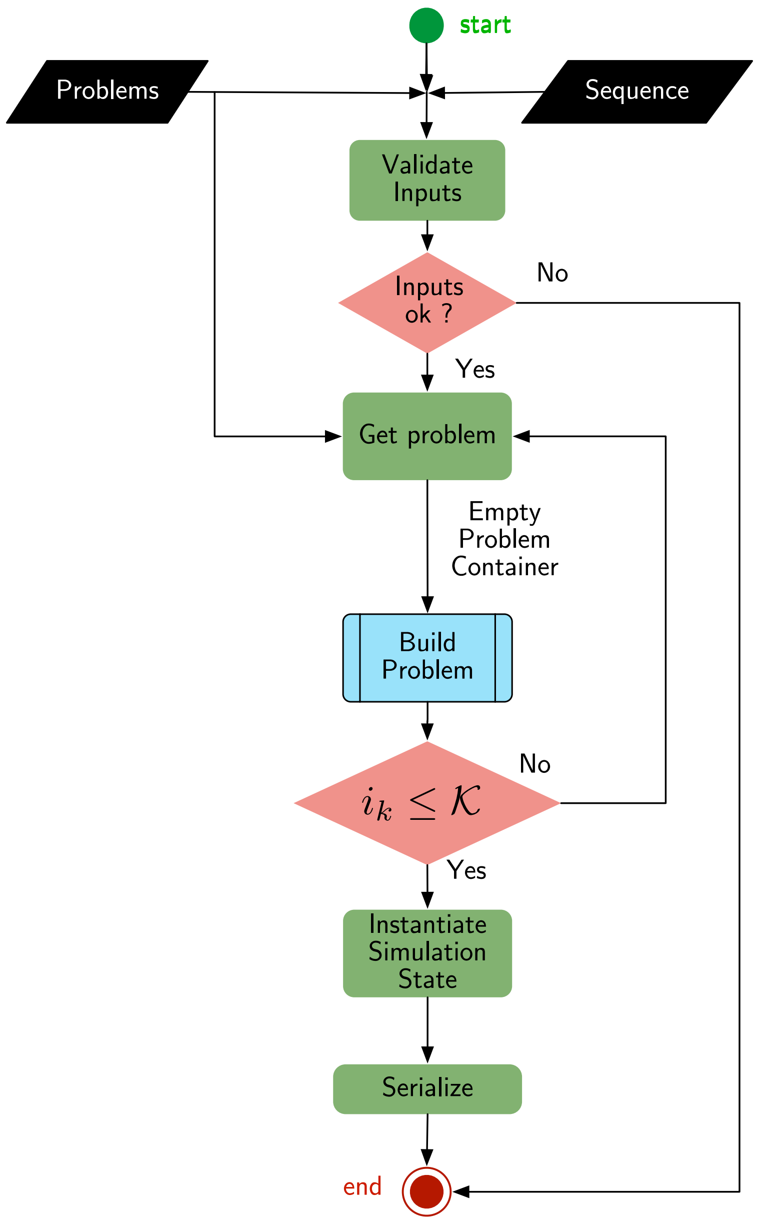


Figure 3.7: Simulation Build Flow Chart.

built and the optimizers instantiated, the simulation state is built by pre-allocating the data structures to keep the values of $\vec{u}$ and $\vec{x}$ in-memory.

The execution flow of the simulation is shown in Fig. 3.8. The first operation in the execution of a simulation is the instantiation of the *results store*. We developed two implementations of a store: in-memory and HDF5. The in-memory implementation is sufficient for situations where the total data size is smaller than system memory and the analyst doesn't need to access the data beyond one working session. For most cases, however, users

```
sim = Simulation(
    name="simulation",
    steps=2,
    models=models,
    sequence=sequence,
    initial_time=DateTime("2020-01-01T00:00:00"),
    simulation_folder=pwd(),
)

build_status = build!(sim)
solve_status = execute!(sim)
```

Code 3.4: Definition of a Simulation and build/solve call

will want to store outputs for the long term and so the data must be written to files. We chose the HDF5 format over alternatives like CSV for these reasons:

- **Supports multidimensional arrays:** The limitation of 2-dimensional data formats limits the implementation of models that require larger structure. For instance, Stochastic Optimization (SO) models where there is a requirements to add a scenario dimension.
- **Supports inline compression:** This features reduces efficient storage of the data at the moment of writing avoiding post-simulation compression steps.
- **Hierarchical storage**: allows storing data in a self-describing way that follows the model tree structure.

Further, to address the aforementioned shortcomings of write-to-disk approaches, `PSI.jl` features a caching layer above the HDF5 file in order to 1) Ensure that all file system writes are at least 1 MiB in size and avoid excessive disk write operations and Provide a read cache for frequently-read outputs.

The instantiation of the `SimulationSequence` pre-calculates the problem solve execution order in each step. For a simulation of N_{steps} at every simulation step, s a total of $N_{problems}$ are solved sequentially as shown in Fig. 3.4.

Prior to executing a model, its parameters and initial conditions ought to be updated to match the simulation timestamp. The models' time-series updating uses the caching mechanism from `PowerSystems.jl` described in Chapter 2 to reduce latency and memory overhead. The feedforward parameters read the latest information from the system state. Once the parameters are updated, the optimizer is executed. After each optimizer successfully reaches a termination condition, the results are first written into the store and the system states updates based on the solution of the optimization model results and the cache of solutions from other models.

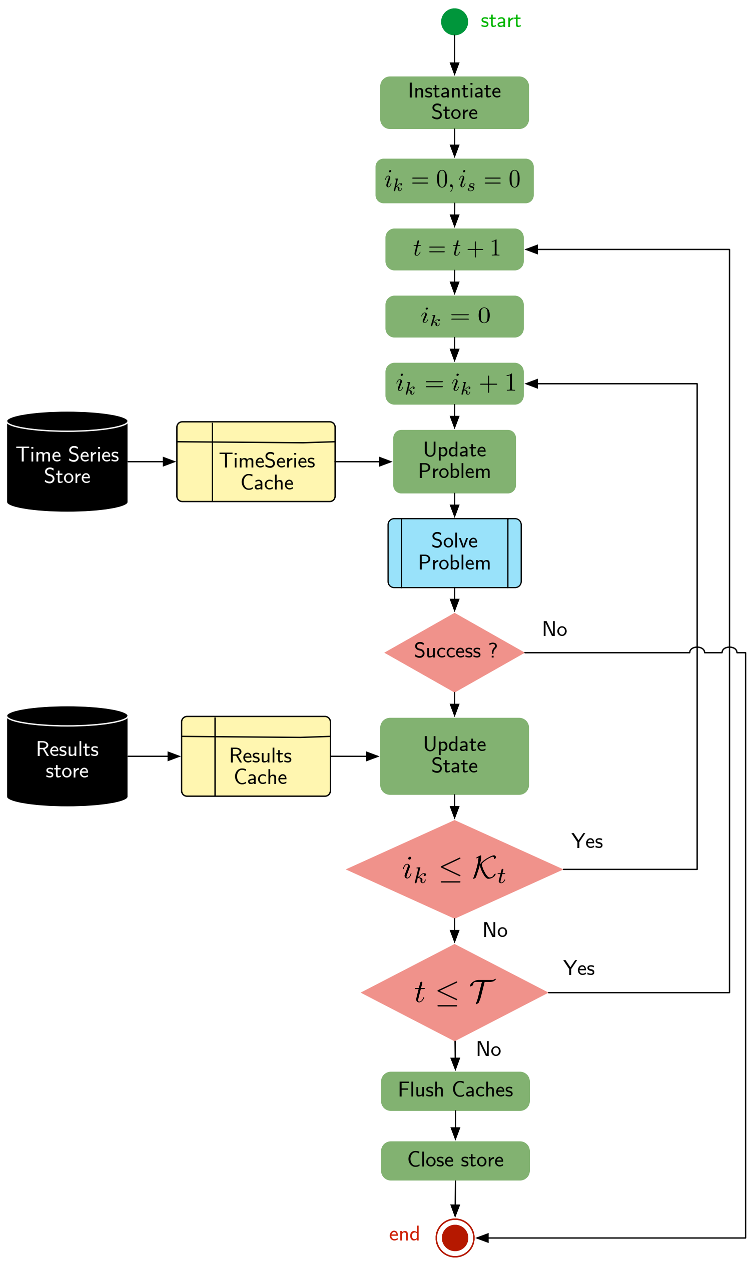


Figure 3.8: Simulation Execution Process Flow Chart.

Results storage and post-processing

All of the objects generated during the simulation build and solution are stored such that the simulation can be reproduced as originally specified. The folder structure post-simulation has been designed such that details about the operations, code logs and files can be analyzed after the simulation is finished.

3.3 Validation and Case Studies

This section presents comparisons of simulation using `PSI.jl` and the PCM simulation `PLEXOS`. These simulation results do not employ an emulation model since this is a feature not present in commercial tools, the implementation of multi-level simulation with emulation is implemented in sub-section 3.4. In this section, we show the verification of results using a modified version of the PJM five-Bus system [94] and a larger scale verification using the updated RTS-96 Bus system [7] where we compare with `PLEXOS` the output of the decision using a simulation with two operation problems [3].

Five-Bus Case

For validation purposes, the 5-bus system depicted in Figure 3.9 will be used to compare the resulting outputs of `PSI.jl` and `PLEXOS`. The system features the five original coal generation units, one solar and one wind power plants. The system transmission network has been further restricted with respect to the original to force the commitment of units in buses D and C.

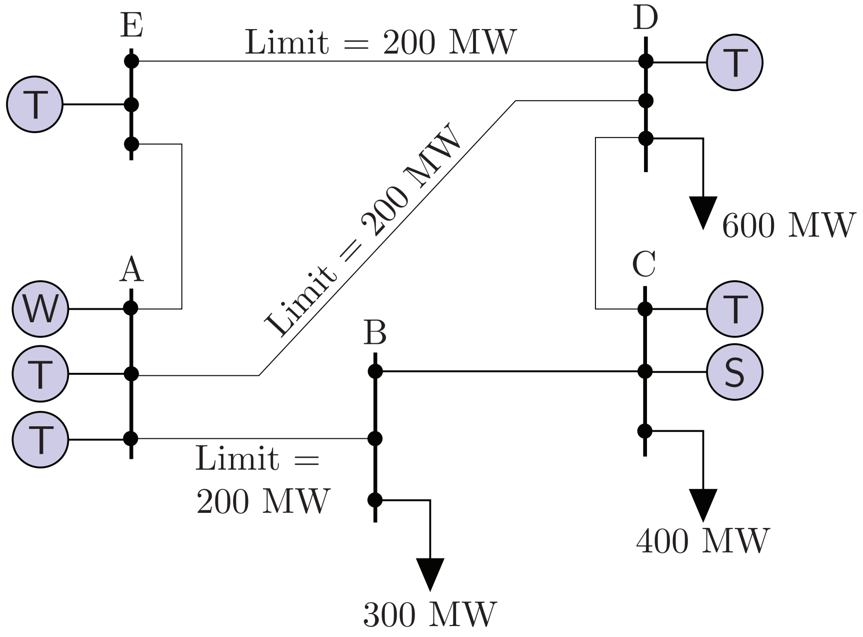


Figure 3.9: Modular Optimization Build Flow Chart.

Firstly, for this system we conducted a single level PCM simulation using a UC model to verify the results between `PSI.jl` and PLEXOS. The model employed a PTDF network model and allowed curtailing power from the VRE resources. Table 3.1 shows the total

[3]Source code and results are available at https://github.com/NREL-SIIP/PowerSimulationsTPWRS

operating cost per generator and total VRE for the simulation period. The results of the single level models where `PSI.jl` matches the results from `PLEXOS` within the numerical tolerances of the solvers. Figure 3.10 shows the fuel stack for the single level simulation.

Secondly, we conducted a simulation using a two-level model featuring DC and AC networks to show the relevance of the network assumption in operations simulations and the easiness to switch network models in `PSI.jl`. The simulations were conducted using the open-source solver `HiGHS` [72] using the default solve parameters.

Figures 3.10 and 3.11 show the fuel stacks for the simulation using a two-model strategy.

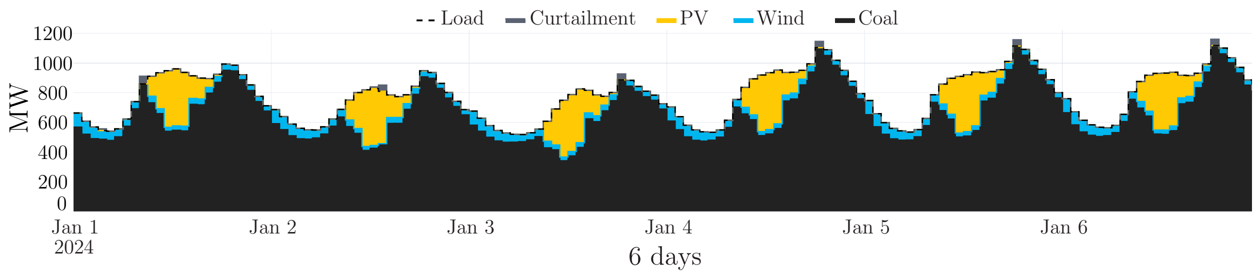


Figure 3.10: Fuel Stack for the simulation of 5-bus system DC Power Flow UC.

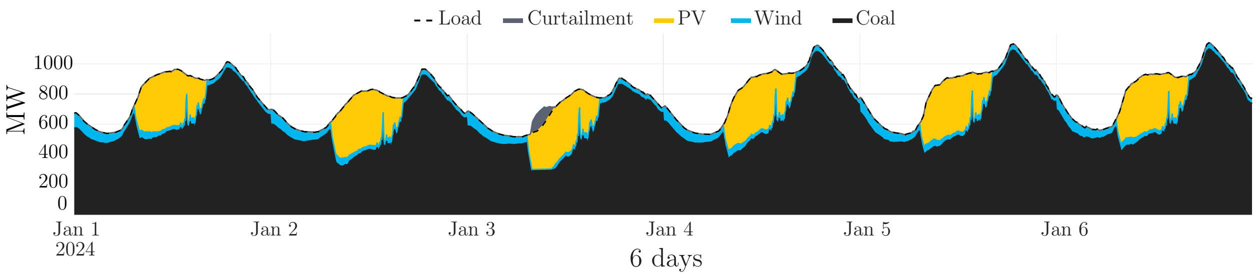


Figure 3.11: Fuel Stack for the simulation of 5-bus system DC Power Flow ED.

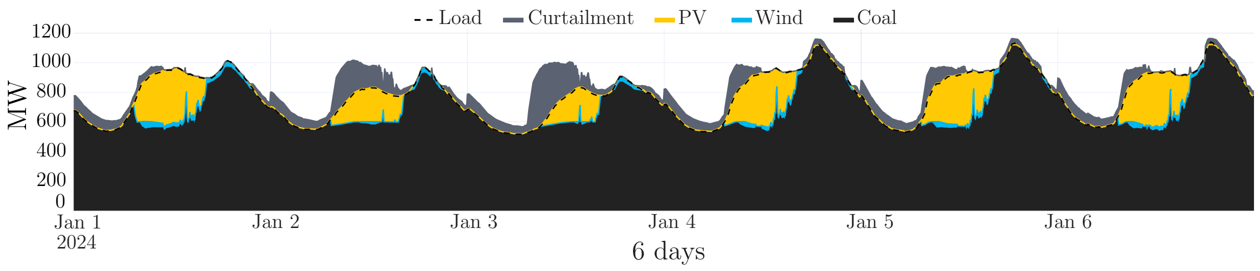


Figure 3.12: Fuel Stack for the simulation of 5-bus system AC Power Flow ED.

The cost results in Table 3.1 show that the forecast error between the first level and the second level reduces the overall generation with Coal significantly and the system is

Table 3.1: Results comparison between simulation cases for 5-bus case

Variable	PLEXOS	PSI UC	PSI UC - ED DC	PSI UC- ED AC
Sundance [$]	98980.1	98980.3	80345.3	90597
Solitude [$]	1.24171×10^6	1.24177×10^6	8.74726×10^5	9.30366×10^5
Park City [$]	0.0	0.0	0.0	0.0
Alta [$]	2276.25	2261.7	2268.31	7070.3
Brighton [$]	5.14102×10^5	5.14102×10^5	5.93361×10^5	6.61518×10^5
Renewables [MW]	17824.95	17826.08	26626.84	18280.63

able to integrate more VRE. However, this result is predicated on the assumption of DC power transmission, once the assumptions about the network are switched to AC power, there is a significant reduction in the amount of VRE integrated with respect to the DC assumption. Note that a large amount of the curtailment when the network is modeled in AC comes from the inability of the system to transmit power between areas of large amount of generation and the loads which causes a lot of solar power to just be utilized to feed loads localized in Buses D and C.

96 Bus RTS Case

The second validation case is the RTS-system originally published in [7] which is at the moment of writing the only published PCM dataset actively maintained and with input data for other modeling platforms. We compare the simulation with `PLEXOS` “interleaved” mode which is meant to perform a simulation where the optimization problems inform each other’s initial conditions. Given that this data set features Piece-wise Linear (PWL) cost functions and several units share the same slopes, there is more propensity for numerical degeneracy. Hence, we will validate the results based on the solver’s objective function and broad generator type dispatch quantities. The simulation is set with a `CopperPlatePowerModel` and over a 14 day period between July 1st and July 14th. The simulation set-up uses a 15-minute interval in the ED as an approximation of the real-time stage in the simulation which is a common practice. The simulations in `PLEXOS` use the solver Xpress version 8.0.1 while `PSI.jl` uses Gurobi 9.0.1, in both cases the MIP gap tolerance was set to $1 \times 10^{-}3$. In this simulation it was not possible to set-up the same solver due to licensing restrictions of the commercial solvers.

The output results show the results of a simulation under the assumption that the ED outputs correspond to the realization of the system. Figures 3.13 and 3.14 show the fuel plot for the UC and ED decision stages of the simulation where the additional variability of the VRE can be observed more clearly in the July 7 to July time frame.

Table 3.2 shows that both software has very close results in the UC stage in terms of the total cost and the fuel choices. In the case of the ED, the comparisons about costs are more difficult to assess due to reporting differences between the software; however, the resource

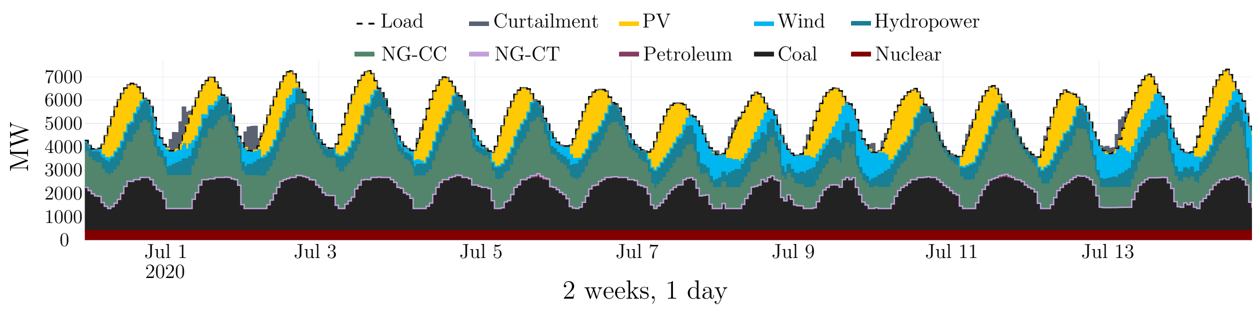


Figure 3.13: Fuel Stack for the simulation of RTS system UC.

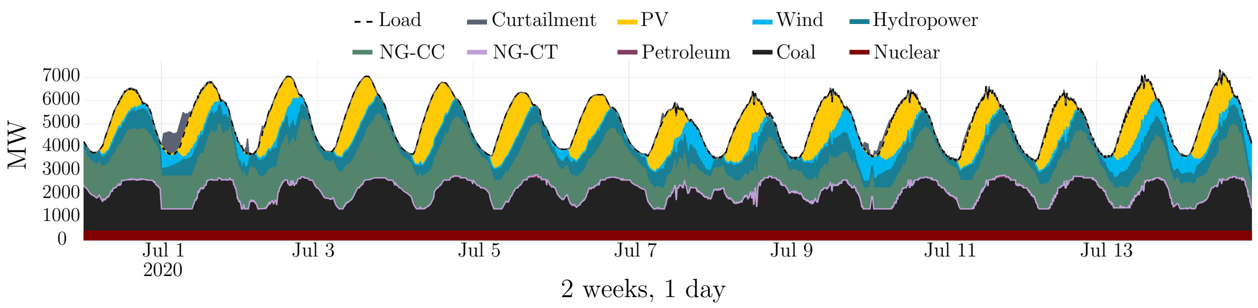


Figure 3.14: Fuel Stack for the simulation of RTS system ED.

Table 3.2: Results comparison between simulation cases for RTS system case

Variable	PLEXOS UC	PLEXOS ED	PSI UC	PSI ED
Gas - CC [MW]	5.26228×10^5	5.06635×10^5	5.29018×10^5	5.05629×10^5
Combustion Turbine [MW]	1751.0	1807.75	1651.0	1550.35
Hydropower [MW]	2.21594×10^5	2.21534×10^5	2.21594×10^5	2.21534×10^5
Nuclear [MW]	1.44×10^5	1.44×10^5	1.44×10^5	1.44×10^5
Steam [MW]	6.15146×10^5	6.142×10^5	6.13056×10^5	6.14076×10^5
Renewables [MW]	3.90612×10^5	3.53815×10^5	3.90013×10^5	3.54877×10^5
Total Cost [$]	5.56758×7	2.708285×7[†]	5.56247×7	2.68851×7[†]

[†]The total cost comparison for the ED stage is done for the fuel cost only due to the reporting from PLEXOS

utilization quantities are within the expected orders of magnitude. It is worth noting that the addition of a second stage makes it more challenging to compare simulation tools since PLEXOS conducted "model repairs" during the simulation, which implies the relaxation of some constraints without too much information about the procedure specifics. On the other hand, PSI.jl did not result in an infeasible solution, nor did it make any modification to the base formulations. Given that the objective of PSI.jl is not to replicate the results from PLEXOS, the results from Table 3.2 can be considered acceptable in terms of verification that the outcomes of a PSI.jl simulations are reasonably similar to other simulators.

3.4 Implementing `PowerSimulations.jl` Automatic Generation Control (AGC) simulations

The results in Section 3.3 showcases the capabilities of `PSI.jl` to model power systems operations under conditions commonly used in practical analysis. However, these results do not exploit the decision-emulation model framework. One of the reasons analyses with emulation models are not frequently used is due to the limited availability of emulation models. In this section, we introduce a model developed explicitly as an emulation problem for the evaluation of reserve deployment mechanisms. The model and simulation developed in this section follows from the concepts developed in Section 3.1.

The increased integration of variable VRE resources in the electricity system introduces several operational challenges for balancing electricity supply and demand which is normally handled through the deployment of reserves. A primary issue is the allocation and deployment of Frequency Regulation Reserve (FRR). In reserve allocation studies, it is essential to consider the FRR deployment mechanisms as well as an estimation of the potential frequency deviations [68]. AGC is used during normal operations when frequency deviations are relatively small ($< 1\%$). These types of studies are a clear candidate for the use of the decision- emulation-model framework in `PSI.jl` where the decision models allocate reserves and the AGC model is used to emulate the deployment of the reserves.

Under a normal operations, if the system cannot be balanced, the ACE increases, and if sustained, persistent imbalances carry penalties [126]. Ensuring adequate AGC simulations is critical for providing a comprehensive evaluation of FRR models — a topic which is rarely evaluated in the literature. Assessing the reserve allocation requires an approach more comprehensive than an instantaneous power balance. If the prevalent source of reserve dynamics is slow, undesirable frequency excursions may occur. But if the prevalent source of reserve dynamics is fast, excess allocation can lead to inefficient costs.

AGC simulations have been used to evaluate the impact of large shares of VRE in several studies. In [102], the authors use an instantaneous power balance calculation of ACE to simulate potential AGC operations and assess FRR needs in CAISO. A more detailed model of AGC is presented in [47] where the authors present several "*control modes*" for the ACE, including a Proportional Integral (PI)-based signal, similar to the one found in [100]. The approach in [47] relies on a heuristic forward-stepping algorithm that resemble a first-order Euler method, thus intertwining the simulation model with the solution algorithm. Wang et al. [186] use the same model [47] to assess the value of improved forecasting under the assumption that CPS2 violations result from reserve saturation. The AGC model described in [4] uses a simplified energy balance model to study long-term stability approximated using algebraic equation; however, there is no integration with upstream dispatch models.

Existing models include simplifications about the limitations of regulating units. Such units respond to AGC commands, which in turn can induce generator saturation and require additional reserve deployments [158, 101]. Existing models do not take into account reserve deployment mechanisms, such as participation factor assignment and reserve re-

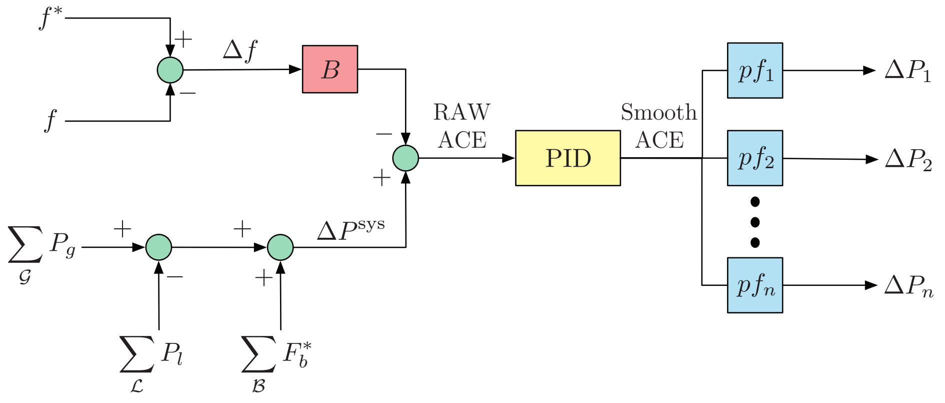


Figure 3.15: AGC control block diagram.

deployment due to saturation within the balancing area. Such models also do not consider the impact of FRR shortfall. Finally, these models do not distinguish between the AGC model and the algorithm used to simulate the system, eliminating the possibility of using LP solvers.

This section presents a model to address these limitations, proposing an AGC simulation model that incorporates mechanisms to account for reserve deployment and generator saturation. The proposed model in this chapter uses an LP formulation to enable flexible algorithm choices to obtain fast solutions and be easily incorporated into a UC and ED simulation. The proposed discrete AGC simulation model can approximate the effects of frequency deviations, limited regulation capabilities, and the interactions with PID control.

In this section the notation is as follows: variables with (t) represent continuous time variables, while variables with the subscript t represent discrete time variables. Δt and Δh indicate the resolution of the AGC and ED discretization, respectively. $\mathcal{G}_t$ represents the sets of available generators at time t, and $\mathcal{T}$ the set of time steps in the AGC simulation. We denote $\forall g \in \mathcal{G}_t, \forall t \in \mathcal{T}$ as $\forall g, t$. The vector notation $\vec{x}$ has been dropped in this section to aid with the visual presentation.

AGC modeling overview

The primary goal of AGC is to maintain (near) instantaneous balance between supply and demand by responding to the ACE signal. The secondary goal of the AGC is to minimize system operational costs [74]. ACE has two components: to account for the frequency error within the balancing area, and another to account for the tie line bias [15]. In systems with large shares of VRE, the power error results in deviations from scheduled interchanges and, if sufficiently severe, significant deviations from nominal frequency.

The instantaneous ACE for a balancing area is composed of two terms [158, 100]:

$$\text{ACE}(t) = \left[P^{\text{export}}_{\text{actual}}(t) - P^{\text{export}}_{\text{scheduled}}(t)\right] - 10B(f(t) - f_s) \tag{3.6}$$

Where $\left[P^{\text{export}}_{\text{actual}}(t) - P^{\text{export}}_{\text{scheduled}}(t)\right]$ represents the deviations between scheduled power transfers between balancing areas, and $10B(f(t) - f_s)$ represents the system-wide power imbalance due to deviations from the system frequency with respect to its nominal value f_s. In this chapter we focus on a single area implementation of AGC, thus $P^{\text{export}}_{\text{actual}}(t) = 0$ and $P^{\text{export}}_{\text{scheduled}}(t) = 0$.

AGC control

It is important to note that the ACE cannot be used directly for control and real AGC systems contain several filters and dead-bands to avoid control reactions to fast variations [102, 158]. As a result, AGC simulation models introduce the definition of the SACE, calculated as a function of ACE such that the resulting signal will steer the ACE towards zero while filtering fast changes [158, 15]. Classic models consider an integral function:

$$\text{SACE}(t) = -K_i \int \text{ACE}(t)dt. \tag{3.7}$$

but realistic AGC systems can feature different PID structures and include extra signal smoothing mechanisms. Detailed descriptions of alternate structures are shared in [74].

AGC operates at a time resolution of 2-6 seconds. At such short time scales there are several modeling challenges, including: possible crossovers to dynamic behavior modeling of the prime movers, consistent data inputs from UC and ED models, and limited guidance about modeling requirements. At a 4-second resolution, AGC can be modeled in continuous time or as a quasi-steady state depending on the level of detail required (e.g., stability analysis vs. reserve deployment).

For the analysis of VRE integration and FRR deployment, AGC modeling is often replaced by a discrete model, and the ACE is simplified to the system's net power balance [47, 186]. In existing models, frequency effects and dynamics are ignored.

Frequency Regulation Reserve Deployment Mechanisms

The design and implementation of an AGC simulation model should also consider the characteristics of the power system regulation fleet, as well as the types and nature of the energy transactions used to allocate regulation reserves. The sequence of function that set the system's commitment and reserve allocation inform the devices deployment responsibilities of the (see Fig. 3.3). In this case study, the characteristics we focus on individual generator participation factors and saturation.

$$P^*_g(t) = P^{\text{ED}}_g + \text{pf}_g\text{SACE}(t) \tag{3.8}$$

until the economic dispatch base-points P^{ED}_g and the participation factors pf_g are updated. This $P^*_g(t)$ is the input to the prime mover, which does not necessarily represent the same instantaneous generator output.

Participation factors are optimized in the generation scheduling model. These are not discussed in previous contributions about AGC simulation models [47, 101], but in practice they are used to allocate the SACE in the regulating units. Participation factors are therefore a critical aspect to model because they determine if any given generator will reach its operational limit (i.e., saturate), which would trigger the re-deployment of reserves from non-saturated generators. The proposed model defines supplementary reserves to account for the required re-deployments induced by saturation.

Given that the SACE can take positive or negative values depending on the regulation direction, the term $\text{pf}_g\text{SACE}(t)$ can also be positive or negative, signaling upward or downward reserve deployments, respectively. In many balancing areas (e.g., ERCOT), generators are assigned upward pf_g^{up} and downward pf_g^{dn} participation factors according to the direction of the deployed regulation reserves.

Generator saturation is another important feature to consider. If a generator is bounded by an operational limit when deploying reserves, which might occur if the power mismatch variations cannot be met with the set participation factors, the remaining units need to cover the shortfall up to their limits. Conversely, if all generators operate at saturation, there will not be enough system-wide regulatory capacity to cover the power mismatch (whether the imbalance reveals a deficit or an excess). When all regulating devices exhaust their capabilities prior to the balance being restored, the missing power must be delivered from neighboring networks or induce a more significant steady-state deviation in the frequency (Δf).

AGC Simulation Model Description

The quasi-steady-state approach assumes that if a sufficiently large simulation step is used, fast dynamics have attained an equilibrium. In the proposed model, we assume that the local prime mover dynamics reach equilibrium by the end of AGC step based on the results in [187]. Given that the model's objective is to simulate the FRR deployment, simulation of prime mover dynamics are not required leading to a reduction of variables and equations. The model is presented here for a single area case in order to focus on the relationship of control mechanisms and generator models. For simplicity, power losses induced by system imbalances are not considered in this chapter.

AGC Control Model

Given the quasi-steady-state assumption, frequency is represented by its steady state model. Let $\mathcal{G}$ denote the set of active generators with droop parameter R_g and let D denote the load-frequency damping. The steady-state frequency deviation Δf_t stemming from the system-wide power deviation ΔP_t^{sys}, which only includes the frequency response of the governor's droop and the area's load damping, is computed as:

$$\Delta f_t = \frac{-\Delta P_t^{\text{sys}}}{\sum_{g\in\mathcal{G}} \frac{1}{R_g} + D}, \qquad \forall t \in \mathcal{T}, \tag{3.9}$$

ΔP_t^{sys} is calculated using time-series differences between generated power $P_{g,t}$, the load $P_{l,t}$, and a FRR deployment R_t:

$$\Delta P_t^{\text{sys}} = \sum_{g\in G} P_{g,t} - \sum_{l\in L} P_{l,t} + R_t, \qquad \forall t, \tag{3.10}$$

equation (3.10) assumes that generators can deploy the reserves R_t in the same AGC cycle.

Based on eqs. (3.9) and (3.10), the discrete version of the ACE for a single area is defined as follows:

$$\text{ACE}_t = -10B\Delta f_t, \qquad \forall t. \tag{3.11}$$

In power systems literature, B is assumed to be equivalent to $\frac{1}{10}\sum_{g\in\mathcal{G}} \frac{1}{R_g} + D$, which makes $\text{ACE}_t = \Delta P_t^{\text{sys}}$. From a steady state perspective, the terms $10B$ in (3.11) and $(\frac{1}{R_g} + D)^{-1}$ in (3.9) have no effect on reserves, since the control will always try to zero out the ACE. However, a good estimate of frequency deviations requires a properly calibrated ratio between the AGC bias and the area's total droop response and damping.

Similar to the continuous time model, a PID is used to drive the ACE signal to zero. The PID is discretized using the step length of the AGC (Δt). The model corresponds to a incremental algorithm that is the most common implementation of discrete PID with integral terms in micro controllers, not prone to error accumulation, thus avoiding the issue of anti wind-up reset [3].

$$\text{SACE}_t = \text{SACE}_{t-1} + K_p\left[\left(1+\frac{\Delta t}{T_i}+\frac{T_d}{\Delta t}\right)\text{ACE}_t + \left(-1-\frac{2T_d}{\Delta t}\right)\text{ACE}_{t-1} + \frac{T_d}{\Delta t}\text{ACE}_{t-2}\right], \forall t. \tag{3.12}$$

Following the same relationship as in (3.8), the participation factors are used to determine the regulation device reserve deployment. An additional term $\Delta Pe_{g,t}$ is included to account for supplementary reserve deployment due to saturation:

$$\Delta P_{g,t} = \text{SACE}_t \cdot \text{pf}_{g,t} + \Delta Pe_{g,t}, \quad \forall g, t. \tag{3.13}$$

If all the generators operate within their limits, $\Delta Pe_{g,t} = 0$. However, if a generator saturates, $\Delta Pe_{g,t} > 0$ for the remaining generators as they are required to deploy reserves beyond their initial target $\text{SACE}_t \cdot \text{pf}_{g,t}$.

Traditionally, the participation factors are set using economic criteria that are dependent upon the relationships between ancillary services bids and energy markets bids. To approximate this allocation process, participation factors are determined proportional to the reserve allocations from the UC [15, 158].

The generator's set-point at each time step $P^*_{g,t}$ is calculated by adding the reserve deployment to the interpolation between the ED base-points $(P^{\text{ED}}_{g,h}, P^{\text{ED}}_{g,h+1})$:

$$P^*_{g,t} = \hat{P}_{g,t} + \Delta P_{g,t} \qquad \forall g \in \mathcal{G}^{\text{agc}}, \forall t \tag{3.14}$$

$$\hat{P}_{g,t} = P^{\text{ED}}_{g,h} + (t-h)\frac{P^{\text{ED}}_{g,h+1} - P^{\text{ED}}_{g,h}}{\Delta h}, \qquad \forall g, t. \tag{3.15}$$

Equations (3.13)-(3.14) only apply for generators participating in the AGC. Other generation units are modeled using only the interpolation values as in Eq. (3.15):

$$P^*_{g,t} = \hat{P}_{g,t} \qquad g \in \mathcal{G}^{\text{no-agc}}, \forall t. \tag{3.16}$$

The total system reserve deployed by the AGC is defined as the sum of the scheduled deployments and the re-deployments:

$$R_t = \sum_{g \in \mathcal{G}^{\text{agc}}} \Delta P_{g,t} + \sum_{g \in \mathcal{G}^{\text{agc}}} \Delta Pe_{g,t}, \qquad \forall t. \tag{3.17}$$

If $|R_t| < |\text{SACE}_t|$, then there is a reserve shortfall u_t, computed as:

$$-\text{SACE}_t = R_t + u_t, \qquad \forall t, \tag{3.18}$$

on which the variable u_t serves as a system-wide slack variable, and represents the size of the imbalance that *would* have been covered, were it not for saturation.
Remark: If there is enough reserve and no generator is saturated, then $u_t = \Delta Pe_{g,t} = 0$ for all time steps. In such case, the model is equivalent to the classic AGC model that allocates its regulation only via participation factors. Also note that, because reserves are deployed according to SACE and not ACE, there is always some system imbalance even if $u = 0$.

Generator model

Each generator participating in the AGC has a limit on how much FRR they can provide. The regulation capability model is defined by limits on the regulation allocated by the UC and ED solutions and set-point limitations:

$$-\overline{R}^{\text{dn}}_{g,t} \le \Delta P_{g,t} \le \overline{R}^{\text{up}}_{g,t}, \qquad \forall g, t \tag{3.19}$$

$$(P^{\text{min}}_g - \hat{P}_{g,t}) \le \Delta P_{g,t} \le P^{\text{max}}_g - \hat{P}_{g,t}, \qquad \forall g, t \tag{3.20}$$

$\overline{R}^{\text{up}}_{g,t}$ and $\overline{R}^{\text{dn}}_{g,t}$ in (3.19) represents the reserve allocation from the UC/ED problems. The upper and lower bounds on (3.20) are computed based on the operational limits and base-points defined via the economic dispatch problem. For ramp-constrained generators such as thermal units, which are denoted by the set $\mathcal{G}^{\text{r}}_t$, additional constraints reflecting upwards $\overline{\text{RUP}}_{g,t}$ and downwards $\overline{\text{RDN}}_{g,t}$ maximum response rates are also considered:

$$-\overline{\text{RDN}}_{g,t} \le \Delta P_{g,t} \le \overline{\text{RUP}}_{g,t}, \qquad \forall g \in \mathcal{G}^{\text{r}}_t, \forall t \tag{3.21}$$

Optimization-based implementation

Because regulation can take upwards and downwards directions, the total regulation is split into four variables as $R_t = (R_t^{\text{up}} - R_t^{\text{dn}}) + (Re_t^{\text{up}} - Re_t^{\text{dn}})$, on which $R_t^{\text{up}}, R_t^{\text{dn}}, Re_t^{\text{up}}, Re_t^{\text{dn}} \geq 0$. Similarly, the regulation for each generator participating in the AGC is also split for the regular reserve and supplementary reserve $\Delta P_{g,t} = \Delta P_{g,t}^{\text{up}} - \Delta P_{g,t}^{\text{dn}}$, and $\Delta Pe_{g,t} = \Delta Pe_{g,t}^{\text{up}} - \Delta Pe_{g,t}^{\text{dn}}$ respectively.

