University of Central Florida
STARS

Electronic Theses and Dissertations, 2020-

2021

Parameter Calibration and Optimization in Smart Grid for Synchronous Generators and Converters

Seyyed Rashid Khazeiynasab
University of Central Florida

 Part of the Electrical and Computer Engineering Commons
Find similar works at: https://stars.library.ucf.edu/etd2020
University of Central Florida Libraries http://library.ucf.edu

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2020- by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation
Khazeiynasab, Seyyed Rashid, "Parameter Calibration and Optimization in Smart Grid for Synchronous Generators and Converters" (2021). *Electronic Theses and Dissertations, 2020-*. 889.
https://stars.library.ucf.edu/etd2020/889

PARAMETER CALIBRATION AND OPTIMIZATION IN SMART GRID FOR SYNCHRONOUS GENERATORS AND POWER ELECTRONIC CONVERTERS

by

SEYYED RASHID KHAZEIYNASAB

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2021

Major Professor: Issa Batarseh

© 2021 Seyyed Rashid Khazeiynasab

ABSTRACT

In power systems, monitoring, protection, and control are usually model based; an accurate dynamic model for either synchronous generators or power electronic converters is essential. Besides, renewable energies, smart loads, energy storage, and new market behavior add new sources of uncertainty to power systems. Therefore, planning in real-time and developing high-quality models is crucial to adapt to uncertainties. In this thesis, we propose a framework for validating and calibrating power system models using novel methods. At the first step, we developed the nonlinear sensitivity-based method to find the critical parameters. Then, we propose an Approximate Bayesian Computation (ABC) based method which is a simulation-based method. By proposing an adaptive kernel function and a threshold sequence, we reduce the computational complexity of a ABC with a sequential Monte Carlo sampler (ABC-SMC). Using deep learning to improve the estimation accuracy, we overcome the curse of dimensionality in our proposed method. Via event playback, we build the simulations for training our model and using a parallel multi-modal long short-term memory (PM LSTM), we improve the accuracy for the high dimensional cases, but to build a model, we need to have a lot of samples. Our next proposal was a conditional variational autoencoder (CVAE), which has a great performance and requires a much smaller sample for training. The proposed methods are comprehensively evaluated; all results show that the proposed approaches have great performance in time and accuracy. We demonstrate the effectiveness of the proposed models on a synchronous generator with its controllers, a DC-DC buck converter, and a three-port inverter.

I dedicate this thesis to all my nephews and nieces.

ACKNOWLEDGMENTS

There is a large number of people that I would like to thank individually for making my Ph.D. graduation happens.

First of all, I am very grateful to my advisor, Prof. Issa Batarseh, for all his support. I learned from him how to approach fundamental real-world problems and how to address them systematically. I will forever be thankful to him for his helpful career advice. Besides I would like to express my deepest appreciation to my previous advisor, Dr. Junjian Qi for his consistent support, invaluable guidance and encouragement in my professional life during the Ph.D. I will forever be thankful to him for his helpful career advice.

I am also thankful to my dissertation committee members, professors Aleksander Dimitrovski, Wei Sun, Qifeng Li, and Zhaomiao Guo for providing me invaluable feedback on my research.

Likewise, I extend my thanks to my family for all their support and help throughout the years.

I would like to especially thank my dear Maryam, for her endless love and supports.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

In power systems, monitoring, protection, and control are usually model-based, an accurate dynamic model for either synchronous generators [14–16] or inverters [17] is thus essential. The inaccuracy of the power system model has been witnessed in the blackout that occurred in the Western U.S. in 1996 [1], in which model simulations showed a stable response while the system became unstable [18, 19]. Following the outage, Western Systems Coordinating Council (WSCC) required that all generators larger than 10 MW be tested for model data verification [1]. Besides, reliability standards such as NERC MOD-033 in North America require dynamic models to be validated with phasor measurement unit (PMU) data once every two years to verify the accuracy of the planning models [20]. Dynamic models play an important role in predicting the system response under contingencies. An accurate model for the synchronous generator is essential for a valid evaluation of power systems' dynamic performance and stability. The recent work on dynamic state estimation also needs accurate enough parameters to be able to provide reliable estimations of the dynamic states [21–23]. The synchronous generator is one of the most critical components in power systems, and its accurate modeling is important for studying the dynamics of the system. This is no trivial task because: 1) The models may not be available for all components; 2) Even if the models are available, the parameters of the models may not be available; and 3) Even if the models and the parameters are available, the parameters may have changed over time.

Power System Models

There are several categories of models that need to be developed for modeling a large power system [14]:

- **Transmission System**: This includes transmission lines, mechanically switched shunt capacitors and reactors, power transformers, flexible AC transmission systems, phase-shifting transformers, and high-voltage direct current transmission systems.
- **Generating Units**: This includes the entire spectrum of supply resources, hydro, steam, gas, and geothermal generation, along with rapidly emerging wind and solar power plants.
- **Load**: This includes the electrical load in the system, which ranges from simple light-bulbs to large industrial facilities.
- **Power electronics components**: This includes the inverters and other power electronics components in the system.

Different types of the models

Based on the study's goal, the different types of models are used in the power system analysis. The different types of models in the power system are as follows:

- **Steady-state Models**: Each of the models in Section 1 can be represented in the steady-state models. For developing the steady state models, different accurate calculations of the impedance and rating are considered.
- **Dynamics Models**: Dynamical models of the component represent the dynamic of the components. The dynamical models are usually used in stability studies, which have time constants in the range of few tens of milliseconds to many seconds [1].
- **Short circuit models**: Short circuit studies are one of the critical studies in the power system. The steady-state model can be used for this critical goal. The models which used in this type of study include negative sequence and zero sequences.

Importance of Model Validation

Power system simulations are used in grid planning and operating decisions. Models of power systems are used to predict system performance under expected disturbance conditions. Under-investment and unsafe operations might result from optimistic models. Pessimistic models can also cause unnecessary capital investment, thereby making electric power more expensive. As a result, we need realistic models for power system operation to ensure reliability and efficiency. Therefore, periodically validating the power system models becomes critically important. In the following section, we consider two occurred blackout cases where the models' inaccuracy leads to the outage.

- **2011 blackout in San Diego**:

  One of the most well-known blackouts is the 2011 blackout in San Diego. This blackout's primary cause was attributed to significant weaknesses in the power industry models: Simulations showed a well-dampened system response, but in reality, the system eventually collapsed because of critical contingencies [19, 24].

- **1996 Western Interconnection Outages**:

  Same as the San Diego blackout, the main reason for the 1996 western interconnection outages is the deficiencies in the model were the industry used in their simulations. Figure 1.1 shows the outputs of the simulated models and the system's actual response. As it can be seen, simulations showed a well-dampened system response, but in reality, the response is totally different and eventually collapsed. Following the outage, Western Systems Coordinating Council (WSCC) required that all generators larger than 10 MW be tested for model data verification [1].

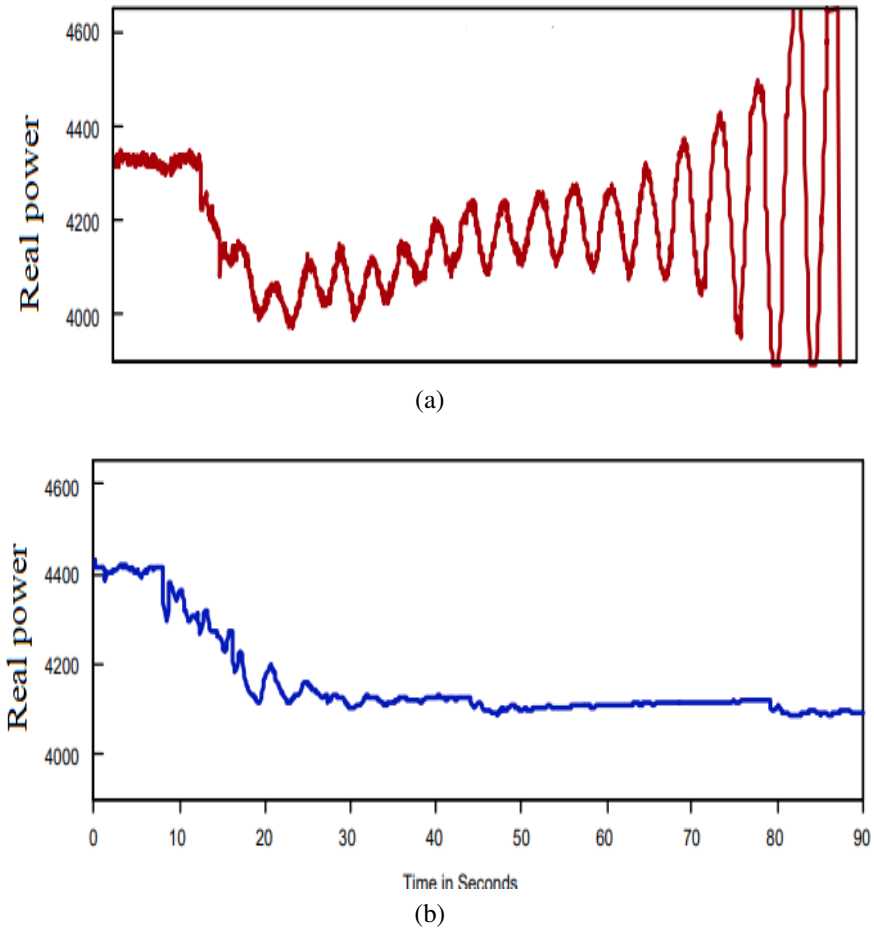


Figure 1.1: Simulated and the observed system real power during August 10, 1996 (a) Actual response and (b) Simulated response [1].

Dissertation Overview and Contributions

This thesis organized as follow:

CHAPTER 2: We begin in this CHAPTER with a comprehensive review of research studies for model validation in the power system and power electronic converters (PEC).

CHAPTER 3: We developed a black-box optimization based on a surrogate model. Then, a

hydro generator with a governor and an exciter are modelled. The critical parameter is identified with sensitivity analysis, and then a general framework for the black-box optimization is provided. Finally, the proposed black-box optimization-based generator parameter calibration method based on a stochastic radial basis function (RBF) surrogate model is being validated and demonstrated through testing on a real hydro generator with eight critical parameters. The main contributions of this CHAPTER are:

- Developed a black-box optimization-based generator parameter calibration method based on a stochastic RBF surrogate model.
- Formulated the generator parameter estimation and provided numerical results for determining an objective function which computationally and effectively works for the parameter estimation problem.

CHAPTER 4: We provide an overview on Approximate Bayesian Computation (ABC)-based method and then, we consider a hydro generator with an exciter, a governor, and a power system stabilizer. Then, we provide a framework to identify the critical parameters of the power system models. We propose a synchrophasor measurement-based generator parameter calibration method by adaptive Approximate Bayesian Computation with Sequential Monte Carlo sampler (A-ABC-SMC) that avoids directly dealing with likelihood functions. In this CHAPTER , we provided the different case studies; we consider the different errors in the parameters' initial values, and we also compare our proposed method with the different methods. We also consider a high-dimensional case with fourteen parameters. The main contributions of this CHAPTER are:

- Instead of using a sensitivity-based approach, an empirical Gramian based approach is proposed directly for the nonlinear generator model to identify the critical parameters with the highest identifiability more accurately.