The reserve provision balance represented by Eqs. (3.13), (3.17) (3.18) are codified as follows:

$$R_t^{\text{up}} - R_t^{\text{dn}} + Re_t^{\text{up}} - Re_t^{\text{dn}} = -\text{SACE}_t - u_t, \qquad \forall t \tag{3.22}$$
$$\Delta P_{g,t}^{\text{up}} = \text{pf}_{g,t}^{\text{up}} \cdot R_t^{\text{up}} + \Delta Pe_{g,t}^{\text{up}}, \qquad \forall g,t \tag{3.23}$$
$$\Delta P_{g,t}^{\text{dn}} = \text{pf}_{g,t}^{\text{dn}} \cdot R_t^{\text{dn}} + \Delta Pe_{g,t}^{\text{dn}}, \qquad \forall g,t \tag{3.24}$$
$$R_t^{\text{up}} = \sum_{g \in \mathcal{G}_t} \Delta P_{g,t}^{\text{up}}, \quad R_t^{\text{dn}} = \sum_{g \in \mathcal{G}_t} \Delta P_{g,t}^{\text{dn}}, \qquad \forall t \tag{3.25}$$
$$Re_t^{\text{up}} = \sum_{g \in \mathcal{G}_t} \Delta Pe_{g,t}^{\text{up}}, \quad Re_t^{\text{dn}} = \sum_{g \in \mathcal{G}_t} \Delta Pe_{g,t}^{\text{dn}}, \qquad \forall t \tag{3.26}$$

where Re_t^{up} and Re_t^{dn} are the components of R_t that represent the total supplementary reserve deployments.

The generator model (3.19)-(3.21) must also be updated for the non-negative upwards and downward variables:

$$\Delta P_{g,t}^{\text{up}} \leq \overline{R}_{g,t}^{\text{up}}, \qquad \forall g,t \tag{3.27}$$
$$\Delta P_{g,t}^{\text{up}} \leq P_g^{\max} - \hat{P}_{g,t}, \qquad \forall g,t \tag{3.28}$$
$$\Delta P_{g,t}^{\text{dn}} \leq \overline{R}_{g,t}^{\text{dn}}, \qquad \forall g,t \tag{3.29}$$
$$\Delta P_{g,t}^{\text{dn}} \leq \hat{P}_{g,t} - P_g^{\min}, \qquad \forall g,t \tag{3.30}$$
$$\Delta P_{g,t}^{\text{up}} \leq \overline{\text{RUP}}_{g,t}, \qquad \forall g,t \tag{3.31}$$
$$\Delta P_{g,t}^{\text{dn}} \leq \overline{\text{RDN}}_{g,t}, \qquad \forall g \in \mathcal{G}_t^{\text{r}}, \forall t \tag{3.32}$$

Supplementary reserves are added to the objective function with a cost proportional to the inverse of the participation factor $\gamma_{g,t}$. This formulation forces the supplementary reserve deployment to follow an allocation similar to the one originally established:

$$\gamma_{g,t} = \begin{cases} 1/\text{pf}_{g,t}, & \text{if } \text{pf}_{g,t} \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.33}$$

Finally, the absolute value of the imbalance $|u_t|$ is added to the objective function with a sufficiently high cost c to force the problem to push the reserves shortfall to zero. The problem is cast as a LP by properly replacing the absolute value $|u_t|$. In summary, the

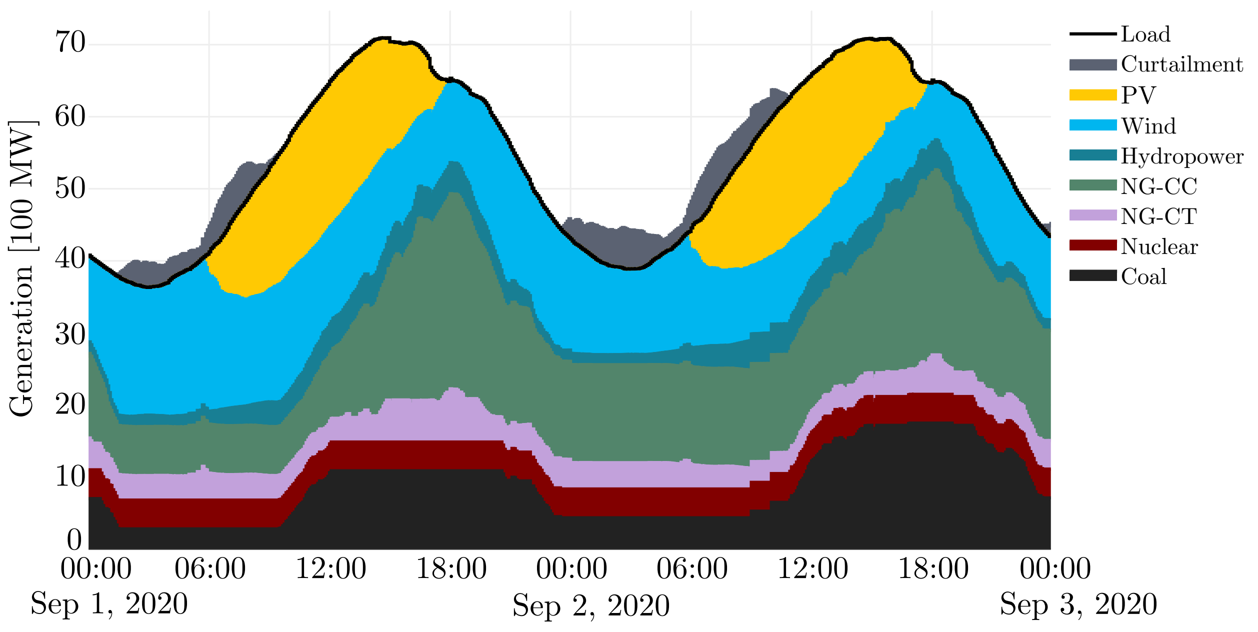


Figure 3.16: Economic Dispatch (ED) stack for the experiment two day period.

optimization problem of the AGC problem $\mathcal{P}^{\text{AGC}}$ can be written as:

$$\min_{\substack{\boldsymbol{R},\Delta\boldsymbol{P},\Delta\boldsymbol{Pe},\\ \Delta\boldsymbol{f},\boldsymbol{u}}} \sum_{t\in\mathcal{T}} \left\{ c|u_t| + \sum_{g\in\mathcal{G}_t} \gamma^{\text{up}}_{g,t}\Delta Pe^{\text{up}}_{g,t} + \gamma^{\text{dn}}_{g,t}\Delta Pe^{\text{dn}}_{g,t} \right\}$$

s.t. Frequency error: (3.9) – (3.10)

ACE and SACE computation: (3.11) – (3.12)

FRR allocation and device model: (3.22) – (3.32)

Remark 2: The problem is cast as a LP by properly replacing the absolute value $|u_t|$ adding new variables z_t that replace the absolute value $|u_t|$, and adding the constraints $z_t \geq u_t$ and $z_t \geq -u_t,\ T = 1,\ldots,T$.

3.5 Simulation of the proposed Automatic Generation Control (AGC) model

The AGC modeling capabilities are demonstrated through an example simulation of coordinated UC and ED problems. First, the simulation solves the UC problem with a 1-hour resolution and forwards the results into an ED problem to obtain the generation base points. The participation factors $\text{pf}^{\text{up,dn}}_{g,t}$ are calculated based on the UC reserve allocation $\overline{R}^{\text{up,dn}}_{g,t}$ to emulate FRR reserve auctions set before the ED. In both UC and ED minimum FRR requirements are based on the pre-calculated profiles, representing current practices of calculating reliability requirements off-line.

The simulation comprises two 24-hour periods using a modified version of the updated RTS-96 bus system which include load, VRE, and reserve requirement profiles [9]. The system's dispatch stack is shown in Fig. 3.16. Each dispatch and AGC problem was solved using CPLEX v12.10.

The original dataset required the implementation of several changes to achieve operational conditions that resemble AGC operations and reserve deployment. The primary modifications include:

- Reallocation the thermal fleet into reserve products and removing VRE from FRR provision.
- Adding the system load frequency response $D = 1.16$MW/Hz equivalent to 1% of peak load (7000 MW) per-unit change of the frequency.
- A droop for generators set to 4%. Together with the load damping, this is equivalent to an area frequency bias of $B = 90.8$ MW/0.1 Hz

The system reaches up to 50% instantaneous power output from VRE and requires VRE curtailment of controllable VRE resources during low load hours. The main source of variability comes from the renewable energy, shown in Fig. 3.17 and the dominant source of power is combined-cycle gas generation. Fig. 3.17 shows large forecast errors between the UC and the ED in the morning periods, which leads to aggressive curtailment in the ED. Since time series for modeling at the AGC timescale are not generally available, the load and VRE time series were up-sampled to 4-seconds. Up-sampling was conducted by interpolating 5-minute data and adding white noise with variance of 2.5% for the load and 10% for VRE.

AGC Simulation Results

PID performance

Fig. 3.18 shows the performance of the discrete PID given two combinations of parameters: $K_p = 1.8$ and $T_i = 1800$ for a fast response, and $K_p = 0.1$ and $T_i = 500$ for a slower response (both PID use $T_d = 0$). The slow response PID has a larger filtering effect and less variation over the simulation window. The control implemented with larger values in the proportional/integral gains leads to a closer imbalance tracking.

In both cases, the SACE responds according to the direction of the forecast errors depicted in Fig. 3.17. The periods after 00:00 in the simulation feature large deviations between the ED profile (blue) and the UC forecast (red), resulting in positive SACE values because the system is deploying downward reserve to restore balance. However, downward reserve participation factors from the UC are not well-calibrated because they are defined by

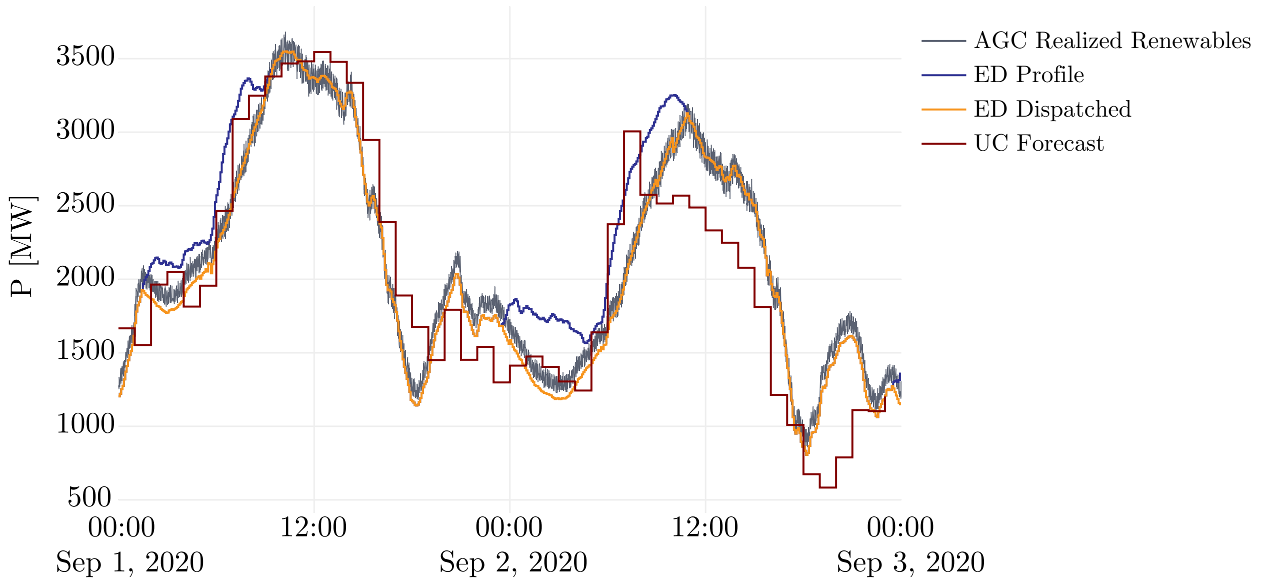


Figure 3.17: Renewable energy profiles for different simulations.

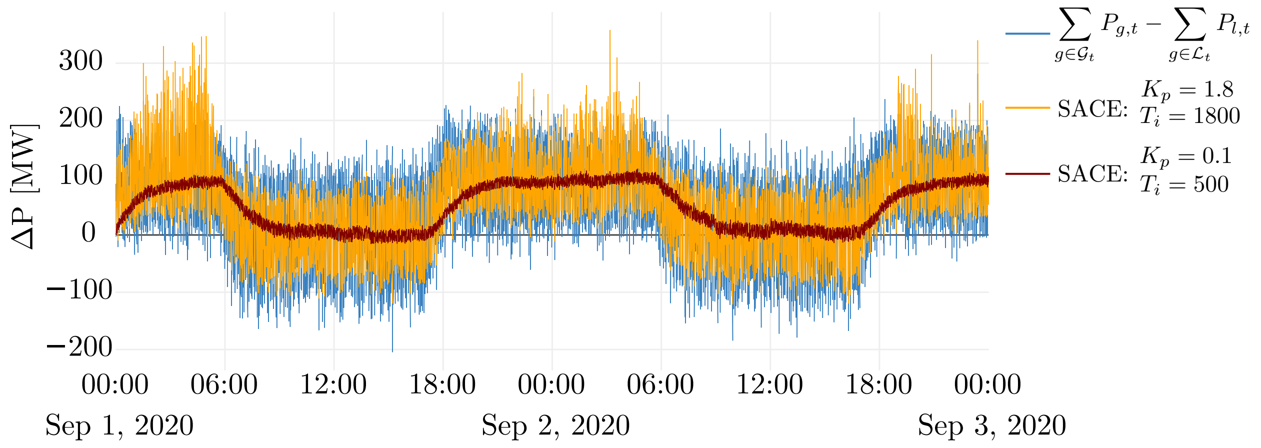


Figure 3.18: Comparison of SACE PID performance.

the UC forecast, leading to generator saturation. During periods of substantial curtailment, the model reflects the deployment of downward reserves (i.e., $\Delta P > 0$), and during periods of near system balance, the control is only deployed to handle short-term variation.

The system's performance in terms of frequency regulation can be seen in Fig. 3.19. In the faster PID (green), the frequency error is reduced further than the slower PID (purple). This result reflects the model's ability to provide frequency deviation estimates in terms of the PID control parameters.

Although the faster PID is capable of a larger reduction of Δf, it does so at the expense of introducing system reserve saturation, since the term $u_t \neq 0$ (see equation (3.18)).

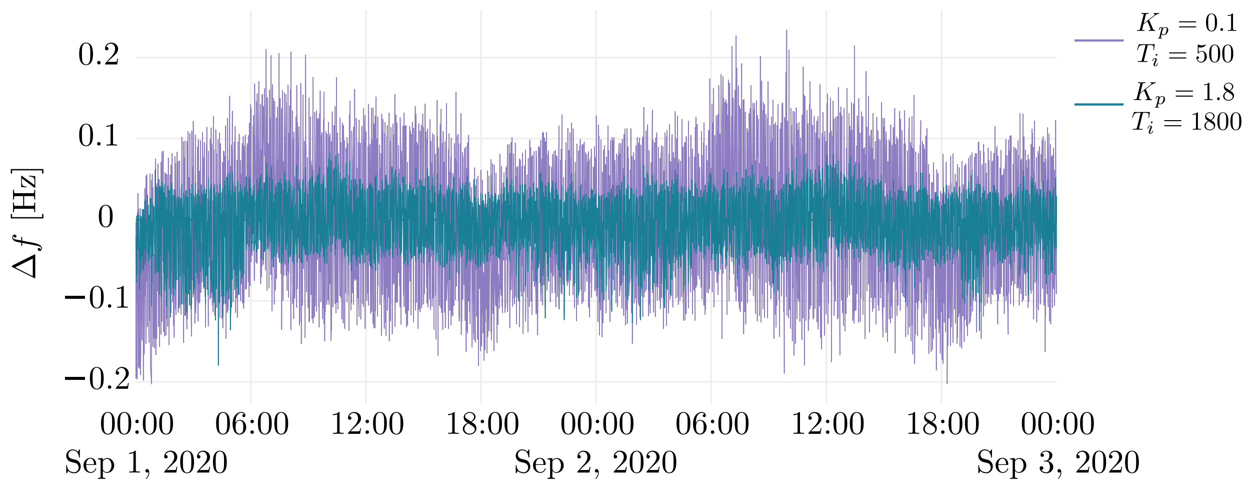


Figure 3.19: Δf for different PIDs.

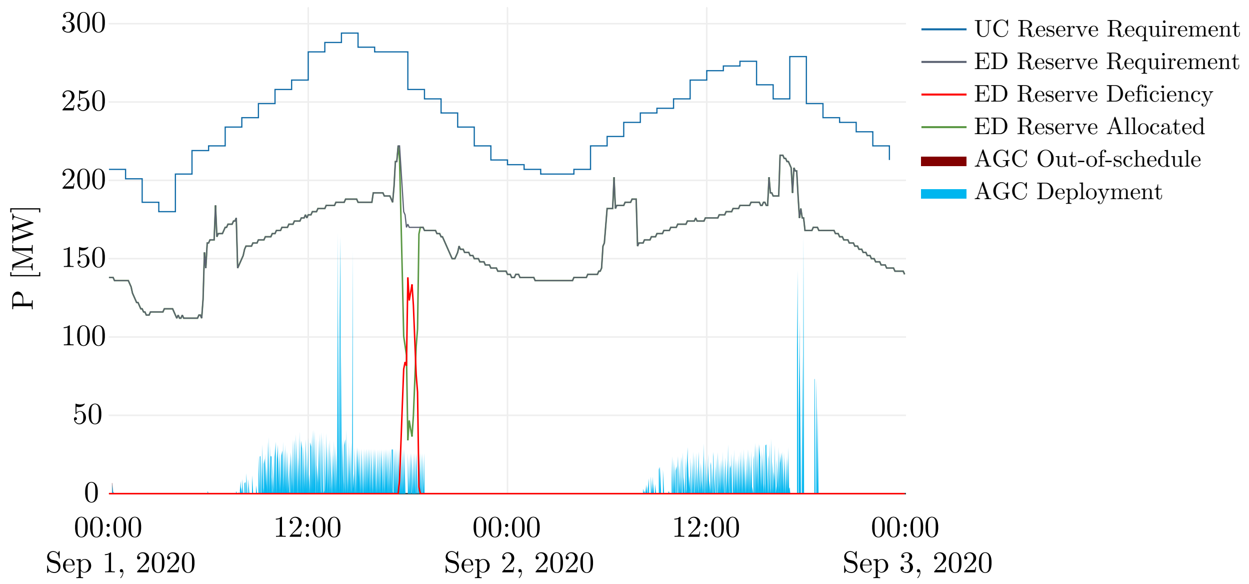


Figure 3.20: Upwards reserve deployment for $K_p = 0.1, T_i = 500$.

Reserve Deployment

The upward reserve allocations along with the requirements from the UC and ED are shown in Figs. 3.20 and 3.21. Although the UC procurement was insufficient to meet the ED requirement at the peak times, the results show that the shortage did not affect frequency performance since the FRR deployment was below the limited allocation. In this case, a combination of forecast error and inadequate reserve requirements in the UC lead to a deficient level of reserves at the ED stage.

Figs. 3.22 and 3.23 demonstrates the deployment of downward reserve, which in

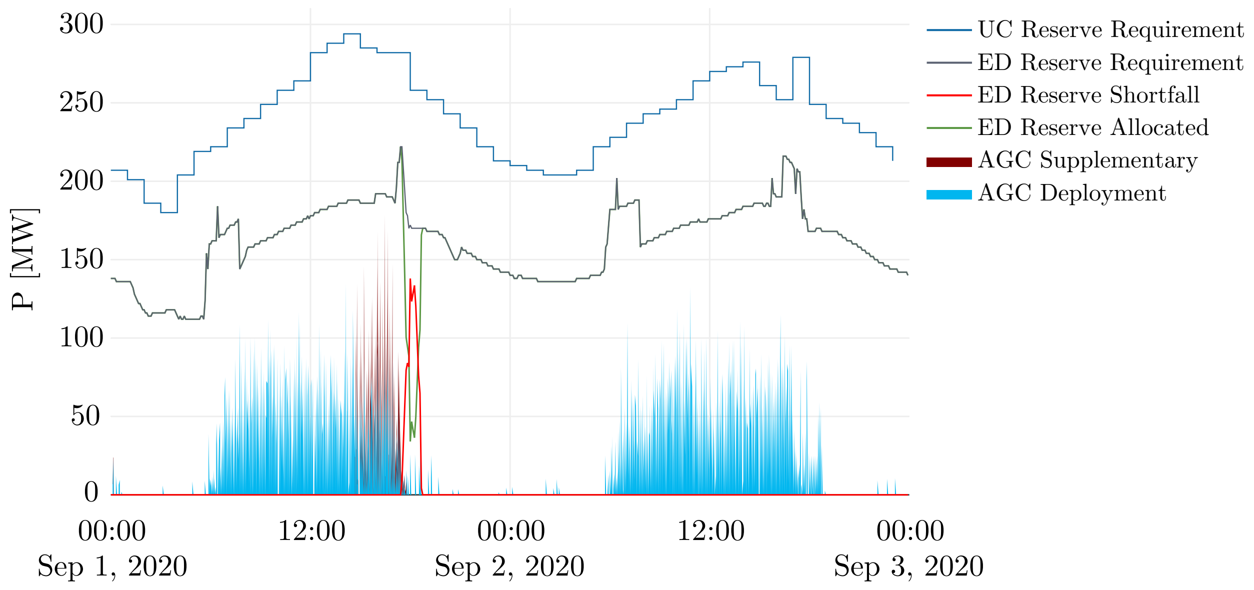


Figure 3.21: Upwards reserve deployment for $K_p = 1.8, T_i = 1800$.

the example case is critical due to a large amount of forecast error. In both cases, the deployment causes reserve saturation and large amounts of supplemental deployment that cannot be accounted for in simplified AGC models. This behavior stems from the significant forecast errors observed in Fig. 3.17 during the hours of high VRE generation. The model's ability to reflect issues in the mechanism to assign the participation factors can be observed here, together with supplemental regulation actions.

This result showcases the importance of considering the deployment mechanism when assessing FRR. The simulation shows that although the requirement was not met, there was sufficient FRR capacity to handle power imbalances. The proposed AGC model successfully assessed the impact of these limited reserve events.

The resulting reserve allocations also show the effect of the different PID tuning. The smaller frequency variations from the faster PID are obtained by deploying a larger amount of reserves, which can result in generator saturation, as shown in Fig. 3.21. Once one or more generators saturate, the control is forced to deploy supplemental reserves from other generators.

In both the upward and downward cases, the results show how the model is able to capture the increased regulation effort exerted from the faster PID that leads to 4.5 times increase in upwards reserve and 16% larger deployment of downwards supplemental reserve.

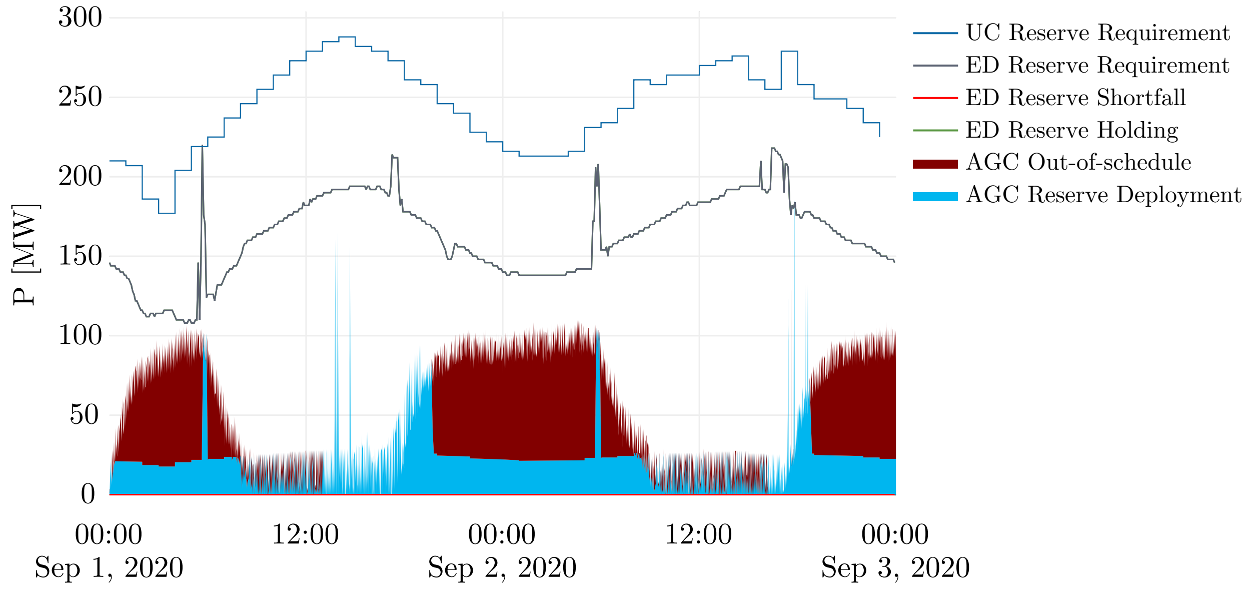


Figure 3.22: Downwards reserve deployment for $K_p = 0.1, T_i = 500$.

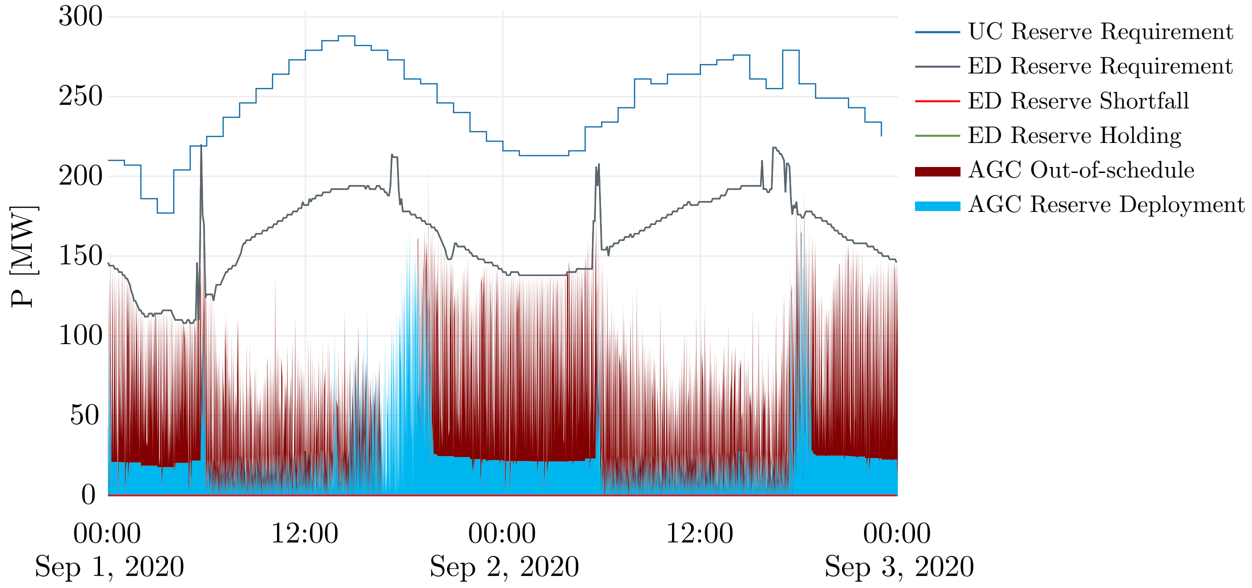


Figure 3.23: Downwards reserve deployment for $K_p = 1.8, T_i = 1800$.

3.6 Conclusions

- This chapter introduces the definitions required to formulate and solve operations simulations of systems with multiple decision stages and requires modeling flexibility.
- We showcase the software design principles that enable the construction of modular

simulations and the implementation of the required routines to conduct operations simulations. Namely, build, initialization and solution.

- The validation and study cases show that `PSI.jl` is able to replicate the results of commercial tools with reasonable levels of confidence.
- This chapter also presents a simulation model for the AGC geared towards reflecting reserve deployments and handling large shares of VRE. Where existing tools that use AGC simulation for FRR deployment analysis are limited to instantaneous power balance models, our approach enables a comprehensive evaluation of FRR deployment schedules.
- The proposed AGC simulation model in this chapter captures interactions between the control and the deployment of FRR in a consistent and tractable way. The optimization-based implementation enables modeling capabilities that previous literature implements only through heuristics.
- The LP formulation takes into account generator limits that lead to generator saturation as well as the effects of system reserve saturation. This leads to improved insights concerning the different processes involved in the FRR allocations and deployment.
- The example simulation using the framework of decision-emulation model in a synthetic system with significant VRE shares shows that this setting can capture key challenges in the large scale integration of VRE. The results show that the model can simulate deployment of supplemental reserves when the participation factors are not well calibrated due to forecast errors. The model can also estimate frequency deviations according to the AGC PID parameters. Current models that feature only power balance cannot provide insight about generator saturation or frequency deviations estimations.

Chapter 4

A Multi-Stage Stochastic Risk Assessment with Renewable Power Markovian Representation

Forecasting techniques have been fundamental to the improvement of power systems operations in the presence of substantial shares of VRE [171]. Projecting a system's state is critical for operators to plan the corrective actions required to maintain that system's secure operation. Yet operators generally employ point forecasts, which estimate the expected VRE power output without providing information on the uncertainty or *volatility* of the VRE resource. In contrast, probabilistic forecasts provide a *distribution* of possible VRE power outputs. This representation of the VRE uncertainty furnishes the possibility of assessing the system's response to VRE variability using a risk framework. However, there are misunderstandings about such uses of probabilistic forecasts, and there has thus far been very limited uptake of probabilistic forecasts in control rooms [191]. Current applications utilize probabilistic forecasts by focusing on a particular quantile in isolation, such as the 80th percentile of excedance, and using that quantile as a lower bound of the VRE across the forecast window [14, 48]. Focusing on a single quantile, however, results in an unrealistically smooth visualization of the power forecast that obscures VRE volatility. In actuality, the VRE realization typically jumps among different forecast quantiles over time.

Uncertainty forecasts are also used extensively in two-stage power system scheduling models, reviewed in detail by [92]. Two-stage prescriptive models are commonly proposed because most applications of probabilistic forecasts have focused on the stochastic Day-ahead Unit Commitment (DAUC) problem, which introduces significant computational challenges when solved in a multi-stage fashion. However, in two-stage formulations, the assumption is that the uncertainty is observable in a single instant, instead of being gradually revealed over time. When used for situational awareness, this assumption obscures how the system will adapt to the true realization of uncertainty, and it underestimates the true risk to the system because the recourse decisions are made with perfect knowledge of the realization [154].

Handling the time dependency of the decision-making process requires a multi-stage modeling framework because the decisions at a specific time period should not depend on knowledge of the future realization of the stochastic data process [132]. Although such models are common in long-term operations and planning [141], there is a limited usage of multi-stage models in operations, and most innovations focus on day-ahead problems. For example, in [189] the authors highlight the importance of time coupling to set reserve policies considering spatial and temporal correlation of prediction errors. Recently, a model of continuous unit commitment UC that relies on scenario trees was proposed in [70]. There, the authors generate an equivalent large-scale MILP to solve the multi-stage problem.

There are a variety of methods to capture the temporal dependence of probabilistic VRE forecasts into scenarios for multi-stage stochastic models: copulas [110], Gaussian kernels [144], or quantile “cut points” that emphasize certain regions of the cumulative distribution function over time [195]. These methods require several computational steps: (1) scenario generation, (2) tree generation, and (3) scenario reduction. These methods are commonly used to formulate day-ahead optimization problems, and there are limited applications which directly use probabilistic forecasts (i.e., quantiles or percentiles). To avoid creating decision trees, the authors of [97] present a multi-stage RUC model and a specialized algorithm to solve the problem. Markov Decision Problems (MDPs) are another alternative to avoid scenario generation in multi-stage decision making, and these were used by [57] to solve a day-ahead storage-scheduling problem in distribution systems using a Markov model for time-series modeling.

In short-term decision making, there are few applications of multi-stage modeling. For instance, in [98], the authors use a multi-stage stochastic program to co-dispatch generation and spinning reserves. The authors propose a joint uncertainty framework for contingencies and renewable generation. However, the proposed model relies on a pre-calculated scenario tree structure and does not mention potential integration with forecast data. Most applications of the prescriptive models that focus on suggesting optimal decisions for operators have found limited application by Independent System Operators (ISOs) due to their computational, data, and model complexity.

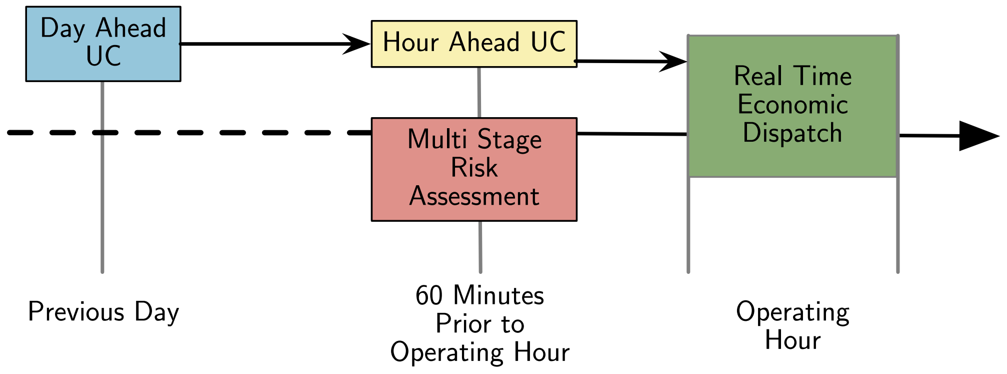

Day Ahead UC
Hour Ahead UC
Real Time Economic Dispatch
Multi Stage Risk Assessment
Previous Day
60 Minutes Prior to Operating Hour
Operating Hour

Figure 4.1: System operation flow including multi-stage risk assessment

Although probabilistic forecasts in prescriptive models have not been widely adopted, this chapter proposes a model and use case of probabilistic forecasts to improve situational risk awareness. Currently, operational risk awareness tools can provide only a limited representation of the system's response to uncertainty realizations, which results in limited situational awareness. To properly model the ability of operators to change their planned decisions as new information is revealed, the model must be multi-stage, yet it must also be solvable by a tractable solution technique that does not rely on the enumeration of all scenarios. To fill this gap, this chapter proposes a tractable multi-stage stochastic linear program to predict reserve deployment at a 5-minute resolution. The model integrates probabilistic forecasts using a Markov chain and is run based on the the outputs of the Hour-ahead Unit Commitment (HAUC) as shown in Fig. 4.1. It can be executed parallel to the ISOs market operation.

Currently, the use of probabilistic forecasts is limited to descriptive analytics for situational awareness. For example, the Solar and Wind Integrated Forecast Tool (SWIFT) system in Hawaii and the situational awareness desk at the Electric Reliability Council of Texas (ERCOT) use uncertainty forecasts to improve risk assessment [61, 92].

The proposed operational framework maps *input* short-term VRE probabilistic forecasts to *output* probabilistic forecasts of corrective actions, such as reserve deployments, during the operating hour. The availability of probabilistic information about future system states enhances the operator's situational awareness and could prompt preventive actions, such as reserve substitution or curtailment. Given the various available risk management mechanisms, we do not prescribe specific adjustment actions to reduce the system's exposure. Instead, the focus is on using probabilistic forecasts to describe future risks one hour before the operating hour. The model proposed in this chapter will help operators to become more acquainted with multi-stage models and probabilistic forecasts, demonstrating the implementation of prescriptive models with endogenous uncertainty consideration.

We propose a novel representation of the probabilistic forecasts using a Markov chain as shown in Fig. 4.2. The temporal correlations between the different quantiles in the probabilistic forecast can be represented by the probability of transitioning among quantiles between time steps. Using a *Markovian graph* [41], we exploit the uncertainty structure to solve the multi-stage stochastic program within reasonable computing times. All the cases from the literature discussed so far rely on scenario trees to capture the evolution of the uncertain quantities, but a 99th-quantile probabilistic forecast over 48 5-minute periods (as we use in this chapter) results in $99^{24} \approx 10^{48}$ scenarios and a model with a similar magnitude of variables and constraints. Such a problem is intractable to formulate as a single model; therefore, the proposed approach does not employ probabilistic forecasts to generate the scenario trees. We exploit the Stochastic Dual Dynamic Programming (SDDP) solution method to tackle this computational challenge.

To demonstrate the use and applicability of the proposed approach in a realistic setting, we have developed a simulation that resembles realistic operating conditions. We use a simulation of the cascading sequence of decisions followed by ISOs. We apply the model in a synthetic system with over 300 thermal generators using realistic area-wide solar-power forecasts and realization data. Although the proposed approach can be used with any probabilistic forecast, the experimental results focus only on solar power uncertainty because the development of area-wide net-load probabilistic forecasts is a nascent research area unto itself and is outside the scope of the proposed model. The results show the information at different operating hours and an example case of its use for situational awareness. To summarize, the main contributions of this chapter are:

1. A practical application of multi-stage stochastic programming and probabilistic forecasting for real-time risk assessment utilizing a Markovian forecast representation of VRE.

2. A formulation of an ED problem considering generation reserve deployment constraints as a risk-averse multi-stage stochastic program.

3. A proposed workflow for integrating probabilistic forecasts and our multi-stage stochastic programs into the risk management operations of an ISO.

4. A method for generating the transition matrix by ranking past observations among the quantiles of corresponding probabilistic forecasts and the implementation for updating this representation in real time.

5. Scoring metrics to calibrate and benchmark the application of Markovian approaches to probabilistic forecasts.

The rest of the chapter is organized as follows: Section 4.1 presents operational practices used by operators to handle system balancing risks and the proposed modeling approaches. Section 4.2 describes the multi-stage model and the solution techniques. The experiment design and results for the statically calculated matrix are detailed in Section 4.3. The extension of the proposed method to incorporate online updating of the Markov matrix is described in Section 4.4, including a comparison of the prediction results compared to the static matrix approach. Finally, Section 4.5 discusses future extensions.

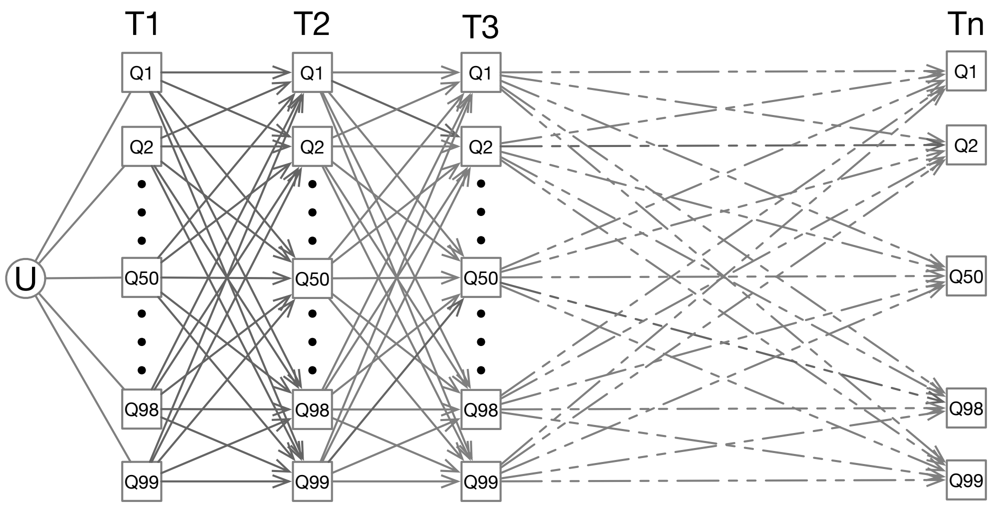


Figure 4.2: Markov representation of the probabilistic forecast

4.1 Risk model and probabilistic forecasting

There is a mismatch between risk mitigation practices: for example, reserve requirement settings are still mostly based on heuristics and historical information that often result in the under- or over-procurement of reserves [19], and these settings do not provide information about the system's exposure in the short term. The approach we introduce in this section describes the applications of probabilistic forecasting to an uncertainty-adapted representation of the possible corrective actions according to the operator's risk preferences.

Developing a risk assessment framework requires identifying two significant components: (1) a representation of the uncertainty, and (2) a model of the system's exposure to uncertainty [67]. In this chapter, we present a short-term ED model with reserve deployment to capture the system's exposure, and we demonstrate the use of a Markovian framework to represent the VRE uncertainty. The proposed framework uses an area-wide uncertainty forecast of VRE derived from plant-level probabilistic forecasts.

The operational model assumes the following setting: an operator executes an HAUC as soon as a VRE probabilistic forecast for the balancing area is issued and determines two quantities: (1) The commitments for the units, and (2) the Ancillary Services (AS) assignments for the regulation generators. Using these results as inputs, we employ an ED model that accounts for the deployment of regulation reserves to minimize the future expected ACE to estimate the effect of VRE variability in the reserve deployment. The ED model does not include line flow constraints given that it represents a single balancing area. The model assumes, as is typical in practice [43], that the eligibility of devices to participate in AS has been determined beforehand and that if the network imposes significant constraints on the deployment of the reserves, these constraints are captured by upper bounds on the AS variables.

Application of Markovian models to probabilistic forecasting

Probabilistic forecasts are a representation of the stochastic process which describes generation from VRE over a finite set of time steps $t = 1, \dots, T$. For a specific period $t \in \mathcal{T}$, the probabilistic forecast is represented as a random variable $\tilde{p}_{a,t}$ approximated through a discrete set of j quantiles, where each quantile $p^{j}_{a,t} \in \sqrt{}^{a}$ might occur with probability $\omega_j = \frac{1}{|\mathcal{J}|}, \; \forall j \in \mathcal{J}$. These quantiles discretize a cumulative distribution function, $F_t(p_{a,t})$, that forecasts the area-wide VRE power, $p_{a,t}$, at time t. The j^{th} quantile forecasts the power level that will be exceeded with probability $\frac{j}{100}$ $(0 \leq j \leq 100)$ [14, 12]:

$$p^{j}_{a,t} = \inf \left\{ p_{a,t} : F_t(p_{a,t}) \geq \omega_j \right\}. \tag{4.1}$$

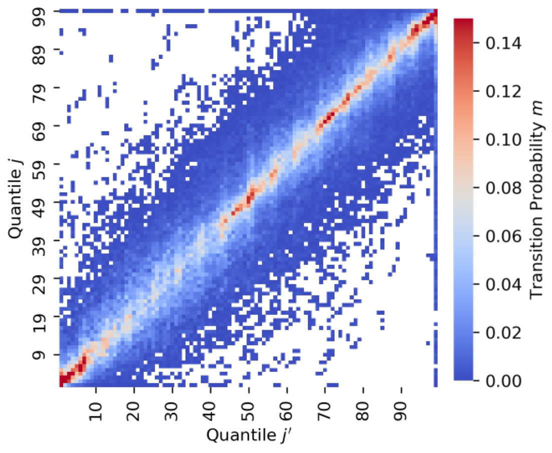
To conduct a multi-stage stochastic evaluation, the forecasting process must account for time dependencies in the forecasts. However, probabilistic forecasting providers usually limit the information to specific quantiles that does not account for time dependencies [180]. In the case of short-term forecasts, providers often rely on heuristics to "*perturb*" the hourly ensemble members in order to generate intra-hour forecasts based on ground measurements. These perturbation methods and the data used to generate the forecasts are generally proprietary.

In this case, we use an alternative approach to integrate the temporal evolution of the uncertainty into the multi-stage model. We employ a Markov transition matrix M (edges in Fig. 4.2) with quantile-to-quantile transition probabilities between time steps. Each entry $m_{j,j'}$ in M describes the probability of transitioning from quantile j at time t to quantile j' at time $t+1$ according to the quantile forecast representation in (4.1).

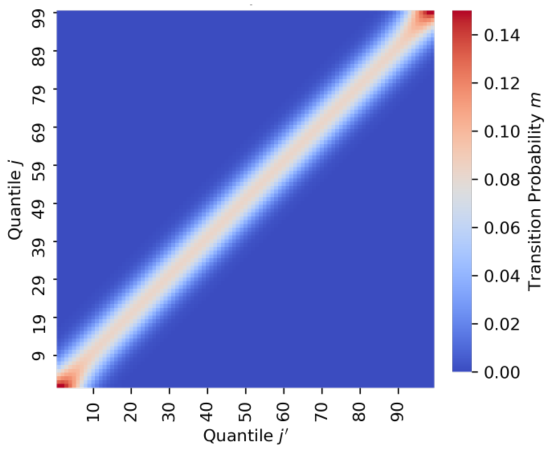
Based on prior knowledge of the behavior of VRE, we can anticipate that the matrix for the area forecast will be highly diagonal due to the persistence of weather conditions over short periods of time, while also reflecting some movement among quantiles due to random variability. This approach affords operators the flexibility of obtaining a matrix $\boldsymbol{M}$ based on the forecast, historical data, or current operating conditions. The Markovian model can accommodate other variants of uncertainty forecasts by changing the formulation of the transition matrix M. In one bounding case, an identity transition matrix (i.e., $m_{j,j'} = 0, m_{j,j} = 1$) would model smooth trajectories that remain entirely within one quantile, i.e., the trajectories persist through the forecast window. On the other extreme, one can also represent extreme volatility by assigning the same probability $\frac{1}{|\mathcal{J}|}$ to each entry in M.

The first approach to the development of the transition matrix M estimation is assuming a truncated normal distribution centered on the diagonal, with the standard deviation chosen based on an exploratory analysis of the realization of VRE data. We estimate the average likelihood that the actual area-wide power will transition from one forecast quantile to another. Synthetic "actuals" are calculated from 5-minute National Solar Radiation Database (NSRDB) data and then transformed from irradiance to power through PVWatts® with the same plant specifications as were used for the forecasts [162, 172]. Over the course of the year 2018, these synthetic actuals are compared to the probabilistic forecast quantiles to count the number of transitions that occur from one quantile to another. An online approach to the estimation of M is further discussed in Section 4.4.

Based on this empirical transition data, shown in Fig. 4.3(a), we estimate the transition matrix to have a standard deviation of 5 percentiles. This results in the smooth transition matrix illustrated in Fig. 4.3(b). The interpretation of the transition matrix for synthetic actuals is that a given forecast quantile has, on average, a 68% chance of staying within a 5% deviation of the 50th quantile.



(a) Empirical quantile transitions



(b) Transition normal distribution with $\sigma = 5$

Figure 4.3: Transition matrix used in the experimental set-up

Economic dispatch formulation considering Area Control Error (ACE)

Power system operations, whether centralized or in the context of energy markets, can be abstracted into multiple stages. These stages are decision-making processes with different resolutions and time horizons [132]. First comes the day-ahead stage; in competitive electricity markets, this stage is primarily for participants to hedge their exposure to real-time prices. In centralized systems, the operator estimates production scheduling.

In the later stages shown in Fig. 3.3, the real-time or balancing stage is where differences between day-ahead forecasts and short-term realizations are balanced. In these stages, the operator plays a larger role to guarantee that the operation of the system is reliably imposing constraints according to security criteria and making sure that there is enough recourse to perform corrective actions. Most operators execute an HAUC (also named RUC) at the top of the hour before execution of the real-time dispatch to make sure enough reserve will be available at the operating hour. The ED stage then calculates the *base points* for the generators. Finally, in the AGC (also called Load Frequency Control (LFC)) stage, the operator communicates the generation *base points* and regulation reserve deployments to compensate for short-term power deviations. Although the base points are usually updated every 5 minutes, large shares of renewable energy have demonstrated significant impacts on time scales below 5 minutes [47, 52]. This has led to increased reserve requirements at the ED stage.

The proposed multi-stage stochastic program sits between Level 1 and Level 2 to estimate the risk of insufficient reserve deployments in Level 3. The system exposure corresponds to large mismatches in the system balance reflected in the ACE in a fashion similar to [47]. The proposed model takes the unit-commitment decisions from Level 1 as inputs, along with the most recent probabilistic forecast, and outputs a distribution of possible reserve deployments and ACE.

Managing Risk with Area Control Error

Usually the ACE is calculated at a much faster rate $\delta\tau$ (4 seconds) than the solution times of the ED δt (5 minutes), $\delta\tau \ll \delta t$. However, in this case we are interested in estimating the reserve deployment due to forecasting error. In balancing studies, AGC modeling is often replaced by a discrete model that ignores dynamic effects such as generators' governor response [47, 186]. For the purposes of this chapter, we consider the ACE at the resolution of the time period t, and ignore the higher-frequency terms and faster system dynamics. Thus, ACE is simplified to the system's net power imbalance and the following reserve terms:

$$\tilde{\mathrm{ACE}}_t = \sum_{l\in\mathcal{L}} p^*_{l,t} - \sum_{g\in\mathcal{G}} p_{g,t} - \tilde{p}_{a,t} - \sum_{g\in\mathcal{G}^{bal}} \left(\Delta p^+_{g,t} - \Delta p^-_{g,t}\right), \tag{4.2}$$

where $p_{g,t}$ are the base point values from the ED result. The deployment of regulation reserves $\Delta p^+_{g,t}, \Delta p^-_{g,t}$ is used to counteract the renewable power deviations and keep the ACE at zero. The operator's objective is to ensure that the system operates reliably during the dispatch stage of the operation. In the context of this chapter, this implies that there is enough reserve to handle sub-5-minute VRE variations.

Dispatch Model

For clarity, we start by describing the deterministic formulation of a dispatch problem considering reserve deployments to minimize the ACE without uncertainty. This model will establish the risk exposure of the system at the ED time scale before the uncertainty is unveiled. It is used to represent the atomic sub-problems in the multi-stage formulation. Model (3) also corresponds to the formulation used in each of sub-problems in forthcoming section III-B.

The joint dispatch and modeling approach that follows is similar to the two-stage robust optimization formulation proposed in [204]. In that chapter, the authors calculate the generator's base points in the first stage subject to the worst-case AGC realization to minimize

ACE in the look-ahead. The cost function includes two components: the PWL generation cost and the occurrence of ACE.

Our goal is to choose the generator power output $p_{g,t}$ along with up- and down-reserve deployments $\Delta p^+_{g,t}$ and $\Delta p^-_{g,t}$, for each generator $g \in \mathcal{G}$ and time period $t \in T$, in a way that minimizes the total cost of generation and provides an estimate of the ACE. We assume that different renewable generators use distinct forecasting methods or forecast providers. A deterministic version of the model using the expectation $\mathbb{E}$ yields the following linear program:

$$\min_{\boldsymbol{p},\boldsymbol{\Delta p}} \quad \sum_{t\in T}\left(\sum_{g\in\mathcal{G}}\sum_{k\in\mathcal{K}} p_{g,k,t}C_{g,k} + \gamma|\mathrm{ACE}_t|\right) \tag{4.3a}$$

s.t.

$$p_{g,t} = \sum_{k\in\mathcal{K}} p_{g,k,t} \qquad \forall g\in\mathcal{G}\ \forall t\in T \tag{4.3b}$$

$$P^{min}_g \le p_{g,t} \le P^{max}_g \qquad \forall g\in\mathcal{G}\backslash\mathcal{G}^{bal}\ \forall t\in T \tag{4.3c}$$

$$p_{g,t} - p_{g,t-1} \le R^{UP}_g \qquad \forall g\in\mathcal{G}\backslash\mathcal{G}^{bal}\ \forall t\in T \tag{4.3d}$$

$$p_{g,t-1} - p_{g,t} \le R^{DN}_g \qquad \forall g\in\mathcal{G}\backslash\mathcal{G}^{bal}\ \forall t\in T \tag{4.3e}$$

$$P^{min}_g \le \Delta p^+_{g,t} + p_{g,t} - \Delta p^-_{g,t} \le P^{max}_g \qquad \forall g\in\mathcal{G}^{bal}\quad \forall t\in T \tag{4.3f}$$

$$p_{g,t} - p_{g,t-1} \le R^{UP}_g - \Delta p^+_{g,t} \qquad \forall g\in\mathcal{G}^{bal}\quad \forall t\in T \tag{4.3g}$$

$$p_{g,t-1} - p_{g,t} \le R^{DN}_g - \Delta p^-_{g,t} \qquad \forall g\in\mathcal{G}^{bal}\quad \forall t\in T \tag{4.3h}$$

$$0 \le \Delta p^+_{g,t} \le Q^{UP}_g \qquad \forall g\in\mathcal{G}^{bal}\quad \forall t\in T \tag{4.3i}$$

$$0 \le \Delta p^-_{g,t} \le Q^{DN}_g \qquad \forall g\in\mathcal{G}^{bal}\quad \forall t\in T \tag{4.3j}$$

$$\sum_{g\in\mathcal{G}} p_{g,t} + \sum_{r\in\mathcal{R}} p^*_{r,t} - \sum_{l\in\mathcal{L}} p^*_{l,t} = 0 \qquad \forall t\in T \tag{4.3k}$$

$$\mathrm{ACE}_t = \sum_{l\in\mathcal{L}} p^*_{l,t} - \sum_{g\in\mathcal{G}} p_{g,t} - \mathbb{E}[\tilde{p}_{a,t}] - \sum_{g\in\mathcal{G}^{bal}} \left(\Delta p^+_{g,t} - \Delta p^-_{g,t}\right), \tag{4.3l}$$

where (4.3b)–(4.3h) define the feasibility $\mathcal{X}_\mathcal{G}$ set for the thermal generators, and (4.3i)–(4.3l) define the expected system balance. In the evaluation model, the point forecast of VRE in the area at time t is represented by the sum over the deterministic forecasts of individual plants, $\sum_{r\in\mathcal{R}} p^*_{r,t}$. To aggregate the plant-level forecasts to the area level, we took each Numerical Weather Prediction (NWP) ensemble member and summed the power over the individual plants, a process described in [14]. Due to the ordering operation, the area-aggregate forecast is not guaranteed to be the same as the order of members for individual plants, such that, in general, $\sum_{r\in\mathcal{R}} p^*_{r,t} \neq \mathbb{E}[\tilde{p}_{a,t}]$.

Model (3) is representative of current practices that use a deterministic equivalent at the plant level to resolve the values $p_{g,t}$, but reserves are deployed based on the area aggregate $\tilde{p}_{a,t}$ which we represent as an uncertain quantity. The purpose of this model is to assess the potential deployments of reserve required to minimize $|ACE_t|$. In Section III, we introduce the probabilistic uncertainty representation into the model for risk assessment.

Operators schedule regulation reserve provisions Q_g^{UP} and Q_g^{DN} before the operating day [48]. To mimic this decision process in the simulation, we have obtained the generator reserve assignments from a reserve co-optimization DAUC model. The resulting reserve schedules are later used as inputs to the HAUC which estimates any supplemental AS, if required. The final AS assignments resulting from the HAUC correspond to the upper-bound reserve deployment values Q_g^{UP} and Q_g^{DN} in the model. The model does not include line flow constraints given that it represents a single balancing area. If the network flow limits reduce the capability of a generator to hold reserves, these constraints are by the values of Q_g^{UP} and Q_g^{DN} that bound the maximum participation from the balancing units.

If the system cannot be balanced, the ACE increases, and persistent imbalances carry penalties or require extreme corrective actions such as load-shedding. To represent the cost of imbalances, the objective function penalizes the value of $|ACE|$ with the parameter γ, which must be set at a value higher than the cost of the most expensive unit in the system. For instance, a value for γ slightly above the offer price ceiling is an appropriate amount, so that the model will dispatch all the resources possible in order to minimize the ACE.

The addition of risk measures to the objective functions of multi-stage models such as (4.3) is not sufficient to obtain results that reflect how the system will adapt to the uncertainty [154, 69]. Section 4.2 shows the re-formulation of (4.3) into a multi-stage model.

4.2 Multi-stage risk-averse model

Model (4.3) uses the expected VRE at each time step. In this section, we extend Model (4.3) to a multi-stage stochastic program, using the Markovian model of VRE discussed in Section 4.1. To aid readability, we first assume the VRE is independent in each time period, and we then extend that model to add the Markov chain.

Adding uncertainty and risk

Model (4.3) can be decomposed using dynamic programming into a sequence of recursive sub-problems as follows:

$$\begin{aligned} V_t(p_{t-1}) = \min_{\substack{p_{g,t},\\ \Delta p^+_{g,t},\Delta p^-_{g,t}}} \quad & C(p_{g,t},\Delta p^+_{g,t},\Delta p^-_{g,t}) + V_{t+1}(p_t) \\ \text{s.t.} \quad & (p_{g,t},\Delta p_{g,t}) \in \mathcal{X}(p_{g,t-1}) \end{aligned} \tag{4.4}$$

where $V_{T+1}(\cdot)=0$, and C and $\mathcal{X}$ are defined as necessary to make $V_1(p_{g,0})$ equivalent to (4.3). The generation levels $p_{g,t}$ are *state* variables that flow through time. The reserve deployments are *control* variables that respond to the uncertainty-adapted corrective actions.

To introduce uncertainty into the decision model, we assume the probabilistic forecast stochastic process $\tilde{P}_{a,t}$ is represented by a set of discrete quantile-probability pairs $(P^j_{a,t}, \omega_j)$. Incorporating risk requires the addition of a *risk measure* $\mathbb{F}$ to map the random variable $\tilde{\mathrm{ACE}}_t$ to a real value [1]. The choice of risk measure as well as its parameterization allows us to calibrate the operator's risk aversion. For example, if $\mathbb{F}[\tilde{\mathrm{ACE}}_t]{=}\mathbb{E}[|\tilde{\mathrm{ACE}}_t|]$, the resulting solution is a risk-neutral policy that minimizes the expected operating cost plus the expected absolute value of ACE. Another common choice for $\mathbb{F}[\tilde{\mathrm{ACE}}_t]$ is the worst-case risk measure $\max[|\tilde{\mathrm{ACE}}_t|]$, which results in a conservative policy that minimizes the expected operating cost plus the worst-case ACE. As a middle ground between these two extremes, we use the Conditional Value-at-Risk (CV@R) [152] of the absolute value of ACE. CV@R can be represented as the linear program:

$$\begin{aligned} \mathrm{CV@R}[\mathrm{ACE}_t]{=}\min_{\alpha_t,z_j} \quad & \alpha_t{+}\tfrac{1}{\varepsilon}\textstyle\sum_{j\in\mathcal{J}}\omega_j z_j \\ \text{s.t.} \quad & z_j{\ge}|ACE^j_t|{-}\alpha_t \quad \forall j{\in}\mathcal{J} \\ & z_j{\ge}0 \quad\quad\quad\quad\quad \forall j{\in}\mathcal{J} \end{aligned} \tag{4.5}$$

CV@R is parameterized by $\varepsilon \in (0{,}1]$, and (4.5) corresponds to the expected value of the worst ε-fraction of quantiles. When $\varepsilon{=}1$, CV@R is equivalent to the use of the risk measure, and in the limit $\lim\limits_{\varepsilon\to 0}\mathrm{CV@R}^{\varepsilon}[\mathrm{ACE}_t]{=}\max[|\mathrm{ACE}_t|]$, which makes CV@R equivalent to the maximum risk measure.

Based on the discrete representation of the uncertainty and assuming that the VRE in each period is independent, problems (4.4) and (4.5) can be combined to produce:

$$\begin{aligned} V_t(p_{t-1}){=}\min_{\substack{z,\alpha_t,p_{g,t},\\ \Delta p^+_{g,t},\Delta p^-_{g,t}}} \; & \sum_{j\in\mathcal{J}}\omega_j\Big[C(p^j_{g,t},\Delta p^{j,+}_{g,t},\Delta p^{j,-}_{g,t}){+}V_{t+1}(p^j_{g,t})\Big] \\ & {+}\gamma\left(\alpha_t{+}\frac{1}{\varepsilon}\sum_{j\in\mathcal{J}}\omega_j z_j\right) \\ \text{s.t.} \quad & \\ & (p^j_{g,t},\Delta p^{j,+}_{g,t},\Delta p^{j,-}_{g,t}) \in \mathcal{X}(p_{g,t-1}) \quad \forall j{\in}\mathcal{J} \\ & |ACE^j_t|{=}\sum_{g\in\mathcal{G}}p_{g,t}{-}\sum_{l\in\mathcal{L}}p^*_{l,t}{+}P^j_{a,t} \\ & \quad {+}\sum_{g\in\mathcal{G}^{bal}}\left(\Delta p^{j,+}_{g,t}{-}\Delta p^{j,-}_{g,t}\right) \quad \forall j{\in}\mathcal{J} \\ & z_j{\ge}|ACE^j_t|{-}\alpha_t \quad \forall j{\in}\mathcal{J} \\ & z_j{\ge}0 \quad \forall j{\in}\mathcal{J} \end{aligned} \tag{4.6}$$

where $\Delta p^{j,+}_{g,t}$ and $\Delta p^{j,-}_{g,t}$ are the recourse variables in each discrete realization j in the uncertainty forecast.

Markovian uncertainty

We now model the area-wide VRE $P_{a,t}$ as governed by a Markov chain with quantile-to-quantile transition probabilities. Instead of a single value function V_t for each time period t, the Markovian model forms a lattice of value functions V_t^j where t is the column and j is the quantile of the Markov chain in Figure 4.2. The objective associated with each value function is modified to account for probabilistic transitions between value functions.

Moreover, there is a direct extension of CV@R from the model (4.6) to the Markovian case via the expected-conditional risk measures of [69]. This extension involves a reformulation of the problem. The V_t sub-problem is split into $|\mathcal{J}|$ sub-problems denoted V_t^j, the variable α_t is added as a state variable to the $t{-}1$ stage to ensure it takes the same value in each of the V_t^j sub-problems, and a conditional expectation is added to account for the cost of $V_{t+1}^{j'}$, conditional on being in V_t^j. Thus, our recursive sub-problems can be modified as follows:

$$\begin{aligned}
V_t^j(p_{t-1},\alpha_t)=\min_{\substack{z,\alpha_{t+1},p_{g,t},\\ \Delta p^+_{g,t},\Delta p^-_{g,t}}} \; & C(p^j_{g,t},\Delta p^{j,+}_{g,t},\Delta p^{j,-}_{g,t})+\gamma(\alpha_t+\frac{1}{\varepsilon}z)\\
& +\mathbb{E}_{j'|j}[V_{t+1}^{j'}(p_t,\alpha_{t+1})]\\
\text{s.t.}\;&\\
&(p^j_{g,t},\Delta p^{j,+}_{g,t},\Delta p^{j,-}_{g,t})\in\mathcal{X}(p_{g,t-1})\\
&ACE^{+j}_{\ t}-ACE^{-j}_{\ t}=\sum_{g\in\mathcal{G}}p_{g,t}-\sum_{l\in\mathcal{L}}p^*_{l,t}+P^j_{a,t}\\
&+\sum_{g\in\mathcal{G}^{bal}}\left(\Delta p^{j,+}_{g,t}-\Delta p^{j,-}_{g,t}\right)\\
&z\geq ACE^{+j}_{\ t}-ACE^{-j}_{\ t}-\alpha_t\\
&ACE^{+j}_{\ t}\geq 0\\
&ACE^{-j}_{\ t}\geq 0\\
&z\geq 0,
\end{aligned} \tag{4.7}$$

where $\mathbb{E}_{j'|j}$ is the expectation of transitioning to quantile j', conditional on being in quantile j in stage t (i.e., the $\mathbf{M}$ matrix). We also introduce a 0^{th} stage,

$$V_1(p_0)=\min_{\alpha_1}\mathbb{E}_j[V_1^j(p_0,\alpha_1)], \tag{4.8}$$

to solve for the first-stage value α_1.

Multi-stage stochastic programs with a Markovian structure, such as problem (4.8), are well-studied in the literature, and generic open-source solvers are available. For this application we employ the SDDP algorithm implemented in the package `SDDP.jl` [42].

The SDDP algorithm is an iterative algorithm that works by approximating the cost-to-go term $\mathbb{E}_{j'|j}[V_{t+1}^{j'}(p_t,\alpha_{t+1})]$ by the point-wise maximum of a collection of linear basis functions.

The approximated sub-problems are therefore finite-dimensional linear programs. The approximation is improved in a two-phase approach. First, a simulation of the policy (called the *forward pass*) is conducted to obtain a set of candidate solutions $\hat{p}_{t-1}$ and $\hat{\alpha}_t$. Then, a *backward pass* is conducted, adding new basis functions for each node V_t^j using the candidate solutions from the forward pass and a sub-gradient of the incoming p_t and α_{t+1} state variables in $V_{t+1}^{j'}$, which are obtained from the linear programming dual. Readers are referred to [141, 163, 41] for further reference.

4.3 Results

We have developed a simulation that resembles realistic operational conditions to demonstrate the applicability of the proposed approach in a practical setting. This section provides full details of the input data and simulation specifications.[1]

Computational experiment specification

Considering that the probabilistic forecast inputs are critical to the proposed model's risk assessment ability, the data used have been carefully developed to represent forecasts of realistic quality. Given the computational burden and time-intensiveness of generating forecast data for individual locations in a large system, as well as isolating the impacts of a single forecasting technology, the experiments only include uncertainty forecasts for solar power VRE. Because the simulation's objective is to demonstrate the application of the proposed approach, forecasting errors and uncertainty from other VRE resources have not been considered.

[1]The code and complete computational environment to reproduce the experiments is available at `https://github.com/Energy-MAC/MultiStageCV@R`.

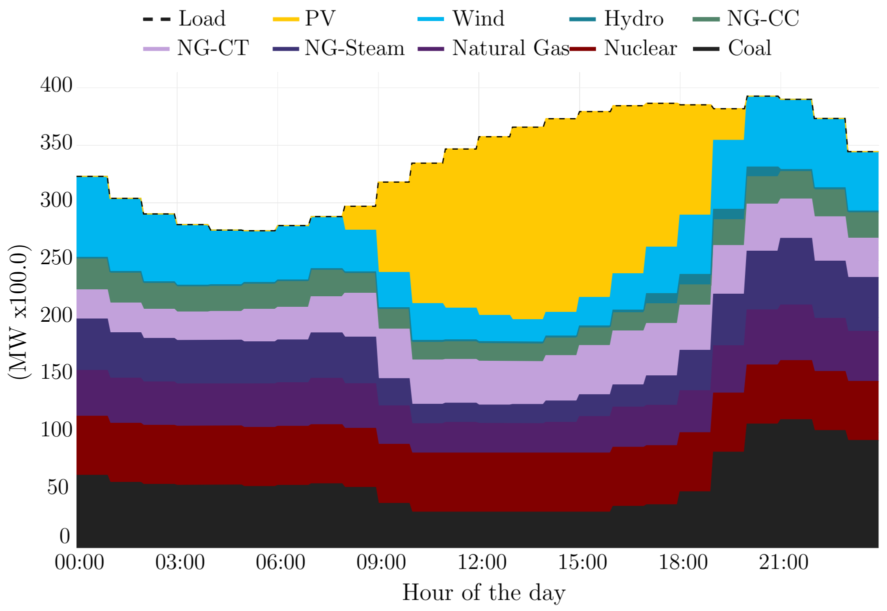


Figure 4.4: Simulation experiment Fuel stack (April 2018)

The system used to develop the proposed approach is based on the ACTIVSg2000 data set geographically located in the footprint of Texas [18]. The system has been extensively modified to accommodate large amounts of solar VRE in comparison with the original version and is publicly available. Solar VRE was increased from 22 solar plants (total installed capacity 938 MW) to 133 (total installed capacity 22,567 MW). Figure 4.4 shows the dispatch stack of the modified system developed for this research during the sample day used in the results, and Fig. 4.5 shows the static reserve requirements.

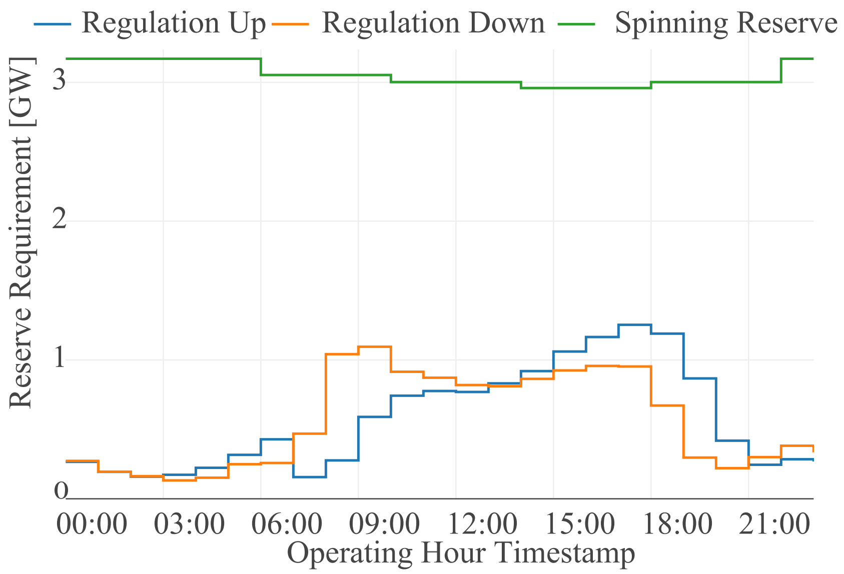


Figure 4.5: Simulation experiment Reserve requirements (April 2018)

To generate the probabilistic solar forecasts, weather data are collected from a large 116-member NWP model ensemble documented in [76] and prepared by the prediction system in [75]. Weather forecasts are translated to power forecasts through the System Advisor Model's PVWatts® power calculator [172], using realistic plant specifications based on recent technological trends and the ERCOT interconnection queue. The 116-member ensemble, now in units of power, is processed to a Probability Density Function (PDF) for each time step according to the 'NWP raw ensemble" method in [40]. The 50th percentile of the probabilistic forecasts for individual power plants is used as a point forecast, $p^*_{r,t}$, in the HAUC.

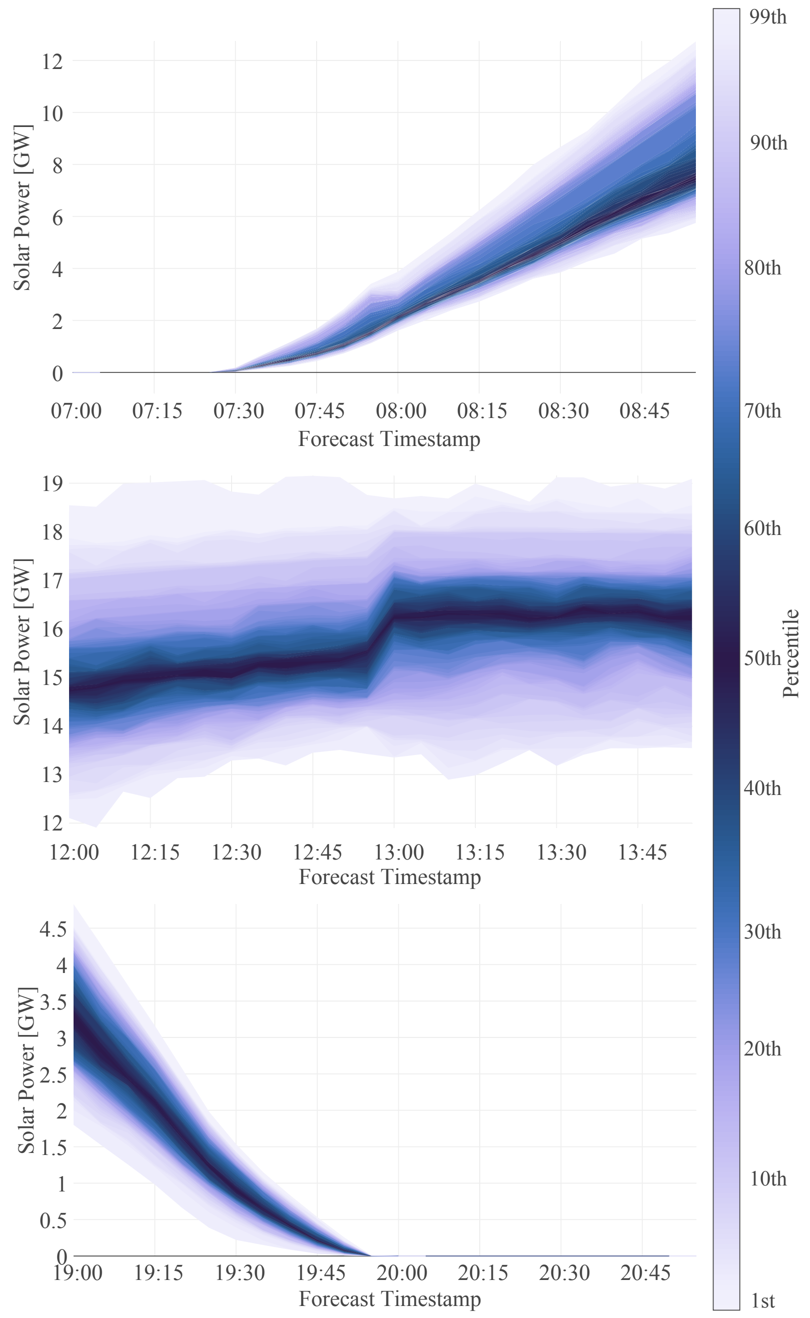


Figure 4.6: Probabilistic forecast representative windows for simulation experiment

To generate the area-wide probabilistic forecast, the 116-member power ensemble is first summed over the solar plants to generate a 116-member ensemble of area-wide power. The 99 quantiles in the set $\tilde{p}^{j}_{a,t}$ are similarly extracted from the empirical PDF of this summed ensemble. This approach is the simplest method to find the joint distribution over the solar plants because it retains the spatial correlations inherent in each NWP ensemble member. Utilizing other techniques to obtain the joint distributions can result in more refined joint forecasts; however, the development of joint forecasts is outside the scope of this chapter.

Figure 4.6 showcases the system's input probabilistic forecasts at three different periods of the day. We have selected a day in the month of April because, in Texas, spring is the season with the most variability in the VRE output. The reserve requirements for the system were calculated using ERCOT's methodology [19], which is based on a heuristic dependent on the amount of installed VRE. The system data contain time-series requirements for spinning, regulation-up, and regulation-down reserves. We use `PowerSystems.jl` [89] to develop the data set to provide full scientific reproducibility of the results.[2]

Simulation sequence and parameters

Capturing the cascading decision processes represented in Figs. 4.1 and 3.3 requires solving multiple optimization problems in a sequence that resembles the ISO's practices. The simulation was conducted as follows:

1. Solve the HAUC problem at the top of the hour using a 2-hour forecast look-ahead. Obtain the commitment decisions and regulation reserve assignment values Q_g^{UP} and Q_g^{DN} for the participating devices for use in upcoming stages. Figure 4.5 shows the system's reserve requirements obtained from ERCOT's methodology [19]. The requirement allocates up to 1.1 GW of down-regulation reserves in the morning hours to deal with the early ramp and 1.25 GW of regulation up for the evening solar ramp down. Although the HAUC model allocates spinning and regulation reserves, only the regulation reserves are used in the risk assessment model. This model uses the same intra-hour deterministic forecast data that will be used in the ED stage.

2. Based on the solution of Step 1, we solve the multi-stage problem (4.7) using `SDDP.jl` v0.3.17 and generate the results for situational awareness. The multi-stage problem uses a 2-hour horizon to match the horizon of the intra-day probabilistic forecast and current ERCOT operating procedures [49]. However, the model is not constrained to a specific look-ahead as long as the forecasting data are available.

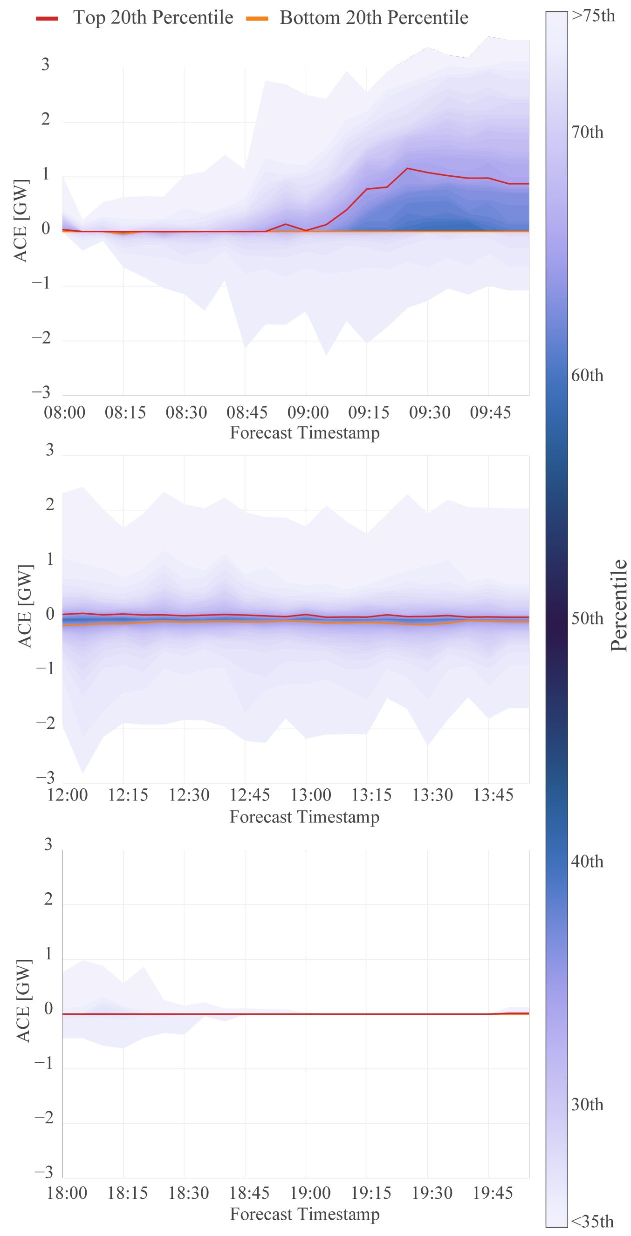
3. If necessary, we conduct a risk reduction step by increasing the reserve requirement and re-running step 2).