- We perform generator parameter calibration by A-ABC-SMC. This likelihood-free method does not directly explore the parameters' likelihood surface but instead estimates the posterior distributions of the parameters by a simulation-based procedure.
- The proposed A-ABC-SMC approach significantly improves the computational efficiency of ABC-SMC through adaptive threshold sequences and perturbation kernel function, and carefully chosen distance function.

CHAPTER 5: Since the industrial generator has more parameters, the proposed A-ABC-SMC has an acceptable performance when the prior distribution for the parameters lies in a narrow range. Also, it suffers from the curse of dimensionality. To overcome the mentioned problems, we proposed a deep-learning-based method in this CHAPTER . We developed a parallel multimodal long short term memory (PM-LSTM) method, estimating a high dimensional case. The proposed method has an acceptable accuracy for top eighteen critical parameters, even when the parameter space is large enough.

CHAPTER 6: Training the PM-LSTM model need a lot of samples. Then, we proposed a conditional variational autoencoder (CVAE) method in this CHAPTER . Instead of using a sensitivity-based approach, an elementary effects (EE) approach is proposed directly for the nonlinear power system models to identify the critical parameters with the highest identifiability. We then propose a CVAE method. Compared to the existing methods, the characteristics of the CVAE method are as follows:

1. Only a small number of samples is needed, and the computational efficiency is high.
2. It can provide accurate parameters when the generator model has significant model discrepancies due to gross errors in the parameters and even when prior distribution does not support the true values.

3. It can estimate the high dimensional cases accurately even when the parameter space is large enough.

4. The implementation is more straightforward and reliable without many complications as compared to the existing reinforcement and deep learning methods.

5. The proposed method is simulation-based and does not need a likelihood function or state-space model of the generator. As commercial software already has stability models, implementation is much easier, and it can be used in real-world applications.

Insightful case studies are presented and discussed in detail. The impact of the different prior distributions, different lower/upper bounds on the prior distributions, and the error in the parameters initial values are analyzed. A specific case where the true values are not in support of the prior distributions is studied. Finally, a high-dimensional case with eighteen critical parameters under multiple events is presented.

CHAPTER 7: We have used the proposed A-ABC-SMC to calibrate a DC-DC buck converter. The results of PEC calibration shows that even when the value of the parameters are so small, the proposed method can find the accurate values of the parameters very well.

CHAPTER 8: In this CHAPTER we consider a Three Port Multilevel Inverter (TPMI). Based on its model, we provide an event and generate the data for training a CVAE model. The results of the calibration show that the well-train model can estimate the parameters of the model with high accuracy.

CHAPTER 9: This CHAPTER concludes this thesis and identifies the key contributions and the future directions.

CHAPTER 2: PREVIOUS WORKS

Generator Parameter Estimation and Model Calibration

For traditional parameter calibration methods (offline methods or field tests), such as standstill frequency response method [25], open-circuit frequency response method [26], and DC excitation method [27], the generator has to be out of service, which is time-consuming and not economically viable [28]. To overcome these deficiencies, online methods based on synchrophasor measurements have been proposed [19]. Simple swing equations have been used to analyze the dynamic response and estimate the generator moment of inertia in [29–31]. The proposed method in [32] uses PMU measurements to estimate four parameters of the five-machine dynamic equivalent electro-mechanical model of the Western Electricity Coordinating Council (WECC) power system network. However, these methods may not be able to represent the actual dynamic response of the generator. Online approaches based on linear and nonlinear curve fitting have also been proposed for estimating parameters [33, 34], but multiple solutions may exist for the same model performance [19, 33]. A particle swarm optimization (PSO) based approach is proposed in [35], which has good performance in the case with small disturbances but does not work as well for large disturbances. In [36], the amplitude and damping of oscillation are used as indicative factors for the mismatch between measurements and simulation values. However, this method involves extensive trial-and-error. Gradient-based optimization methods have been proposed in [37, 38], which, however, have to calculate the gradient of the nonlinear objective function with respect to the parameters based on the generator dynamic model in each iteration and may get stuck at the local optima.

Kalman filter (KF) methods have been proposed for generator parameter calibration [30, 39, 40]. The extended KF (EKF) in [39, 40] maintains an elegant and computationally efficient recursive

update. However, the high nonlinearity of the power system model may violate the local linearity assumption and EKF estimation may become unreliable. The ensemble KF (EnKF) as a Monte Carlo implementation of KF is applied to calibrate generator parameters in [19], which avoids linearization but may suffer from slow convergence rate and Gaussian assumption of the process and measurement errors [41]. To address these problems, a Bayesian inference method with polynomial-chaos-expansion-based surrogate models is proposed [41]. However, under large parameter errors, the reliability of the estimation may degrade. Then a two-stage hybrid Markov chain Monte Carlo (MCMC) method is further developed to recover the posteriori distribution of the parameters [42] and an importance sampling technique is introduced to more efficiently estimate the posterior [43].

Besides, the multistage genetic algorithm optimization has been proposed in [44] to estimate the generator parameters. However, there is no guarantee of finding global optima and the efficiency of the algorithm still needs to be improved. To address these problems, in this CHAPTER we develop a black-box optimization based generator parameter calibration method based on a stochastic radial basis function (RBF) surrogate model [45]. The proposed method does not require an explicit objective function, can solve non-convex problems by building a global model of the objective function, and guarantees convergence to the global optimum from a theoretical standpoint if the number of iterations is large enough [44].

Machine learning has been applied for parameter calibration [46–49]. The method in [46] generates extensive simulation data to train a multi-output convolutional neural network model and predict a small number of generator parameters. The q-learning-based method in [47] only works for cases, where a few parameters need to be calibrated. The proposed deep q-learning-based method in [48] performs well in low-dimensional cases under different events. However, the q-value affects the policy significantly, and a small change in the q-value affects the policy a lot. Furthermore, its implementation is complicated since a lot of efforts are needed to make the neural network function

approximation work and the deep q-learning be stabilized. The Soft Actor-Critic-based method proposed in [49] has a good performance for calibration but requires a lot of hyper-parameter tuning to converge.

In the above section, we only focus on the generator parameter estimation. However, there are parameter estimation problems in bulk power systems and also in the emerging power electronics problems especially with the increasing penetration of power electronics interfaced renewable generation. Then, having a unified approach that could be effective for both problems is needed. In the following section, we focus on the previous works in the PEC area.

PEC Parameter Calibration

PEC are broadly used in different power electronics applications, including motor drives, computers, portable electronics, domestic appliances, or in power conversion systems for renewable generation, variable-speed ac machine drives, uninterruptible power supplies, aerospace power systems among others [50–52]. Monitoring the conditions of the PEC and analyzing their outputs in the system plays an important role in the operation and reliability of power system. Estimate the parameters of PEC can improve the mathematical models of the converters which are based on the linear analysis [53], and also to design a good controller, the exact model of the PEC is needed, which relays on the exact parameters. On the other hand, the parameters of the PEC change with age, manufacturing tolerance, parasitic elements, and load changes. Consequently, these uncertainties must be considered during the modeling stage of the power converter. For example, it has been reported in [54] that capacitors cause 30% of the failures in converter circuits [50, 55, 56].

In utility-scale photovoltaic (PV) systems, PV inverters are responsible for about 37% of unscheduled maintenance events [57]. Power electronics converters are responsible for about 13% of wind

tower downtime and 18% of all failures in wind turbines under observation. Power semiconductor switches and capacitors are the most vulnerable types of components of PECs [58]. Failures can occur due to temperature stresses, mechanical vibrations, and humidity, in that order [59].

In particular, power electronic converters are used in active distribution grids consisting of commercial-off-the-shelf converters that need to be modeled. Unfortunately, some field failure reports reveal that power electronic converters represent one of the weakest points in the systems [59]. The converters usually do not come with a behavioral model, making it impossible to conduct theoretical studies or simulations for interconnection and control purposes. One problem of the used converters in the system is that the converters which are used are usually made by the different brands. Most of these converters can control power flow in the grid element, but because of their high penetration in grid elements, they can be in conflict with conventional devices installed in power networks. It is, therefore, necessary to develop comprehensive control systems with which all operations can be coordinated. For the purpose of analyzing, designing, and controlling active networks with high integration of electronic converters, the electronic power converter model is indispensable [60]. Power electronic converter failures account for a large percentage of system failures, affecting system reliability [59]. Failure of the converter or other power electronics component can even cause blackouts in the power system [18]. Then, the conditions of the PEC should be monitored [61]. Besides, the limitations of statistical reliability prediction give rise to the concept of prognostics and health management (PHM). The PHM approach determines how well components or systems are performing by monitoring the operating conditions alone or operation conditions plus key performance and degradation parameters. A huge amount of data can also be produced by closely monitoring components and systems. Manufacturers can utilize this data to figure out how best to handle new designs and products in the field.

The white-box based method in [55] has good accuracy, but its computational time is high, and for a complex system, its implementation is a big problem. The proposed methods in [62, 63] use a

polynomial interpolation method with the least-squares (LS) algorithm to estimate the parameters of the converter, but these methods can not find the global optimal, and under the different load changes tests, the estimated parameters may be different from the true parameters. The subspace-based method proposed in [64] has good accuracy, but its final solution needs heavy difficulties to implement. In [65, 66] the parameters of the DC-DC converter are estimated based on the LS technique, and finally, a non-linear black-box model of the converter proposed. But, in these methods, the physical parameters do not have meaning, and based on these types of models, we can not analyze the model correctly.

The measurement-based approaches which are based on acquiring the instantaneous values of the input and output at the terminals of the power converters can be applied for parameter estimation [63]. These approaches have an advantage being compatible with non-invasive online monitoring of the input/output signals. Then, using these types of methods do not to conflict with the operation of the converters.

CHAPTER 3: PMU MEASUREMENT BASED GENERATOR PARAMETER ESTIMATION PARAMETER CALIBRATION BY BLACK-BOX OPTIMIZATION WITH A STOCHASTIC RADIAL BASIS FUNCTION SURROGATE MODEL

Khazeiynasab, S. R., Qi, J. (2020). PMU measurement based generator parameter calibration by black-box optimization with a stochastic radial basis function surrogate model. In 2020 North American Power Symposium (NAPS).

Introduction

In this CHAPTER , we propose a synchrophasor measurement based generator parameter calibration method by a black-box optimization approach with a stochastic radial basis function (RBF) surrogate model. Based on comparison between the outputs of the generator model with estimated parameters and the phasor measurement unit (PMU) measurements, we define an objective function for the black-box optimization problem, which is approximated by a RBF surrogate model. The prior information of the parameters is treated as constraints in the black-box optimization problem. The formulated black-box optimization problem is then solved by a Stochastic Response Surface Method (MSRSM). The effectiveness of the proposed method is tested and validated on a hydro generator. The simulation results show that the proposed approach can accurately and efficiently estimate the generator parameters subject to gross errors in the prior distributions of the parameters.

Generator Dynamic Model

The dynamical model of a synchronous generator can be written in a general form as:

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}) & \text{(3.1a)} \\ \boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\alpha}), & \text{(3.1b)} \end{cases}$$

where $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ are the state transition and output functions, $\boldsymbol{u} \in \mathbb{R}^v$ is the injected measured signals (voltage magnitude, phase angle, and frequency), $\boldsymbol{y} \in \mathbb{R}^o$ is the output variables (including active and reactive power of the generator), $\boldsymbol{\alpha} \in \mathbb{R}^z$ is the parameter vector, and $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector that could include rotor angle, rotor speed, transient or sub-transient voltages, and controller states. The following state transition and observation equations are presented for a synchronous generator and its controllers. In all of the following equations, $df = f_t - f_{t-1}$. The following differential equations represent the GENTPJ synchronous machine model [3, 19]:

$$\begin{aligned}
T'_{\mathrm{do}}\frac{dE'_{\mathrm{q}}}{dt} &= S_{\mathrm{d}}\frac{E'_{\mathrm{q},t-1}(X''_{\mathrm{d}}-X_{\mathrm{d}})+E''_{\mathrm{q},t-1}(X_{\mathrm{d}}-X'_{\mathrm{d}})}{X'_{\mathrm{d}}-X''_{\mathrm{d}}}+E_{\mathrm{fd}}, \\
T'_{\mathrm{qo}}\frac{dE'_{\mathrm{d}}}{dt} &= S_{\mathrm{q}}\frac{E'_{\mathrm{d},t-1}(X''_{\mathrm{q}}-X_{\mathrm{q}})+E''_{\mathrm{d},t-1}(X_{\mathrm{q}}-X'_{q})}{X'_{\mathrm{q}}-X''_{\mathrm{q}}}, \\
T''_{\mathrm{do}}\frac{dE''_{\mathrm{q}}}{dt} &= E'_{\mathrm{q},t-1}-S_{\mathrm{d}}E''_{\mathrm{q},t-1}-(X'_{\mathrm{d}}-X''_{\mathrm{d}})I_{\mathrm{d},t}, \\
T''_{\mathrm{qo}}\frac{dE''_{\mathrm{d}}}{dt} &= E'_{\mathrm{d},t-1}-S_{\mathrm{q}}E''_{\mathrm{d},t-1}+(X'_{\mathrm{d}}-X''_{\mathrm{d}})I_{\mathrm{q},t}, \\
2H\frac{d\omega}{dt} &= P_{\mathrm{me},t-1}-\frac{E''_{t-1}V_t}{X'_{\mathrm{d}}+X_{\mathrm{tr}}}\sin(\delta_{t-1}-\theta_t)-D\frac{d\delta}{dt}, \\
\frac{d\delta}{dt} &= \omega_{t-1}-\omega_0,
\end{aligned}$$

where S_{d}, S_{q} are the saturation coefficients for d-axis and q-axis, E'_{d} and E'_{q} are d- and q-axis transient voltages, E''_{d} and E''_{q} are d- and q-axis sub-transient voltages, δ is rotor angle, and ω is rotor speed, which are considered as state variables. I_{d} and I_{q} are the d- and q-axis currents of the

generator, $E_{\rm fd}$ is the field voltage, and $P_{\rm me}$ is the mechanical power, and V_t and θ_t are the voltage magnitude and phase angle at the high-voltage side of the step-up transformer, and are the injected measured signals. $T'_{\rm do}$ is d-axis transient rotor time constant, H is inertia constant, $X_{\rm q}$ is q-axis synchronous reactance, $X'_{\rm d}$ is d-axis transient reactance, $X'_{\rm q}$ is q-axis transient reactance. More details of the generator model and its parameters can be found in [3, 19]. The IEEEG3 governor model is represented by the following differential equations [2]:

$$
\begin{aligned}
T_{\rm P}\frac{ds_{\rm g0}}{dt} &= P_{\rm r} - d\omega - s_{{\rm g1},t-1}(R_{\rm P} - R_{\rm T}) - s_{{\rm g2},t-1} - s_{{\rm g0},t-1},\\
T_{\rm G}\frac{ds_{\rm g1}}{dt} &= s_{{\rm g0},t-1},\\
T_{\rm R}\frac{ds_{\rm g2}}{dt} &= -R_{\rm T}T_{\rm R}s_{{\rm g1},t-1} - s_{{\rm g2},t-1},\\
T_{\omega}\frac{ds_{\rm g3}}{dt} &= \frac{1}{a_{11}}\big(a_{23}(1+a_{11})s_{{\rm g1},t} - s_{{\rm g3},t-1}\big),
\end{aligned}
$$

where $s_{\rm g0}, s_{\rm g1}, s_{\rm g2}$, and $s_{\rm g3}$ are considered as state variables and $P_{\rm r}$ is the power reference. $T_{\rm G}$ is gate servo time constant, $T_{\rm P}$ is pilot servo valve time constant, $R_{\rm P}$ is permanent speed droop coefficient, $R_{\rm T}$ is transient speed droop coefficient, a_{23} is turbine coefficient, $T_{\rm R}$ is time constant, and $T_{\rm w}$ is water starting time. More details of the governor model and its parameters can be found in [2]. The ESST1A exciter model is represented by the following differential and algebraic

equations:

$$
\begin{aligned}
T_{\mathrm{r}}\frac{ds_{\mathrm{e0}}}{dt} &= V_{\mathrm{c},t-1} - s_{\mathrm{e0},t-1},\\
T_{\mathrm{b1}}\frac{ds_{\mathrm{e2}}}{dt} &= T_{\mathrm{c1}}\frac{ds_{\mathrm{e1}}}{dt} + s_{\mathrm{e1},t-1} - s_{\mathrm{e2},t-1},\\
T_{\mathrm{b}}\frac{ds_{\mathrm{e1}}}{dt} &= T_{\mathrm{c}}\frac{d}{dt}(V_{\mathrm{s},t} + V_{\mathrm{r},t} + s_{\mathrm{e0}} - s_{\mathrm{e4}}) - s_{\mathrm{e0},t-1} - s_{\mathrm{e3},t-1} - s_{\mathrm{e1},t-1},\\
T_{\mathrm{A}}\frac{ds_{\mathrm{e3}}}{dt} &= K_{\mathrm{A}}s_{\mathrm{e2},t-1} - s_{\mathrm{e3},t-1},\\
E_{\mathrm{fd},t} &= s_{\mathrm{e3},t-1} + V_{\mathrm{s},t} + K_{\mathrm{lr}}(I_{\mathrm{lr}} - I_{\mathrm{fd},t}),\\
T_{\mathrm{f}}\frac{ds_{\mathrm{e4}}}{dt} &= K_{\mathrm{f}}\frac{dE_{\mathrm{fd}}}{dt} - s_{\mathrm{e4},t-1},
\end{aligned}
$$

where $s_{\mathrm{e0}}, s_{\mathrm{e1}}, s_{\mathrm{e2}}, s_{\mathrm{e3}}, s_{\mathrm{e4}}$ are state variables. V_{r} and V_{s} are the voltage reference and control input signal, respectively. T_{c}, T_{c1}, T_{b}, and T_{b1} are exciter time constants, K_{A} is AVR steady state gain. More details of the exciter model and its parameters can be found in [2, 3].

Event Playback

Let $\boldsymbol{z}^* = [\boldsymbol{P}_{\mathrm{meas}}^\top \ \boldsymbol{Q}_{\mathrm{meas}}^\top]^\top$ be the measurements from PMU and $\boldsymbol{z} = [\boldsymbol{P}_{\mathrm{model}}^\top \ \boldsymbol{Q}_{\mathrm{model}}^\top]^\top \in \mathbb{R}^{2K}$ be the time-series outputs of model (3.1) where K is the number of time steps. In generator model validation and parameter calibration, "event playback" illustrated in Figure 3.1 utilizes a dynamic simulation with play-in signals to check the model performance against the actual PMU measurements [19]. The PMU is installed at the subsystem point of common coupling (PCC) to collect measurements of voltage magnitude and phase angle/frequency, and current phasors of the branches connected to the PCC. Real and reactive power measurements can be obtained from the voltage phasor and current phasor measurements. Voltage magnitude and phase angle/frequency are used as inputs while real and reactive powers are the outputs. The external system is modeled as a regulated voltage source and its voltage is equal to the quantity measured by PMU [19, 39].

For an event captured by PMU, if large model deficiency is identified between model simulation and PMU measurements, the parameters need to be calibrated. In real application, different events can lead to different system responses. For the additional events, if the simulated response and PMU measurements are significantly different, the calibration procedure should be repeated, or multiple events can be used together for parameter calibration [49].

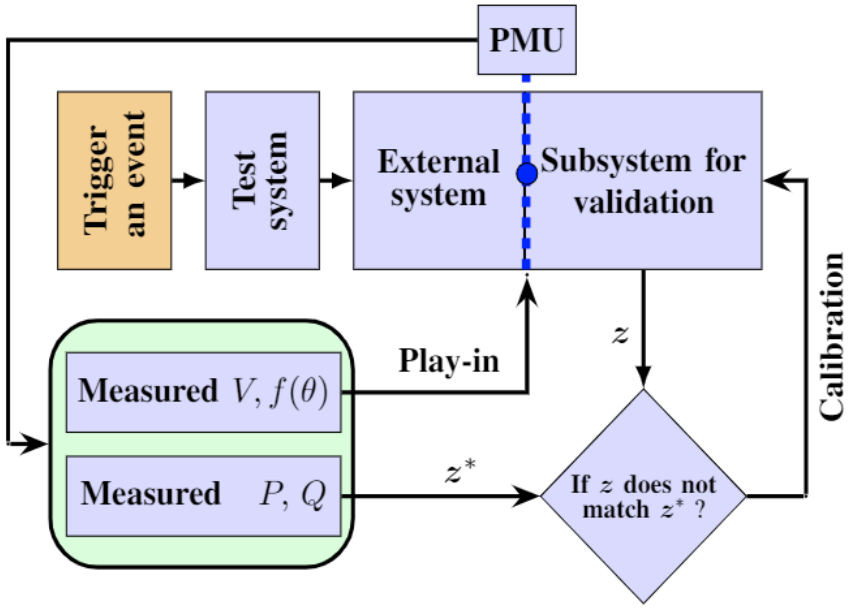


Figure 3.1: Framework of the playback event for model validation and calibration.

At time step k the output of the model is given by:

$$P_{\text{model},k} = \frac{E''_k V_k}{X'_{\text{d}} + X_{\text{tr}}} \sin(\delta_k - \theta_k) \tag{3.2}$$

$$Q_{\text{model},k} = \frac{V_k^2 - E''_k V_k \cos(\delta_k - \theta_k)}{X'_{\text{d}} + X_{\text{tr}}}, \tag{3.3}$$

where E''_k and δ_k are, respectively, the generator sub-transient voltage and rotor angle at time step k, X'_{d} is the d-axis transient reactance, X_{tr} is the reactance of the step-up transformer, and V_k and θ_k are the voltage magnitude and phase angle at high-voltage side of the step-up transformer at

time step k.

Identifying Critical Parameters

After a model deficiency has been revealed, the next step is to identify the problematic parameters. A generating unit with its control can have many parameters. Calibrating all parameters could be computationally challenging and also not every parameter is identifiable. Trajectory sensitivity has been used to identify the most critical parameters [39]. Based on trajectory sensitivities, we can have insights about how the changes in parameters influence the system response. If a parameter exerts a large influence on the response, the corresponding sensitivity will be large.

Specifically, the sensitivity of the outputs P and Q with regard to parameter α_i can be calculated as [19]:

$$S(\alpha_i) = \sum_{k=1}^{K} \frac{|P_k(\alpha_i^+) - P_k(\alpha_i^-)| + |Q_k(\alpha_i^+) - Q_k(\alpha_i^-)|}{K(\alpha_i^+ - \alpha_i^-)/\alpha_i},$$

where K is the number of time steps, $\alpha_i^+ = \alpha_i + \Delta\alpha_i$ and $\alpha_i^- = \alpha_i - \Delta\alpha_i$, and $\Delta\alpha_i$ is a small perturbation of α_i. After the sensitivity analysis, the parameters selected to be estimated are those with a large sensitivity [19, 39].