The model is parametrized using an α value of 0.2, i.e., hedging against the expected value of the top and bottom 20th percentiles. The model is executed on April 1st because spring time in Texas usually features VRE volatility and large shares of generation. The

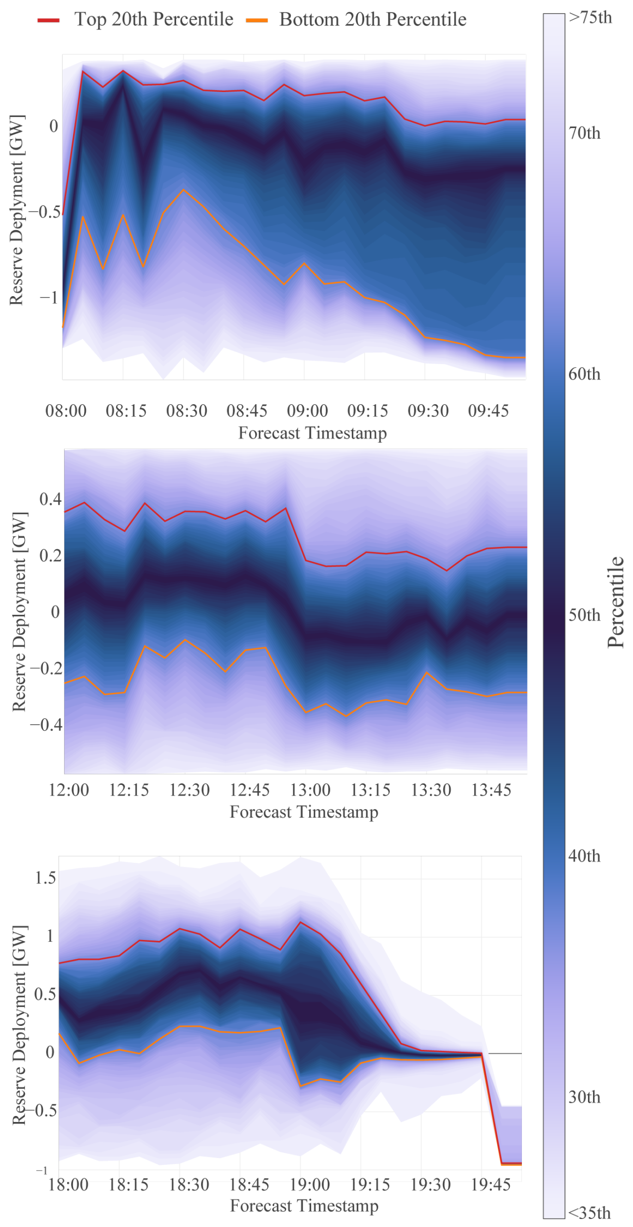
[2]Additional information about the data and code necessary reproduce the construction of the system is available at `https://github.com/NREL-SIIP/ExtremeSolarTexas`.

simulation sequence is implemented using `PowerSimulations.jl` in v1.6.2 of the Julia programming language [16]. We use CPLEX 12.10 to solve all optimization problems.

Risk awareness results



(a) ACE Values



(b) Reserve Deployment Values

Figure 4.7: Risk awareness results

Figure 4.7(a) visualizes the estimated distribution of the ACE from the multi-stage model for the periods shown in Fig. 4.6. For every 5-minute operating period, we show the distribution of the ACE, where the positive values represent generation shortfalls and negative values represent excess generation. For reference, we highlight the top and bottom 20th percentiles, which are commonly used by operators as the threshold values for risk. The forecasts show both the forthcoming operating hour and the following operating hour, as is standard practice [61]. Thus, the operator can evaluate the immediate risk (i.e., in the forthcoming operating hour) and future risks. The ACEs observed in the extremes of Fig. 4.7(a) are a product of low-probability transitions (i.e., favoring very large ramps) in M. As we can observe in Fig. 4.6, the probabilistic forecast spans over a 5-GW range of uncertainty with a central value of 16 GW, and this is also reflected in the ACE forecast. These results highlight the importance of further explorations to refine the calculation of M when the temporal interdependence in the probabilistic forecast might not be well-calibrated.

For most of the quantiles, the forecast ACE values are 0.0 since the system is able to deploy enough reserves to handle VRE variability. However, we can observe distinct risk behavior in each forecasting period. During the morning ramp at 8:00, am the system can maintain the ACE at 0.0 for the first operating hour, and during the second hour, the model predicts that there is a significant probability that the ACE is larger than zero and could reach 1 GW, which would be considered unacceptable. On the other hand, at the noon period, we can observe that there is a probability of having a sustained ACE of 100 MW ACE. Lastly, we can see that during the evening ramp the ACE risk is reduced to 0 for the majority of the operating hour. Figure 4.7(b) shows the probabilistic forecast of the reserve deployments. We can see that the morning and mid-day forecast predicts that there is a high probability of saturating the reserves. This behavior is produced because the system does not have perfect foresight of the uncertainty realization, and due to system constraints it cannot adapt fast enough to the realization, despite the spare reserve capacity. Based on the results, this system's regulation reserve requirements seem to be insufficient to completely hedge against the uncertainty at the operating hour, and this is reflected in the results of the ACE as well.

The multi-stage model predicts that for the realizations between the top and bottom 20th percentile of solar power, output the reserves are very likely to reach their maximum limits. In the case of the morning operating window, the system reaches its maximum reserves at the following operating hour (9:00), giving the operator information to adjust the operational parameter before the next execution of the HAUC. We can also observe that the model predicts that there is a higher probability of having to deploy downwards reserve starting at 9:00. In line with the ACE behavior from Fig. 4.7(a), the following operating hour exhibits a high risk of saturation. This is not surprising given the the morning ramp-up of the solar power and the ramp-up of the gas generation (see Fig. 4.5). The system is therefore under a lot of stress, and a forecast error might cause a thermal unit to be turned off too soon. On the other hand, the evening window shows that there is enough reserve capacity to be deployed to keep the system's ACE at 0.0 even for the lowest-probability quantiles.

The reserve saturation has a different profile at the noon hour, where it is possible to observe that the reserves can saturate for most of the operating window and produce a persistent ACE. This extreme behavior results from the fact that model (4.3) can only use $\Delta p^{+}_{g,t}$ and $\Delta p^{-}_{g,t}$ as recourse to bring the system back to balance. The results highlight that the regulation reserve requirements do not provide the system enough recourse options to reduce the risk significantly if the uncertainty is too extreme. Operators usually have at their disposal additional mechanisms to re-balance the system if the regulation reserves reach saturation, such as reserve substitution or renewable generation curtailment. However, accounting for additional regulation mechanisms is outside the scope of the current manuscript.

Figures 4.7(a) and 4.7(b) can be incorporated into the operator's risk dashboards such as those already common in many ISOs (e.g., SolarView [50] or the examples shown in [61]). For instance, an operator observing that the bottom and top 20th percentiles will saturate the reserve like in Fig. 4.7(b) can initiate corrective actions before the operating hour, or plan for the following operating hour. In the case of down-reserve saturation, the operator can choose to curtail, or offer energy to other interconnected balancing areas.

Reducing system risk

In this section, we show the execution of a preventive action to reduce the system's exposure to risk. Based on the results from Figs. 4.7(a) and 4.7(b) we mimic the decision process available to operators working with increased awareness. During the morning period, the ACE projection is large at the end of the forecast horizon. However, the increase in ACE happens in the second hour of the forecast after another execution of the HAUC. As a result, an operator can choose not to perform any risk reduction measure for the following operating hour. On the other hand, during the mid-day period, there is a sustained ACE in both directions during the operating hour that can be considered unacceptable. As a result, the operator can perform a risk reduction measure. In this case, we chose to increase the regulation reserves by 30% from 0.83 GW regulation up and 0.81 GW regulation down to 1.079 GW and 1.053 GW, respectively, as an example.

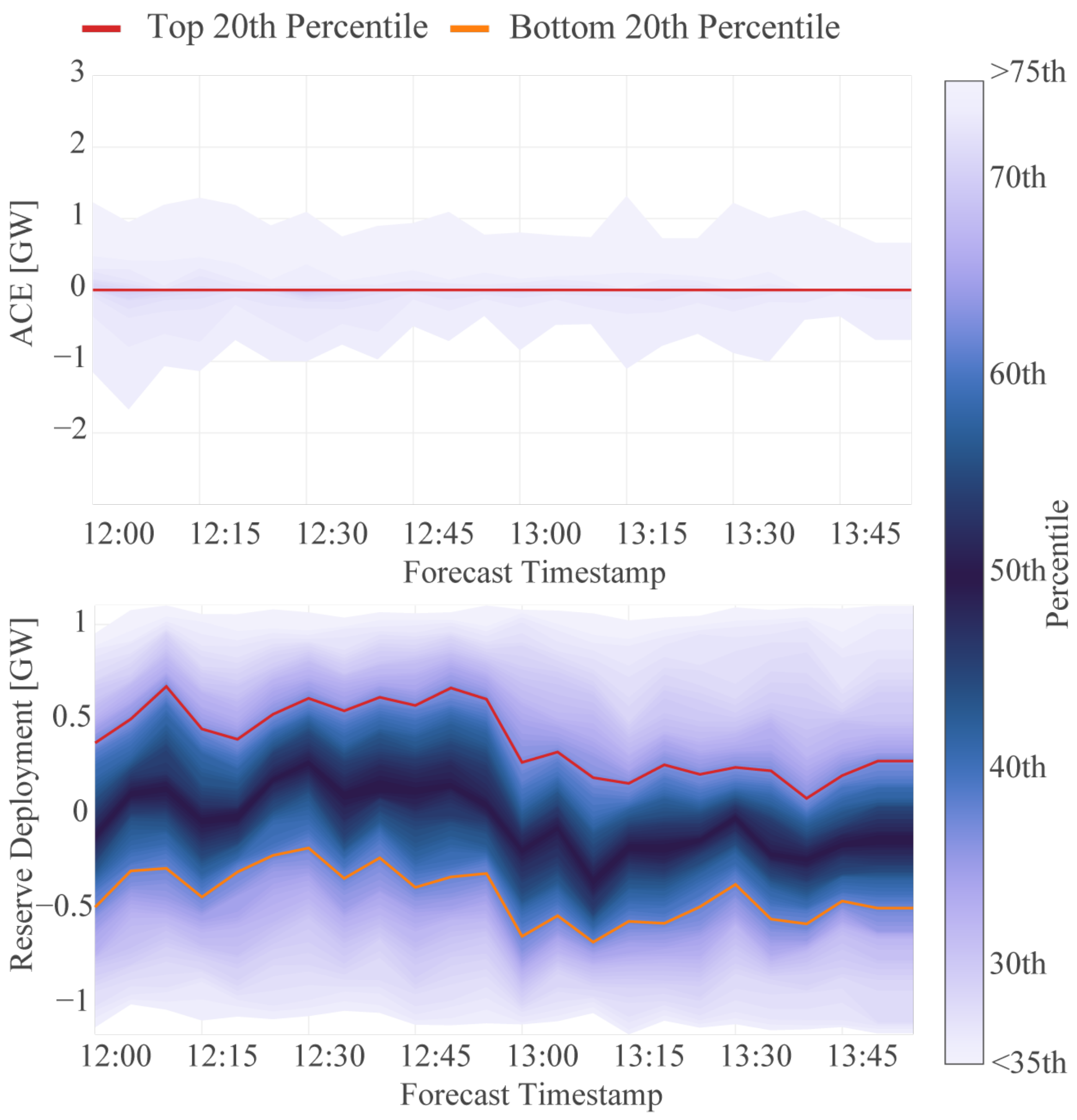


Figure 4.8: Probabilistic result after increasing reserve requirements during the mid-day period

The outcome of this adjustment in shown in Fig. 4.8, where the system now has significantly more of a margin to balance the system variability. We can observe that the negative ACE value is now 0.0 for most of the top 20th percentile (i.e., avoiding over-generation). Figure 4.7(a) shows that even though the system has more reserves to handle variability and reduce the risk at the top and bottom quantiles, it is still close to saturation. Note that although the reserve still has some probability of saturation, the extreme ACE values have also been reduced from a 2 GW value to 1 GW.

Results Under Low Variability

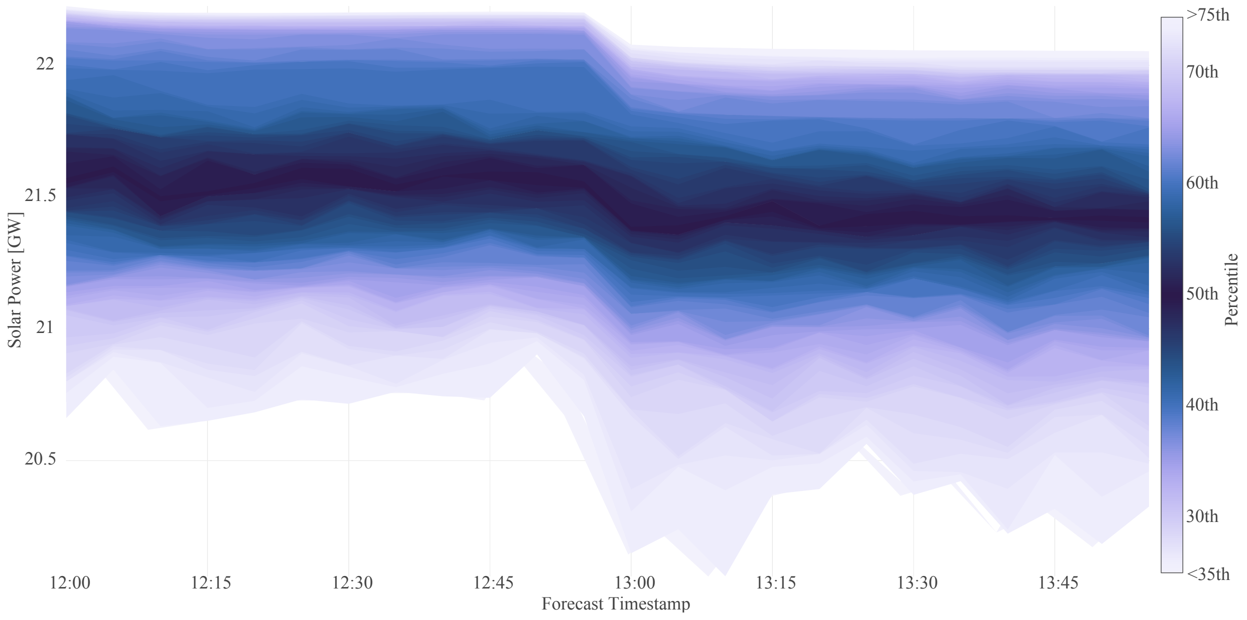


Figure 4.9: Low variability forecast, 12:00 pm, July 22nd, 2018

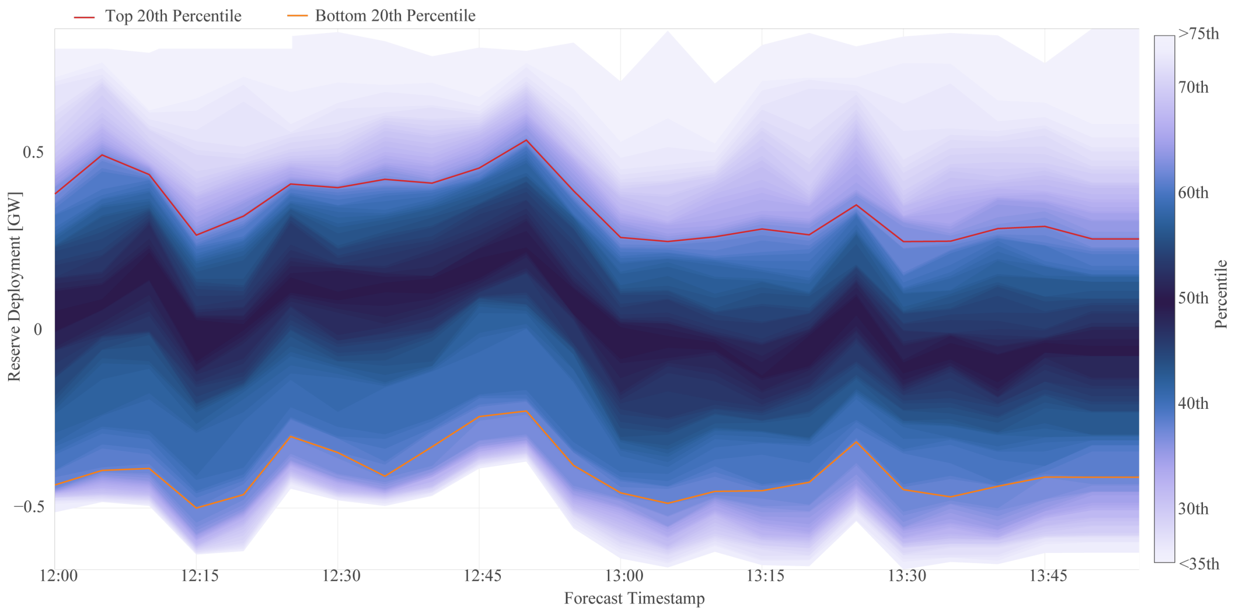


Figure 4.10: Reserve deployment forecast, 12:00 pm, July 22nd, 2018

The results shown so far are representative of a highly variable spring day where the system is at higher risk. Figure 4.9 shows a forecast for noon during a summer day. When compared with the forecast in Fig. 4.6, we can see the spread of the power in the probabilistic forecast is 1.5 GW during the 2-hour period, whereas in April the spread is 7 GW during the same period. Given the relatively low variability and the higher reserve requirements for the summer season, the forecast of reserve deployments is well within the $\pm$ 1.2 GW regulation reserve allocation (see Fig. 4.10). Given that the system has enough regulation capacity, the ACE is projected to be 0.0 for the entire horizon. Such a result allows the operator to know that the system can handle even the most extreme potential variations of VRE within the forecast window.

Computational performance

Although the proposed approach is not yet suitable for online decision making due to the computational burden, it is critical that the operators get the risk assessment in time to perform preventive actions if needed. We ran the model in commercial hardware. The CPU was an Intel(R) Xeon 10-core Ivy Bridge 2.5 GHz with 64 GB of RAM on Scientific Linux 7. As a stopping criterion, we used a relative tolerance of 1e-3 after five consecutive training iterations (see `SDDP.jl` documentation for more detail). We used CPLEX 12.10 as the linear programming solver for the algorithm parametrized with the the option `"CPX-PARAM_Emphasis_Numerical" = 1` in order to avoid numerical issues in the execution of the SDDP algorithm. Additional details about the implementation are available in the code repository.

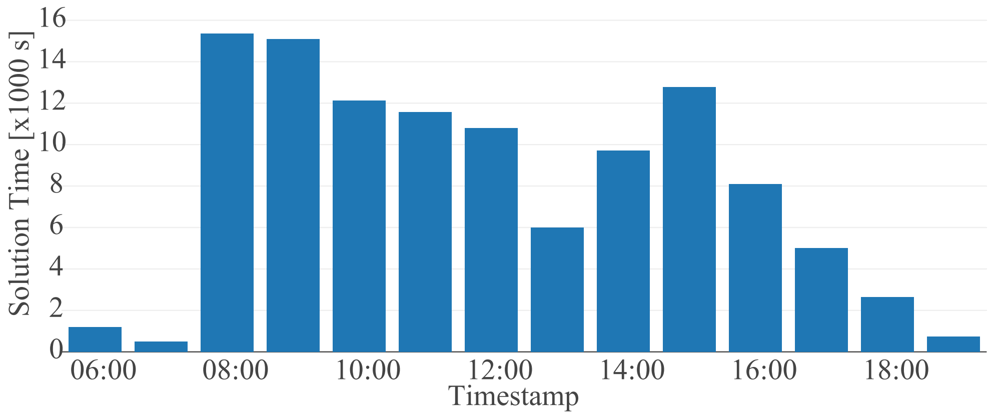


Figure 4.11: Solution times of the `SDDP.jl` algorithm execution including forward and backwards pass

Figure 4.11 shows the full solution time of the model. The number of state variables differs at every hour because the system will have a different number of units online, impacting the algorithm's performance. At each solution call, the SDDP model solves over 100 state variables, and during the ramp times the value can be as high as 165 state variables.

The worst-case solution time is 15000 seconds (4 hours) at the 8:00 timestamp when the system experiences the most stress with the solar ramp and the largest number of state variables. This solution time is reasonable for an initial proof of concept done under realistic operation conditions using commercial hardware similar to that available to operators and with a generic implementation of the SDDP algorithm. However, it will need improvements before it can be used by system operators in real time. To reduce computation times, a time limit (e.g., 60 minutes) could be used at the trade-off of sub-optimal policies. Alternatively, improvements such as parallelism, tuning of solver parameters, and heuristics to construct an initial approximation of the value function could be investigated. We leave these to future work.

The computational cost of the SDDP solution technique scales with the number of state variables, stages, and scenarios. Increasing the size of the individual linear programs by adding additional constraints such as network or other generator constraints will increase the solve time of the individual problems, but it should have only a minimal impact on the overall complexity of the algorithm. The impact of the additional constraints is highly dependent on the structure of the new constraints and the solver's capability to simplify the resulting problem. We leave an investigation of different formulations to future work.

4.4 Online transition matrix estimation

In this section, we extend the Markovian representation of the temporal dependence structure discussed in Section 4.1, where we employed a fixed transition matrix created assuming a truncated normal distribution centered on the diagonal. This method estimated the average likelihood that the actual area-wide power would transition from one forecast quantile to another and fit a normal distribution. In the extension, developed in this section, we look at updating the transition matrix online using a subset of recent probabilistic forecasts and observations, hoping this will better capture uncertainty. In line with ERCOT's real-time electricity market, a new solar power observation and probabilistic forecast are released every 5 minutes. We update the transition matrix using the following steps:

1. Specify the historical time window and number of lead times to use in creating the online transition matrix.

2. For each day in the historical time window, use the forecasts issued within an hour of the current time (e.g., if the current forecast starts at 7:00, use forecast from the previous day(s) starting at 7:00, 7:05, ..., 7:55) to populate the transition matrix using transitions from the specified number of lead times.

3. For rows containing no observations, set $m_{j,j}$=1.

However, there is no clear mechanism to specify the historical information used such as time window and the number of lead times required to find the best fit. Hence, we develop a set of metrics to evaluate the possible parameterizations of the fitting procedures.

Transition matrix scoring metrics

To determine which transition matrix performs the best, we sample trajectories from the probabilistic forecast using the transition matrix and score them using the VS [160] as well as two new metrics based on the principle of band depth [170]. Many other studies use the energy score [54] to assess performance, but recent work shows that this score cannot diagnose misspecifications of the dependence structure [109], so we choose to omit this score.

Variogram Score

Scheuerer and Hamill introduced the VS [160], which is a negatively oriented proper scoring rule [54] that captures each unique pairwise difference (i.e., time lags from 5 minutes to 115 minutes in this application) considering all the components in the multivariate forecast. The VS of order p can be written as

$$\begin{gathered}\text{VS}_p(F_t,\mathbf{o})=\sum_{i,j=1}^{d} w_{ij}\left(|o_i-o_j|^p-\frac{1}{S}\sum_{k=1}^{S}|x_i^{(k)}-x_j^{(k)}|^p\right)^2 \\ \forall i,j=1,2,\dots,d\end{gathered} \tag{4.9}$$

when the forecast distribution F_t is approximated by the ensemble consisting of the S solar power trajectories sampled from the transition matrix $\mathbf{x}^{(1)},\dots,\mathbf{x}^{(S)}$. Here, $x_i^{(k)}$ is the i^{th} component of the k^{th} trajectory, $\mathbf{o}$ is the d-variate observation vector, and w_{ij} gives weights to each pair. We set $p=0.5$ and use the inverse of the time lag between components in minutes as weights. Note that vector quantities are in bold.

Band Depth Scores

The band depth is a statistic that provides a way to order a set of curves [96]. Curves within a set that have larger band depths are more central, whereas curves with smaller band depths are more outlying. We propose a scoring metric that consists of the normalized band depths of the set of trajectories $\vec{x}^{(k)}$ together with the observation vector $\vec{o}$. We calculate the Band Depth Score (BDS) by normalizing this set of band depths with respect to the highest band depth in the set.

Following the notation from López-Pintado [96], we represent the graph of a function with $y(t),i=1,\dots,n,t\in\mathcal{I}$, which is a subset of plane $G(y)=\{(t,y(t)):t\in\mathcal{I}\}$ where $\mathcal{I}$ is an interval in $\mathbb{R}$. For simplicity, we assume that a band in $\mathbb{R}^2$ is delimited by only two curves $(y_{i_1}, y_{i_2},)$ which we represent by $B(y_{i_1},y_{i_2})=\{(t,x(t)):t\in\mathcal{I},\quad \min_{r=1,2}y_{i_r}(t)\leq x(t)\leq \max_{r=1,2}y_{i_r}(t)\}$. The sample band depth is then given by the fraction of all possible bands that contain the full curve $y(t)$:

$$\mathrm{BD}(y)=\binom{n}{2}^{-1}\sum_{1\leq i_1<i_2\leq n} I\{G(y)\subseteq B(y_{i_1},y_{i_2})\} \tag{4.10}$$

where $I\{\cdot\}$ is the indicator function.

López-Pintado also offers a more flexible formulation called the modified band depth, which measures the fraction of time for which a curve $y(t)$ falls within each band

$$\mathrm{MBD}_n(y)=\binom{n}{2}^{-1}\sum_{1\leq i_1<i_2\leq n} \lambda_r\{A(y;y_{i_1},y_{i_2})\}, \tag{4.11}$$

where $A(y)\equiv A(y;y_{i_1},y_{i,2})\equiv\{t\in\mathcal{I}:\min_{r=i_1,i_2}y_r(t)\leq y(t)\leq\max_{r=i_1,i_2}y_r(t)\}$, $\lambda_r(t)=\lambda(A(y))/\lambda(\mathcal{I})$, and λ is the Lebesgue measure on $\mathcal{I}$. Similar to the BDS, we propose an additional scoring metric, the Modified Band Depth Score (MBD), by normalizing the set of modified band depth with respect to the highest modified band depth in the set.

Online transition matrix performance

To determine the most effective online transition matrix, we iterated through a subset of the parameter space comprising the number of days and number of transitions used from each historical forecast. We tested 1 to 45 days using 1, 6, 12, and 23 transitions per forecast. Since each forecast has a 5-minute resolution and a 2-hour horizon, 23 transitions use data from the full forecast, whereas 6 and 12 transitions use data from the first half-hour and hour, respectively. Five hundred solar power trajectories were sampled using each transition matrix, and the VS, the BDS, and the MBDS were calculated using the observations. This process was repeated 10 times to obtain average scores for one day in each season.

Online transition matrices associated with the best VSs for four date and time combinations are shown in Fig. 4.12(a)–(d) and illustrate how the sparsity and density of the transition matrix change under different conditions. With the help of fan plots depicting the probabilistic forecast and observations, shown in Fig. 4.12(e)–(h), we can understand why these matrices performed well. Looking at Fig. 4.12(e), initially, the forecast under-predicts the observations, which is common near sunrise, but the observations fall to the lower quantiles by the end of the time horizon. In such a case, the transition matrix has greater density above the diagonal — corresponding to a downward quantile traverse — and some density around high and low quantiles where the forecast remains for several lead times. Fan plots in Fig. 4.12(f)–(g) both have observations that remain near the same quantile range for the duration of the forecasts. Here, the density in the transition matrix corresponds to the quantiles among which the observations rank. In other words, the more clustered a transition matrix is around a group of quantiles, the more likely a trajectory starting out in this quantile range is to remain there. An extreme example of this occurs at 15:30 on November 16th, shown in Fig. 4.12(h). Near sunset, the forecast becomes compressed and often under-predicts the observations (e.g., Fig. 4.12)).

Subplots (a) through (d) contain online transition matrices created for different times to illustrate several major trends that appear in the online transition matrices. Each subplot title provides the time for which the transition matrix was created, the number of transitions used per forecast, and the number of days of historical forecasts used to create the transition matrix. Subplots (e) through (h) contain fan plots depicting the forecast starting at the same times for which the transition matrices were created. Probabilistic quantile forecasts are shown in red with darker colors corresponding to more central quantiles, and the benchmark observations are shown with a solid black line.

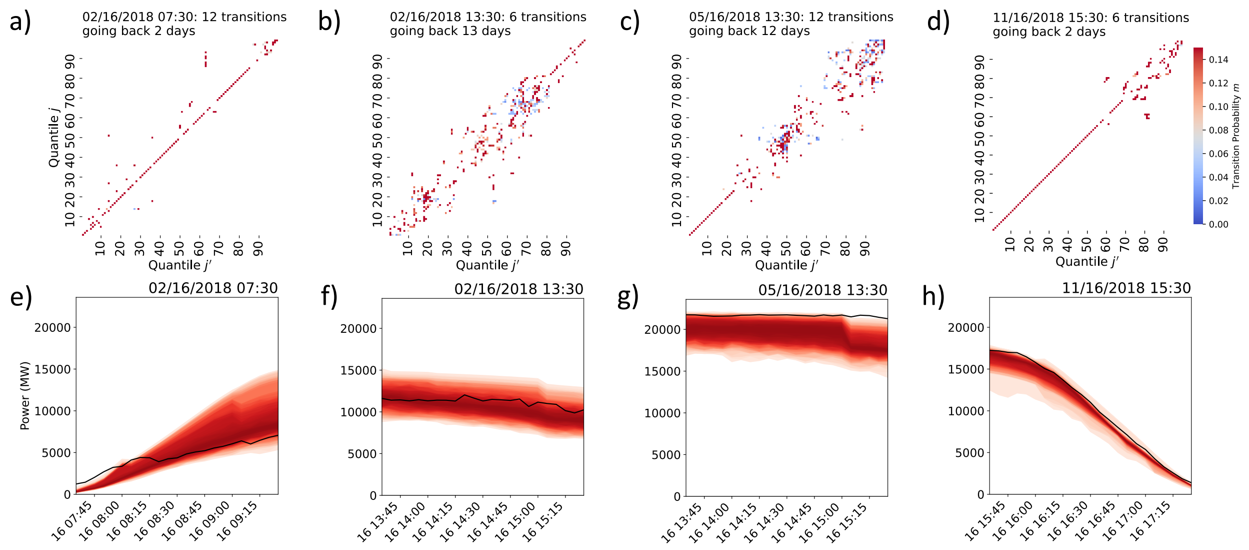


Figure 4.12: Online transition matrices created for different times

Heat maps showing the VS and MBDS for transition matrices created using all parameter combinations at six times on each of the four days are provided in Fig. 4.13. We observe both diurnal and seasonal variation in the VS (left panel) and the MBDS (right panel) along with variation associated with the amount of data used to create the transition matrix at a given date and time as shown by each individual heat map. Since the VS is negatively oriented, lower values are better. However, the opposite is true for the MBDS. To simplify the visual interpretation, we orient the color maps in Fig. 4.13 such that lighter values indicate better scores and darker values indicate worse scores. The best VSs tend to occur later in the day when the sun is going down and the magnitude of any misprediction is lower. However, this same period has poor MBDSs, because forecasts often under-predict the solar power output near sunset, meaning that the observations fall away from the most central trajectory. The worst VSss occur in the beginning to middle of the day on February 16th (see the first, third, and fourth heat map slices on the top left row of Fig. 4.13) and at 13:30 on May 16th (see the fourth heat map slice on the second left row of Fig. 4.13). We can attribute the poor scores on February 16th to higher variability present in the observations, which is visible in Fig. 4.12(e)–(f). Those at 13:30 on May 16th we attribute to a sharp change in the forecast that is not realized in the observations (see Fig. 4.12(g)). The worst MBDSs occur at 17:30 on February 16th, at 15:30 on May 16th, and at 17:30 on November 16th. In all of these cases, the forecasts under-predict the observations — similar to the case shown in Fig. 4.12(h).

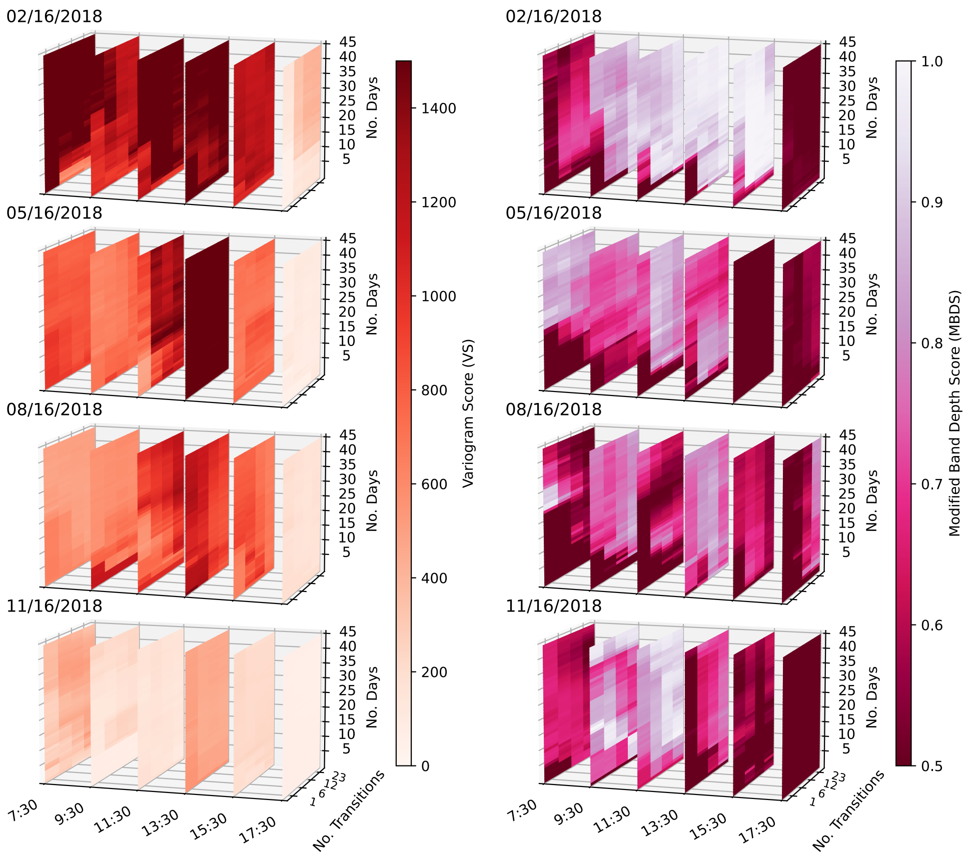


Figure 4.13: VS (left) and MBDS (right) for each online transition matrix that was created

While we presented some of the best-performing transition matrices as judged by the VS, the BDS and the MBDS provide operators with additional useful information. For example, if an operator schedules reserves based upon the 20th and 80th quantiles, the BDS could be tuned to indicate whether a trajectory generated using the transition matrix would fall outside these quantiles at any point, and the MBDS could indicate what fraction of the time a trajectory would fall outside these quantiles.

We found that the optimal amount of data for creating the transition matrix varied by both season and time of day. While we didn't observe a clear trend in the length of the historical data window, we can recommend that using fewer transitions from each forecast in the spring and summer months produces better transition matrices. Finally, we demonstrated this approach using a multi-stage stochastic risk-assessment model under uncertainty.

Naturally, an individual solar facility will display greater inter-hourly variability than the balancing area aggregation due to localized variations in cloud cover and aerosols. Future work could apply this methodology to forecasts for an individual solar plant, and the resulting transition matrix would reflect the increased variability. This would extend the applicability of this method to entities such as solar facility operators and vertically integrated utilities. Further, such entities could likely leverage ground-based observations in place of the NSRDB-based observations used here, which would more accurately capture the variability at the site. Using local observations may also result in a transition matrix showing more variability than that created using the NSRDB-based observations.

Using online transition matrices operationally

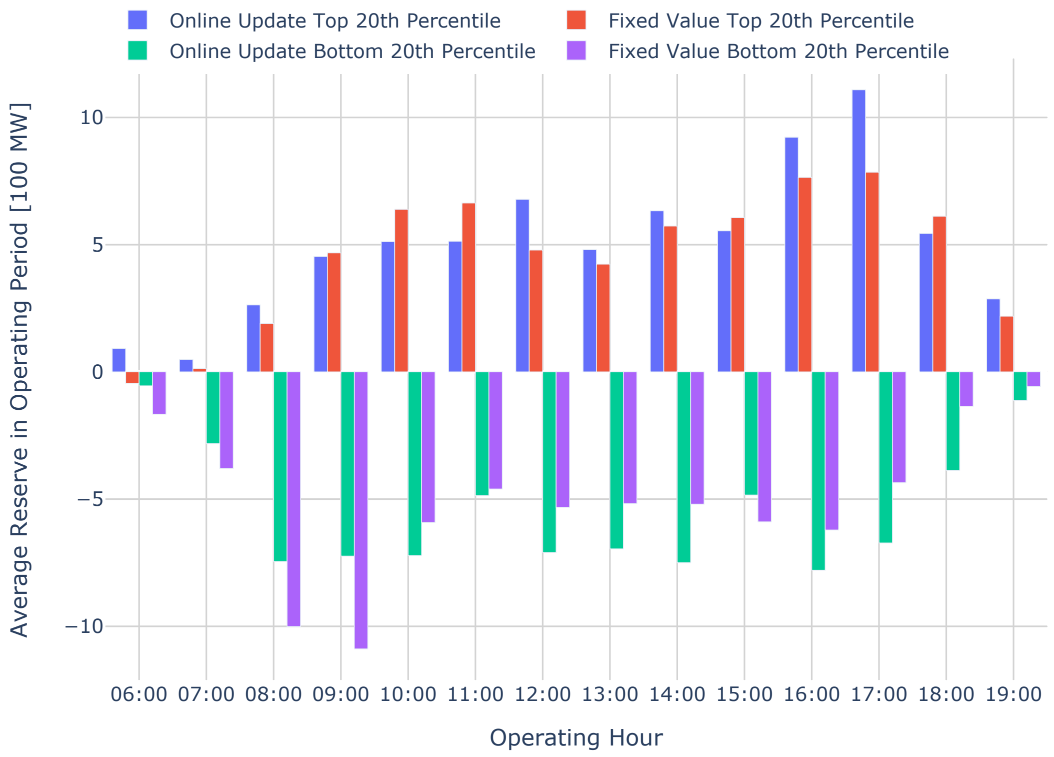


Figure 4.14: Comparison of the predicted average reserve deployment for the operating hours with solar power production

To showcase the online transition matrix in operation, we ran the model specified in [86] on April 1st, 2018, a highly variable spring day in Texas. We present results for the top and bottom 20th percentiles, which operators commonly use as thresholds. The online method predicts larger variability during the late afternoon hours (Fig. 4.14), which results in larger reserve deployments — particularly between 16:00–17:00, where the model estimates up to 120 MW of necessary additional reserves. The opposite occurs in the morning, where the online method consistently predicts less variability than the fixed matrix approach. Positive results represent the deployment of upward regulation reserves, and negative results represent the deployment of downward regulation reserves.

The expected ACE varies proportionately to the reserve deployments, with higher expected ACE values occurring at the same time as higher reserve requirements (Fig. 4.15). During the morning, the system reaches its maximum reserves at the same time as maximum ACE (9:00). However, the model predicts that the downwards reserve deployment at 9:00 is significantly larger using the fixed matrix, which results in saturation, creating a large discrepancy in the predicted ACE values between the online and the fixed matrix methods. In contrast, ACE values around noon are large for both methods due to an under-allocation of reserves. Positive results represent excess generation, and negative values represent generation shortfall.