Black-Box Optimization for Parameter Estimation

A single-objective optimization problem for parameter estimation can be written in a general form as:

$$\min g(\boldsymbol{\alpha}_{\mathbf{c}}) \tag{3.4a}$$

$$\text{s.t. } \boldsymbol{\alpha}_{\mathrm{c}} \in \chi, \tag{3.4b}$$

where $\boldsymbol{\alpha}_{\mathrm{c}}$ is the critical parameter vector to be estimated and $\chi = \{\boldsymbol{\alpha}_{\mathrm{c}} \in \mathbb{R}^m \cap [\boldsymbol{\alpha}_{\mathrm{c}}^{\mathrm{L}}, \boldsymbol{\alpha}_{\mathrm{c}}^{\mathrm{U}}]\}$. Here m is the number of critical parameters to be estimated and $\boldsymbol{\alpha}_{\mathrm{c}}^{\mathrm{L}}$ and $\boldsymbol{\alpha}_{\mathrm{c}}^{\mathrm{U}}$ are the lower and upper bounds of the predefined prior distribution for $\boldsymbol{\alpha}_{\mathrm{c}}$. When the analytical expression of the objective function is not known, the problem is a black-box optimization problem. The black-box optimization framework can be applied to the problem of generator parameter calibration. In our particular problem, $\boldsymbol{\alpha}_{\mathrm{c}}$ is the vector of parameters of a generator model which can be defined within a reasonable range.

Many approaches have been proposed for solving black-box optimization. Some methods rely on metaheuristics, such as genetic algorithms [67], particle swarm [67], and simulated annealing [68]. However, these methods suffer from slow convergence in many applications [44]. Other methods based on direct search or surrogate work faster than metaheuristics methods [45]. For approximating a function for g, several methods have been proposed. Trust regions method builds local models for function g [69]. In contrast, Radial Basis Function (RBF) method [13] and stochastic RBF method [70] build a global mode of the function g [44].

Objective Function

By comparing $\boldsymbol{z}^*$ with $\boldsymbol{z}$, the parameters of the model can be estimated. In this CHAPTER , we choose the following L_1 norm based objective function:

$$\rho(\boldsymbol{z}, \boldsymbol{z}^*) = \frac{1}{2K}\|\boldsymbol{z} - \boldsymbol{z}^*\|_1, \tag{3.5}$$

where $\|\cdot\|_1$ is the 1-norm of a vector.

Radial Basis Function Model

Given k distinct points $\boldsymbol{\alpha}_{\mathrm{c}}^1, \cdots, \boldsymbol{\alpha}_{\mathrm{c}}^k$ where their function values are known as $g^1, \cdots, g^k$, in RBF approximation we seek a function s_k as follow to approximate g [13]

$$s_k(\boldsymbol{\alpha}_{\mathrm{c}}) = \sum_{i=1}^{k} \lambda_i \phi(\|\boldsymbol{\alpha}_{\mathrm{c}} - \boldsymbol{\alpha}_{\mathrm{c}}^i\|) + p(\boldsymbol{\alpha}_{\mathbf{c}}), \tag{3.6}$$

where $\|\cdot\|$ is the Euclidean norm, $\boldsymbol{\lambda} \in \mathbb{R}^k$, and $p(\boldsymbol{\alpha}_{\mathbf{c}})$ is a polynomial of degree d. The polynomial guarantees existence and uniqueness of an interpolant of the form (3.6), and its minimum degree depends on the type of RBF ϕ [44]. Table 3.1 lists the common RBFs $\phi(r)$ and the corresponding $d_{\min}$. The parameter $\gamma > 0$ for multiquadric function can change the shape of the function, but is usually set to be 1 [45, 71].

Algorithm for Solving the Black-Box Optimization Problem

1. Choosing Initial Points: Choose Initial Points: To guarantee existence of a unique interpolant of the form (3.6), the algorithm needs at least $m + 1$ distinct interpolation points. Different methods have been proposed to choose the initial points. One method is to pick the 2m

Table 3.1: Common RBFs [13]

RBF	$\phi(r)$	$d_{\min}$
linear	r	0
cubic	r^3	1
multiquadric	$\sqrt{r^2+\gamma^2}$	0
thin-plate spline	$r^2\log(r)$	1

points of the corner of the parameter space [72]. In [13] the m+1 points of the corner of the parameter space are chosen. However, these methods only work for a small parameter space. Another strategy is to use Latin Hypercube experimental design [73]. In this method, different samples are built and the one that maximizes the minimum Euclidean distance between the sample points is selected. In [44], fifty Latin Hypercube designs are built and the one that has the maximum of minimum Euclidean distance is used for choosing the m+1 initial points. The chosen $m+1$ points comprise a set $\mathcal{S}$.

2. Determine the Next Search Points: After choosing the initial points, in every iteration the algorithm fits a surrogate model for g that interpolates the points in $\mathcal{S}$ and chooses the next evaluation point $\boldsymbol{\alpha_c}$ according to two criteria. The first one is to minimize the Euclidean distance from $\boldsymbol{\alpha_c}$ to the points in $\mathcal{S}$. The second one is to minimize the value of the surrogate model at $\boldsymbol{\alpha_c}$. In other words, the following two objective functions need to be minimized:

$$h_1(\boldsymbol{\alpha_c}) = \min_{i=1,\cdots,k}\|\boldsymbol{\alpha_c} - \boldsymbol{\alpha_c^i}\| \tag{3.7}$$

$$h_2(\boldsymbol{\alpha_c}) = s_k(\boldsymbol{\alpha_c}), \tag{3.8}$$

where k is the cardinality of the $\mathcal{S}$ in the current iteration. During implementation, the algorithm should make a tradeoff between exploration and exploitation. Exploration implies trying to improve the surrogate model in the unknown parts of the parameter space, whereas exploitation implies trying to find the best objective function value based on the current surrogate model [20]. The algorithm uses a weight $\beta \in [0, 1]$ that made a trade-off between h_1 and h_2 and convert two objectives optimization problem to a single objective function. For each point of the population P, a score is calculated based on the metric defined below [44]:

$$\beta \frac{h_1(\boldsymbol{\alpha_c}) - \min_{\tau \in P} h_1(\tau)}{\max_{\tau \in P} h_1(\tau) - \min_{\tau \in P} h_1(\tau)} + \frac{h_2(\boldsymbol{\alpha_c}) - \min_{\tau \in P} h_2(\tau)}{\max_{\tau \in P} h_2(\tau) - \min_{\tau \in P} h_2(\tau)}. \tag{3.9}$$

The points in the population with a larger score retained and randomly mutated [44]. When the point $\boldsymbol{\alpha_c}$ is determined, the function g is evaluated at $\boldsymbol{\alpha_c}$, the point is added to $\mathcal{S}$, and the iteration is complete [74]. More details about the implementation of the trade-off between the exploration and exploitation and its implementation can be found in [44, 45].

3. Algorithm: By defining the objective function and the prior of parameters, the Metric Stochastic Response Surface Method (MSRSM) [44] is presented as Algorithm 1.

Simulation Results

We implement our method based on PSS/E and Python 2.7 and test it on the system used in [75]. All tests are carried out on a PC with Intel(R) Core(TM) i7-8700 and 8 GB RAM. A PMU is installed at the 230-kV level of the substation. The sampling rate of the PMU is 30 sample/s. We play-in voltage and frequency signals to the network and use the active and reactive power of network as PMU measurements.

Algorithm 1: MSRSM algorithm for estimating parameters and T is the maximum number of iteration [13, 44].

Choose $m+1$ independent points $\boldsymbol{\alpha}_{\mathrm{c}}^{1}, \cdots, \boldsymbol{\alpha}_{\mathrm{c}}^{m+1}$ based on Section IV-C1.
$\mathcal{S} \leftarrow \{(\boldsymbol{\alpha}_{\mathbf{c}}^{\mathbf{1}}, g(\boldsymbol{\alpha}_{\mathrm{c}}^{1})), \cdots, (\boldsymbol{\alpha}_{\mathrm{c}}^{m+1}, g(\boldsymbol{\alpha}_{\mathrm{c}}^{m+1}))\}$
for $1 \leq i \leq T$ **do**
    Compute the RBF interpolant s_k to the points in $\mathcal{S}$ according to IV-C2.
    Determine the trade-off between exploration and exploitation with $\beta \in [0, 1]$ in Section IV-C2.
    Find a point $\boldsymbol{\alpha}_{\mathrm{c}}^{k+1}$ that satisfies (3.7)–(3.8) and using the weight β to balance these two criteria.
    Evaluate g at $\boldsymbol{\alpha}_{\mathrm{c}}^{k+1}$ to obtain $g(\boldsymbol{\alpha}_{\mathrm{c}}^{k+1})$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(\boldsymbol{\beta}_{\mathbf{c}}^{\boldsymbol{k+1}}, g(\boldsymbol{\alpha}_{\mathrm{c}}^{k+1}))\}$
end for
Return the best solution among those in $\mathcal{S}$

Critical Parameter Identification

From the trajectory sensitivity in Section 3, it is found that among 16 machine parameters only three parameters, d-axis transient rotor time constant (T_{pdo}), inertia constant (H), and d-axis synchronous reactance (X_{d}) have considerable effects on the model's response. For the exciter with 18 parameters, only three parameters, exciter gain (K_{A}), and time constants of the exciter (T_{c} and T_{b}) and for the governor with 14 parameters, only two parameters, turbine coefficient (a_{23}) and dashpot time constant (T_{R}) have considerable impacts on the model's response. The top eight parameters and their sensitivities are listed in Table 3.2.

Table 3.2: Sensitivity Analysis of the Parameters

Parameter	**Sensitivity**
K_{A}	1.75
T_{b}	1.35
a_{23}	1.13
T_{pdo}	1.11
X_{d}	0.85
T_{c}	0.67
H	0.62
T_{R}	0.57

Setup of the Proposed Method

In addition to the objective function in (3.5), we also consider the following two objective functions:

$$g_1(\boldsymbol{z}^*, \boldsymbol{z}) = \frac{1}{2K} \sum_{k=1}^{2K} (z_k - z_k^*)^2 \tag{3.10}$$

$$g_2(\boldsymbol{z}^*, \boldsymbol{z}) = \sqrt{\frac{1}{2K} \sum_{k=1}^{2K} (z_k - z_k^*)^2}. \tag{3.11}$$

In Table 3.3 we compare the number of function evaluations which are needed to reach the estimated values under different objective functions (g_1 and g_2). It can be seen that the chosen objective function in (3.5) needs fewer number of simulations. Therefore, throughout this CHAPTER we choose (3.5) as the objective function.

Based on [45], thin-plate spline performance is better than the other types of RBFs, then we use

Table 3.3: Number of Function Evaluation which Required to Find the Estimated Parameter for Different Objective Functions

Objective function	g_1 [76]	g_2 [6]	g in (3.5)
Number of simulations	785	980	675

the thin-plate as RBF. We use the RBFOpt [45] and its parameters are left to their default values. The details of the choosing β can be found at [45]. The maximum number of iterations is 200 and 800 for two- and eight-parameter, respectively.

Validation under Different Prior Distributions

The estimated parameters of the black box optimization are determined by the objective function and the prior distribution. For the practice implementation, the prior distribution of the parameters can be chosen using the data provided by the manufacturer, which is considered to be reasonable in general. In this section, we analyze the estimation accuracy under different prior information. We consider two parameters (the moment of inertia H for synchronous generator and the amplifier gain K_{A} for the exciter) with their original values as $H = 5.4$ and $K_{\mathrm{A}} = 125$. We choose the prior distributions as uniform distribution with lower bound as 0 and upper bound as 100% and 300 % greater than the mean values to account for parameter uncertainties. The results of the identification for the the different prior distributions are shown in Table 3.4. It can be seen that proposed method can provide accurate estimation of parameters with different prior distributions.

Table 3.4: Parameter Calibration Under Different Prior Distributions

H ($H_{\text{True}} = 5.40$)		K_{A} ($K_{\text{A}_{\text{True}}} = 125$)	
Prior	**Estimated** (Error (%))	**Prior**	**Estimated** (Error (%))
$\mathcal{U}(0\ ,\ 10.8)$	5.40 (0)	$\mathcal{U}(0,\ \ 250)$	125 (0)
$\mathcal{U}(0,\ 16.2)$	5.404 (0.08)	$\mathcal{U}(0,\ 375)$	124.95 (0.04)

Validation for High Dimension

In this section, we are demonstrating the performance of the proposed method with eight key parameters that are known to influence the dynamic response of the system and can not be directly measured easily. These are the gains in the exciter K_{A}, time constant of exciter (T_{b} and T_{c}), turbine coefficient (a_{23}), d-axis transient rotor time constant (T_{pdo}), moment of inertia (H), d-axis synchronous reactance (X_{d}), and dashpot time constant (T_{R}). We assumed the parameters have 10% greater than their true values. The prior of these parameters are assumed to follow uniform distributions. We choose the lower/upper bounds of the uniform prior distributions as 30% mean values. Table 4.7 lists the estimated values. It can be seen the algorithm has good accuracy under high dimension. Figure 3.2 shows the model validation results for real and reactive power for the true and estimated parameters.

Validation Performance Under the Tolerance

Here we show the calibration in the presence of different deviations. We consider two parameters $\{H_{\text{True}} = 5.4, K_{A_{\text{True}}} = 125\}$. We consider the priors as a uniform distribution with considering the true parameters at tail regions of the assumed prior distributions. As presented in Table 3.6,

Table 3.5: Calibration of Eight Key Parameters

Parameter	**True value**	**Estimated value**	**Error (%)**
K_{A}	125	125.47	0.4
T_{b}	3.86	3.84	0.5
a_{23}	1.102	1.110	0.7
T_{pdo}	5.40	5.45	0.9
X_{d}	0.57	0.575	0.6
T_{c}	0.90	0.90	0
H	5.40	5.38	0.4
T_{R}	2.40	2.43	1.5

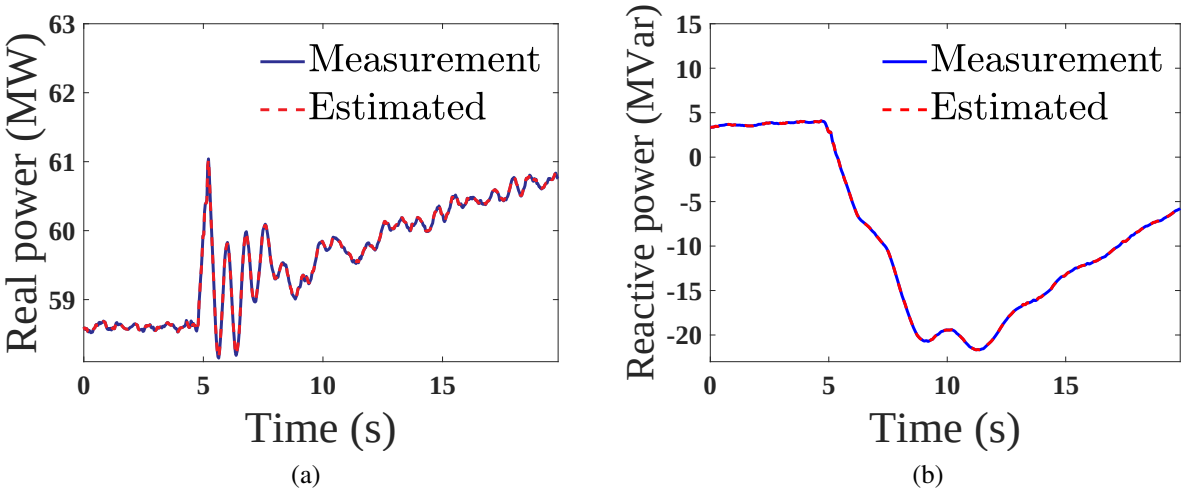


Figure 3.2: Measurement and model's output for the the model with the estimated parameters (a) Real power and (b) Reactive power.

even when the deviation is large, the proposed approach has a good accuracy.

For the highlighted first row in Table 3.6, the prior, the estimated and true values of the parameters are shown in Figure 3.3. It is seen that even when the true values are at the tail regions of the assumed prior distributions, the proposed algorithm still has very good accuracy.

Table 3.6: Parameter Calibration Under Different Deviations

H ($H_{\text{True}} = 5.40$)		K_{A} ($K_{\text{A}_{\text{True}}} = 125$)	
Prior	**Estimated** (Error (%))	**Prior**	**Estimated** (Error (%))
U(0 , 5.41)	**5.39 (0.1)**	**U(0, 125.01)**	**125 (0)**
$\mathcal{U}(5.39, 10.8)$	5.40 (0)	$\mathcal{U}(124.99, 250)$	125 (0)

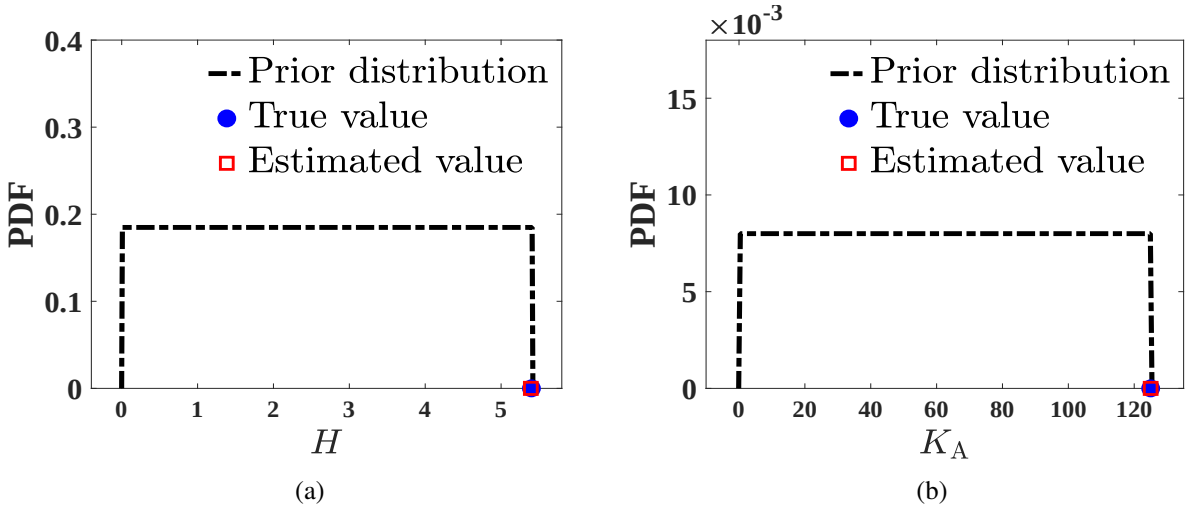


Figure 3.3: Prior distribution, true and estimated values for (a) H and (b) K_{A}.

Time Efficiency

The synchronous generator model used in this CHAPTER has 36 parameters. Running simulations for the cases of two and eight keys parameters take around 30 seconds and 10 minutes, respectively.

Conclusion

This CHAPTER proposes a PMU measurement based generator parameter calibration approach by black-box optimization with a stochastic radial basis function surrogate model. The proposed algorithm is applied to calibrate the decentralized dynamic model of a generator. The simulation results show that the proposed method can accurately and efficiently estimate the parameters. Even when the true values are at the tail regions of the assumed prior distributions, the proposed algorithm can still very accurately estimate the parameters. In our future work we will further improve the proposed method as follows.

- In this CHAPTER critical parameters are identified based on trajectory sensitivity. Since the generator model is highly nonlinear, a better approach to quantify the parameter-output behavior from the perspective of nonlinear systems and further identify critical parameters will be developed.
- Since the generator models used in the industry could have a large number of parameters, in our future work we will further improve the time efficiency of the proposed method for high dimensional cases.
- Improving the robustness and estimation accuracy of the proposed method when there exists an extremely large deviation between the true parameters and the prior information will also be studied in the future.

CHAPTER 4: GENERATOR PARAMETER CALIBRATION BY A-ABC-SMC

S. R. Khazeiynasab and J. Qi, "Generator Parameter Calibration by Adaptive Approximate Bayesian Computation With Sequential Monte Carlo Sampler," in IEEE Transactions on Smart Grid, vol. 12, no. 5, pp. 4327-4338, Sept. 2021, doi: 10.1109/TSG.2021.3077734.

Introduction

Secure power system operation relies on accurate steady-state and dynamic system models. It is thus crucial to carefully validate the models in power systems, in particular the generator models. The phasor measurement unit (PMU) technologies provide a low-cost option for generator model validation and parameter calibration without interfering with their operation. In this CHAPTER , we propose a synchrophasor measurement based generator parameter calibration method by A-ABC-SMC that avoids directly dealing with likelihood functions. An empirical parameter sensitivity Gramian based approach is developed to identify the critical parameters from a nonlinear system perspective. We propose an ABC SMC approach for generator parameter calibration with an adaptive threshold sequence scheme and perturbation kernel function for high computational efficiency. The effectiveness of the proposed method is validated for a hydro generator against multiple system events. The results show that the proposed approach can accurately and efficiently estimate the full probabilistic posterior distributions of the generator parameters.

In this CHAPTER we focused on the Bayesian-based method. The Bayesian inference methods in [41–43] still need to evaluate the likelihood function of the parameters. By contrast, Approximate Bayesian Computation (ABC) has been proposed to infer posterior distributions where like-

lihood functions are computationally intractable or too costly to evaluate [12]. ABC exploits the computational efficiency of modern simulation techniques by replacing the calculation of the likelihood function with a comparison between the observed and the simulated data. The simplest ABC algorithm is ABC rejection sampler [77], which, however, could be very inefficient. Many techniques have been proposed to improve the efficiency of ABC, such as ABC based on MCMC (ABC MCMC) [78] and ABC with sequential Monte Carlo sampler (ABC SMC) [4, 5]. For ABC MCMC the chain could be very long and a chain may get stuck in regions of low probability for a very long time [79], which can be avoided by ABC SMC [4].

In this CHAPTER we propose an adaptive ABC SMC (A-ABC-SMC) based generator parameter calibration approach and improve the efficiency of ABC SMC by developing systematic methods to adaptively choose the threshold sequence and the perturbation kernel function. The contributions of this CHAPTER are summarized as follows.

1. Instead of using a sensitivity-based approach, an empirical Gramian based approach is proposed directly for the nonlinear generator model to more accurately identify the critical parameters with the highest identifiability.

2. We perform generator parameter calibration by A-ABC-SMC, a likelihood-free method that does not directly explore the likelihood surface of the parameters but instead estimates the posterior distributions of the parameters by a simulation-based procedure.

3. The proposed A-ABC-SMC approach significantly improves the computational efficiency of ABC SMC through adaptive threshold sequences and perturbation kernel function, and carefully chosen distance function.