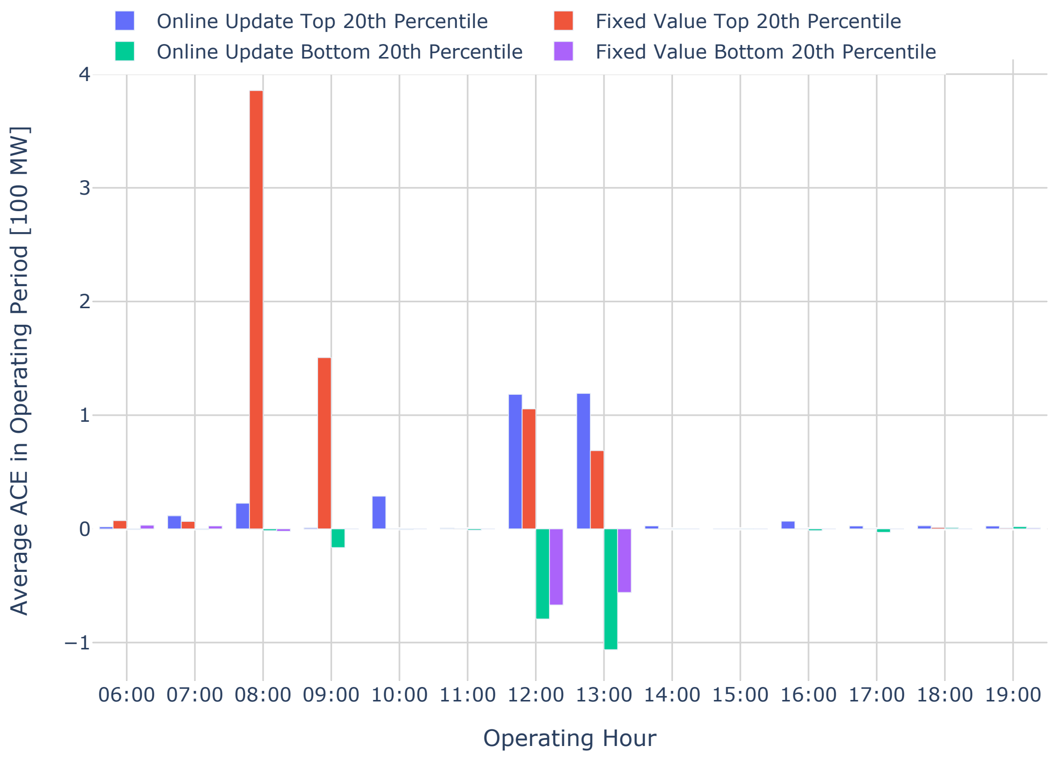


Figure 4.15: Comparison of the predicted average ACE for the operating hours with solar power production

4.5 Conclusions

- System operators still face significant challenges in integrating probabilistic forecasting into operational workflows. Emergency management system operators (operator EMSs still have technical limitations that create information technology challenges in receiving probabilistic forecasts and processing them into valuable insights for floor operators. Additionally, the need for institutional approval from regulatory organizations can become a major roadblock to the integration of probabilistic information in the operator's control room; hence, uncertainty forecasts are currently an underutilized tool in operating rooms. We propose a modeling framework to incorporate uncertainty forecasts for short-term risk assessment in ISO operations that can be integrated into risk assessment platforms such as SolarView [50] to provide additional insights about the system's risk and help close some of the gaps that still prevent probabilistic forecasts from being more widely adopted in power system operations. We showcase that the inherent time dependencies in the probabilistic forecasts can be used to formulate a multi-stage problem as a *Markovian graph*.

- Using the Markovian graph, it is possible to incorporate joint time dependencies of the control variables and the forecast and to find solutions in reasonable computing times. The results show that time consistency in the sequential decision modeling can represent limitations in the deployment of corrective actions over the operating hour. We present an implementation of the risk assessment stage in the operator's sequence of decisions in a system with realistic forecasting data and a large generation fleet size.

- In this chapter, we offered two distinct methods for capturing the temporal dependence structure of probabilistic solar forecasts based on the Markov property and presented a strategy for updating the Markov transition matrix using recent forecasts and observational data. We developed two multivariate scoring metrics based on the band depth and modified band depth of a vector of observations with respect to a set of trajectories and compared them to the well-known variogram score.

Chapter 5

Simulation of Power Systems Dynamics

For several decades, computational simulations have been the primary tool for studying power systems dynamics and stability. The scale and complexity of interconnected power systems limits the applicability of simpler analytical techniques. Despite a wealth of knowledge about models, simulation techniques, and software implementations[117], numerous advancements in computer hardware, programming languages, and numerical integration techniques make for more changes and advancements. The changing nature of the grid also catalyzes innovation in simulation models and techniques.

The increasing integration of generation sources via power electronics is changing power systems. It is generally agreed that new dynamics in the controls of Inverter-based Resources (IBRs) change modeling requirements for system-wide stability studies that rely on time-domain simulations [133, 60, 120]. In power systems dominated by synchronous generators, physical phenomena (magnetic fluxes, electromechanics, mechanical control reaction times, or thermo-dynamic processes) drive dynamic behavior. The control logic associated with synchronous generators is commonly tuned to the timescale of the relevant processes, creating a natural separation between the dynamic behaviors attributable to physics and controls. On the other hand, IBR dynamics are dominated by their controls, including modulation, PLLs, voltage, current, and power controllers. Therefore, the practical requirements of the control design – often, cascading PID – define the relationships between timescales. In fact, interactions at higher frequencies have recently been recognized as a new stability category [60], highlighting the exigent need to revisit the understanding of system dynamics with high shares of IBRs.

Determining the level of modeling detail required to accurately capture the phenomena of interest is a key challenge in power systems simulation in the presence of IBR. The modeling choices determine the algorithmic and computational requirements that must be met to execute the time-domain simulation. In systems supplied predominantly by synchronous generators, IEEE Std 1110-2019 [73] guides the generator model complexity requirements based on the category of stability under study and the severity of the perturbation. No such modeling guidance exists for IBR-dominated systems.

Several articles have explored the modeling requirements for systems with IBRs and shed light on the theoretical [106, 121, 66] and practical [71] requirements for time-domain simulations in large systems, particularly where network circuit dynamics are concerned. Decisions about simulation models are often framed in terms of “slow" and “fast" dynamics, though formal definitions of these two categories depend on the context in which they are used. This chapter presents two formalizations of “slow" and “fast" in terms of a signal's bandwidth and model order reduction through Singular Perturbation Theory (SPT).

Challenges concerning time-domain simulations in the context of IBR dominated systems have recently been discussed by the authors in [133] where they study the applicability of specific techniques to the simulation and modeling of systems with IBRs from the signal processing perspective. A recent review [169] discusses existing and novel methods to accelerate Electro-magnetic Transient (EMT) simulations in systems with IBRs. Composite system models for network, generators and inverters of different types are described in [95, 184].

The new control and analysis challenges that come with the integration of IBRs also bring modeling and solution difficulties for incumbent approaches and highlight the need for fundamental changes in the domain of power system simulation. Practitioners in the power system community have reported limitations of existing commercial tools for systems that have sizable IBR share and/or weak interconnections. For instance, an ERCOT report [149] notes existing non-convergence and numerical instabilities in weak grid situations.

Moreover, nearly all power system dynamic simulation packages integrate the modeling layer (where the behavior of system elements is specified) and the simulation layer (where the algorithmic solution approach is defined). In almost all cases, the details of these layers are inaccessible to the developer and limit model development to the pre-defined structures of the solution algorithms available. The close integration between the algorithm and the models, combined with the development hurdles, licensing, and software limitations described above, restricts the scientific study of power systems with high share of IBRs. As a result, it is necessary to develop simulation tools with enough flexibility to explore the modeling and solution aspects of systems with large shares of IBRs.

This chapter first clarifies the semantics used to describe time-domain simulation models and their applications to system-wide simulations in the presence of IBRs given the importance of developing novel simulation techniques that can capture IBR dynamics more accurately. The objective in this chapter is to interpret the assumptions and modeling approaches implicitly embedded in widely used definitions and use it to develop a simulation tool that addresses the shortcoming of existing approaches. Particular emphasis is provided to reviewing *simplifications* and *transformations* inherent in various time-domain simulation methods as well as discussing the capabilities and limitations of these approaches.

This chapter further describes a new modeling platform, `PowerSimulationsDynamics.jl` (`PSID.jl`) based on the Julia language. This particular package provides the computational tools needed to develop models and algorithms to simulate systems with a focus on IBRs based on a simulation model's theoretical capabilities. The tool has the flexibility to formulate models with different levels of detail. To date, `PSID.jl` is the only open-source

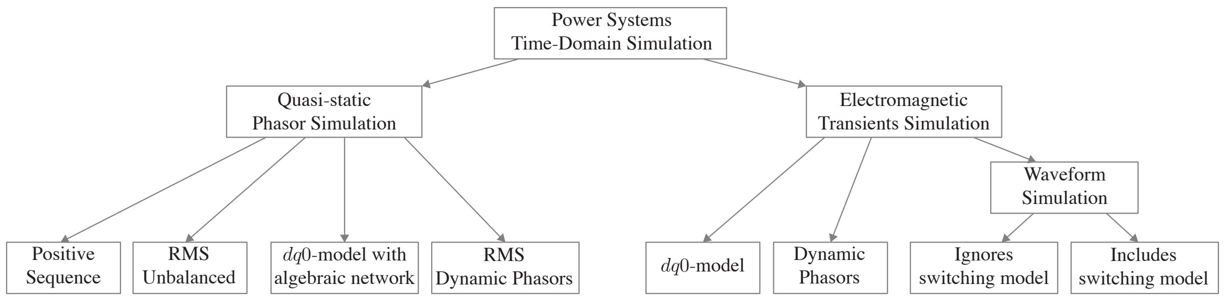


Figure 5.1: Taxonomy of power systems time-domain simulation models.

power systems dynamics tool developed explicitly to enable extensibility at multiple points of the modeling stack, from the component to the integration algorithm.

This chapter will:

- Review power systems time-domain simulation modeling approaches with applications to IBRs.
- Present a systematic discussion of the origins and underlying assumptions of different power systems dynamic modeling techniques, including examples of their application.
- Survey modeling approaches and terminology used in the literature, and their level maturity of their implementation in commercial and research applications.
- Provide an overview of the design and implementation details of `PowerSimulations-Dynamics.jl`, a package which helps researchers easily assess the trade-off between model complexity and computational requirements.
- Describe an implementation of IBR and machine models based on generic data models that supports developing encapsulated sub-component models. The modular design enables code and model reuse; this reduces development requirements and enables fast and simple prototyping of controls and models.

The chapter will first provide readers with the relevant definitions that concern simulation models and the methods involved. Specifically, section 5.1 discusses the definition of power systems time-domain simulation methods and introduces the taxonomy of the simulation models described in this chapter. Section 5.2 reviews common simplifications and transformations used in time-domain simulations relevant to defining the scope of simulation model definitions. In Section 5.3, we discuss systems simulation focusing on definitions and categorizations. Section 5.4 introduces the implementation and modeling techniques in `PSID.jl`. Section 5.4 discusses the simulation models implemented in `PSID.jl` including the residuals and mass-matrix formulations. Section 5.4 discusses the details of the software design and structure as well as the modeling details for generators, inverters, loads and network. The simulation case studies and verification results are shown in Section 5.5 where we compare the results from `PSID.jl` with commercial tools to verify the validity of the results. Finally, conclusions are presented in Section 5.6.

Notation

Lower-case letters x denote one-dimensional real variables, parameters, and functions; upper-case letters in the form F and $F()$ are used for matrices and functions respectively; arrows (as in $\vec{x}$) represent vectors of variables or parameters; $\langle\cdot\rangle$ denote phasors; and lower-case letters in the form $\boldsymbol{x}$ denote complex-valued variables, functions, or parameters. Finally, $\jmath{=}\sqrt{-1}$.

5.1 Time-domain Simulation

In this chapter time-domain simulation methods means models of the dynamics of power systems as a function of time. This type of simulation model is typically used to determine the transient effects of sudden changes to the system state.

A time-domain simulation requires specifying two “layers": (i) a system model, including differential equations that describe the system's physics and controls, and (ii) a time-stepping (or integration) algorithm. The choices made around the system model inform integration algorithm requirements. This section precisely defines what we mean by “simulation model" and provides a taxonomy of widely used methods.

Definition

A system simulation model is a set of dynamic equations in causal form for a collection of interconnected components expressed with explicit differential equations:

$$\frac{d\vec{\boldsymbol{x}}(t)}{dt}{=}F(\vec{\boldsymbol{x}}(t),\vec{\boldsymbol{y}}(t),\vec{\eta},t), \quad \vec{\boldsymbol{x}}(t_0){=}\vec{\boldsymbol{x}}^0, \tag{5.1a}$$

$$\frac{d\vec{\boldsymbol{y}}(t)}{dt}{=}G(\vec{\boldsymbol{x}}(t),\vec{\boldsymbol{y}}(t),\vec{\psi},t), \quad \vec{\boldsymbol{y}}(t_0){=}\vec{\boldsymbol{y}}^0, \tag{5.1b}$$

where $\vec{\boldsymbol{x}}(t)$ and $F(\cdot)$ represent the device (e.g., IBRs, machines, loads) states with parameters $\vec{\eta}$. The circuit dynamics of the network are represented as the subsystem $\vec{\boldsymbol{y}}(t)$ and $G(\cdot)$ with network parameters $\vec{\psi}$.[1] The simulation model (5.1a)-(5.1b) can be used to represent real- and complex-valued signal analysis.

Given the system model (5.1a)–(5.1b), a simulation can be defined as follows: from an initial condition for the device and network states $\vec{\boldsymbol{x}}(t_0),\vec{\boldsymbol{y}}(t_0)$, advance the solution in time t from one point to the next considering a discrete timeline $\{t_0,t_1,...,t_n,...,T\}$. A

[1]Simulation models could include a spatial component, as in the case of circuit models that consider traveling waves. In cases where the behavior of traveling waves is relevant, time-delay equivalents such as the Bergeron model are commonly used and can be implemented via (5.1a)–(5.1b).

simulation requires a stepping algorithm that finds the solution at time t_{n+1} given values of the involved variables over $\{t_0,t_1,...,t_n\}$.

In steady state, voltage and current state variables in the model defined by (5.1a)–(5.1b) vary sinusoidally in time, making the dynamic response dependent on the value of t. Without some form of transformation, differential equations describing time-invariant system dynamics do not have well-defined equilibria.

Solving time-varying nonlinear systems is costly because integration algorithms use fixed time-steps, whereby equilibrium points cannot be defined. Additionally, modeling three-wire three-phase power system components can result in intractable expressions with cross-coupling terms. (For example, in the case of asymmetrical circuits or unbalanced signals.)[2] As we will see, *simplifications* and *transformations* influence the maximum required discretization step, Δt, and the number of states required to model the dynamics of interest.

Taxonomy

In power systems, time-domain simulation is roughly classified into two groups: (1) *QSP Domain Simulations* in which the transmission system circuits dynamics are represented algebraically as discrete changes between steady-state operating points; and (2) *EMT Simulations* which include sufficient detail to capture fast dynamic phenomena. QSPs are used for the study of low-frequency phenomena that ranges from inertial response to frequency regulation. On the other hand, EMT simulations are used in settings where one wishes to capture the impact of line dynamics, converter switching, machine fluxes, and/or lightning surges.

Figure 5.1 depicts a taxonomy of time-domain simulation models found in the literature and covered in this chapter. Though we have organized model types as subcategories of QSP domain models and EMT models, there are various types of models within each category. We have organized model types roughly by the fastest-timescale phenomena they are intended to capture. As we will explain, although modeling faster-timescale phenomena requires more modeling detail, there are also other important differences between these models. Their mathematical formulations, the algorithms available to numerically integrate them, and the assumptions required to interface machines and other energy conversion interfaces across a network differ in important ways within the taxonomy. These differences can introduce fundamental changes in a model's properties and output, influence the complexity of initializing model runs, and dictate the maximum allowable time step (and, therefore, the computing time and data generated).

For practitioners, the definition of a simulation model is intertwined with the software environment. Each simulation category (QSPs or EMT) has highly specialized models, algorithms, and modeling practices developed over many decades. As a result, choosing a

[2]The terminology adopted follows [199], which uses *balanced* in the context of signals and *symmetric* in the context of circuits.

simulation model and its methods is inextricably tied to to the software environment and its capabilities. Therefore, in order to clearly talk about the difference between models in the taxonomy, we will introduce two broad classes of modeling choices: *simplifications* and *transformations*. The following section systematically reviews the broad space of potential simplifications and transformations. Subsequently, we discuss specific simulation tools in the context of these simplifications and transformations. This enables us to begin to understand the entire range of assumptions behind different simulation tools and to move beyond the simple distinction of whether or not they are designed to study electromagnetic transients and inform several design choices when developing simulation software.

5.2 Simplifications and Transformations

By “simplification" and “transformations," we mean mathematical manipulations of state variables that preserve the validity of some portion of a model’s characterization of system physics and control loops. Transformations can have multiple equivalent formulations, and several works have developed a comprehensive analysis of transformations [119, 129] and arrived at equivalent conclusions to the ones presented here. Different from transformations, simplifications are aimed to reducing model complexity at the cost of an approximation to the real system dynamics.

Simplification: Averaging Dynamics

Averaging methods focus on determining how the behavior of a complicated time-varying system can be approximated by a time-invariant system [77]. Averaging reduces model bandwidth requirements, which, in turn, enables simulations to increase the maximum allowable time-step Δt. Averaging can be used to simplify switching dynamics and analysis of power electronic converters: see, for example, [199].

Averaging techniques are critical to reducing the computational cost of conducting simulations. For instance, [202] notes that with the application of averaging techniques, the integration step of a power system model can go from 50ms to as large as one second for low-frequency dynamics. Phasor representations of signals are regularly used as simplifications in power system simulations, and we discuss several approaches in the remainder of this subsection, along with their connection to averaging.

The remaining of this section reviews several simplifications and transformations commonly found in the power systems literature. The focus of the review is on their applications for the modeling if IBRs.

Steady State Phasors

Steady state phasors are commonly defined in engineering. Given the signal $s(t){=}s_{pp}\cos(\varrho t{+}\theta){=}\sqrt{2}\ \mathrm{Re}\{Se^{\jmath\varrho t}\}$, the “phasor" $S{=}\frac{s_{pp}}{\sqrt{2}}e^{\jmath\theta}$ is expressed using the root-mean-square value of the wave instead of the instantaneous value. This definition requires stationary conditions and homogeneous frequency.[3]

Dynamic Phasors

Dynamic phasor as a concept was introduced in the early 1990s and has received renewed attention as a mechanism for simulating systems, including fast IBR dynamics. There are at least three independent but similar definitions of “dynamic phasor" in the literature, each developed using different approaches. The three approaches are motivated by the observation from averaging theory that the study of signal envelope variations is sufficient to derive the systems’ stability properties. Sanders *et al.* [156] introduced *dynamic phasors* via generalized averaging involving a time-dependent sliding-window interval $\mathcal{T}(t)=[t-T,t]$ for time-varying systems. In particular,[4] consider the following expansion for a *nearly periodic* signal $s(t)$:

$$s(t)=\sum_{k=-\infty}^{\infty} \langle s\rangle_k(t)\; e^{\jmath k\varrho t}. \tag{5.2}$$

where ϱ is the *carrier frequency* of the signal. The time-varying Fourier coefficients $\langle s\rangle_k(t)$, one for each harmonic component k, are recoverable as

$$\langle s\rangle_k(t)=\frac{1}{T}\int_{\tau\in\mathcal{T}(t)} s(\tau)e^{-\jmath k\varrho\tau}\mathrm{d}\tau, \tag{5.3}$$

and these are regarded as *dynamic phasors*. The time dependency of $\langle s\rangle_k(t)$ enables the formulation of a differential rule to capture continuous time behavior as follows:

$$\left\langle \frac{ds(t)}{dt}\right\rangle_k=\frac{d\langle s\rangle_k(t)}{dt}+\jmath k\varrho\langle s\rangle_k(t). \tag{5.4}$$

We will refer to this approach to dynamic phasors as the *Fourier approach*.

Another definition of dynamic phasors is derived from a communications-theoretic perspective [34, 182, 183]. Venkatasubramanian introduced *time-varying phasors* in [182], defining such a phasor via a linear operator $P()$, such that $P(s(t))=\langle s\rangle(t)$. In other words, $P()$ maps a signal $s(t)$ to a phasor. The definition of *time-varying phasors* begins with the idea that given a dynamic phasor of the form:

$$\langle s\rangle(t)=s_{pp}(t)e^{\jmath\theta(t)}, \tag{5.5}$$

there exists a unique band-pass modulated signal

$$s(t)=\mathrm{Re}\Big\{\langle s\rangle(t)e^{\jmath\varrho t}\Big\}=s_{pp}(t)\cos(\varrho t+\theta(t)), \tag{5.6}$$

[3]Refer to Section 1.4.1 [119] and its references for further details on the origins of the term “phasor."

[4]In this section, ϱ represents signal frequency, and ω_s denotes system frequency.

where s_{pp} is the signal's amplitude. The operator $P()$ is defined for signals where it is possible to find a unique phasor $\langle s\rangle(t)$ such that $P^{-1}(P(s(t)))=s(t)$. The key property of $P()$ is that if the spectral content of $s(t)$ is slower than the carrier frequency ϱ, the phasor operator maps is unique, i.e., there is only one possible signal onto which the phasor maps back. Hence, the method is akin to a modulation operation to go from a time representation to a complex phasor representation.

From (5.6), it is possible to define the differential property of $P()$ as follows:

$$\begin{aligned} \frac{ds(t)}{dt} &= \mathrm{Re}\left\{\left(\frac{d\langle s\rangle(t)}{dt}+\jmath\varrho\langle s\rangle(t)\right)e^{\jmath\varrho t}\right\} \\ P\left(\frac{ds(t)}{dt}\right) &= \frac{d\langle s\rangle(t)}{dt}+\jmath\varrho\langle s\rangle(t). \end{aligned} \tag{5.7}$$

The differential property in (5.7) is the same as the Fourier dynamic phasor in (5.4) for $k=1$. We will refer to this approach to defining a dynamic phasor as the *linear operator approach*.

The third approach to deriving dynamic phasors is rooted in concept of Shifted Frequency Analysis (SFA) developed by Martí *et al.* [107, 202], closely related to the definition of the linear operator $P()$ introduced in [182].[5] SFA begins with the assumption that a power system dynamic signal is a band-pass, real-valued and can be represented by a Fourier decomposition as follows:

$$s(t)=s_{pp,d}(t)\cos(\varrho t)-s_{pp,q}(t)\sin(\varrho t), \tag{5.8}$$

and the dynamic phasor is defined as $\langle s\rangle(t)=s_{pp,d}(t)+\jmath s_{pp,q}(t)$. It is composed of low-pass functions in quadrature $s_{pp,d}(t)=s_{pp}(t)\cos(\theta(t))$ and $s_{pp,q}(t)=s_{pp}(t)\sin(\theta(t))$. The procedure described by SFA provides a method to obtain $\langle s\rangle(t)$ through a *shifted frequency* representation of $s(t)$:

$$\langle s\rangle(t)=(s(t)+\jmath\mathrm{H}\{s(t)\})e^{-\jmath\varrho t}, \tag{5.9}$$

where $\mathrm{H}()$ is the Hilbert transform [62]. Equation (5.9) shifts signal $s(t)$ with the form (5.8) by $-\varrho$. As a result, it centers all the signal's dynamics around 0 Hz. The differential property for (5.9) can be derived from differentiating (5.9), resulting in (5.7).

$$\begin{aligned} &\frac{d\langle s\rangle(t)}{dt}=\frac{d}{dt}\Big[(s(t)+\jmath\mathrm{H}\{s(t)\})e^{-\jmath\varrho t}\Big] \\ \Longrightarrow &\left\langle\frac{ds(t)}{dt}\right\rangle=\frac{d\langle s\rangle(t)}{dt}+\jmath\varrho\langle s\rangle(t) \end{aligned} \tag{5.10}$$

In each case, the definition of "phasor" corresponds to the complex envelope of a signal with a modulating frequency $k\varrho$ (with $k=1$ only in the linear operator and SFA approaches,

[5]In fact, Venkatasubramanian mentions that the time-varying phasor approach is inspired by the Hilbert transform, which is central to SFA.

and arbitrary integer k in the Fourier approach). The three approaches reach the same operation for the differentiation of phasorial representation of a signal and its properties.

The definitions above provide several properties required to leverage dynamic phasors for time-domain simulations in average models. Note that a dynamic phasor operation is interpreted as a band-pass modulation [182, 202] where a given time-varying signal $s(t)$ is demodulated into a set of dynamic phasors $\langle s \rangle_k(t)$ with a given carrier frequency ϱ. The implication is that the signal $s(t)$ needs to possess a well-defined Fourier transform, which requires that: (1) The integral of $s(t)$ needs to be well-defined over the period $\mathcal{T}{=}[t{-}T{,}t]$, i.e., (5.3) to bounded; and (2) for a given frequency $k\varrho$, the signal $s(t)$ is narrow-band with a bandwidth $\mathcal{B}{=}[(k{-}1)\varrho{,}(k{+}1)\varrho]$ [35]. This demodulation perspective provides a definition of “slow" and “fast" dynamics of the system with respect to the modulating frequency ϱ and its harmonics as noted in [34]. In particular, the bandwidth (or speed) of the dynamic phasor is limited by the modulating signal frequency $k\varrho$ in order for the reverse transformation to be one-to-one. This discussion highlights that under certain conditions, it is possible to use an average model of the dynamics without any information loss.

Using dynamic phasors for simulation model averaging comes with potential trade-offs, since the increase in the number of states can dilute the gains derived from using larger Δt [35]. This is particularly problematic in cases where the resulting average models are still time varying. The practical implication of these limitations is that the dynamic phasor representation of a high-frequency (i.e., fast) signal could require many k^{th} harmonic components. It also implies that the representation of high-frequency dynamic behavior using a single carrier frequency may introduce error in the demodulation process [201].

Representation of three-phase signals and models

The definitions in Section 5.2 focus on “single-phase" complex signals. However, there is not a unique application of dynamic phasors to poly-phase systems, contributing to the diversity of definitions and formulations when the technique is used for time-domain simulation. For instance, SFA and the linear operator are applied phase-by-phase to a three-phase abc in [202, 182] to study unbalanced systems. In this case, each phase needs to comply with the requirements described in Section 5.2 to enable an exact representation of the underlying signals. Another implementation is used in [35], where the Fourier approach is used to develop models for each k^{th} harmonic required when modeling machines, FACTS, and HVDC-links. Similarly, the Fourier dynamic phasors were implemented in conjunction with sequence transformations in [164] to study unbalanced AC machines. The myriad implementations of dynamic phasors to three-phase signals and systems makes it difficult to point to a universally accepted set of properties and limitations.

Transmission-level power systems analysis is generally concerned with the modeling of multi-phase system quantities with a common *system frequency* ω_s and an offset θ for each phase. For simplicity in the discussion that follows, we consider a three-wire three-phase signal $\boldsymbol{s}_{abc}(t)$ (that satisfies $\boldsymbol{s}_{abc}(t)^\top\mathbf{1}{=}0$) of the form:

$$\vec{s}_{abc}(t){=}\begin{bmatrix} s_a(t)\cos(\omega_s t{+}\theta_a(t)) \\ s_b(t)\cos(\omega_s t{+}\theta_b(t)) \\ s_c(t)\cos(\omega_s t{+}\theta_c(t)) \end{bmatrix}, \tag{5.11}$$

where $s_a(t)$, $s_b(t)$, and $s_c(t)$ are real-valued wave amplitudes, $\theta_a(t)$, $\theta_b(t)$, and $\theta_c(t)$ represent the phase difference between the waves, and the signal's frequency is the *system frequency* ω_s, which is commonly assumed as a uniform "system frequency". $\boldsymbol{s}_{abc}(t)$ can be used to represent voltages, currents, electromagnetic flux or physical values like inductance.

In this chapter, we approach the study of dynamic phasors for three-phase signals using a *space vector* representation that provides a complex signal. The equivalent representation is derived as follows:

$$\boldsymbol{s}(t){=}c\begin{bmatrix} s_a(t)\left(e^{\jmath(\omega_s t+\theta_a(t))}{+}e^{-\jmath(\omega_s t+\theta_a(t))}\right) \\ s_b(t)\left(e^{\jmath(\omega_s t+\theta_b(t))}{+}e^{-\jmath(\omega_s t+\theta_b(t))}\right) \\ s_c(t)\left(e^{\jmath(\omega_s t+\theta_c(t))}{+}e^{-\jmath(\omega_s t+\theta_c(t))}\right) \end{bmatrix}^\top \begin{bmatrix} e^{\jmath 0} \\ e^{\jmath\frac{2\pi}{3}} \\ e^{\jmath\frac{-2\pi}{3}} \end{bmatrix}, \tag{5.12}$$

where the constant c is set to scale the vector's magnitude. For an arbitrary three-phase signal, (5.12) permits representation in a two-dimensional vector as follows: [199, 129]

$$\boldsymbol{s}(t){=}c\Big(\boldsymbol{s}_+(t)e^{\jmath\omega_s t}{+}\boldsymbol{s}_-(t)e^{-\jmath\omega_s t}\Big) \tag{5.13}$$

where the terms:

$$\begin{aligned}\boldsymbol{s}_+(t){=}&s_a(t)e^{\jmath\theta_a(t)}{+}s_b(t)e^{\jmath(\theta_b(t)+\frac{2\pi}{3})}\\ &{+}s_c(t)e^{\jmath(\theta_c(t)-\frac{2\pi}{3})}\end{aligned} \tag{5.14}$$

and

$$\begin{aligned}\boldsymbol{s}_-(t){=}&s_a(t)e^{-\jmath\theta_a(t)}{+}s_b(t)e^{-\jmath(\theta_b(t)-\frac{2\pi}{3})}\\ &{+}s_c(t)e^{-\jmath(\theta_c(t)+\frac{2\pi}{3})},\end{aligned} \tag{5.15}$$

are complex-valued multipliers for the positive and negative frequency components, respectively. This shows that any three-phase signal without a zero-sequence component and with frequency ω_s can be modeled without loss of information using two complex-value dynamic phasors as long as $\boldsymbol{s}_+(t)$,$\boldsymbol{s}_-(t)$ comply with the requirements described in Section 5.2. This approach is used in [164] where the authors develop a model for signals of the form (5.13) for an unbalanced machine using (5.2)–(5.4) with $k{=}1$ and $k{=}{-}1$ components.

The implication for time-domain simulation is that it is not possible to directly model an arbitrary three-phase signal (5.11) with one complex phasor quantity – put more formally, $\langle s(t)\rangle{\neq}\langle s(t)\rangle_+{+}\langle s(t)\rangle_-$. Nevertheless, if the signal is balanced – $s_a(t){=}s_b(t){=}s_c(t){=}s(t)$ and $\theta_a(t){=}\theta(t)$ $\theta_b(t){=}\theta_a(t){-}\frac{2\pi}{3}$ $\theta_c(t){=}\theta_a(t){+}\frac{2\pi}{3}$ – then the components $\boldsymbol{s}_+(t)$, $\boldsymbol{s}_-(t)$ reduce to:

$$\boldsymbol{s}_+(t){=}3s(t)e^{\jmath\theta(t)}, \quad \boldsymbol{s}_-(t){=}0, \tag{5.16}$$

which implies that $\langle s(t)\rangle = \langle s(t)\rangle_+$. The result in (5.16) showcases the effectiveness of averaging techniques in balanced systems: *the envelope of a balanced three-phase signal is completely determined by the positive-frequency phasor*. Even if a dynamic simulation requires modeling phasors for components at multiple frequencies, however, phasors can still improve solution times with respect to modeling directly in abc if the increase in Δt compensates for the increase in the number of states [35].

Reference Frame Transformations

The application of reference frame transformations in simulation models is ubiquitous since they are the basis of rotating electric machinery analysis and power electronic converter control. In simulation models, the Park transform (or dq0 transform) [136] is used to obtain a time-invariant dynamic model of the rotational mutual inductance between the rotor and the stator of the machine. Park's transform is defined as:

$$\vec{s}_{dq0}(t) = CT_p(\theta(t))\vec{s}_{abc}(t) \tag{5.17}$$

where $\theta(t)$ is the angle between the "reference axes" and the axes of rotation. The transform matrix $T_p(\theta(t))$ can be defined arbitrarily for any $\theta(t)$ which leads to different formulations depending on the leading-lagging relationships between the d and q axis. In this chapter, we follow the convention in Standard IEEE-1110 [73]. whereby the q-axis lags the d-axis and the a-phase is aligned with the d-axis, to define the transformation matrix:

$$T_p(\theta(t)) = \begin{bmatrix} \cos\theta(t) & \cos(\theta(t)-\frac{2\pi}{3}) & \cos(\theta(t)+\frac{2\pi}{3}) \\ -\sin\theta(t) & -\sin(\theta(t)-\frac{2\pi}{3}) & -\sin(\theta(t)+\frac{2\pi}{3}) \\ 1 & 1 & 1 \end{bmatrix}. \tag{5.18}$$

While manipulations involving dynamic phasors hinge on assumptions concerning the frequency content involved in signals, the transform (5.17) can be applied to any three-phase signal (in both directions). Although the reference transform is valid and reversible for any combination of $\boldsymbol{s}_{abc}(t)$ and $\theta(t)$, transforming the system is *useful* for the purposes of simulation only with careful selection of $\theta(t)$.

In simulation models, the calculation of the angle $\theta(t)$ is usually done by integrating the frequency of the signal (5.11):

$$\theta(t) = \int_{t_o}^{t} (\omega_s + \Delta\omega_s(\tau))\mathrm{d}\tau + \theta^0, \tag{5.19}$$

where, respectively, ω_s and $\Delta\omega_s(t)$ are the frequency and its time-dependent variation. In a balanced system, the effectiveness of the transformation relies on using a reference frame with a constant frequency, i.e., $\Delta\omega_s(t)=0$, such that we can obtain:

$$\vec{s}_{dq0}(t) = cT_p(\theta(t))\vec{s}_{abc}(t) \triangleq \boldsymbol{s}_+(t). \tag{5.20}$$

Under balanced conditions, Park's transform maps the three-phase signal's dynamic phasor $\boldsymbol{s}_+(t)$. The transformation reduces the simulation models by transforming the variables of the components to a rotating reference frame [11] which requires fewer variables without loss of information.

In interconnected device simulations, we can define the reference frame to reference transformation:

$$T_p(\theta_1(t))T_p(\theta_2(t))^{-1}=\begin{bmatrix}\cos\Delta\theta(t) & \sin\Delta\theta(t) & 0\\ -\sin\Delta\theta(t) & \cos\Delta\theta(t) & 0\\ 0 & 0 & 1\end{bmatrix} \tag{5.21}$$

where $\Delta\theta(t)=\theta_1(t)-\theta_2(t)=\int_{t_0}^{t}(\Delta\omega_1(\tau)-\Delta\omega_2(\tau))d\tau+\Delta\theta^0$. The above property implies that only the local reference frames' frequency deviations are required for the system simulation. Moreover, the selection of common system frequency ω_s is of no consequence to the simulation results as long as there is a local $\Delta\omega(t)$ variable at the device model.

In the case of machine models, it is possible to define the change of the local reference frame frequency in terms of the rotor speed changes, i.e., $\Delta\omega(t)=\Delta\omega_r(t)$. This property was identified in [201], where the authors recognized that the machine rotor is a *perfect demodulator of the network signals*. This implies that each synchronous machine imposes, through its rotor angular speed, the frequency at the bus. Further, the "system frequency" is not a modeled quantity, nor is it required for accurate representation of the system dynamics.

However, in contrast to electrical machine models, an IBR does not necessarily have a local reference frame. Hence, when simulating IBRs, the relationship of the device model with ω_s is dictated by the controls and their representation. For instance, many IBR control structures rely on frequency measurements at the local bus for the internal controls, introducing a requirement heretofore unproblematic in system simulations: namely, reliably modeling frequency dynamics at the bus. When modeling control schemes that require frequency measurements, the idealized frequency measurement is modeled as follows:

$$\Delta\omega_{\text{bus}}(t)=\frac{d}{dt}\tan^{-1}\frac{V_{q,\text{bus}}(t)}{V_{d,\text{bus}}(t)}. \tag{5.22}$$

The implication of having to estimate the local frequency in a simulation is that (5.22) relies on the value of the voltage states to estimate the local bus frequency. As a result, the frequency deviations' impact on the simulation becomes dependent on the network model's assumptions. Further, the frequency measurement model (5.22) suggests that simplifications in the network dynamics can mischaracterize the effects of frequency deviations on the controls and introduce further deviations from real-world behavior of the IBR.

Recent works have looked at the theoretical underpinnings of the system's modeling and the definition of system frequency in simulation models to develop novel theories about the relationships between energy and system frequency [116, 119]. This is done in order to develop a rigorous perspective that considers the inclusion of IBRs. It should be noted, however, that this is still an area under active development and there are currently no widely adopted practices to manage the selection of frequency for large-scale simulations.

Singular perturbation theory (SPT)

In the power systems literature, it is common to make approximations that reduce the number of states modeled [167]. The justifications for several reductions are based on practical knowledge; however, these approximations have also been formalized in terms of timescale separation arguments derived from SPT of system dynamics Singular Perturbation Theory (SPT) of system dynamics [26, 80].[6]

In the context of the model (5.1a)–(5.1b), SPT identifies two state categories: “slow" states ($\overline{\boldsymbol{x}}_s$, $\overline{\boldsymbol{y}}_s$) and “fast" states ($\overline{\boldsymbol{x}}_f$, $\overline{\boldsymbol{y}}_f$). Fast states are multiplied by a small positive real scalar ε that represents all the small parameters to be neglected. The dynamics are then presented as:

$$\dot{\vec{\boldsymbol{x}}}_s = F_s(\vec{\boldsymbol{x}},\vec{\boldsymbol{y}},\vec{\eta},\varepsilon), \qquad \dot{\vec{\boldsymbol{y}}}_s = G_s(\vec{\boldsymbol{x}},\vec{\boldsymbol{y}},\vec{\psi},\varepsilon) \tag{5.23}$$
$$\varepsilon\dot{\vec{\boldsymbol{x}}}_f = F_f(\vec{\boldsymbol{x}},\vec{\boldsymbol{y}},\vec{\eta},\varepsilon), \qquad \varepsilon\dot{\vec{\boldsymbol{y}}}_f = G_f(\vec{\boldsymbol{x}},\vec{\boldsymbol{y}},\vec{\psi},\varepsilon) \tag{5.24}$$

SPT involves finding the trajectories of a dynamical system as $\varepsilon\to 0$. If the conditions are met, it is possible to set $\varepsilon=0$ and reduce (5.24) to algebraic equations. The requirement to perform this reduction is that the algebraic equations (5.24) in the domain of interest have distinct roots:

$$\vec{\boldsymbol{x}}_f=R_x(\vec{\boldsymbol{x}}_s,\vec{\boldsymbol{y}}_s,\vec{\eta}), \quad \vec{\boldsymbol{y}}_f=R_y(\vec{\boldsymbol{x}}_s,\vec{\boldsymbol{y}}_s,\vec{\psi}). \tag{5.25}$$

If the roots of (5.25) are unique, it is possible to guarantee, via eigenvalue analysis, that during a transient the fast dynamics will not diverge. As described in [80], most of this analysis is done on the “boundary layer system" that imposes requirements on the eigenvalues of the Jacobian of F_f and G_f. The uniqueness of the roots enables the use of variable substitution in a way that results in a reduced model:

$$\dot{\vec{\boldsymbol{x}}}_s=F_s(\vec{\boldsymbol{x}}_s,\vec{\boldsymbol{y}}_s,R_x(\vec{\boldsymbol{x}}_s,\vec{\boldsymbol{y}}_s,\vec{\eta}),\vec{\eta}), \tag{5.26a}$$
$$\dot{\vec{\boldsymbol{y}}}_s=G_s(\vec{\boldsymbol{x}}_s,\vec{\boldsymbol{y}}_s,R_y(\vec{\boldsymbol{x}}_s,\vec{\boldsymbol{y}}_s,\vec{\psi}),\vec{\psi}). \tag{5.26b}$$

System (5.26a)–(5.26b) is the *quasi-steady-state model*. These definitions based on SPT provide a different definition of “slow" and “fast" in a simulation, this time, in terms of the “speed" of some states with respect to others. SPT provides conditions under which the fast states may rapidly converge to a root that approaches the final value of the state, i.e., $\vec{\boldsymbol{x}}_f\to R_x(\vec{\boldsymbol{x}}_s,\vec{\boldsymbol{y}}_s,\vec{\eta})$ and $\vec{\boldsymbol{y}}_f\to R_y(\vec{\boldsymbol{x}}_s,\vec{\boldsymbol{y}}_s,\vec{\psi})$.

The theoretical underpinnings of SPT justify the practice of neglecting line capacitance and the resulting voltage dynamics in short-length lines due to the small value of the

[6]The reader is referred to [80] for the foundations of SPT.

current drawn from the shunt impedance. SPT is used in [159, 158] to show that modeling current flows over the network via the admittance matrix $\boldsymbol{Y}$ is a reasonable approximation of the integral manifold. Such approximations have also been shown to be valid for small signal analysis and are typically used for studying intra- and inter-area electromechanical oscillations, as well as stabilizer parameter tuning [158].

5.3 Simulation Categories

Table 5.1 summarizes common simulation categories and associated characteristics based on the taxonomy in Fig. 5.1. The model properties are split between the larger categories of (i) Quasi-Static Phasor (QSP) and (ii) Electro-magnetic Transient (EMT) given the commonalities in modeling assumptions. The main difference between the two categories stems from how network circuits are represented. We have identified the modeling of the network circuits as the main determinant of a simulation model's numerical properties, available solution techniques, and resulting computational requirements.

Quasi-Static Phasor (QSP)

QSP models are narrow-band simulation models that focus on dynamics that do not deviate significantly from the steady-state frequency. In most software implementations, the integration method independently solves for the device and network equations using an algebraic representation of (5.1b).

Positive Sequence

Also called *Root Mean Square (RMS) balanced* simulation, this is a QSP simulation where the balanced network is modeled using a single phase, the positive sequence. These models are formulated as DAEs and use solution strategies derived from the Partitioned-Explicit solution methods, relying on the assumption that the differential portion of the DAE is not stiff. In [167], B. Stott presents a detailed account of the solution techniques applicable to this simulation. These type of simulators are primarily designed for the analysis of machine angles in balanced transmission systems. This category of simulation is also commonly called *electromechanical* or *transient stability* simulation.

RMS Unbalanced

This is a simulation used in the analysis of non-symmetrical network circuits. The network is modeled using symmetrical components while keeping the representation algebraic. In this simulation, the dynamic components feed into the positive sequence subset of equations, and the three sequence networks are simplified using circuit algebra depending on the type of fault [182]. These models are commonly used to evaluate the effects of unbalanced faults in the transient behavior of the system.

RMS Dynamic Phasors

This approach is based on constructing the model using Fourier Dynamic Phasor as defined in Section 5.2. However, the derivative terms of the network portion of the models (left-hand side) are neglected based on SPT, since for most models they are close to zero. This model produces a subset of algebraic equations for each k^{th} harmonic component that can be integrated as a system of DAEs. The accuracy of the model depends on the number of harmonics considered in the model and whether the dynamics around each harmonic possess the required properties to neglect the left-hand side derivative [35].

dq0-model with algebraic network

This type of simulation model is obtained by performing a dq0 transformation and assuming steady state. In the balanced case, this model is equivalent to *positive sequence*. When modeling an asymmetric network and/or harmonics, the method can increase the minimum Δt, but its usefulness is limited since it results in a time-varying model.

QSP simulation software tools are highly developed for system-level balanced analysis within the positive sequence category. For instance, PTI PSS®E, GE PSLF, PowerTech Lab's TSAT, and PowerWorld, among others, are tools that share common solution methods and models. The model libraries are generic and developed in coordination with ISOs standards and practices. Most commercial tools use Partitioned-Explicit solution methods and fixed time-stepping. The use of partitioned methods has many merits from the computational performance perspective. However, it also has mathematical drawbacks that can lead to numerical instabilities [117]. Partition methods used by conventional transient stability simulators are inherently tied to the quasi-stationary assumption of the network variables, which imposes limitations on the modeling detail that can be included in a simulation.

Electromagnetic Transients

EMT models are simulation models that can consider a diverse range of dynamic phenomena and are commonly implemented in studies that can significantly deviate from steady-state frequency. These are usually employed to study high frequency phenomena, such as overvoltages, harmonic propagation, sub-synchronous resonance, and transient recovery voltages, among others. Within this category there is a large variety of modeling and computational approaches that depend on the techniques used to limit the computational cost of executing the simulations.

(1) Waveform simulation: A *waveform* simulation, also known as *point-on-wave* simulation or abc simulation, is the most comprehensive implementation of EMT simulations. These simulation models represent the full wave throughout the entire simulation, which results in a time-variant model. As a result, these models require special consideration in the initialiation of simulation and the integration technique choice. The solution methods commonly employ the numerical integration substitution technique using the trapezoidal

rule for integration [36]. To capture fast electromagnetic phenomena, these simulations usually feature more detailed transmission line models than the π model. A *waveform* simulation allows the use of distributed models such as the Bergeron or frequency-dependent models. A waveform simulation can include detailed converter switching dynamics; it uses similar methods but employs piece-wise continuous models between switching instants and requires additional equations and solution methods to capture the ON/OFF switching of converter transistors.

(2) dq0-model: A dq0-model is used to model the network in a rotating reference frame commonly modeled using the *transformed* π model. In the balanced case, also known as a *fast time-varying phasor* simulation [183], the model results in a time-invariant formulation that yields the same initialization routine of the positive-sequence model, but the simulation is able to represent the network dynamics limited only by the π model assumptions. Since the network is now modeled using only ODEs (i.e., there are no longer algebraic equations in the network model), the system model becomes a series of stiff ODEs because of the multi-rate nature of the system dynamics. As a result of the increased stiffness, finding the solution to the model might require Simultaneous-Implicit solution methods [167]. Most of the benefits of this formulation are lost when used in unbalanced networks or with harmonics, since the model results in time-varying signals that have similar requirements to *waveform simulations*.

(3) Dynamic Phasors: This approach requires that the model be formulated using dynamic phasors (see Section 5.2). Dynamic phasors are useful when the frequency spectrum of power system transients are multiple narrow banded signals centered around multiple harmonics of a fundamental frequency ϱ. The choice of ϱ depends on the application; when employed to model converters, it could be the switching frequency, whereas in the case of system-level analysis, ω_s is a common choice. This approach can be implemented in several ways depending on the application, and if done correctly, the resulting model will be time-invariant but at the expense of additional states. A time-variant model enables other analytical techniques such as sensitivity studies via small-signal analysis at equilibrium points making the approach practical beyond the formulation of simulation models. As shown in [35], dynamic phasors are helpful if the computational efficiency gained by reducing the model's bandwidth (and usage of adaptive time-stepping techniques) outweighs the cost of increasing the number of states.

The most developed software solutions for EMT simulations focus on waveform dynamics with and without converter switching and include models for very detailed component-level analysis. Examples of software environments are PSCAD, EMTP®, Simulink, and PLECS, among others.

Table 5.1: Simulation category review and classification

Simulation Category	**Name**	**Multiple Frequencies / Asymmetrical / Unbalanced**	**Network Representation**	**Software Availability**
Quasi-static phasor Simulation	• Positive Sequence • RMS Balanced • Electromechanical[†] • Transient Stability[†]	No	Single Phase	Mature
	• RMS Unbalanced	No[*]	Symmetrical Components	Mature
	• $dq0$-model with algebraic network	Yes	$dq0$	Research
	• RMS Dynamic Phasor	Yes	abc or $dq0$	Research
Electromagnetic Transients (EMT) Simulation	• Waveform • Point-on-Wave • abc-model	Yes	abc	Mature
	• $dq0$-model • Fast Time-Varying Phasor	Yes	$dq0$	Research
	• Dynamic Phasor	Yes	abc or $dq0$	Research

[†]These are commonly used for balanced systems, but they can be formulated for unbalanced networks as RMS Unbalanced model.
[*]The unbalanced network is handled algebraically by connecting the symmetrical circuits into a single phase.

5.4 `PowerSimulationsDynamics.jl` - An Open Source Modeling Package for Modern Power Systems with Inverter-Based Resources

As discussed in previous sections, there are several approaches to simulating interconnected systems with IBRs that have shown promise but require additional exploration. Currently the common practice when developing new algorithms or models is to implement bespoke dynamic simulation, which have a high setup cost, which often will not support large-scale, reproducible experiments that are easily extensible to new models or methods. There are important open-source efforts that have been used successfully by the research community. Notably, the Matlab-based library `PSAT` has been a pioneer in extensible dynamic modeling for power systems[115]. The Python-based tool `ANDES` [29] offers a modeling approach where a symbolic layer describes the components and a numeric layer performs vector-based numerical computations. Dynaωo is a hybrid C++/Modelica open-source suite of a simulation tool for power systems [58].

In the Julia ecosystem, `PowerDynamics.jl` [145] also uses symbolic representations of dynamic models to provide a two-stage process of symbolic and compiled representations of dynamic network similar to the approach used in `ANDES`. However, the level of customizability and modularity of open source tools has some limitations. In most cases, implementing new models requires directly modifying the underlying simulation logic source code. Although there is support for component-level flexibility for generator models, there is a limited focus on inverters. Existing libraries are deeply integrated with solution methods and rather difficult to use when attempting to explore integration algorithms.

Julia is a scripting language like Python and Matlab, but offers the performance of low-level compiled languages [66]. Our choice of Julia is central to enabling many of the desired features of a power systems simulation software as described in [117]. The software's support for multiple dispatch and composition has allowed us to design a software and model library that is computationally efficient, easy to use, and extendable through method overloading [111]. Julia's multiple dispatch is particularly useful for mathematical modeling since methods can be defined based on abstract data structures that enable code re-use and easy interfacing with existing models [16]. `PSID.jl` evaluates different models by selecting the appropriate method version based on the signature of arguments passed into the function. This approach is different from traditional scripting languages, where dispatch is based on a special argument syntax and is sometimes implied rather than explicitly written.

`PSID.jl` exploits Julia's *multiple dispatch* to support its modeling flexibility. The approach developed in this chapter uses a data model for inverters and generators for component-level customization, and flexible ODE and algebraic representations of network circuit dynamics. The system model is also flexible and allows for different numerical integration routines via the `DifferentialEquations.jl` [148] common API. `PSID.jl` uses novel automatic differentiation techniques to calculate the Jacobians of existing and custom models without user input.

Simulation Models in `PSID.jl`

The simulation model in `PSID.jl` stems from the general simulation model (5.1a)-(5.1b) and can now be implemented in software using only real variables. `PSID.jl` solves a real-valued initial-value-problem formulated as follows:

$$\frac{d\vec{x}}{dt}=F(\vec{x},\vec{y},\vec{\eta}), \quad \vec{x}(t_0)=\vec{x}^0, \tag{5.27a}$$

$$\frac{d\vec{y}}{dt}=G(\vec{x},\vec{y},\vec{\psi}), \quad \vec{y}(t_0)=\vec{y}^0, \tag{5.27b}$$

where $\vec{x}$ and $F(\cdot)$ represent the devices (e.g., IBRs, machines, loads) states and equations with parameters $\vec{\eta}$. The circuit states and dynamics of the network are represented as the subsystem $\vec{y}$ and $G(\cdot)$ with network parameters $\vec{\psi}$. The general `PSID.jl` model is a standard current injection model; therefore, $\vec{y}$ represent the network voltages and currents. The numerical advantages of current injection models outweigh the complexities of implementing constant power loads for longer-term transient stability analysis and support the modeling of fast network dynamics [117]. The network is defined in a common Synchronous Reference Frame (SRF) rotating at frequency ω_{sys}, which as discussed in Section 5.2 can be defined as a constant value, a reference from another device or the center of inertia.

Fully power system dynamic models tend to be large-scale and stiff due to the multi-rate nature of power systems dynamics. As a result, model (5.27a)-(5.27b) needs to be reformulated to employ appropriate integration algorithms that can find a solution to the system dynamics.

Because the challenges of integrating IBRs stem from both interactions across time scales and interactions with line dynamics [106], `PSID.jl` implements a formulation suitable for the exploration of simultaneous solution methods. This design choice is geared towards the exploration of implicit solution methods that can handle unknown stiffness *a priori*, since explicit methods are not A-stable [2]. Further, the potential challenge of stiffness requires different approaches depending on the properties of the system being simulated – for example, stiffness arising from eigenvalues that exhibit large negative parts versus cases with large imaginary eigenvalues. As a result, a software package that relies on a single approach to stiffness in its solution method can limit the models and situations that can be simulated. The need to provide flexible formulations for power systems simulation models is described in detail in [118], where the author notes the flexibility and algorithmic improvements afforded by a semi-implicit formulation of the dynamic model. `PSID.jl` is able to formulate DAE models of dynamical equations in two different ways which adapt to the requirements of implicit numerical methods using error control and variable step properties.

• **Residual Model:**[7]. This model is implemented for methods that find the solution to H$(t,z_t,z_t')=0$ at each time-step t. This formulation distinguishes between a subset of differential states z_d and algebraic states z_a, where the differential states are described by at least one derivative, resulting in the following system formulation:

$$\vec{r}_{x_d}=\frac{d\vec{x}_d}{dt}-F_d(\vec{x}_d,\vec{x}_a,\vec{y}_d,\vec{y}_a,\vec{\eta}) \tag{5.28a}$$
$$\vec{r}_{x_a}=F_a(\vec{x}_d,\vec{x}_a,\vec{y}_d,\vec{y}_a,\vec{\eta}) \tag{5.28b}$$
$$\vec{r}_{y_d}=\frac{d\vec{y}_d}{dt}-G_d(\vec{x}_d,\vec{x}_a,\vec{y}_d,\vec{y}_a,\vec{\psi}) \tag{5.28c}$$
$$\vec{r}_{y_a}=G_a(\vec{x}_d,\vec{x}_a,\vec{y}_d,\vec{y}_a,\vec{\psi}) \tag{5.28d}$$

where $\vec{x}_d$ and $\vec{x}_a$ are respectively the differential and algebraic states of the device and $\vec{y}_d$ and $\vec{y}_a$ are network differential and network algebraic counterparts. Functions F_d, F_a, G_d and G_a are the subsystems of equations that define the system of non-linear equations solved at each time-step t. The terms $\vec{r}_{x_d}$, $\vec{r}_{x_a}$, $\vec{r}_{y_d}$, $\vec{r}_{y_a}$ correspond to the residuals of the non-linear system of equations.

It is important to note that models like (5.28a)-(5.28d) arise in power systems from the application of SPT to (5.27a)-(5.27b), in order to reduce overall model stiffness. By selectively zeroing out some of the differential terms and transforming the model into a system of index-1 DAE, the “slow" states maintain their differential representation while the “fast" states are simplified into the algebraic terms. However, in `PSID.jl` there is no requirement that the separation between differential and algebraic states matches slow and fast simplifications.

• **Mass Matrix Model:** This model is implemented for methods derived from the solution of mechanics problems where the differential terms z' are multiplied by constants. It represents system dynamics with the equation $Mz'=h(z)$. Although mass matrix models may have an arbitrary structure, in `PSID.jl` the focus is on models where M is a diagonal matrix[8]. The resulting model is as follows:

$$M_x\frac{d\vec{x}}{dt}=F(\vec{x},\vec{y},\vec{\eta}) \tag{5.29a}$$
$$M_y\frac{d\vec{y}}{dt}=G(\vec{x},\vec{y},\vec{\psi}) \tag{5.29b}$$

where M_x corresponds to the mass matrix for the device states and M_y corresponds to the matrix for the network states. The diagonal elements of M_x are determined by the time constants of the device models and can be 0 if the dynamics of the state are not included; this commonly occurs in large datasets where filtering dynamics are ignored. On other hand, since `PSID.jl` employs a current injection model, M_y has 0 when a line is

[7]In the differential equations literature, this model is also named *semi-explicit* system

[8]These classes of mass matrix models are also named *Lumped mass matrix* models

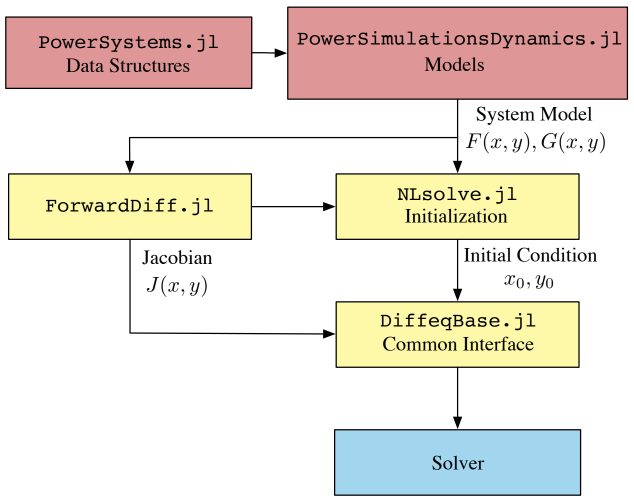


Figure 5.2: Software dependencies in `PSID.jl`.

modeled using an algebraic relationship or when the constant multiplying the node voltage derivative terms. Similar to the residual model, SPT can be used by setting the entries M diagonal to $\epsilon \to 0$ in (5.29a)-(5.29b) and analyzing the conditions under which those entries reduce model stiffness.

The flexibility in the model formulation enables the use of solvers that employ Backwards Differentiation Formula (BDF) and Backward Euler approaches as well as collocation algorithms derived from the Radau, Rosenbrock, and Rodas methods. This wide variety of solvers is enabled by `PSID.jl`'s integration with the `DifferentialEquations.jl` API [147] supports a wide variety of solution methods for both residual and mass-matrix formulations in the simulation model. Integrating the model through a generic solver API allows the `PSID.jl` framework to be used to formulate large-scale stiff power systems simulation models. This in turn supports the development of novel integration algorithms and improvements in existing methods.

Software design and structure

`PSID.jl` needs to support (1) package users who want to develop scripts to perform analysis and (2) users that want to develop new devices or branch models. The usability objectives require that `PSID.jl` handle the routines for initialization, Jacobian calculations, interfaces with `DifferentialEquations.jl` execution and result post-processing.

Figure 5.2 shows the relationships among the different packages used in `PSID.jl`. The system model is used to obtain the Jacobian using `ForwardDiff.jl` [150] and `NLSolve.jl` to find the initial conditions. Finally, the Jacobian and initial conditions are interfaced into the integrator through the common interface. The following sections describe how the flow in Fig. 5.2 maintains flexibility and scalability in the development of simulation experiments and models.

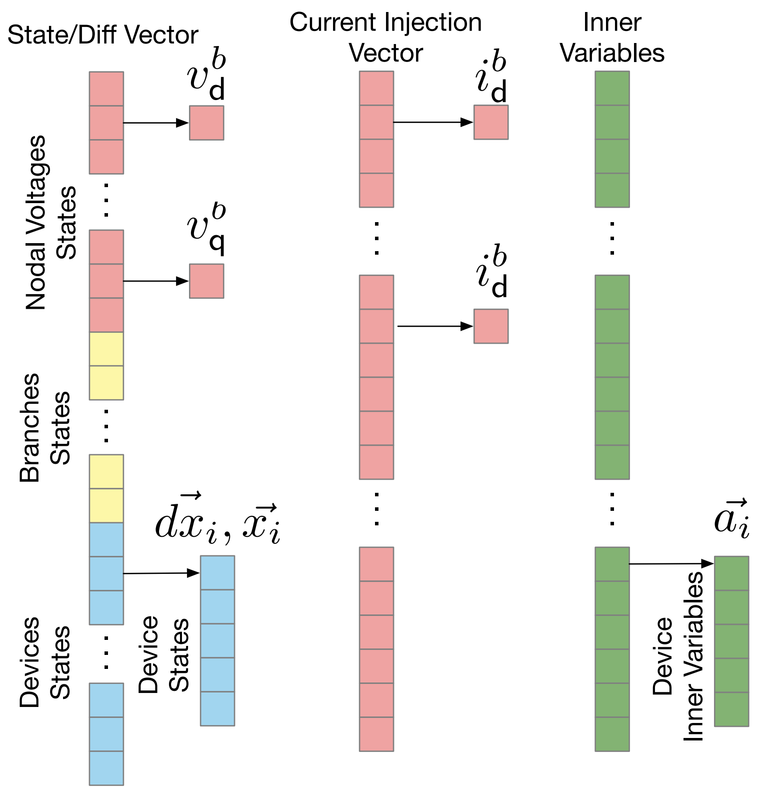


Figure 5.3: Implementation of the state space indexing.

System model implementation

Equations (5.28a)-(5.28d) and (5.29a)-(5.29b) are implemented in the software as a Julia structure with pre-allocated vectors to perform in-place updates of $\frac{d\vec{x}}{dt}$, $\frac{d\vec{y}}{dt}$, $\vec{x}$, $\vec{y}$. These vectors are cached in a system container to support in memory updates of their values and avoid unnecessary memory allocations. The additional vector $\vec{a}$ is used for intermediate inner variables when it is necessary to share information between a `DynamicInjector` model components. This design avoids the unnecessary addition of algebraic equations to the state vector, which is a critical consideration since the computational complexity of an implicit method is dominated by the linear solution method. The linear solution method computational cost is a function of the required precision and the number of variables with an upper bound $O(n^3)$ [20]. The system model also keeps track of global variables in which all models have visibility; one example of this is the system frequency ω_{sys} for reference frame transformations.

The challenge when implementing the caching design in conjunction with a modular approach for modeling is determining the distribution of the state vectors to each component model and preventing the user from implementing the state vector slice. To facilitate the interfacing with minimal computational costs, the caches get instantiated during the simulation build where we collect each device's states, its inner variables, and bus location to generate an index of the cached objects. The index is a tree structure to provide for a rapid search, while the device is on the first level and the component on the second level. Figure 5.3 demonstrates the organization of the internal cached vectors in the simulation `struct`.

Automatic Differentiation (AD) of Jacobians

Implementing Jacobian computations in simulation models has become widespread over the last two decades, with its application critical in machine-learning and optimization. Performant Jacobian computations is an important procedure in obtaining implicit solution methods for stiff DAE systems since it is used in the non-linear solve. Further, performing Jacobian computations is required when implementing adaptive timestepping techniques. Furthermore, there is a wealth of sparsity techniques that help reduce Jacobian computational costs. These techniques also present opportunities to solve challenges in simulating systems with IBRs. [56]. Computing the Jacobian function requires deriving the matrix function:

$$J(\vec{x},\vec{y},\vec{\eta},\vec{\psi})=\left[\begin{array}{cc} \frac{\partial}{\partial\vec{x}}F(\vec{x},\vec{y},\vec{\eta}) & \frac{\partial}{\partial\vec{y}}F(\vec{x},\vec{y},\vec{\eta}) \\ \frac{\partial}{\partial\vec{x}}G(\vec{x},\vec{y},\vec{\psi}) & \frac{\partial}{\partial\vec{y}}G(\vec{x},\vec{y},\vec{\psi}) \end{array}\right] \tag{5.30}$$

The flexibility afforded by `PSID.jl` makes it challenging to know *a priori* the structure of (5.30). Tools that use symbolic layers such as `ANDES` can exploit symbolic operation libraries to calculate Jacobian expressions. However, symbolic differentiation isn't always effective since the length of the expressions grows exponentially.

`PSID.jl`, employs modern AD techniques using `ForwardDiff.jl` [150] as the backend. `ForwardDiff.jl` employs a dual number approach for computing Jacobian vectors. Since the Jacobians of dynamic simulation models are square, forward-differentiation is considered the most efficient method for AD since forward AD computes a column-wise Jacobian, whereas reverse AD schemes use row-wise computations [103].

Providing users with a performant Jacobian matrix function evaluations using AD techniques that rely on a dual number implementation means that the caching design specified in Section 5.4 needs to be implemented for `Float` and `Dual` number types. Additionally, since dynamic power systems problems result in sparse Jacobian matrices, it it possible to generate and cache the Jacobian matrix and provide an in-place calculation for the Jacobian system model during the integration process. Since the Jacobian is obtained from the indexed system model described in Section 5.4 it is possible to map the entries of the AD-derived Jacobian directly to the states to perform the reductions necessary for small signal stability.

Small-Signal stability analysis

Take the residual formulation (5.28a)-(5.28d) from dynamic components $\vec{x}{:=}(\vec{x}_d^\top,\vec{y}_d^\top)$ and algebraic variables $\vec{y}{:=}(\vec{x}_a^\top,\vec{y}_a^\top)$. These are used to define $S(\cdot){:=}(F_a(\cdot)^\top,G_a(\cdot)^\top)$ as the vector of differential equations and $R(\cdot){:=}(F_d(\cdot)^\top,G_d(\cdot)^\top)$ as the vector of algebraic equations in the entire system. When residuals are set to zero, the resulting non-linear differential algebraic system can be written as follows:

$$\begin{bmatrix} \frac{d}{dt}\vec{x} \\ 0 \end{bmatrix} = \begin{bmatrix} S(\vec{x},\vec{y}) \\ R(\vec{x},\vec{y}) \end{bmatrix}. \tag{5.31}$$

We are interested in the stability that surrounds an equilibrium point $\vec{x}_{eq},\vec{y}_{eq}$ and one that satisfies $d\vec{x}/dt{=}0$, or equivalently $S(\vec{x}_{eq},\vec{y}_{eq}){=}0$, while satisfying $R(\vec{x}_{eq},\vec{y}_{eq}){=}0$. To achieve this equilibrium we use a first order approximation:

$$\begin{bmatrix} \frac{d}{dt}\Delta\vec{x} \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} S(\vec{x}_{eq},\vec{y}_{eq}) \\ R(\vec{x}_{eq},\vec{y}_{eq}) \end{bmatrix}}_{=0} + J[\vec{x}_{eq},\vec{y}_{eq}] \begin{bmatrix} \Delta\vec{x} \\ \Delta\vec{y} \end{bmatrix}. \tag{5.32}$$

Depending on the specific variables and functions modeled, the Jacobian matrix $J[\vec{x}_{eq},\vec{y}_{eq}]$ can be split into four blocks:

$$J[\vec{x}_{eq},\vec{y}_{eq}] = \begin{bmatrix} S_x & S_y \\ R_x & R_y \end{bmatrix}. \tag{5.33}$$

If R_y is not singular, we can eliminate the algebraic variables to obtain the reduced Jacobian:

$$J_{\text{red}} = S_x - S_y R_y^{-1} R_x \rightarrow \frac{d}{dt}\Delta\vec{x} = J_{\text{red}}\Delta\vec{x} \tag{5.34}$$

defines our reduced system for the differential variables and allows for the use eigenvalues to analyze local stability. These results imply that under the condition that there is a power flow solution (i.e., R_y singularity) it possible to use a QSP model to derive overall small-signal stability of the system [140].

Injection device modeling

Depending on the model requirements, device models in `PSID.jl` can be implemented in their own SRF or in the network's SRF. The modeling methods employ the data structures and type hierarchy from `PowerSystems.jl` [89] where generators and inverters are defined as a composition of components (see Fig. 5.4). This design enables the interoperability of components within a generic device container. As a result, it is possible to implement custom component models and interface them with other existing components and models

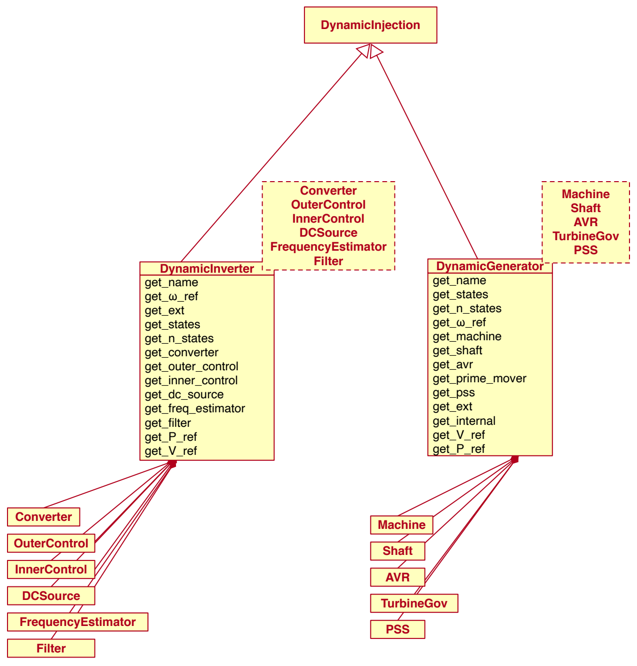


Figure 5.4: `DynamicInjection` data structures from `PowerSystems.jl`.

by simply overloading the `device!`[9] method. Whenever the function iterates over the vector containing the entire state space, only the relevant portions are passed to the `device!` function. Listing 5.1 shows a prototype of method overload required to develop a model for a `MyCustomDevice`. The method `device!()` is dispatched based on the type of the `device` field which is defined the user at the top of Listing 5.1 (L1-L4). At the device modeling layer, the sub-components from the meta-models share local information internal to each device. For instance, in an IBR model, the current reference variable i_{olc}^{ref} from the outer loop needs to shared with the inner loop control. In a generator model, the mechanical torque variable τ_m is required by the shaft model. This design further modularizes the implementation of an injection device's sub-components. Intra-device information passing is standardized via port variables that enable efficient information passing between component modeling functions. The arrows in Figs. 5.6 and 5.5 showcase the port variables and their source-destination relationships for the inverter and generator respectively.

Generator Modeling

Generator models follow common practices that are well-established in the literature [83, 117]. The machine model in `PSID.jl` benefits from the existing standardization in generator models already implemented in most software solutions based on the underlying physics of the generator. Figure 5.5 depicts component levels used in `PSID.jl` to describe

[9]In Julia's diction, functions that mutate arguments have a `!`

```julia
struct MyCustomDevice <: DynamicInjection
    field1
    field2
end

function device!(
    states::AbstractArray{T},            # x_i
    diff::AbstractArray{T},              # dx_i
    v_d::T,
    v_q::T,
    i_d::AbstractArray{T},
    i_q::AbstractArray{T},
    global_vars::AbstractArray{T},
    inner_vars::AbstractArray{T},        # a_i
    device::MyCustomDevice,
    t,
) where {T <: ACCEPTED_REAL_TYPES}
    # Update diff vector
    diff[1] = f_1(states, v_d, v_q,...)
    diff[2] = f_2(states, v_d, v_q,...)

    # Update inner vars
    inner_vars[1] = inner_var_1(states, v_d, v_q,...)
    inner_vars[2] = inner_var_2(states, v_d, v_q,...)

    # add to the bus currents
    i_d += current_d(states, v_d, v_q,...)
    i_q += current_q(states, v_d, v_q,...)
    return
end
```

Code 5.1: Injection device method prototype

synchronous generators where each generator is defined by a machine model, an excitation circuit and associated controls (Automatic Voltage Regulator (AVR) and Power System Stabilizer (PSS)), a shaft model, and a prime mover. The metamodel in Fig. 5.5 also allows the implementation of the initialization routine for generators described in [117].

Inverter Modeling

IBR simulation models are structured according to the cascaded control architectures used in the field. IBR controls employ transformations to reduce the number of control loops and produce and regulate three-phase signals (of the form (5.11)). In this setting, transformations facilitate avoiding sinusoidal signals regulations, which requires high-order controls [199]. IBRs controls commonly employ Park's Transform since it has the property

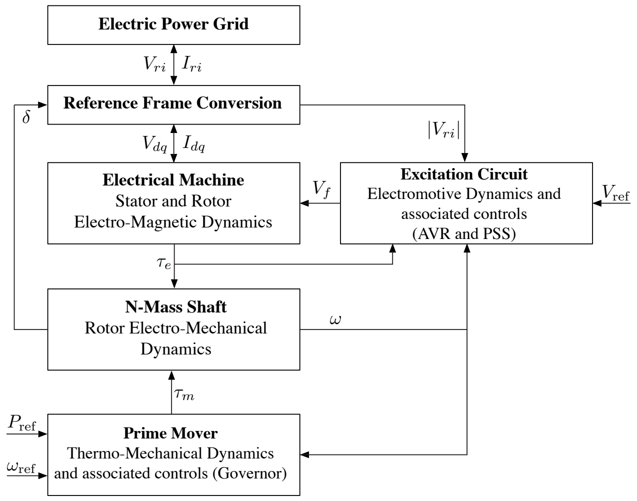


Figure 5.5: Generator metamodel.

of reducing the controls' bandwidth requirements, which in turn achieves a zero steady-state error. The model developer should not only care about model precision, but be aware of simplification effects and whether control performance is reliable [184].

Figure 5.6 depicts the relationships for a `DynamicInverter` model. The sub-component separation and variable sharing in the inverter is able to support both grid-following industrial models commonly used in QSP[10] and grid-following models. Both the sub-component separation and variable sharing function in the inverter further allow for more advanced control architectures such as Virtual Synchronous Machine (VSM) and droop controls. Each inverter is defined by a filter, converter model, inner loop control, outer loop control, frequency estimator (typically PLL), and a primary energy source model.

The proposed component decomposition deviates from the one implemented for QSP modeling, which distinguishes between three main modeling blocks: generator, electrical, and plant-level controls. However, the QSP structure cannot integrate frequency measurement modeling and additional control functions. Furthermore, the representation of the filter in this approach is limited to an algebraic representation which restricts the implementation of the model in EMT settings.

Since there is no well-established initialization method for generic inverter models, `PSID.jl` employs the sequence described in Fig. 5.7 to initialize inverter states. The initialization routine generalizes to any inverter implemented in a compatible fashion with the metamodel.

[10]The adaptation of industrial models to the proposed meta model is discussed in detail in https://github.com/NREL-SIIP/PowerSystems.jl/discussions/767

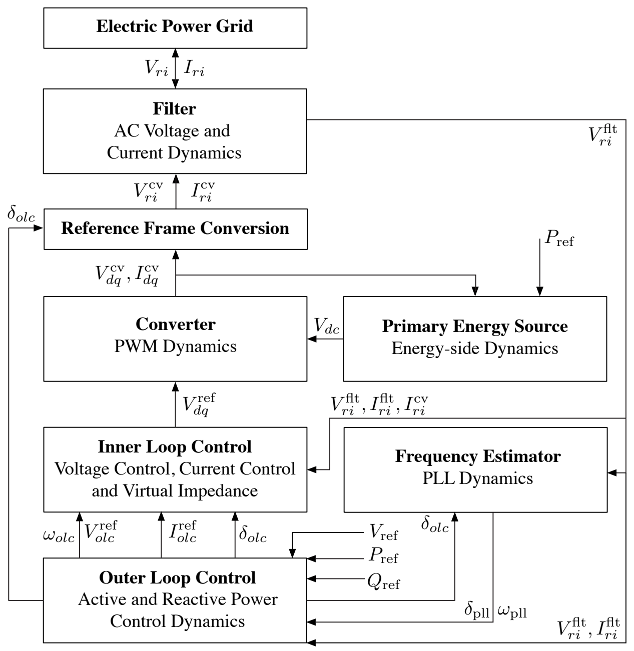

Electric Power Grid
V_{ri} I_{ri}
Filter
AC Voltage and Current Dynamics
V_{ri}^{flt}
V_{ri}^{cv} I_{ri}^{cv}
δ_{olc}
Reference Frame Conversion
P_{ref}
$V_{dq}^{\text{cv}}, I_{dq}^{\text{cv}}$
Converter
PWM Dynamics
V_{dc}
Primary Energy Source
Energy-side Dynamics
V_{dq}^{ref}
$V_{ri}^{\text{flt}}, I_{ri}^{\text{flt}}, I_{ri}^{\text{cv}}$
Inner Loop Control
Voltage Control, Current Control and Virtual Impedance
Frequency Estimator
PLL Dynamics
ω_{olc} V_{olc}^{ref} I_{olc}^{ref} δ_{olc}
V_{ref} P_{ref} Q_{ref}
δ_{olc}
Outer Loop Control
Active and Reactive Power Control Dynamics
δ_{pll} ω_{pll}
$V_{ri}^{\text{flt}}, I_{ri}^{\text{flt}}$

Figure 5.6: Generator metamodel.

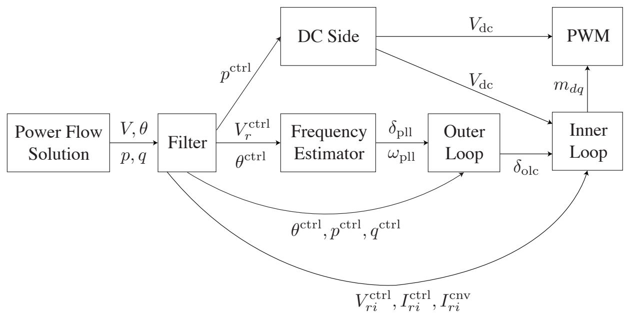


Figure 5.7: Inverter initialization model.

Load Models

PSID.jl supports two major static load models: ZIP and Exponential. Load models are implemented by aggregating the individual loads located at a bus b and using the network SRF. Each aggregated load collects all the currents from static loads by using a rectangular model. All load models follow the same methods as the injection devices by updating the total current at each bus b. For example, the total load currents in a bus with ZIP loads is implemented as follows:

$$i_{\mathsf{d}}^b = \frac{1}{|v^b||v_0^b|^2} \sum_{l \in \mathcal{L}_i^b} |v_0^b| \left(p_l v_{\mathsf{d}}^b + q_l v_{\mathsf{q}}^b\right) + \sum_{l \in \mathcal{L}_p^b} |v_0^b|^2 \left(p_l v_{\mathsf{d}}^b + q_l v_{\mathsf{q}}^b\right) + \sum_{l \in \mathcal{L}_z^b} |v^b| \left(p_l v_{\mathsf{d}}^b + q_l v_{\mathsf{q}}^b\right) \tag{5.35a}$$

$$i^b_{\mathsf{q}}=\frac{1}{|v^b||v^b_0|^2}\sum_{l\in\mathcal{L}^b_i}|v^b_0|\left(p_l v^b_{\mathsf{q}}-q_l v^b_{\mathsf{d}}\right)+\sum_{l\in\mathcal{L}^b_p}|v^b_0|^2\left(p_l v^b_{\mathsf{q}}-q_l v^b_{\mathsf{d}}\right)+\sum_{l\in\mathcal{L}^b_z}|v^b|\left(p_l v^b_{\mathsf{q}}-q_l v^b_{\mathsf{d}}\right) \tag{5.35b}$$

where $\mathcal{L}^b_p$, $\mathcal{L}^b_i$, $\mathcal{L}^b_z$ are the sets of constant power, constant current and constant impedance loads at each bus b respectively. v^b_0, p_l, q_l are the bus voltage and load's power from the power-flow used to estimate the total load currents i^b_{d} and i^b_{q} at the bus. In addition to the static load models, two models of dynamic loads are available: 5- and 3-state induction machine models.