Generator Dynamic Model

The nonlinear dynamical model of a synchronous generator and its controls includes a generator (GENTPJ), an exciter (ESST1A), a governor (IEEEG3), and a power system stabilizer (PSS, IEEEST) can be written as (3.1). The block diagram of the generator, exciter, governor, and PSS are shown in Figs. 4.1–4.4.

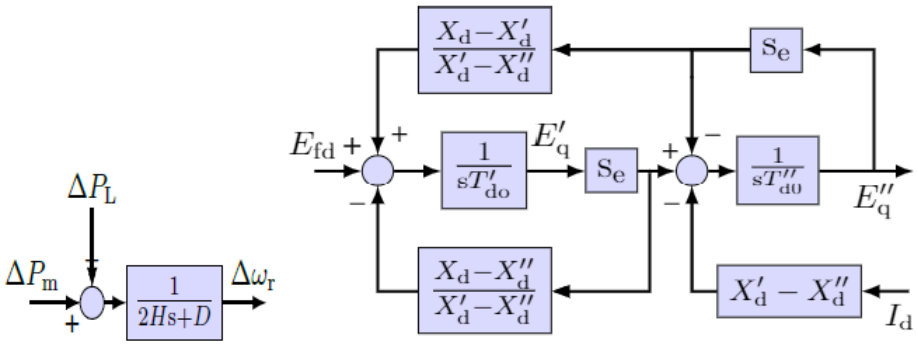


Figure 4.1: GENTPJ generator model. H is inertia constant, D is damping factor, T'_{do} is d-axis transient rotor time constant, T''_{do} is d-axis sub-transient rotor time constant, X_d is d-axis synchronous reactance, X'_d is d-axis transient reactance, X''_d is d-axis sub-transient reactance, s_e is saturation factor, E_{fd} is excitation voltage, E'_q is q-axis transient voltage, and E''_q is q-axis sub-transient voltage [2, 3].

In model validation, the time-series real and reactive powers from the model, denoted by vectors $\boldsymbol{P}_{\text{model}}$ and $\boldsymbol{Q}_{\text{model}}$, are compared with the time-series real and reactive power measurement vectors, denoted by $\boldsymbol{P}_{\text{meas}}$ and $\boldsymbol{Q}_{\text{meas}}$. If the outputs from the model do not match the PMU-measured outputs, the generator parameters need to be calibrated [19].

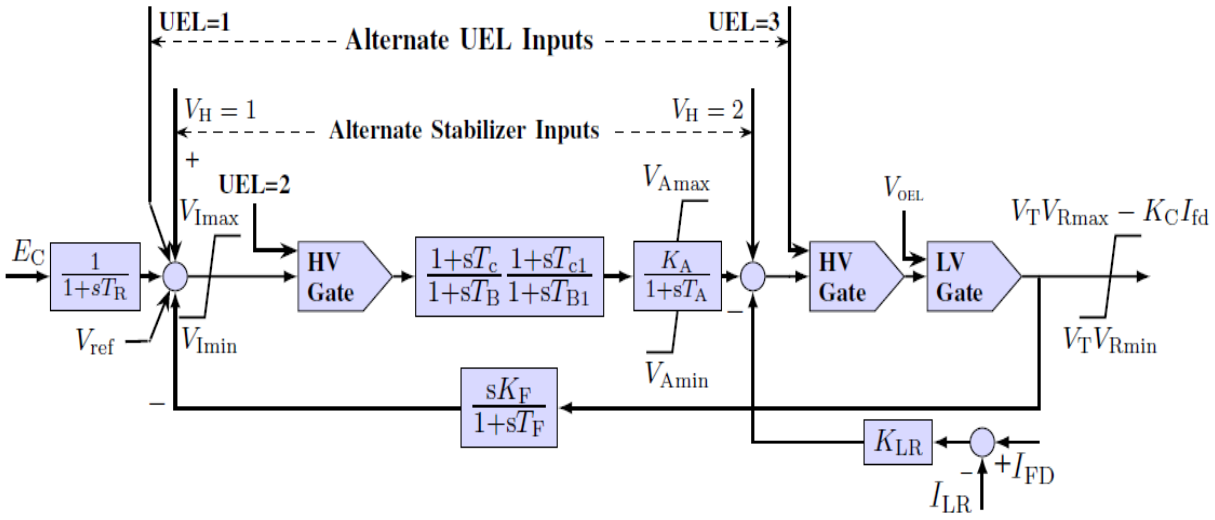


Figure 4.2: ESST1A exciter model. T_R is voltage transducer time constant, V_{Imax} is maximum voltage error, V_{Imin} is minimum voltage error, T_C, T_B, T_{C1}, T_{B1} are time constants, K_A is AVR steady state gain, T_A is rectifier bridge equivalent time constant, V_{Amax} is maximum of AVR output, V_{Amin} is minimum of AVR output, V_{Rmax} is maximum rectifier bridge output, V_{Rmin} is minimum rectifier bridge output, K_C is commutation factor for rectifier bridge, K_F is stabilizer feedback gain, T_F is stabilizer feedback time constant, K_{LR} is field current limiter gain, I_{LR} is field current instantaneous limit, and I_{fd} is excitation current [2].

Identifiability of Parameters

A synchronous generator has many parameters. Calibrating all parameters could be computationally challenging and also not every parameter is identifiable. Therefore, the parameters with highest identifiability should be first identified [19, 28].

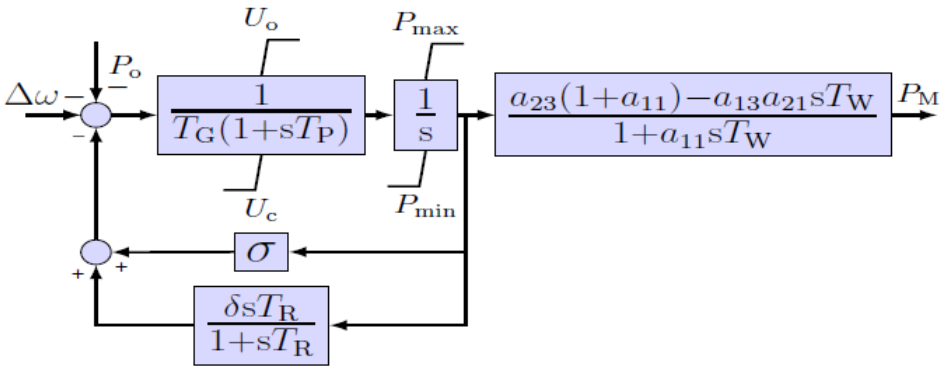


Figure 4.3: IEEEG3 governor model. T_{G} is gate servo time constant, T_{P} is pilot servo valve time constant, U_{o} is opening gate rate limit, U_{c} is closing gate rate limit, $P_{\max}$ is maximum gate position, $P_{\min}$ is minimum gate position, σ is permanent speed droop coefficient, δ is transient speed droop coefficient, ω is velocity of the rotor, a_{11}, a_{13}, a_{21}, a_{23} are turbine coefficients, T_{W} is water starting time, and T_{R} is governor time constant [2].

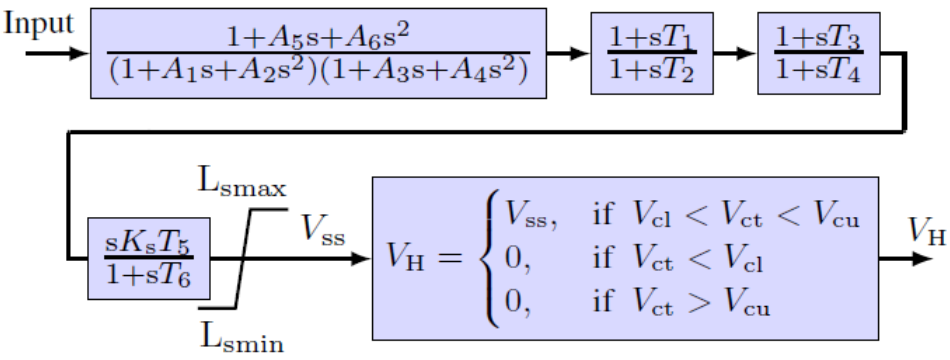


Figure 4.4: IEEEST PSS model. $A_1, A_2, A_3, A_4, A_5, A_6$ are filter coefficients, $T_1, T_2, T_3, T_4, T_5, T_6$ are lead/lag time constants, K_{s} is PSS gain, L_{smax} is PSS output maximum limit, L_{smin} is PSS output minimum limit, and V_{cl} and V_{cu} are voltage cutoff limiters [2].

Sensitivity Based Approach

In [19], the sensitivity of P and Q with regard to parameter α_i can be calculated to help identify critical parameters:

$$S(\alpha_i) = \sum_{k=1}^{K} \frac{|P_k(\alpha_i^+) - P_k(\alpha_i^-)| + |Q_k(\alpha_i^+) - Q_k(\alpha_i^-)|}{K(\alpha_i^+ - \alpha_i^-)/\alpha_i},$$

where K is the number of time steps, $\alpha_i^+ = \alpha_i + \Delta\alpha_i$ and $\alpha_i^- = \alpha_i - \Delta\alpha_i$, and $\Delta\alpha_i$ is a small perturbation of α_i. The parameters with the largest sensitivities will be considered as having the highest identifiability [19].

Empirical Gramian Based Approach

The sensitivity in Section 4 is only locally defined for one operating point and the nonlinear behavior of the generator model is inevitably lost. Here we adapt the empirical Gramian approach [80,81] to analyze the sensitivity of the outputs to parameters. Specifically, we define an empirical parameter sensitivity Gramian (EPSG) which perturbs the parameters to reveal the parameter-output behavior [82]. Note that EPSG is defined for a reasonable region of the parameters directly for the original nonlinear dynamical model and can thus better reflect the identifiability of the parameters [81]. The following sets are defined for EPSG:

$$T = \{\boldsymbol{T}_1, \cdots, \boldsymbol{T}_r;\, \boldsymbol{T}_j \in \mathbb{R}^{v\times v},\, {\boldsymbol{T}_j}^\top \boldsymbol{T}_j = \boldsymbol{I}_v,\, j = 1, \ldots, r\}$$

$$M = \{c_1, \cdots, c_s;\, c_m \in \mathbb{R},\, c_m > 0,\, m = 1, \ldots, s\}$$

$$E = \{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_v;\, \text{standard unit vectors in}\, \mathbb{R}^v\},$$

where T defines the initial parameter perturbation direction, r is the number of matrices for perturbation directions, M defines the perturbation sizes, s is the number of perturbation sizes for each direction, $\boldsymbol{I}_v$ is an identity matrix with dimension v, and E defines the parameter to be perturbed.

For the ith parameter, α_i, in the nonlinear system (3.1), with fixed initial states $\boldsymbol{x}_0$, EPSG can be defined as

$$\boldsymbol{W}^{\text{con}}(\alpha_i) = \sum_{j=1}^{r}\sum_{m=1}^{s} \frac{1}{rsc_m^2} \int_0^\infty \boldsymbol{\Phi}^{jm}(t)\, dt, \tag{4.1}$$

where $\boldsymbol{\Phi}^{jm}(t) \in \mathbb{R}^{o\times o}$ is given as $\boldsymbol{\Phi}^{jm}(t) = (\boldsymbol{y}^{jm}(t) - \boldsymbol{y}_0^{jm})(\boldsymbol{y}^{jm}(t) - \boldsymbol{y}_0^{jm})^\top$, $\boldsymbol{y}_0^{jm}$ refers to the

outputs corresponding to the unperturbed initial parameter $\boldsymbol{\alpha}_0$, and $\boldsymbol{y}^{jm}(t)$ is the output of the nonlinear system under parameter $\boldsymbol{\alpha}^{jm} = \boldsymbol{\alpha}_0(1 + c_m \boldsymbol{T}_j \boldsymbol{e}_i)$.

The discrete form of EPSG can be defined as [81]

$$\boldsymbol{W}(\alpha_i) = \sum_{j=1}^{r} \sum_{m=1}^{s} \frac{1}{rsc_m^2} \sum_{k=1}^{K} \boldsymbol{\Phi}_k^{jm} \Delta t_k, \tag{4.2}$$

where $\boldsymbol{\Phi}_k^{jm} \in \mathbb{R}^{o \times o}$ is given by $\boldsymbol{\Phi}_k^{jm} = (\boldsymbol{y}_k^{jm} - \boldsymbol{y}_0^{jm})(\boldsymbol{y}_k^{jm} - \boldsymbol{y}_0^{jm})^\top$, $\boldsymbol{y}_k^{jm}$ is the output of the nonlinear system at time step k corresponding to the parameter $\boldsymbol{\alpha}^{jm} = \boldsymbol{\alpha}_0(1 + c_m \boldsymbol{T}_j \boldsymbol{e}_i)$, and Δt_k is the time step size. We choose $\Delta t_k = 0.033$ s since the reporting rate of PMU measurement is assumed to be 30 samples/s. T is chosen as

$$T = \{\boldsymbol{I}_v, -\boldsymbol{I}_v\} \tag{4.3}$$

so that we perturb each parameter in both positive and negative directions. We choose a linearly scaled set $M = \{0.25, 0.5, 0.75\}$ for the perturbation sizes.

The identifiability of the ith parameter can be indicated by the trace of $\boldsymbol{W}(\alpha_i)$, denoted by $\mathrm{tr}\big(\boldsymbol{W}(\alpha_i)\big)$. Based on our numerical experiments, trace as a metric that measures the overall identifiability of a parameter is a good choice among several available metrics [83, 84].

Note that some parameters do not change much during the generator lifetime based on engineering experience. If that is the case we can either choose not to calibrate those parameters even if they are identified as critical parameters or we can utilize this information to help properly assign prior distributions for those parameters (for uniform prior distribution we can reduce the width of the interval and for normal prior distribution we can reduce the standard deviation).

ABC for Parameter Estimation

In many areas there is either no closed-form likelihood function or it is computationally too expensive to calculate such a function [79]. Particularly for the parameter calibration problem, the computational expense for having explicit likelihood function is the major challenge. For this reason, some papers such as [41,42] approximate the likelihood function by using a polynomial chaos expansion (PCE)-based response surface to represent system response. However, when the number of parameters increases, the number of terms involved in PCE will also increase which will reduce the efficiency.

A more straightforward approach is to perform simulations for the model using different parameters, compare the simulated results with the observed data, and estimate the likelihood of a given parameter set to generate outputs that match the observed data. However, in real applications the probability of an exact match is very low [79]. ABC based approaches provide a framework to generate approximations to the true posterior distribution by systematic comparisons between real and simulated data instead of directly evaluating the likelihood function [5, 10].

ABC SMC

Let $\boldsymbol{z}^*$ be the measurements and $\boldsymbol{z}(\boldsymbol{\alpha}_\mathrm{c})$ the outputs of the model where $\boldsymbol{\alpha}_\mathrm{c} \in \mathbb{R}^{v'}$ is the vector of the v' critical parameters to be estimated. Assuming the prior distribution for $\boldsymbol{\alpha}_\mathrm{c}$ as $\pi(\boldsymbol{\alpha}_\mathrm{c})$, Bayes's theorem allows us to write its posterior distribution $p(\boldsymbol{\alpha}_\mathrm{c}|\boldsymbol{z}^*)$ in terms of the prior distribution and the likelihood of the observed data $\boldsymbol{z}^*$ for given parameter $\boldsymbol{\alpha}_\mathrm{c}$, $l(\boldsymbol{z}^*|\boldsymbol{\alpha}_\mathrm{c})$, as [10, 85]

$$p(\boldsymbol{\alpha}_\mathrm{c}|\boldsymbol{z}^*) \propto l(\boldsymbol{z}^*|\boldsymbol{\alpha}_\mathrm{c})\pi(\boldsymbol{\alpha}_\mathrm{c}). \tag{4.4}$$

The posterior distribution represents the most updated information/knowledge regarding the parameters given the available data. ABC compares the simulated data $\boldsymbol{z}$ with the real data $\boldsymbol{z}^*$ and accepts only the simulations for which the distance between $\boldsymbol{z}^*$ and $\boldsymbol{z}$, $\rho(\boldsymbol{z}, \boldsymbol{z}^*)$, is less than a predefined tolerance ϵ. The goal is to determine the approximate posteriors as [86]

$$p(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z}^*) \approx p_\epsilon(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z}^*) \propto \int_{\mathbb{R}^{2K}} l(\boldsymbol{z}|\boldsymbol{\alpha}_{\mathrm{c}}) \mathbb{1}\big(\rho(\boldsymbol{z}, \boldsymbol{z}^*) \leq \epsilon\big) \pi(\boldsymbol{\alpha}_{\mathrm{c}}) d\boldsymbol{z}, \tag{4.5}$$

where $p_\epsilon(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z}^*)$ is an approximation of the posterior $p(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z}^*)$, $l(\boldsymbol{z}|\boldsymbol{\alpha}_{\mathrm{c}})$ is the probability density of data $\boldsymbol{z}$ given model parameters $\boldsymbol{\alpha}_{\mathrm{c}}$, and $\mathbb{1}(x)$ is equal to one when condition x is true.

If ϵ is sufficiently small, the distribution $p_\epsilon(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z}^*)$ will be a good approximation of the posterior distribution. ABC SMC samples from a sequence of distributions that increasingly resemble the target posterior. They are constructed by estimating the intermediate distributions $p_{\epsilon_t}(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z})$ for a decreasing sequence of $\{\epsilon_t\}_{1 \leq t \leq N_T}$ where N_T is the maximum number of iterations [87]. The algorithm first generates an initial pool of N particles that satisfy $\rho(\boldsymbol{z}, \boldsymbol{z}^*) \leq \epsilon_1$ by randomly sampling from the prior $\pi(\boldsymbol{\alpha}_{\mathrm{c}})$ and all particles are assigned an equal weight as $1/N$. In the following iterations, successive distributions are randomly constructed by sampling from the previous population with probabilities $\{w^{(i,t-1)}\}_{\{1 \leq i \leq N\}}$ where $w^{(i,t-1)}$ is the weight for the ith particle in iteration $t-1$. A perturbation kernel κ^t is used to perturb them and find $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}$'s. The new particle $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}$ is used to simulate $\boldsymbol{z}$ and if $\rho(\boldsymbol{z}, \boldsymbol{z}^*) \leq \epsilon_t$ is satisfied, the particle is accepted. The process is repeated until N particles are accepted.

At each iteration, new weights are assigned to the particles, and in the next iteration the particles with larger weights become better represented in the population. The importance weights associ-

ated with an accepted population $\{\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}\}_{\{1\leq i\leq N\}}$ are calculated as [5]:

$$w^{(i,t)} \propto \frac{\pi\left(\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}\right)}{\sum_{j=1}^{N} w^{(j,t-1)}\kappa^{t}\left(\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}\middle|\boldsymbol{\alpha}_{\mathrm{c}}^{(j,t-1)}\right)}. \tag{4.6}$$

The efficiency of ABC SMC heavily relies on a proper choice of the distance function, the threshold sequence $\{\epsilon_t\}_{1\leq t\leq N_T}$, and the perturbation kernel function $\kappa^t(\cdot|\cdot)$ [10].

Existing Threshold Sequences

To balance the computational efficiency and the accuracy of the posterior distribution, a threshold sequence is defined as:

$$\mathcal{E} = \{\epsilon_1, \epsilon_2, \cdots, \epsilon_{N_T}\}, \tag{4.7}$$

where $\epsilon_1 > \epsilon_2 > \cdots > \epsilon_{N_T}$. If the threshold is too large, too many proposed particles are accepted; if it is too small, the ABC algorithm is not efficient since many proposed particles will be rejected [88]. The threshold sequence also influences the convergence towards the global mode [10], [8]. It can be chosen in advance [4, 79], which, however, may cause the algorithm to be stuck in local modes or the final ABC posterior to be very different from the true posterior [87]. By contrast, selecting it adaptively based on some quantile of the threshold in the previous iteration has better performance [89].

1. In [4], $\epsilon_1 = 1$, $\epsilon_{N_T} = 0.0025$, and $\epsilon_{2:N_T-1}$ is chosen as

$$\epsilon_{t+1} = \frac{1}{2}(\epsilon_t - \epsilon_{N_T}). \tag{4.8}$$

2. In [5,6], ϵ_1 and ϵ_{N_T} are predefined ($\epsilon_1 = 2$, $\epsilon_{N_T} = 0.01$ in [6]), and ϵ_{t+1} is the qth-percentile of the sorted particle distances in last iteration. However, the optimal q is problem-specific and is decided case by case [89].

3. In [7], for an appropriate $k \in Z^+$, ϵ_1 is calculated by sampling $N_{\text{init}} = kN$ particles from the prior as:

$$\epsilon_1 = \max(\rho_1^{\min}, \cdots, \rho_N^{\min}), \tag{4.9}$$

where $\rho_1^{\min}, \cdots, \rho_N^{\min}$ are the N smallest distances of the N_{init} sampled particles. Then $\epsilon_{2:N_T-1}$ is determined as the 25th percentile of the accepted distances from the previous iteration. However, ϵ_1 tends to be small and many proposed parameters will be rejected, and the choice of $\epsilon_{2:N_T-1}$ has similar problems as in [5, 6].

4. In [8], ϵ_1 is also chosen by (4.9) and ϵ_{t+1} is updated as

$$\epsilon_{t+1} = q_t \epsilon_t, \tag{4.10}$$

where $q_t = 1/\hat{c}_t$ with

$$\hat{c}_t = \sup_{\boldsymbol{\alpha}_c} \frac{p_{\epsilon_t}(\boldsymbol{\alpha}_c)}{p_{\epsilon_{t-1}}(\boldsymbol{\alpha}_c)}. \tag{4.11}$$

Here the density ratio can be calculated by the method in [90]. Note that the next threshold is adjusted based on how rapidly the posteriors are changing. However, when calculating q_t based on (4.11) only the last iteration is considered, which may be inefficient as the estimated posteriors will change only very slowly in later iterations.

Existing Perturbation Kernels

A perturbation kernel defines the distribution of the random variables that will be added to the particles to move them around in the parameter space. A local perturbation kernel generates new particles that are more possible to be accepted, while a widely spread kernel can explore larger parameter spaces with a lower acceptance rate [79].

1. A *component-wise Uniform kernel* is used in [9]. However, it cannot provide a good approximation of the posterior and the algorithm is thus inefficient [10].

2. The *component-wise Normal kernel* in [5] minimizes the Kullback-Leibler distance between the desired posterior and the proposal distribution by a Gaussian distribution with mean $\boldsymbol{\alpha}_{\mathrm{c},j}^{(t-1)}$ and variance $(\sigma_j^t)^2$ [5]. However, it does not consider the correlations between parameters, and thus cannot reflect the structure of the posterior and may search in the space with low probabilities.