Network Modeling

`PSID.jl` implements a lumped parameter π-circuit of a transmission line in a dq reference frame. This general representation of the network can capture the circuit network dynamics depending on a study's requirements. It further enables the modeler to selectively choose which specific branches should be modeled including circuit dynamics. The model is implemented by splitting the real and imaginary portions of the network quantities using a *rectangular* representation as follows:

$$\frac{1}{\Omega_b}\begin{bmatrix}L & 0\\ 0 & L\end{bmatrix}\frac{d}{dt}\begin{bmatrix}\vec{i}^{\ell}_{\mathsf{d}}\\ \vec{i}^{\ell}_{\mathsf{q}}\end{bmatrix}=E_\ell\begin{bmatrix}\vec{v}_{\mathsf{d}}\\ \vec{v}_{\mathsf{q}}\end{bmatrix}-\begin{bmatrix}R & -L\\ L & R\end{bmatrix}\begin{bmatrix}\vec{i}^{\ell}_{\mathsf{d}}\\ \vec{i}^{\ell}_{\mathsf{q}}\end{bmatrix} \tag{5.36}$$

$$\frac{1}{\Omega_b}\begin{bmatrix}\frac{1}{B} & 0\\ 0 & \frac{1}{B}\end{bmatrix}\frac{d}{dt}\begin{bmatrix}\vec{v}^{b}_{\mathsf{d}}\\ \vec{v}^{b}_{\mathsf{q}}\end{bmatrix}=\begin{bmatrix}\vec{i}^{b}_{\mathsf{d}}\\ \vec{i}^{b}_{\mathsf{q}}\end{bmatrix}-\begin{bmatrix}\Re(Y_a) & \Im(Y_a)\\ -\Im(Y_a) & \Re(Y_a)\end{bmatrix}\begin{bmatrix}\vec{v}^{b}_{\mathsf{d}}\\ \vec{v}^{b}_{\mathsf{q}}\end{bmatrix} \tag{5.37}$$

where (5.36) estimates the currents of the branches modeled dynamically, $\vec{i}^b_{\mathsf{d}}$ and $\vec{i}^b_{\mathsf{q}}$ are the total real (d-axis) and imaginary current (q-axis) injections from devices and dynamic branches at the bus b. E_ℓ is the rectangular incidence matrix that returns the difference between the sending and receiving buses, and R and L are the series resistance and inductance matrices in each dynamic branch.

Equation (5.37) is used to represent Kirchoff Voltage Laws across the network. The admittance matrix Y_a corresponds to a modified admittance matrix without the series elements of the branches modeled dynamically. The blocks $1/B$ on the left-hand side correspond to the total lump capacitance at the nodes. The classical network model in QSP models employ SPT to eliminate differential terms, since these terms ℓ/Ω_b,$c/\Omega_b\approx10^{-3}$ have small values in most systems, operating at 50 or 60 Hz. If the model does not account for the lines' dynamics, the left-hand-side of (5.37) becomes zero, yielding the equivalent QSP network representation $0=V-YI$.

Since the square matrices on the left and right hand side are sparse and have identical patterns in each block, `PSID.jl` implements a sparse matrix vector multiplication using `nzrange` to reduce computational effort. Note that effectively, the matrices on the left hand side of the equations are M_y in (5.29b).

Perturbations

Solver-agnostic perturbations are a major challenge in the development of simulation software given that the re-initialization procedure can differ depending on the solver and whether the perturbations introduce a change in the algebraic versus the dynamic portions of the model. `PSID.jl` simplifies the implementation of perturbations (e.g., line faults and step changes) via enabling callbacks to the integrator.

Callbacks enable controlling the solution flow and intermediate initializations without requiring simulation re-starts and other heuristics typically used in power systems dynamic modeling. `PSID.jl` includes built-in features to manipulate perturbation objects that internally define solver callbacks. Modelers can extend the perturbation library by defining custom callback structures. This can allow for further flexibility when modeling.

5.5 Simulation Validation and Case Studies

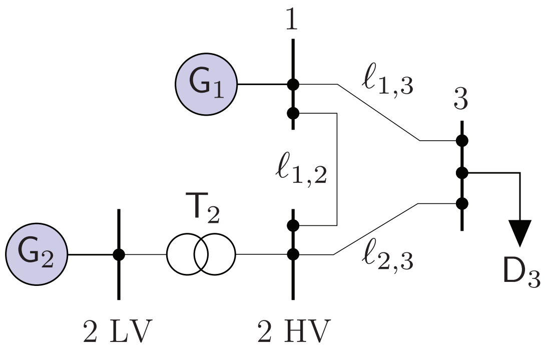


Figure 5.8: Three-bus high-voltage (HV) network system for case studies.

This section presents comparisons of simulations using `PSID.jl`, the QSP simulator PSS®E, and the waveform simulation EMT platform PSCAD. We also show verification results for a large QSP case with high shares of IBRs generation and small-signal capabilities using the results published in [29] [11]. Listing 5.2 shows the `PSID.jl` code used to run the simulations formulated as ResidualModel with the Sundials solver `IDA()`.

Four-bus network case

For validation purposes, the 3-bus HV network system (4-bus) depicted in Figure 5.8 will be used to analyze the dynamic response of `PSID.jl` against other simulation tools. The definitions of G_1 and G_2 will change according to the study under consideration.

[11]All simulation code and results are available at https://github.com/NREL-SIIP/PSIDValidation

```julia
using PowerSystems
using PowerSimulationsDynamics
using Sundials

sys = System("system_data.json")

sim = Simulation(
    ResidualModel, # model type
    sys, # system
    pwd(), # directory
    (0.0, 20.0), # time span
    BranchTrip(1.0, Line, "BUS 1-BUS 2 HV-i_1"), # fault
)

execute!(sim, IDA(), abstol = 1e-10)
results = read_results(sim)
```

Code 5.2: Example of setting up a simulation

Quasi-Static Phasor (QSP) simulation

Table 5.2: Models used for 4-bus QSP validation

Generator	Machine	Excitation	Governor	PSS
1 - G1	GENROU	None	None	
2 - EG/SG/RG	GENROU	SEXS	GAST	
2 - ND	GENROU	SEXS	TGOV1	
2 - WG	GENROU	SEXS	GAST	IEEEST
2 - SH	GENROU	SEXS	HYGOV	
Inverter	**Outer**	**Inner**	**Converter**	**Filter Freq. Est.**
2 - S/SW	REPCA1	REECB1	REGCA1	None

In this case, G_1 represents a large grid connection using a GENROU machine model without additional controllers and G_2 represents a collection of eight generation sources, including industrial grid-following IBRs. The models used are summarized in Table 5.2. This simulation is performed using a residual formulation with the solver `IDA` and abstol$=1\times10^{-10}$.

To perform the validation, we run two perturbations: (1) a generation trip of inverter Bus2-S at t=1s to assess the active power balancing and generator speed modeling; and (2) a trip of the line between buses 1 and 3 to assess voltage control and generator electromechanical modeling. Figure 5.9 presents the rotor speed for the generation units and Figures 5.11 and 5.10 depicts the bus voltage and active power output of the generation

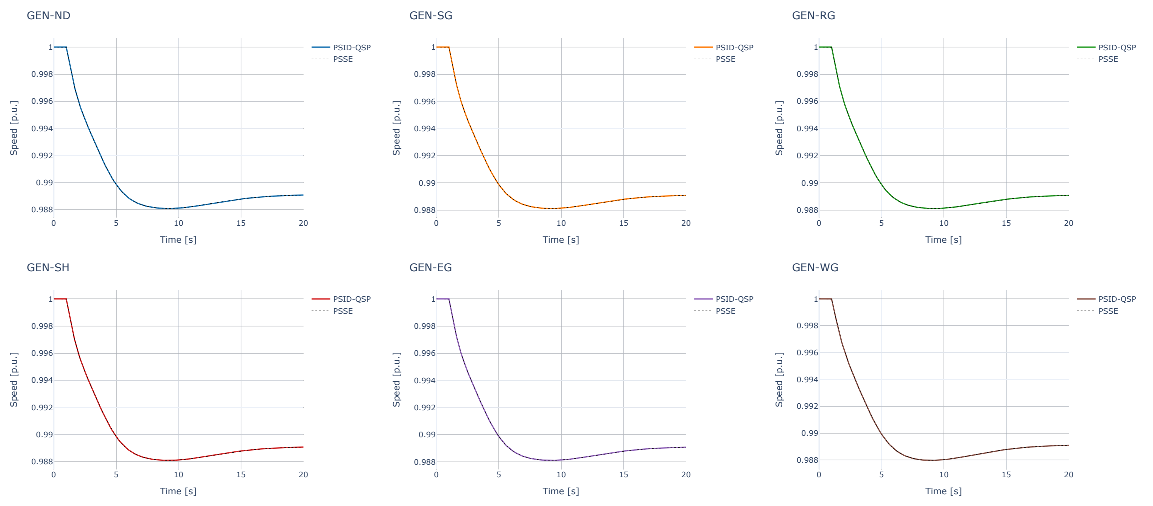


Figure 5.9: Rotor speed comparison with PSS®E.

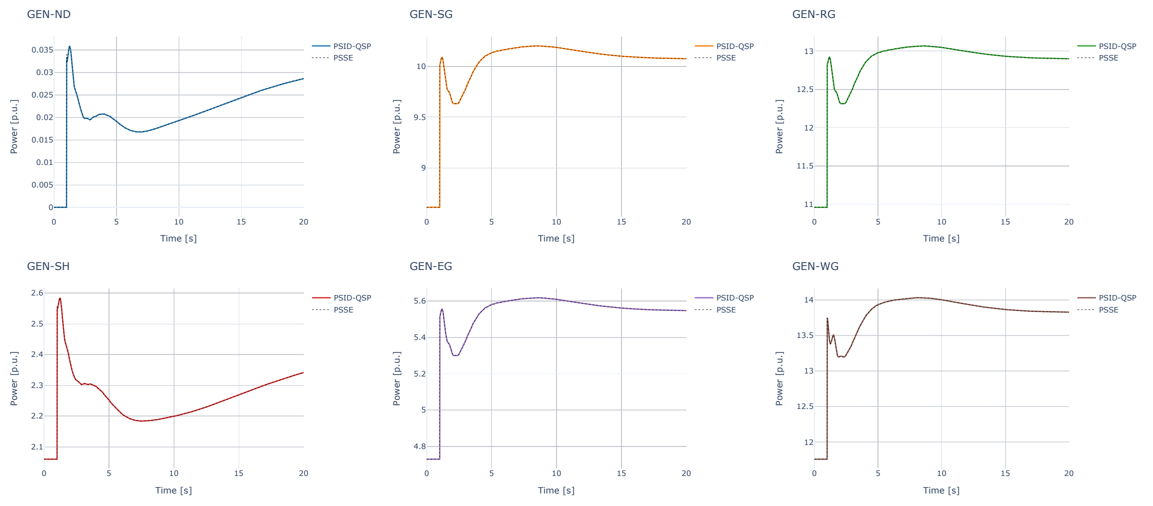


Figure 5.10: Power Output comparison with PSS®E.

units, which are able to recover the lost power and output to prevent a frequency collapse. Figures 5.13 and 5.12 showcases the voltage and field current of the generators after the line trip which follow precisely the trace of the benchmark solution. Note that the field current is not a state in the GENROU model but rather a variable derived from several states, highlighting the accuracy of `PSID.jl`.

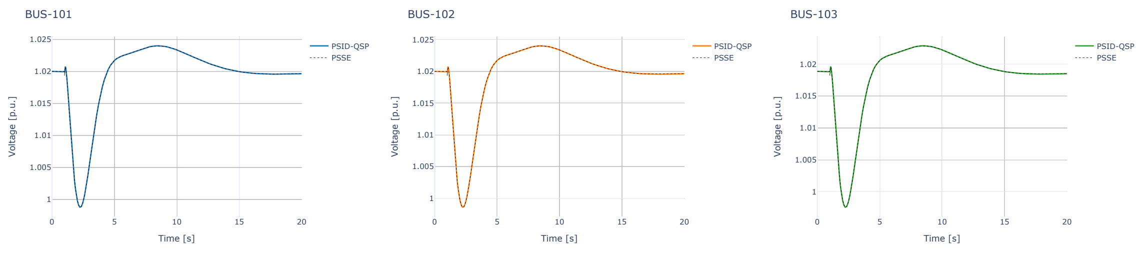


Figure 5.11: Bus voltage comparison with PSS®E with generator trip.

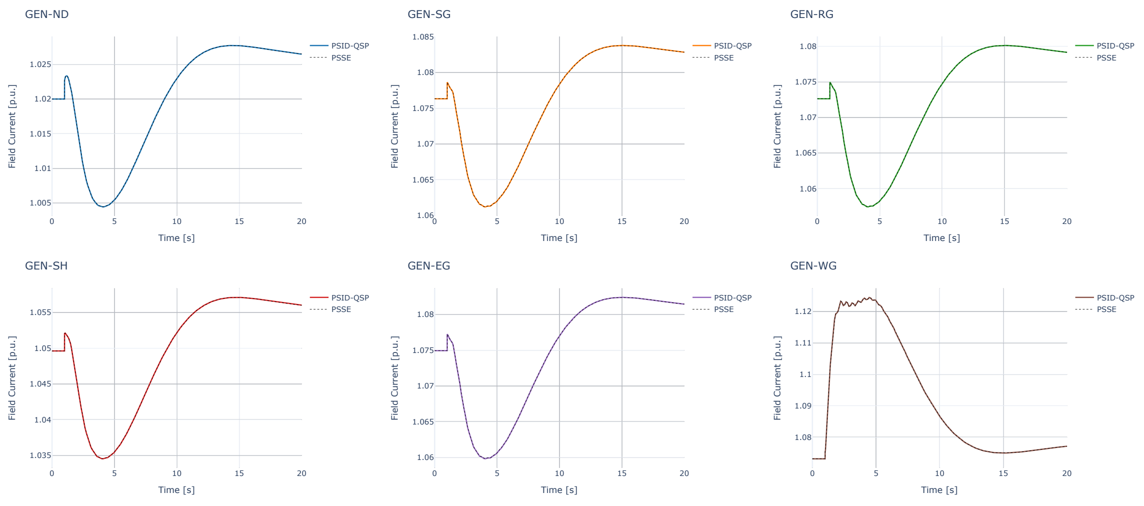


Figure 5.12: Rotor speed comparison with PSS®E.

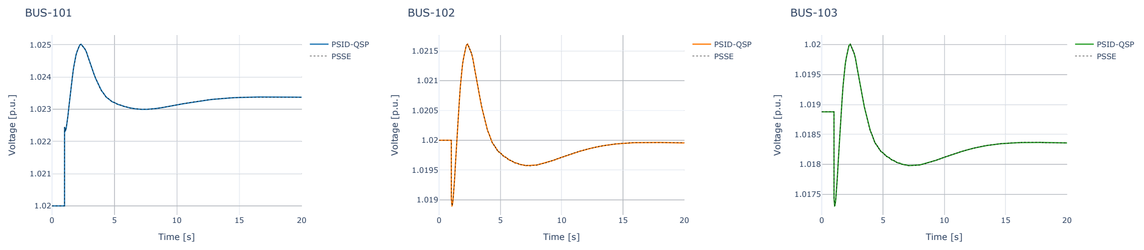


Figure 5.13: Bus voltage comparison with PSS®E with generator trip.

Balanced EMT simulation

To validate the dq -EMT simulation capabilities, we conducted a waveform simulation of the system in Fig. 5.8 using a π-line and substituting G_1 with an inverter featuring VSM control [30] and G_2 with a droop control [105]. The control blocks of the inverters are implemented as custom PSCAD components to match models available in `PSID.jl`. In the PSCAD model, the output of the inner loop control ($V_{\text{dq}}^{\text{ref}}$) and the outer loop control angle (δ_{olc}) define an ideal three-phase voltage source at the converter side of the filter.

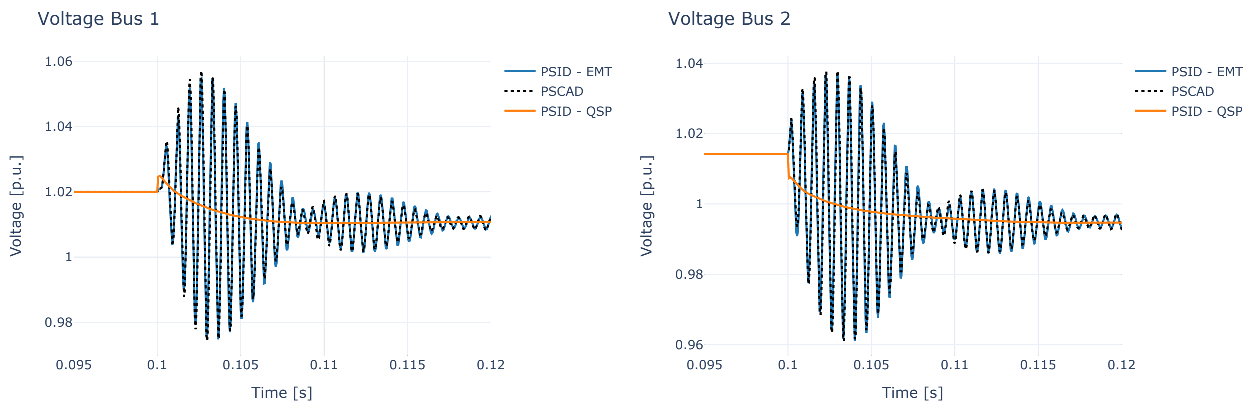


Figure 5.14: Voltage Response PSCAD vs PSID - EMT .

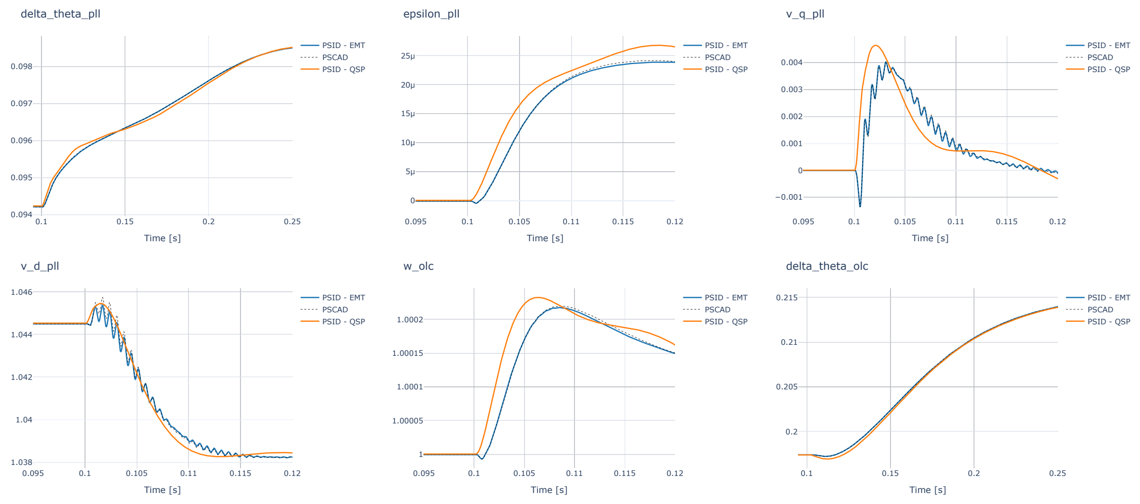


Figure 5.15: System States Response PSCAD vs PSID - EMT .

In this system we conducted a line trip between buses 1 and 2 using a mass-matrix model with abstol=10^{-14} in `PSID.jl`, and setting a 5μs time-step in PSCAD. Additionally, we conducted a simulation specifying the lines to be modeled algebraically, as usually done in QSP simulations. Figure 5.14 showcases the close match between `PSID.jl` and PSCAD for the bus voltages and the capability to capture high frequency dynamics which the algebraic lines cannot capture. Figure 5.15 shows a comparison of the internal states of the PLL in the VSM inverter. These results display the effects of the line models in the internal PLL states which in this case show that the additional dynamics from the lines do not affect the PLL angle estimation $\delta\theta_{olc}$ but the EMT does reflect an oscillatory dynamic in $v_{\mathrm{q},pll}$ that the algebraic model cannot.

240-bus WECC QSP case

To assess its modeling capabilities on a large scale, we tested `PSID.jl` on the 240-bus WECC case for the study of large-scale integration grid following IBR [200]. The system is comprised of devices described in Table 5.3 and 329 transmission lines. The resulting dynamic model has a total of 2420 dynamic states and 506 algebraic states.

Table 5.3: Models used in WECC 240-Bus QSP phasor validation

Count	Machine	Excitation	Governor	PSS	
6	GENROU	SEXS	GAST	IEEEST	
41	GENROU	SEXS	GAST		
3	GENROU	SEXS	HYGOV	IEEEST	
22	GENROU	SEXS	HYGOV		
1	GENROU	SEXS	TGOV	IEEEST	
36	GENROU	SEXS	TGOV		
Inverters	**Outer**	**Inner**	**Converter**	**Filter**	**Freq. Est.**
121	REPCA1	REECB1	REGCA1	None	None

We executed 329 line trips and 195 generator trips in the system using both PSS®E and `PSID.jl`. For each contingency, we calculated and compared the Root Mean Square Difference (RMSD) defined as

$$\text{RMSD} = \frac{||\vec{s}_{psid} - \vec{s}_{ref}||_2}{N} \tag{5.38}$$

where $\vec{s}_{psid}$ is the signal trace result from `PSID.jl` and $\vec{s}_{ref}$ is the signal trace from the reference commercial software. For every bus voltage, angle, and generator speed, resulting in a total of 595 trace comparisons. Overall, we calculated the RMSD of 311,780 individual traces to verify the validity of `PSID.jl` simulation results with respect to a commercial tool for QSP modeling. We employed the `Rodas5P()` solver with adaptive time-stepping and 1×10^{-9} absolute tolerance and PSS®E using Δt=0.005. Table 5.4 shows the results of this exercise; the average difference is calculated as the average of the max difference over the set of perturbations.

Angle values are the largest difference between simulation software, which is not unexpected since the network model and solution techniques can be significantly different. After some analysis, we observed that the partitioned method can result in angle values rotated 360 degrees after the perturbation, which can result in large differences in the trace but do not change the fundamental conclusions on system stability.

Table 5.4: Signal difference analysis of large simulation

	Generator Trip	Line Trip
Max Angle RMSD [deg]	0.41	0.45
Max Voltage RMSD [pu]	2.5929×10^{-5}	2.6177×10^{-5}
Max Speed RMSD [pu]	6.5672×10^{-6}	1.2211×10^{-5}
Average Angle RMSD [deg]	3.1719×10^{-2}	1.569×10^{-4}
Average Voltage RMSD [pu]	1.0818×10^{-5}	7.8307×10^{-7}
Average Speed RMSD [pu]	1.5992×10^{-6}	5.2253×10^{-7}

Small Signal analysis

The `PSID.jl` AD routine for Jacobian evaluation provides the capability to evaluate function (5.30) at a desired operating point to analyze small signal stability in a system. We employ a known small signal unstable system (i.e., the 11-bus Kundur system; see Table 5.5) to verify eigenvalue calculations in `PSID.jl`. We compare the eigenvalues and the damping ζ with outputs from the Python package `ANDES` [29], a package which functions similarly to `PSID.jl`. ANDES eigenvalue routine analyses has already been validated against the commercial tool DSATools SSAT. The RMSD of the eigenvalues between `ANDES` and `PSID.jl` is $\frac{1}{N}||\lambda_{\text{PSID}}-\lambda_{\text{ANDES}}||_2\approx0.2872$. The results confirm that these small signal analyses closely match other QSP models with packages like `ANDES`. The small differences observed are explained by power-torque approximations performed in shafts and turbine governor models.

Table 5.5: Eigenvalue result comparison for 11-bus system

	`PSID.jl`		ANDES	
	Eigenvalue	ζ[%]	Eigenvalue	ζ[%]
#1	$-37.2633+0j$	100	$-37.2633+0j$	100
#2	$-37.1698+0j$	100	$-37.1699+0j$	100
#3	$-36.2028+0j$	100	$-36.2031+0j$	100
⋮	⋮	⋮	⋮	⋮
#39	$0.5381-2.7415j$	–19.26	$0.5537-2.7459j$	–19.77
#40	$0.5581+2.7415j$	–19.26	$0.5537+2.7459j$	–19.77

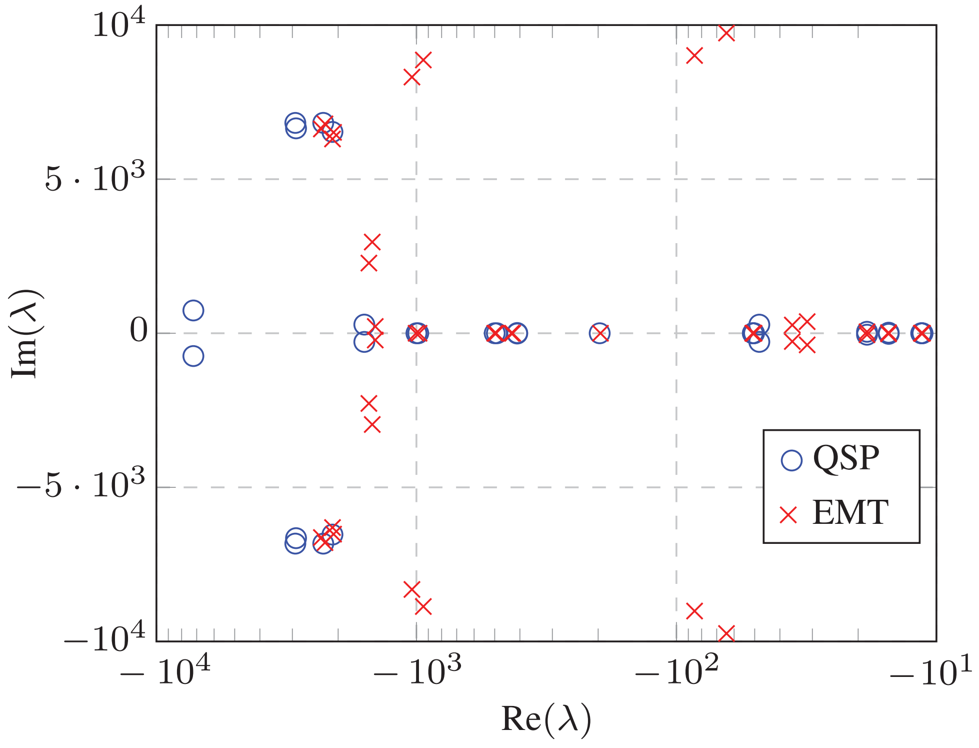


Figure 5.16: Eigenvalue comparison for balanced EMT simulation.

The network modeling flexibility between QSP and EMT also extends to small signal analyses. This in turn allows users to compare the eigenvalues from one model to another. Figure 5.16 presents a comparison of the eigenvalues for both formulations in `PSID.jl` for the system (see Section 5.5). The EMT model has an additional 12 states for the network, with a large imaginary component that explains the fast oscillations observed in Figure 5.14. Results are consistent with research that have shown that the quasi-static assumption in QSP models tends to overestimate the system's damping [66, 106]

5.6 Conclusions

- This chapter made the case that distinctions between QSP and EMT models are insufficient in characterizing the fundamentals of specific simulation models and techniques. Instead, there is a large diversity of transformations and simplifications that yield a range of simulations models, each of which differently balances computational complexities with model fidelity – provided certain operational conditions are met, such as narrow-band phenomena in dynamic phasors. These transformations enable cheaper EMT simulations without losing information. Figure 5.1 and Table 5.1 summarize the proposed classification taxonomy for time-domain simulations.

- This chapter showcases that the applicability of certain simplification techniques, like SPT, depends heavily on the control parameters of IBRs and the time constants in remainder of the system that can't be simplified. As a result, modeling requirements need to be determined from detailed models before any simplifications are performed. These requirements in turn affects the generalizability of specific device models and assumptions in different systems under different disturbances.

- Based on the findings of the simulation classifications, this chapter introduced an open-source simulation toolbox `PSID.jl`, which focuses on providing flexibility in the modeling of system using both QSP and EMT simulations focused on the requirements of systems with IBRs. `PSID.jl` is built on Julia scripting language and incorporates a myriad of power system devices and component models while enabling the analysis of transient responses of dynamical systems that vary in model complexity. The development of device metamodels for generators and inverters standardize ports and states interaction among devices and their internal components. The proposed software design enables researchers to define a new component's model within the proposed metamodel and quickly explore novel architectures and controls.

- Validation results from `PSID.jl` reveal several insights: (1) the toolbox's capacity to model IBR derived from industrial grids; (2) reveal an equivalence between dq modeling and EMT under balanced system conditions; (3) and enable solutions for stiff systems when modeling line dynamics.

- The network circuit model using dq enables the formulation of QSP and balanced EMT models for the study of system stability including fast dynamics and the exploration of stiff model solution integration algorithms in large models.

Chapter 6

Conclusions and future work

The technological changes required to transition the electricity supply from a fossil fueled power system to one that is powered by VRE requires significant improvements in the methods and models used to simulate the grid. The increasing reliance on computer simulations for the study and development of novel technologies requires the adoption of better practices of scientific computing in the power systems field. Chapter 2 highlights the importance of scientific computing principles to tackle these issues by enabling *reproducibility* and *validity* of such simulation experiments in the context of power systems research and proposes a set of practices to improve reproducibility in the power systems field. This dissertation's most relevant result consists of developing an ecosystem for simulating power systems, with two main objectives in mind. The first, to provide users with the capabilities to study a system with large shares of VRE resources, and second, to adhere to best practices where scientific computing are concerned.

This dissertation has described how to implement software in three packages: `PowerSystems.jl`, `PowerSimulations.jl` and `PowerSimulationsDynamics.jl`. These packages are informed by the principles of scientific computing and tackle the computational challenges that come with energy transitions in modeling uncertainty in VREs and handling fast dynamic interactions in systems with large proportions of IBR generation.

`PowerSystems.jl` resolves common data issues in the development of datasets faced by researchers and model developers. Having a consistent data model along with the utilities required to handle simulation data across multiple scales prevents unnecessary data conversions and inconsistencies. Further, `PowerSystems.jl` is the first model-agnostic data management tool developed for power systems computations. As described in Chapter 2, the most commonly used data models in power systems evolved from the power flow problem and thus embedded the specifics of the power flow problem into existing models.

This thesis notes the effectiveness of `PowerSystems.jl`, and its importance as the base data model for implementing simulations that include multi-time scale operations simulations, probabilistic forecasts, and QSP and EMT dynamic simulations.

The variability and uncertainty of renewable energy resources brings operational challenges to the electric system. As these resources become more widely adopted, it is important to examine the conditions that arise when modeling and studying system operations. Although studying the a system's operation is common practice, the definitions and practices of this class of simulations are poorly defined. The lack of definition for operational simulations is due in part to the computational tools used for these studies which fall under two categories: (1) commercial tools (2) *ad-hoc* software projects. In response to this definitional gap, this dissertation introduced `PowerSimulations.jl` a first of its kind simulation tool for *operations simulations* that implements the conceptual framework of decision-emulator model introduced in Chapter 1. `PowerSimulations.jl` software implementation tackles the challenge of model-limited-choice with the introduction of flexible model formulations. It further solves graphs by providing analysts and researchers with a framework to formulate computational experiments with minimal limitations. Further, `PowerSimulations.jl` modeling capabilities were verified against a state of the art PCM simulator. Two study cases depicted the modeling capabilities of `PowerSimulations.jl` in terms of advanced modeling, one that focused on AC network mnodeling and a second one with focus on advanced thermal generation modeling.

A `PowerSimulations.jl` framework was used to implement a novel simulation model for an AGC, and proved that the model can capture key challenges when attempting to integrate VREs on a large scale. Results show that the model can simulate a deployment of supplemental reserves when participation factors are not well-calibrated due to forecast errors. The model can also estimate frequency deviations according to the AGC PID parameters.

System operators still face significant challenges in managing the inherent uncertainty of VREs which lead to the development of advanced operational schemes. On the most promising technologies is probabilistic forecasting, particularly as a tool for risk-based decision making. In Chapter 4 we introduced a modeling framework to incorporate uncertainty forecasts for short-term risk assessments in ISO operations employing a *Markovian graph*. The framework was shown to be effective in an online risk assessment scenario and developed based on the `PowerSystems.jl` and `PowerSimulations.jl` novel capabilities. The results show that with the inclusion of VRE and inter-temporal dependencies help capture the inherent time-dependencies in probabilistic forecasts. Chapter 4 further proposed an an online transition matrix estimation method to capture short-term variability.

In a much smaller time-scale, a very different challenge arises with the fundamental shift in the energy conversion mechanisms used to generate renewable energy with IBRs. Abnormal or unexpected interactions between IBR controls have been identified in multiple locations typically after the facility is operational and a grid event occurs, which demonstrates that the simulation practices used for interconnection studies have flaws that are well understood by the community. Chapter 5 takes a look at the common simulation practices for the study of systems stability, particularly with respect to the embedded assumptions in QSP and EMT simulations. The analysis demonstrates that there are under-explored modeling techniques that can tackle existing simulation challenges.

Findings from the simulation techniques analyses explored in this thesis motivate the development of `PowerSimulationDynamics.jl`, a modeling library that implements the concepts of scientific computing from Chapter 1 to the study of power systems dynamics. `PowerSimulationDynamics.jl` is designed to support flexible simulation model formulation and is similar in its approach with `PowerSimulations.jl`, though with a larger focus exploring novel numerical methods. Results from verifying and validating `PowerSimulationDynamics.jl` demonstrate its ability to reproduce results from commercial QSP and EMT tools on a large scale. The added advantage of `PowerSimulationDynamics.jl` is the additional extensibility it offers at the modeling and algorithmic layer. The simulation results from `PowerSimulationDynamics.jl` reveal that under balanced conditions `dq` models are equivalent to wave-form simulations.

Although this dissertation makes advances in the practice of simulating power systems, there are still various applications left unexplored. This work focuses on what could be described as *analytical* simulations, which primarily study systems with specific operating conditions in mind. Nevertheless, there are other applications that require new simulation techniques, such as resilience and reliability analysis. These applications require extensions to the approaches presented in this work and add another layer of complexity to the existing challenges of simulating systems with large shares of VRE. Another area of focus for future work is the development of black-start simulations for systems powered with large shares of VRE, a critical capability for systems that face threats from changing climates and cyberattacks.

Bibliography

[1] Philippe Artzner et al. “Coherent Measures of Risk”. In: *Mathematical Finance* 9.3 (1999), pp. 203–228 (cit. on p. 87).

[2] Uri M Ascher and Linda R Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. Vol. 61. Siam, 1998 (cit. on p. 126).

[3] Karl Johan Åström and Tore Hägglund. *PID controllers: theory, design, and tuning*. Instrument Society of America Research Triangle Park, 1995 (cit. on p. 67).

[4] Yu Ba and Wei-Dong Li. “A simulation scheme for AGC relevant studies”. In: *IEEE Transactions on Power Systems* 28.4 (2013), pp. 3621–3628 (cit. on p. 63).

[5] David H Bailey, Jonathan M Borwein, and Victoria Stodden. “Facilitating reproducibility in scientific computing: Principles and practice”. In: *Reproducibility: Principles, Problems, Practices, John Wiley and Sons, New York* (2016) (cit. on p. 7).

[6] Lorena A Barba. “Terminologies for reproducible research”. In: *arXiv preprint arXiv:1802.03311* (2018) (cit. on pp. 7, 9).

[7] C. Barrows et al. “The IEEE Reliability Test System: A Proposed 2019 Update”. In: *IEEE Transactions on Power Systems* (2019), pp. 1–1. DOI: 10.1109/TPWRS.2019.2925557 (cit. on pp. 15, 24, 59, 61).

[8] Clayton P. Barrows and Ilya Chernyakhovskiy. *PSI-Cambodia: Analysis Release:1.0.0*. Version 1.0.0. Aug. 2020. DOI: 10.5281/zenodo.4009566. URL: https://doi.org/10.5281/zenodo.4009566 (cit. on p. 40).

[9] Clayton Barrows et al. “The IEEE reliability test system: A proposed 2019 Update”. In: *IEEE Transactions on Power Systems* 35.1 (2019), pp. 119–127 (cit. on pp. 28, 35, 71).

[10] Thomas Bartz-Beielstein and Mike Preuss. “The future of experimental research”. In: *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, 2010, pp. 17–49 (cit. on p. 7).

[11] Juri Belikov and Yoash Levron. “Comparison of time-varying phasor and dq0 dynamic models for large transmission networks”. In: *International Journal of Electrical Power & Energy Systems* 93 (2017), pp. 65–74 (cit. on p. 119).

[12] Zied Ben Bouallègue, Pierre Pinson, and Petra Friederichs. "Quantile forecast discrimination ability and value". In: *Quarterly Journal of the Royal Meteorological Society* 141.693 (2015), pp. 3415–3424 (cit. on p. 82).

[13] Fabien CY Benureau and Nicolas P Rougier. "Re-run, repeat, reproduce, reuse, replicate: transforming code into scientific contributions". In: *Frontiers in neuroinformatics* 11 (2018), p. 69 (cit. on pp. 10, 17).

[14] Ricardo J Bessa et al. "Towards improved understanding of the applicability of uncertainty forecasts in the electric power industry". In: *Energies* 10.9 (2017), p. 1402 (cit. on pp. 77, 82, 85).

[15] Hassan Bevrani. *Robust power system frequency control*. Springer, 2009 (cit. on pp. 64, 65, 67).

[16] Jeff Bezanson et al. "Julia: A Fresh Approach to Numerical Computing". In: *SIAM Review* 59.1 (2017), pp. 65–98 (cit. on pp. 11, 12, 29, 40, 52, 94, 125).

[17] A. B. Birchfield, T. Xu, and T. J. Overbye. "Power Flow Convergence and Reactive Power Planning in the Creation of Large Synthetic Grids". In: *IEEE Transactions on Power Systems* 33.6 (Nov. 2018), pp. 6667–6674. DOI: 10.1109/TPWRS.2018.2813525 (cit. on p. 14).

[18] Adam B Birchfield et al. "Grid structural characteristics as validation criteria for synthetic networks". In: *IEEE Transactions on power systems* 32.4 (2016), pp. 3258–3265 (cit. on p. 90).

[19] Bill Blevins. *2016 Methodology for Determining Minimum Ancillary Service Requirements*. Tech. rep. ERCOT, 2016 (cit. on pp. 81, 93).

[20] A Bojańczyk. "Complexity of solving linear systems in different models of computation". In: *SIAM Journal on Numerical Analysis* 21.3 (1984), pp. 591–603 (cit. on p. 129).

[21] T. Brown, J. Hörsch, and D. Schlachtberger. "PyPSA: Python for Power System Analysis". In: *Journal of Open Research Software* 6.4 (1 2018). DOI: 10.5334/jors.188. eprint: 1707.09913. URL: https://doi.org/10.5334/jors.188 (cit. on p. 43).

[22] Tom Brown, Jonas Hörsch, and David Schlachtberger. "PyPSA: Python for power system analysis". In: *arXiv preprint arXiv:1707.09913* (2017) (cit. on p. 27).

[23] Jonathan B Buckheit and David L Donoho. "Wavelab and reproducible research". In: *Wavelets and statistics*. Springer, 1995, pp. 55–81 (cit. on p. 9).

[24] B. Bush. "Topology-Based Machine-Learning for Modeling Power-System Responses to Contingencies". In: *American Statistical Association 2020 Joint Statistical Meetings* (2020) (cit. on p. 40).

[25] B. Bush et al. “Topological machine learning methods for power system responses to contingencies”. In: *The Thirty-Third Annual Conference on Innovative Applications of Artificial Intelligence, Association for the Advancement of Artificial Intelligence* (2020) (cit. on p. 40).

[26] Joe H Chow et al. *Time-scale modeling of dynamic networks with applications to power systems*. Vol. 46. Springer, 1982 (cit. on p. 120).

[27] Carleton Coffrin et al. “PowerModels. J1: An Open-Source Framework for Exploring Power Flow Formulations”. In: *2018 Power Systems Computation Conference (PSCC)*. IEEE. 2018, pp. 1–8 (cit. on pp. 11, 28, 43, 52).

[28] National Research Council et al. *Review of the Department of Homeland Security’s approach to risk analysis*. National Academies Press, 2010 (cit. on p. 3).

[29] Hantao Cui, Fangxing Li, and Kevin Tomsovic. “Hybrid Symbolic-Numeric Framework for Power System Modeling and Analysis”. In: *IEEE Transactions on Power Systems* (2020) (cit. on pp. 8, 28, 125, 137, 143).

[30] Salvatore D’Arco, Jon Are Suul, and Olav B. Fosso. “A Virtual Synchronous Machine implementation for distributed control of power converters in SmartGrids”. In: *Electric Power Systems Research* 122 (2015), pp. 180–197 (cit. on p. 140).

[31] Joseph F. DeCarolis, Kevin Hunter, and Sarat Sreepathi. “The case for repeatable analysis with energy economy optimization models”. In: *Energy Economics* 34.6 (Nov. 2012), pp. 1845–1853. ISSN: 0140-9883. DOI: `10.1016/j.eneco.2012.07.004`. (Visited on 08/02/2019) (cit. on pp. 42, 52).

[32] Joseph F DeCarolis, Kevin Hunter, and Sarat Sreepathi. “The case for repeatable analysis with energy economy optimization models”. In: *Energy Economics* 34.6 (2012), pp. 1845–1853 (cit. on p. 25).

[33] Ewa Deelman et al. “The future of scientific workflows”. In: *The International Journal of High Performance Computing Applications* 32.1 (2018), pp. 159–175 (cit. on p. 17).

[34] C.L. Demarco and George Verghese. “Bringing Phasor Dynamics into the Power System Load Flow”. In: *1993 North American Power Symposium Washington, D.C.* Oct. 1993, pp. 463–729 (cit. on pp. 114, 116).

[35] Turhan Demiray. “Simulation of power system dynamics using dynamic phasor models”. PhD thesis. ETH Zurich, 2008 (cit. on pp. 116, 118, 122, 123).

[36] Hermann W Dommel. “Digital computer solution of electromagnetic transients in single-and multiphase networks”. In: *IEEE transactions on power apparatus and systems* 4 (1969), pp. 388–399 (cit. on p. 123).

[37] D Donoho and Victoria Stodden. “Reproducible research in the mathematical sciences”. In: *The Princeton Companion to Applied Mathematics, Nicholas J. Higham, Mark R. Dennis, Paul Glendinning, Paul A. Martin, Fadil Santosa, and Jared Tanner, editors, Princeton University Press, Princeton, NJ, USA* (2015), pp. 916–925 (cit. on p. 10).

[38] David L Donoho. “An invitation to reproducible computational research”. In: *Biostatistics* 11.3 (2010), pp. 385–388 (cit. on pp. 7, 10).

[39] Gary Dorris and David Millar. “Making the Right Resource Choice Requires Making the Right Model Choice”. In: *NRRI Insights* (2021), pp. 1–7 (cit. on p. 43).

[40] Kate Doubleday, Vanessa Van Scyoc Hernandez, and Bri-Mathias Hodge. “Benchmark probabilistic solar forecasts: Characteristics and recommendations”. In: *Solar Energy* 206 (2020), pp. 52–67 (cit. on p. 91).

[41] Oscar Dowson. “The policy graph decomposition of multistage stochastic programming problems”. In: *Networks* 76.1 (2020), pp. 3–23. DOI: `10.1002/net.21932`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.21932` (cit. on pp. 80, 89).

[42] Oscar Dowson and Lea Kapelevich. “SDDP.jl: A Julia Package for Stochastic Dual Dynamic Programming”. In: *INFORMS Journal on Computing* 33.1 (2021), pp. 27–33. DOI: `10.1287/ijoc.2020.0987` (cit. on p. 88).

[43] Pengwei Du et al. “New Ancillary Service Market for ERCOT”. In: *IEEE Access* 8 (2020), pp. 178391–178401 (cit. on p. 81).

[44] Iain Dunning, Joey Huchette, and Miles Lubin. “JuMP: A Modeling Language for Mathematical Optimization”. In: *SIAM Review* 59.2 (2017), pp. 295–320. DOI: `10.1137/15M1020575` (cit. on p. 51).

[45] Iain Dunning, Joey Huchette, and Miles Lubin. “JuMP: A modeling language for mathematical optimization”. In: *SIAM Review* 59.2 (2017), pp. 295–320 (cit. on p. 12).

[46] E. Ela, M. Milligan, and M. O’Malley. “A flexible power system operations simulation model for assessing wind integration”. In: *2011 IEEE Power and Energy Society General Meeting*. July 2011, pp. 1–8. DOI: `10.1109/PES.2011.6039033` (cit. on p. 43).

[47] Erik Ela and Mark O’Malley. “Studying the variability and uncertainty impacts of variable generation at multiple timescales”. In: *IEEE Transactions on Power Systems* 27.3 (2012), pp. 1324–1333 (cit. on pp. 63, 65, 66, 84).

[48] Electricity Reliability Council of Texas. *ERCOT Nodal Protocols*. July 2021. URL: `http://www.ercot.com/mktrules/nprotocols/current` (visited on 07/20/2021) (cit. on pp. 77, 86).

[49] Electricity Reliability Council of Texas. *Functional Description of the Core Market Management System (MMS) Applications for Look-Ahead SCED*. July 2011. URL: http://www.ercot.com/content/meetings/metf/keydocs/2012/0927/06_mms_lookahead_sced_phase1_requirements_version_9_11_12_me.doc (visited on 07/20/2021) (cit. on p. 93).

[50] Christine Gamble et al. *WindView*. Oct. 2018. DOI: 10.11578/dc.20190128.1. URL: hhttps://github.com/windview/client (cit. on pp. 96, 107).

[51] Robert C Gentleman et al. "Bioconductor: open software development for computational biology and bioinformatics". In: *Genome biology* 5.10 (2004), R80 (cit. on p. 10).

[52] Vahan Gevorgian, Yingchen Zhang, and Erik Ela. "Investigating the impacts of wind generation participation in interconnection frequency response". In: *IEEE transactions on Sustainable Energy* 6.3 (2014), pp. 1004–1012 (cit. on p. 84).

[53] Martin Glauer et al. "The Open Energy Ontology". In: (2020) (cit. on p. 31).

[54] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. "Probabilistic forecasts, calibration and sharpness". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69.2 (2007), pp. 243–268 (cit. on p. 101).

[55] Steven N. Goodman, Daniele Fanelli, and John P. A. Ioannidis. "What does research reproducibility mean?" In: *Science Translational Medicine* 8.341 (2016), 341ps12–341ps12. ISSN: 1946-6234. DOI: 10.1126/scitranslmed.aaf5027 (cit. on p. 9).

[56] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008 (cit. on p. 130).

[57] Samuele Grillo, Antonio Pievatolo, and Enrico Tironi. "Optimal Storage Scheduling Using Markov Decision Processes". In: *IEEE Transactions on Sustainable Energy* 7.2 (2016), pp. 755–764. DOI: 10.1109/TSTE.2015.2497718 (cit. on p. 78).

[58] Adrien Guironnet et al. "Towards an Open-Source Solution using Modelica for Time-Domain Simulation of Power Systems". In: *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. IEEE, Oct. 2018. DOI: 10.1109/isgteurope.2018.8571872 (cit. on p. 125).

[59] Francis H Harlow and Jacob E Fromm. "Computer experiments in fluid dynamics". In: *Scientific American* 212.3 (1965), pp. 104–111 (cit. on p. 9).

[60] Nikos Hatziargyriou et al. "Definition and classification of power system stability revisited & extended". In: *IEEE Transactions on Power Systems* (2020) (cit. on pp. 5, 108).

[61] Sue Ellen Haupt et al. "The use of probabilistic forecasts: Applying them in theory and practice". In: *IEEE Power and Energy Magazine* 17.6 (2019), pp. 46–57 (cit. on pp. 79, 95, 96).

[62] Simon Haykin. *Communication systems*. John Wiley & Sons, 2008 (cit. on p. 115).

[63] Michael T Heath. *Scientific computing: an introductory survey*. Vol. 80. SIAM, 2018 (cit. on p. 9).

[64] J. M. Henderson. "Automatic Digital Computer Solution of Load Flow Studies [includes discussion]". In: *Transactions of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems* 73.2 (Jan. 1954), pp. 1696–1702. ISSN: 0097-2460. DOI: 10.1109/AIEEPAS.1954.4499023 (cit. on p. 26).

[65] R. Henriquez-Auba et al. "Grid Forming Inverter Small Signal Stability: Examining Role of Line and Voltage Dynamics". In: *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*. 2020, pp. 4063–4068. DOI: 10.1109/IECON43393.2020.9255030 (cit. on p. 40).

[66] Rodrigo Henriquez-Auba et al. "Grid Forming Inverter Small Signal Stability: Examining Role of Line and Voltage Dynamics". In: *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2020, pp. 4063–4068 (cit. on pp. 109, 125, 144).

[67] Glyn A Holton. "Defining risk". In: *Financial analysts journal* 60.6 (2004), pp. 19–25 (cit. on p. 81).

[68] Hannele Holttinen et al. "Methodologies to determine operating reserves due to increased wind power". In: *IEEE Transactions on Sustainable Energy* 3.4 (2012), pp. 713–723 (cit. on p. 63).

[69] Tito Homem-de-Mello and Bernardo K Pagnoncelli. "Risk aversion in multistage stochastic programming: A modeling and algorithmic perspective". In: *European Journal of Operational Research* 249.1 (2016), pp. 188–199 (cit. on pp. 86, 88).

[70] Kari Hreinsson, Anna Scaglione, and Bita Analui. "Continuous time multi-stage stochastic unit commitment with storage". In: *IEEE Transactions on Power Systems* 34.6 (2019), pp. 4476–4489 (cit. on p. 78).

[71] Qiuhua Huang et al. "Effect of accurate modelling of converter interfaced generation on a practical bulk power system". In: *IET Generation, Transmission & Distribution* 14.15 (2020), pp. 3108–3116 (cit. on p. 109).

[72] Qi Huangfu and JA Julian Hall. "Parallelizing the dual revised simplex method". In: *Mathematical Programming Computation* 10.1 (2018), pp. 119–142 (cit. on p. 60).

[73] "IEEE Guide for Synchronous Generator Modeling Practices and Parameter Verification with Applications in Power System Stability Analyses". In: *IEEE Std 1110-2019 (Revision of IEEE Std 1110-2002)* (2020), pp. 1–92. DOI: 10.1109/IEEESTD.2020.9020274 (cit. on pp. 108, 118).

[74] N. Jaleeli et al. "Understanding automatic generation control". In: *IEEE Transactions on Power Systems* 7.3 (1992), pp. 1106–1122 (cit. on pp. 64, 65).

[75] S. D. Jascourt, D. Kirk-Davidoff, and C. Cassidy. “Forecasting Solar Power and Irradiance – Lessons from Real-World Experiences”. In: *Proceedings of American Solar Energy Society National Solar Conference 2016*. Ed. by R. Perez and D. Renne. 2016, pp. 112–120 (cit. on p. 91).

[76] S. Jascourt, K. Doubleday, and L. Munchak. “Probabilistic Solar Power Forecasts Applying Bayesian Model Averaging to a Large Superensemble”. In: *American Meteorological Society 12th Conference on Weather, Climate, and the New Energy Economy*. Jan. 2021 (cit. on p. 91).

[77] Hassan K Khalil. *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002 (cit. on p. 113).

[78] L.K. Kirchmayer. *Economic Operation of Power Systems*. General Electric series. Wiley, 1958 (cit. on pp. 3, 27).

[79] Jack PC Kleijnen et al. “State-of-the-art review: a user’s guide to the brave new world of designing simulation experiments”. In: *INFORMS Journal on Computing* 17.3 (2005), pp. 263–289 (cit. on p. 7).

[80] Petar Kokotović, Hassan K Khalil, and John O’reilly. *Singular perturbation methods in control: analysis and design*. SIAM, 1999 (cit. on p. 120).

[81] D. Krishnamurthy. “PSST: An open-source power system simulation toolbox in Python”. In: *2016 North American Power Symposium (NAPS)*. 2016, pp. 1–6 (cit. on p. 27).

[82] Benjamin Kroposki. “Integrating high levels of variable renewable energy into electric power systems”. In: *Journal of Modern Power Systems and Clean Energy* 5.6 (Nov. 2017), pp. 831–837. ISSN: 2196-5420 (cit. on p. 42).

[83] Prabha Kundur. *Power system stability and control*. McGraw-Hill New York, 1994 (cit. on p. 132).

[84] T. Kwong. *Hands-On Design Patterns and Best Practices with Julia*. Packt Publishing, 2020. ISBN: 9781838648817 (cit. on pp. 26, 30).

[85] Jose Daniel Lara, Daniel E Olivares, and Claudio A Cañizares. “Robust energy management of isolated microgrids”. In: *IEEE Systems Journal* 13.1 (2018), pp. 680–691 (cit. on p. 16).

[86] Jose Daniel Lara et al. “A Multi-Stage Stochastic Risk Assessment with Markovian Representation of Renewable Power”. In: *IEEE Transactions on Sustainable Energy* (2021), pp. 1–1. DOI: 10.1109/TSTE.2021.3114615 (cit. on p. 105).

[87] Jose Daniel Lara et al. “AGC Simulation Model for Large Renewable Energy Penetration Studies”. In: *(to appear) 2020 North American Power Symposium (NAPS)*. 2020, pp. 1–6 (cit. on p. 40).

[88] Jose Daniel Lara et al. "Computational experiment design for operations model simulation". In: *Electric Power Systems Research* 189 (2020), p. 106680. ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2020.106680. URL: http://www.sciencedirect.com/science/article/pii/S0378779620304831 (cit. on pp. 25, 42, 43, 50).

[89] Jose Daniel Lara et al. "PowerSystems.jl A power system data management package for large scale modeling". In: *SoftwareX* 15 (2021), p. 100747 (cit. on pp. 52, 93, 131).

[90] Benoit Legat et al. "MathOptInterface: a data structure for mathematical optimization problems". In: *INFORMS Journal on Computing* (2021). DOI: 10.1287/ijoc.2021.1067 (cit. on pp. 12, 51).

[91] Randall J LeVeque, Ian M Mitchell, and Victoria Stodden. "Reproducible research for scientific computing: Tools and strategies for changing the culture". In: *Computing in Science & Engineering* 14.4 (2012), pp. 13–17 (cit. on p. 7).

[92] Binghui Li and Jie Zhang. "A review on the integration of probabilistic solar forecasting in power systems". In: *Solar Energy* 207 (2020), pp. 777–795 (cit. on pp. 77, 79).

[93] F. Li and R. Bo. "Small test systems for power system economic studies". In: *IEEE PES General Meeting*. July 2010, pp. 1–4. DOI: 10.1109/PES.2010.5589973 (cit. on p. 24).

[94] Fangxing Li and Rui Bo. "Small test systems for power system economic studies". In: *IEEE PES general meeting*. IEEE. 2010, pp. 1–4 (cit. on p. 59).

[95] Yitong Li, Yunjie Gu, and Tim Green. "Mapping of Dynamics between Mechanical and Electrical Ports in SG-IBR Composite Grids". In: *IEEE Transactions on Power Systems* (2022), pp. 1–1. DOI: 10.1109/TPWRS.2022.3141882 (cit. on p. 109).

[96] Sara López-Pintado and Juan Romo. "On the Concept of Depth for Functional Data". eng. In: *Journal of the American Statistical Association* 104.486 (2009), pp. 718–734. ISSN: 0162-1459 (cit. on p. 101).

[97] Alvaro Lorca et al. "Multistage adaptive robust optimization for the unit commitment problem". In: *Operations Research* 64.1 (2016), pp. 32–51 (cit. on p. 78).

[98] Runzhao Lu et al. "Multi-stage stochastic programming to joint economic dispatch for energy and reserve with uncertain renewable energy". In: *IEEE Transactions on Sustainable Energy* (2019) (cit. on p. 78).

[99] Yingbo Ma et al. "A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions". In: *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE. 2021, pp. 1–9 (cit. on p. 13).

[100] Jan Machowski et al. *Power system dynamics: stability and control*. John Wiley & Sons, 2020 (cit. on pp. 63, 64).

[101] Yuri V Makarov et al. *Assessing the value of regulation resources based on their time response characteristics*. Tech. rep. Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2008 (cit. on pp. 63, 66).

[102] Yuri V Makarov et al. "Operational impacts of wind generation on California power systems". In: *IEEE Transactions on power systems* 24.2 (2009), pp. 1039–1050 (cit. on pp. 63, 65).

[103] Charles C Margossian. "A review of automatic differentiation and its efficient implementation". In: *Wiley interdisciplinary reviews: data mining and knowledge discovery* 9.4 (2019), e1305 (cit. on p. 130).

[104] Manuel Marin et al. "Spine-project/Spine-Toolbox: v0.5.0-final.1". In: (Feb. 2021). DOI: `10.5281/zenodo.4501197` (cit. on p. 28).

[105] Uros Markovic et al. "Stability analysis of converter control modes in low-inertia power systems". In: *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. IEEE. 2018, pp. 1–6 (cit. on p. 140).

[106] Uros Markovic et al. "Understanding small-signal stability of low-inertia systems". In: *IEEE Transactions on Power Systems* (2021) (cit. on pp. 109, 126, 144).

[107] J.R. Marti. "Shifted frequency analysis (SFA) for EMTP simulation of fundamental frequency power system dynamics". In: *Internal Report, Power System Laboratory*. The University of British Columbia, 2005 (cit. on p. 115).

[108] Carlo Brancucci Martinez-Anido et al. "The value of day-ahead solar power forecasting improvement". In: *Solar Energy* 129 (2016), pp. 192–203 (cit. on p. 16).

[109] Dennis van der Meer. "A benchmark for multivariate probabilistic solar irradiance forecasts". In: *Solar Energy* 225 (Sept. 2021), pp. 286–296. ISSN: 0038-092X. DOI: `10.1016/J.SOLENER.2021.07.010` (cit. on p. 101).

[110] Dennis van der Meer et al. "Clear-sky index space-time trajectories from probabilistic solar forecasts: Comparing promising copulas". In: *Journal of Renewable and Sustainable Energy* 12.2 (2020), p. 026102 (cit. on p. 78).

[111] *Methods: The Julia Language*. `https://docs.julialang.org/en/v1/manual/-methods/`. Accessed: 2019-09-25 (cit. on p. 125).

[112] F Milano, M Zhou, and GuanJi Hou. "Open model for exchanging power system data". In: *2009 IEEE Power & Energy Society General Meeting*. IEEE. 2009, pp. 1–7 (cit. on p. 29).

[113] Federico Milano. "A python-based software tool for power system analysis". In: *Power and Energy Society General Meeting (PES), 2013 IEEE*. IEEE. 2013, pp. 1–5 (cit. on p. 27).

[114] Federico Milano. “An open source power system analysis toolbox”. In: *IEEE Transactions on Power systems* 20.3 (2005), pp. 1199–1206 (cit. on pp. 8, 27).

[115] Federico Milano. “An open source power system analysis toolbox”. In: *IEEE Transactions on Power systems* 20.3 (2005), pp. 1199–1206 (cit. on p. 125).

[116] Federico Milano. “Complex Frequency”. In: *IEEE Transactions on Power Systems* (2021), pp. 1–1. DOI: 10.1109/TPWRS.2021.3107501 (cit. on p. 119).

[117] Federico Milano. *Power system modelling and scripting*. Springer Science & Business Media, 2010 (cit. on pp. 25, 27, 31, 108, 122, 125, 126, 132, 133).

[118] Federico Milano. “Semi-Implicit Formulation of Differential-Algebraic Equations for Transient Stability Analysis”. In: *IEEE Transactions on Power Systems* 31.6 (2016), pp. 4534–4543. DOI: 10.1109/TPWRS.2016.2516646 (cit. on p. 126).

[119] Federico Milano and Álvaro Ortega Manjavacas. *Frequency Variations in Power Systems: Modeling, State Estimation, and Control*. John Wiley & Sons, 2020 (cit. on pp. 113, 114, 119).

[120] Federico Milano et al. “Foundations and challenges of low-inertia systems”. In: *2018 Power Systems Computation Conference (PSCC)*. IEEE. 2018, pp. 1–25 (cit. on pp. 5, 108).

[121] George S Misyris, Spyros Chatzivasileiadis, and Tilman Weckesser. “Grid-forming converters: Sufficient conditions for RMS modeling”. In: *Electric Power Systems Research* 197 (2021), p. 107324 (cit. on p. 109).

[122] J.M. Morales et al. *Integrating Renewables in Electricity Markets*. First. Vol. 205. Springer International Series in Operations Research & Management Science. Springer US, 2014 (cit. on p. 1).

[123] Robbie Morrison. “Energy system modeling: Public transparency, scientific reproducibility, and open development”. In: *Energy strategy reviews* 20 (2018), pp. 49–63 (cit. on p. 10).

[124] Amin Nasri et al. “Network-constrained AC unit commitment under uncertainty: A Benders’ decomposition approach”. In: *IEEE Transactions on Power Systems* 31.1 (2015), pp. 412–422 (cit. on p. 15).

[125] Engineering National Academies of Sciences, Medicine, et al. *The Future of Electric Power in the United States*. 2021 (cit. on p. 2).

[126] NERC. “Reliability Standards for the Bulk Electric Systems of North America”. In: (2020). URL: https://www.nerc.com/pa/Stand/ReliabilityStandardsCompleteSet/RSCompleteSet.pdf (cit. on p. 63).

[127] Scott Nestler. “Reproducible (Operations) Research-A primer on reproducible research and why the OR community should care about it.” In: *OR/MS Today* 38.5 (2011), p. 22 (cit. on p. 7).

[128] Charles I. Nweke et al. “Benefits of chronological optimization in capacity planning for electricity markets”. In: *2012 IEEE International Conference on Power System Technology (POWERCON)*. 2012, pp. 1–6. DOI: `10.1109/PowerCon.2012.6401421` (cit. on pp. 42, 45).

[129] Colm J O’Rourke et al. “A geometric interpretation of reference frames and transformations: dq0, Clarke, and Park”. In: *IEEE Transactions on Energy Conversion* 34.4 (2019), pp. 2070–2083 (cit. on pp. 113, 117).

[130] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press, 2010 (cit. on p. 9).

[131] *OpenEnergyPlatform/oeplatform*. original-date: 2015-11-20T14:21:02Z. Mar. 2021. URL: `https://github.com/OpenEnergyPlatform/oeplatform` (visited on 03/16/2021) (cit. on p. 28).

[132] Patrick Panciatici et al. “Advanced optimization methods for power systems”. In: *2014 Power Systems Computation Conference*. IEEE. 2014, pp. 1–18 (cit. on pp. 45, 78, 83).

[133] Mario Paolone et al. “Fundamentals of power systems modelling in the presence of converter-interfaced generation”. In: *Electric Power Systems Research* 189 (2020), p. 106811 (cit. on pp. 5, 108, 109).

[134] Anthony Papavasiliou and Shmuel S Oren. “A stochastic unit commitment model for integrating renewable supply and demand response”. In: *2012 IEEE Power and Energy Society General Meeting*. IEEE. 2012, pp. 1–6 (cit. on pp. 14, 16).

[135] Alessandra Parisio, Evangelos Rikos, and Luigi Glielmo. “A model predictive control approach to microgrid operation optimization”. In: *IEEE Transactions on Control Systems Technology* 22.5 (2014), pp. 1813–1827 (cit. on p. 16).

[136] Robert H Park. “Two-reaction theory of synchronous machines generalized method of analysis-part I”. In: *Transactions of the American Institute of Electrical Engineers* 48.3 (1929), pp. 716–727 (cit. on p. 118).

[137] Carl Pechman. “Model-Limited Choice and the Determination of the Need for Generation Capacity”. In: *Regulating Power*. Springer, 1993, pp. 77–97 (cit. on p. 43).

[138] Edo Pellizzari et al. *Reproducibility: A Primer on Semantics and Implications for Research*. RTI Press, 2017 (cit. on pp. 7, 9, 10).

[139] Roger D. Peng. “Reproducible Research in Computational Science”. In: *Science* 334.6060 (2011), pp. 1226–1227. ISSN: 0036-8075. DOI: `10.1126/science.1213847`. eprint: `https://science.sciencemag.org/content/334/6060/1226.full.pdf` (cit. on pp. 9, 10).

[140] George Peponides, PV Kokotovic, and J Chow. “Singular perturbations and time scales in nonlinear models of power systems”. In: *IEEE Trans. on Circuits and systems* 29.11 (1982), pp. 758–767 (cit. on p. 131).

[141] M.V.F. Pereira and L.M.V.G. Pinto. “Multi-Stage Stochastic Optimization Applied to Energy Planning”. In: *Mathematical Programming* 52 (1991), pp. 359–375 (cit. on pp. 78, 89).

[142] Stefan Pfenninger et al. “Opening the black box of energy modelling: Strategies and lessons learned”. In: *Energy Strategy Reviews* 19 (2018), pp. 63–71 (cit. on pp. 25, 42, 52).

[143] Stefan Pfenninger, Adam Hawkes, and James Keirstead. “Energy systems modeling for twenty-first century energy challenges”. In: *Renewable and Sustainable Energy Reviews* 33 (2014), pp. 74–86. ISSN: 1364-0321. DOI: `https://doi.org/10.1016/j.rser.2014.02.003`. URL: `http://www.sciencedirect.com/science/article/pii/S1364032114000872` (cit. on p. 1).

[144] Pierre Pinson et al. “From probabilistic forecasts to statistical scenarios of short-term wind power production”. In: *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology* 12.1 (2009), pp. 51–62 (cit. on p. 78).

[145] Anton Plietzsch et al. “PowerDynamics. jl—An experimentally validated open-source package for the dynamical analysis of power grids”. In: *SoftwareX* 17 (2022), p. 100861 (cit. on p. 125).

[146] Hao Quan et al. “A computational framework for uncertainty integration in stochastic unit commitment with intermittent renewable energy sources”. In: *Appl. energy* 152 (2015), pp. 71–82 (cit. on p. 14).

[147] Christopher Rackauckas and Qing Nie. “Confederated modular differential equation APIs for accelerated algorithm development and benchmarking”. In: *Advances in Engineering Software* 132 (2019), pp. 1–6 (cit. on p. 128).

[148] Christopher Rackauckas and Qing Nie. “DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia”. In: *The Journal of Open Research Software* 5.1 (2017). DOI: `10.5334/jors.151` (cit. on pp. 12, 125).

[149] Ehsan Rehman et al. *Dynamic Stability Assessment of High Penetration of Renewable Generation in the ERCOT Grid*. Tech. rep. Version 1.0. ERCOT, Apr. 2018 (cit. on p. 109).

[150] Jarrett Revels, Miles Lubin, and Theodore Papamarkou. “Forward-mode automatic differentiation in Julia”. In: *arXiv preprint arXiv:1607.07892* (2016) (cit. on pp. 13, 128, 130).

[151] Ciaran Roberts et al. "Grid-Coupled Dynamic Response of Battery-Driven Voltage Source Converters". In: *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE. 2020, pp. 1–6 (cit. on p. 40).

[152] R Tyrrell Rockafellar, Stanislav Uryasev, et al. "Optimization of conditional value-at-risk". In: *Journal of risk* 2 (2000), pp. 21–42 (cit. on p. 87).

[153] Nicolas P Rougier et al. "Sustainable computational science: the ReScience initiative". In: *PeerJ Computer Science* 3 (2017), e142 (cit. on p. 9).

[154] Birgit Rudloff, Alexandre Street, and Davi M Valladão. "Time consistency and risk averse dynamic decision models: Definition, interpretation and practical consequences". In: *European Journal of Operational Research* 234.3 (2014), pp. 743–750 (cit. on pp. 78, 86).

[155] Susan M Sanchez, Paul J Sánchez, and Hong Wan. "Work smarter, not harder: a tutorial on designing and conducting simulation experiments". In: *2018 Winter Simulation Conference (WSC)*. IEEE. 2018, pp. 237–251 (cit. on pp. 7, 13, 14).

[156] Seth R Sanders et al. "Generalized averaging method for power conversion circuits". In: *IEEE Transactions on power Electronics* 6.2 (1991), pp. 251–259 (cit. on p. 114).

[157] Geir Kjetil Sandve et al. "Ten simple rules for reproducible computational research". In: *PLoS computational biology* 9.10 (2013), e1003285 (cit. on p. 25).

[158] Peter W Sauer, Mangalore A Pai, and Joe H Chow. *Power system dynamics and stability: with synchrophasor measurement and power system toolbox*. John Wiley & Sons, 2017 (cit. on pp. 63, 64, 65, 67, 121).

[159] PW Sauer, S Ahmed-Zaid, and PV Kokotovic. "An integral manifold approach to reduced order dynamic modeling of synchronous machines". In: *IEEE transactions on Power Systems* 3.1 (1988), pp. 17–23 (cit. on p. 121).

[160] Michael Scheuerer and Thomas M. Hamill. "Variogram-Based Proper Scoring Rules for Probabilistic Forecasts of Multivariate Quantities". In: *Monthly Weather Review* 143 (4 Apr. 2015), pp. 1321–1334. ISSN: 1520-0493. DOI: 10.1175/MWR-D-14-00269.1. URL: https://journals.ametsoc.org/view/journals/mwre/143/4/mwr-d-14-00269.1.xml (cit. on p. 101).

[161] Greg Schivley, Ethan Welty, and Neha Patankar. "PowerGenome/PowerGenome: v0.4.1". In: (Jan. 2021). DOI: 10.5281/zenodo.4552835 (cit. on p. 28).

[162] Manajit Sengupta et al. "The National Solar Radiation Data Base (NSRDB)". In: *Renewable and Sustainable Energy Reviews* 89 (2018), pp. 51–60 (cit. on p. 82).

[163] Alexander Shapiro. "Analysis of Stochastic Dual Dynamic Programming Method". In: *European Journal of Operational Research* 209.1 (2011), pp. 63–72 (cit. on p. 89).

[164] A.M. Stankovic, S.R. Sanders, and T. Aydin. "Dynamic phasors in modeling and analysis of unbalanced polyphase AC machines". In: *IEEE Transactions on Energy Conversion* 17.1 (2002), pp. 107–113. DOI: 10.1109/60.986446 (cit. on pp. 116, 117).

[165] Philip B. Stark. "Before reproducibility must come preproducibility". EN. In: *Nature* 557 (May 2018), p. 613. DOI: 10.1038/d41586-018-05256-0. (Visited on 08/01/2019) (cit. on p. 10).

[166] Victoria Stodden. "Reproducible research for scientific computing: Tools and strategies for changing the culture". In: *Computing in Science & Engineering* 14.4 (2012), p. 13 (cit. on p. 7).

[167] Brian Stott. "Power system dynamic response calculations". In: *Proceedings of the IEEE* 67.2 (1979), pp. 219–241 (cit. on pp. 120, 121, 123).

[168] Wencong Su, Jianhui Wang, and Jaehyung Roh. "Stochastic energy scheduling in microgrids with intermittent renewable energy resources". In: *IEEE Transactions on Smart Grid* 5.4 (2013), pp. 1876–1883 (cit. on p. 16).

[169] Sunil Subedi et al. "Review of Methods to Accelerate Electromagnetic Transient Simulation of Power Systems". In: *IEEE Access* 9 (2021), pp. 89714–89731. DOI: 10.1109/ACCESS.2021.3090320 (cit. on p. 109).

[170] Ying Sun and Marc G Genton. "Functional Boxplots". In: *Journal of Computational and Graphical Statistics* 20 (2 2011), pp. 316–334. ISSN: 1537-2715. DOI: 10.1198/jcgs.2011.09224 (cit. on p. 101).

[171] Conor Sweeney et al. "The future of forecasting for renewable energy". In: *Wiley Interdisciplinary Reviews: Energy and Environment* 9.2 (2020), e365 (cit. on p. 77).

[172] *System Advisor Model Version 2018.11.11 (SAM 2018.11.11)*. https://sam.nrel.gov/. National Renewable Energy Laboratory (NREL), Golden, CO, USA. 2018 (cit. on pp. 82, 91).

[173] Josh A Taylor, Sairaj V Dhople, and Duncan S Callaway. "Power systems without fuel". In: *Renewable and Sustainable Energy Reviews* 57 (2016), pp. 1322–1336 (cit. on p. 1).

[174] Daniel Thom, Jose Daniel Lara, and Clayton P. Barrows. *NREL-SIIP/InfrastructureSystems.jl: v1.7.4*. Version v1.7.4. May 2021. DOI: 10.5281/zenodo.4780333. URL: https://doi.org/10.5281/zenodo.4780333 (cit. on p. 31).

[175] Leon Thurner et al. "Pandapower-an open source Python tool for convenient modeling, analysis and optimization of electric power systems". In: *IEEE Transactions on Power Systems* (2018) (cit. on p. 27).

[176] G. L. Torres and V. H. Quintana. “An interior-point method for nonlinear optimal power flow using voltage rectangular coordinates”. In: *IEEE Transactions on Power Systems* 13.4 (Nov. 1998), pp. 1211–1218. DOI: 10.1109/59.736231 (cit. on p. 10).

[177] Aidan Tuohy et al. “Unit commitment for systems with significant wind penetration”. In: *IEEE Transactions on Power Systems* 24.2 (2009), pp. 592–601 (cit. on pp. 15, 16).

[178] Mathias Uslar et al. *The Common Information Model CIM: IEC 61968/61970 and 62325-A practical introduction to the CIM*. Springer, 2012 (cit. on p. 18).

[179] Mathias Uslar et al. *The Common Information Model CIM: IEC 61968/61970 and 62325-A practical introduction to the CIM*. Springer Science & Business Media, 2012 (cit. on p. 31).

[180] Dennis van der Meer. “A benchmark for multivariate probabilistic solar irradiance forecasts”. In: *Solar Energy* 225 (2021), pp. 286–296. ISSN: 0038-092X. DOI: https://doi.org/10.1016/j.solener.2021.07.010 (cit. on p. 82).

[181] Luigi Vanfretti et al. “iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations”. In: *SoftwareX* 5 (2016), pp. 84–88 (cit. on pp. 27, 28).

[182] V Venkatasubramanian. “Tools for dynamic analysis of the general large power system using time-varying phasors”. In: *Int. Journal of Electrical Power & Energy Systems* 16.6 (1994), pp. 365–376 (cit. on pp. 114, 115, 116, 121).

[183] Vaithianathan Venkatasubramanian, Heinz Schattler, and John Zaborszky. “Fast time-varying phasor analysis in the balanced three-phase large electric power system”. In: *IEEE Transactions on Automatic Control* 40.11 (1995), pp. 1975–1982 (cit. on pp. 114, 123).

[184] D Venkatramanan et al. “Integrated System Models for Networks with Generators & Inverters”. In: *arXiv preprint arXiv:2203.08253* (2022) (cit. on pp. 109, 134).

[185] J Wang et al. “Wind power forecasting uncertainty and unit commitment”. In: *Applied Energy* 88.11 (2011), pp. 4014–4023 (cit. on pp. 14, 16).

[186] Qin Wang et al. “Quantifying the economic and grid reliability impacts of improved wind power forecasting”. In: *IEEE Transactions on Sustainable Energy* 7.4 (2016), pp. 1525–1537 (cit. on pp. 16, 42, 63, 65, 84).

[187] Xiaozhe Wang and Hsiao-Dong Chiang. “Analytical studies of quasi steady-state model in power system long-term stability analysis”. In: *IEEE Transactions on Circuits and Systems I* 61.3 (2014), pp. 943–956 (cit. on p. 66).

[188] J. B. Ward and H. W. Hale. "Digital Computer Solution of Power-Flow Problems [includes discussion]". In: *Transactions of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems* 75.3 (Jan. 1956), pp. 398–404. ISSN: 0097-2460. DOI: 10.1109/AIEEPAS.1956.4499318 (cit. on p. 26).

[189] Joseph Warrington et al. "Policy-based reserves for power systems". In: *IEEE Transactions on Power Systems* 28.4 (2013), pp. 4427–4437 (cit. on p. 78).

[190] Matthew West. "Part 2 General Principles for Data Models". In: *Developing High Quality Data Models*. Ed. by Matthew West. Boston: Morgan Kaufmann, 2011, p. 61. ISBN: 978-0-12-375106-5. DOI: https://doi.org/10.1016/B978-0-12-375106-5.00019-1 (cit. on p. 18).

[191] Rebecca Widiss and Kevin Porter. *Review of variable generation forecasting in the west: July 2013-march 2014*. Tech. rep. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2014 (cit. on p. 77).

[192] Greg Wilson et al. "Best practices for scientific computing". In: *PLoS biology* 12.1 (2014), e1001745 (cit. on pp. 10, 18, 25, 40).

[193] Greg Wilson et al. "Good enough practices in scientific computing". In: *PLoS computational biology* 13.6 (2017), e1005510 (cit. on p. 25).

[194] Allen J Wood, Bruce F Wollenberg, and Gerald B Sheblé. *Power generation, operation, and control*. John Wiley & Sons, 2013 (cit. on p. 26).

[195] David L. Woodruff et al. "Constructing probabilistic scenarios for wide-area solar power generation". In: *Solar Energy* 160 (2018), pp. 153–167 (cit. on p. 78).

[196] Working Group on Common Format For Exchange of Solved Load Flow Data. "Common Format For Exchange of Solved Load Flow Data". In: *IEEE Transactions on Power Apparatus and Systems* PAS-92.6 (Nov. 1973), pp. 1916–1925. ISSN: 0018-9510. DOI: 10.1109/TPAS.1973.293571 (cit. on pp. 18, 26, 27).

[197] Lei Wu, Mohammad Shahidehpour, and Zuyi Li. "Comparison of scenario-based and interval optimization approaches to stochastic SCUC". In: *IEEE Transactions on Power Systems* 27.2 (2011), pp. 913–921 (cit. on pp. 14, 16).

[198] Yixing Xu et al. "US Test System with High Spatial and Temporal Resolution for Renewable Integration Studies". In: *arXiv preprint arXiv:2002.06155* (2020) (cit. on p. 35).

[199] Amirnaser Yazdani and Reza Iravani. *Voltage-sourced converters in power systems: modeling, control, and applications*. John Wiley & Sons, 2010 (cit. on pp. 112, 113, 117, 133).

[200] Haoyu Yuan et al. "Developing a reduced 240-bus wecc dynamic model for frequency response study of high renewable integration". In: *2020 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)*. IEEE. 2020, pp. 1–5 (cit. on p. 142).

[201] J Zaborszky, H Schattler, and V Venkatasubramanian. “Error estimation and limitation of the quasi stationary phasor dynamics”. In: *Proc. Power Syst. Comput. Conf.* 1993, pp. 721–729 (cit. on pp. 116, 119).

[202] Peng Zhang, Jose R. Marti, and Hermann W. Dommel. “Synchronous Machine Modeling Based on Shifted Frequency Analysis”. In: *IEEE Transactions on Power Systems* 22.3 (2007), pp. 1139–1147. DOI: 10.1109/TPWRS.2007.901288 (cit. on pp. 113, 115, 116).

[203] Yao Zhang et al. “Chance-constrained two-stage unit commitment under uncertain load and wind power output using bilinear benders decomposition”. In: *IEEE Transactions on Power Systems* 32.5 (2017), pp. 3637–3647 (cit. on p. 15).

[204] T Zheng et al. “Robust optimization and its application to power system operation”. In: *CIGRE*. 2012 (cit. on p. 84).

[205] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, Robert John Thomas, et al. “MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education”. In: *IEEE Transactions on power systems* 26.1 (2011), pp. 12–19 (cit. on pp. 8, 27, 28, 38).