3. The *multivariate Normal kernel* in [10] considers the correlations between parameters, and uses a covariance matrix obtained based on a subset of the particles from the previous iteration whose distances are smaller than ϵ_t of the current iteration. However, if any particles are far from the mean of all particles, the algorithm will spend considerable time to find a new particle acceptable for the next iteration.

4. The *multivariate Normal kernel with K_{n}-nearest neighbors* in [10] perturbs $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t-1)}$ based on a Gaussian distribution with mean $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t-1)}$ and the empirical covariance of the K_{n}-nearest particles of $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t-1)}$. However, too large K_{n} does not have obvious advantage compared with using the whole population while too small K_{n} may lead to a lack of exploration of the parameter space.

Parameter Calibration by A-ABC-SMC

By comparing the simulated $\boldsymbol{z}$ with PMU-measured $\boldsymbol{z}^*$, the parameters of the model can be estimated. In this CHAPTER we perform generator parameter calibration by adapting the ABC SMC algorithm in Section 4.

In this CHAPTER , we develop systematic methods to choose proper distance function, the threshold sequence, and the perturbation kernel function in order to greatly improve the computational efficiency, which will be discussed below.

Distance Function

The distance function measures the degree to which the model outputs match the observed data. L_1 distance [91], Euclidean distance [10], chi-squared distance function [92], root mean square (RMS) distance [6] are popular distance functions. In this CHAPTER we choose the following L_1 distance function based on numerical experiments:

$$\rho(\boldsymbol{z}, \boldsymbol{z}^*) = \frac{1}{2K}\|\boldsymbol{z} - \boldsymbol{z}^*\|_1, \tag{4.12}$$

where $\|\cdot\|_1$ is the 1-norm of a vector.

Adaptive Threshold Sequence

Since dynamic simulation is the most computationally expensive part of the A-ABC-SMC algorithm, a properly chosen threshold sequence should reduce the total number of dynamic simulations. This requires a careful balance between the number of simulations in each iteration and

a good approximation of the posterior distribution. Then, it is better to adaptively determine the threshold sequences. We propose the following threshold sequence scheme to achieve this goal:

1. We choose ϵ_1 as

$$\epsilon_1 = \min(\rho_1^{\max}, \cdots, \rho_N^{\max}), \tag{4.13}$$

   where $\rho_1^{\max}, \cdots, \rho_N^{\max}$ are the N largest distances of the N_{init} sampled particles. This avoids the drawback of the too small ϵ_1 in (4.9) that leads to many proposed parameters to be rejected.

2. Let $c_0 = 1$, for iteration $t > 1$ we calculate q_t as

$$q_t = \frac{\sum_{i=1}^{t} \hat{c}_{i-1}}{\sum_{i=1}^{t} \hat{c}_i}. \tag{4.14}$$

   Different from [8], we consider the impact of all previous iterations when calculating q_t. Since $q_t \leq 1$, there is $\epsilon_{t+1} \leq \epsilon_t$. If ϵ_t is less than the predefined smallest threshold, the algorithm will stop. This addresses the inefficiency issue for the method in (4.10)–(4.11) especially for later iterations and greatly improves the efficiency.

Adaptive Perturbation Kernel

All the kernels in Section 4 use $\boldsymbol{\alpha}_{\text{c}}^{(i,t-1)}$ as the mean. Utilizing these kernels ABC SMC may waste a lot of time in sampling the areas of low likelihood. The acceptance rate could be very low and the algorithm may be stuck in local modes. Therefore, we propose to choose the mean of the Gaussian kernel as a function of the distance between $\boldsymbol{\alpha}_{\text{c}}^{i}$ and the mean of all particles, $\bar{\boldsymbol{\alpha}}_{\text{c}}$. For

$\boldsymbol{\alpha}_{\mathrm{c}} = [\alpha_{\mathrm{c},1}, \cdots, \alpha_{\mathrm{c},v'}]^\top$, the range of the particles for the jth parameter is

$$R_j = \max\{d_j^i\}_{1 \le i \le N} - \min\{d_j^i\}_{1 \le i \le N}, \tag{4.15}$$

where d_j^i is the distance between particle i and the mean of the particles for the jth parameter. The mean of the kernel for the jth parameter of particle i is chosen as

$$\tilde{\alpha}_{\mathrm{c},j}^i = \frac{d_j^i}{R_j}\bar{\alpha}_{\mathrm{c},j} + \frac{R_j - d_j^i}{R_j}\alpha_{\mathrm{c},j}^i, \tag{4.16}$$

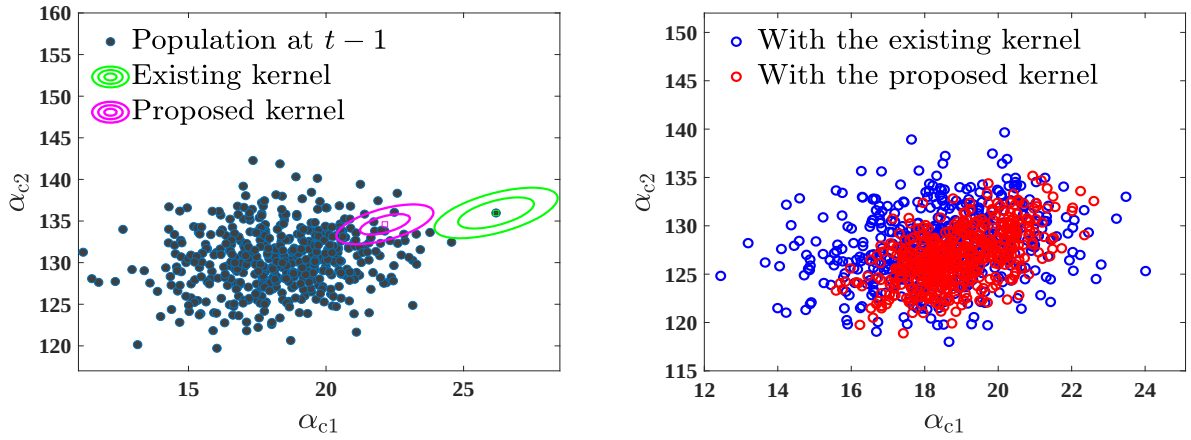
where $\bar{\alpha}_{\mathrm{c},j}$ is the mean of the jth parameter for all particles. For $1 \le i \le N$, $\boldsymbol{\alpha}_{\mathrm{c}}^i$ is perturbed independently according to a Gaussian distribution with mean $\tilde{\boldsymbol{\alpha}}_{\mathrm{c}}^i$.

Figure 4.5 illustrates the populations of two parameters, $\alpha_{\mathrm{c}1}$ and $\alpha_{\mathrm{c}2}$, generated by using the existing kernels and the proposed kernel from two consecutive iterations in our simulations. For the population at iteration $t-1$ shown in Figure 4.5a, an example is given for the existing and proposed kernels for one particle. For the existing kernels $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t-1)}$ is the mean, while for the proposed kernel the mean is chosen as (4.16). For existing kernels the algorithm will search around the particles located in low densities. By contrast, for the proposed kernel the focus is more on the space with higher densities, thus improving time efficiency. Figure 4.5b shows the population at iteration t with the existing and the proposed kernels. It is clear that using the proposed kernel the generated population will be more concentrated with higher efficiency.

The joint distribution first samples a particle from the last population $\boldsymbol{\alpha}_{\mathrm{c}}^{t-1} \sim p_{\epsilon_{t-1}}(\cdot|\boldsymbol{z}^*)$ and obtains a new particle by using the kernel $\kappa^t(\boldsymbol{\alpha}_{\mathrm{c}}^t|\boldsymbol{\alpha}_{\mathrm{c}}^{t-1})$ as:

$$p_{\epsilon_{t-1},\epsilon_t}(\boldsymbol{\alpha}_{\mathrm{c}}^{t-1}, \boldsymbol{\alpha}_{\mathrm{c}}^t|\boldsymbol{z}) \propto p_{\epsilon_{t-1}}(\boldsymbol{\alpha}_{\mathrm{c}}^{t-1}|\boldsymbol{z})\kappa^t(\boldsymbol{\alpha}_{\mathrm{c}}^t|\boldsymbol{\alpha}_{\mathrm{c}}^{t-1}) \int f(\boldsymbol{z}^*|\boldsymbol{\alpha}_{\mathrm{c}}^t)\mathbb{1}(\rho(\boldsymbol{z}^*, \boldsymbol{z}) \le \epsilon_t)d\boldsymbol{z},$$

where $\kappa^t(\boldsymbol{\alpha}_{\mathrm{c}}^t|\boldsymbol{\alpha}_{\mathrm{c}}^{t-1})$ should minimize the following Kullback-Leibler divergence $\mathfrak{C}$ between



(a) (b)

Figure 4.5: The population at two consecutive iterations for the existing and proposed kernels: (a) Population at iteration $t-1$ with the existing and proposed kernels illustrated for one particle; (b) Populations at iteration t using the existing and proposed kernels.

$p_{\epsilon_{t-1},\epsilon_t}(\boldsymbol{\alpha}_{\mathrm{c}}^{t-1}, \boldsymbol{\alpha}_{\mathrm{c}}^{t}|\boldsymbol{z})$ and the desired distribution p_{desire} [5, 93]:

$$\mathfrak{C}\big(p_{\mathrm{desire}} \parallel p_{\epsilon_{t-1},\epsilon_t}(\boldsymbol{\alpha}_{\mathrm{c}}^{t-1}, \boldsymbol{\alpha}_{\mathrm{c}}^{t}|\boldsymbol{z})\big) = \int p_{\mathrm{desire}} \log\left(\frac{p_{\mathrm{desire}}}{p_{\epsilon_{t-1}}(\boldsymbol{\alpha}_{\mathrm{c}}^{t-1}|\boldsymbol{z})\kappa^t(\boldsymbol{\alpha}_{\mathrm{c}}^{t}|\boldsymbol{\alpha}_{\mathrm{c}}^{t-1})}\right) d\boldsymbol{\alpha}_{\mathrm{c}}. \tag{4.17}$$

It has been proved that minimizing (4.17) is equivalent to maximizing $\int p_{\mathrm{desire}} \log\left(\kappa^t(\boldsymbol{\alpha}_{\mathrm{c}}^{t}|\boldsymbol{\alpha}_{\mathrm{c}}^{t-1})\right) d\boldsymbol{\alpha}_{\mathrm{c}}$ [94]. Presumably, p_{desire} should be similar to the joint distribution that picks two particles independently [5], i.e.

$$p_{\mathrm{desire}} = p_{\epsilon_{t-1}}(\boldsymbol{\alpha}_{\mathrm{c}}^{t-1}|\boldsymbol{z}) p_{\epsilon_t}(\boldsymbol{\alpha}_{\mathrm{c}}^{t}|\boldsymbol{z}). \tag{4.18}$$

For a multivariate Gaussian kernel, we assume that all particles in iteration $t-1$ only satisfy

$\rho(\boldsymbol{z}, \boldsymbol{z}^*) \leq \epsilon_{t-1}$ but not $\rho(\boldsymbol{z}, \boldsymbol{z}^*) \leq \epsilon_t$. Then Γ^t can be written as [5]:

$$\Gamma^t \approx \sum_{i=1}^{N} w^{(i,t-1)} \big(\boldsymbol{\alpha}_{\mathrm{c}}^i - \tilde{\boldsymbol{\alpha}}_{\mathrm{c}}\big)\big(\boldsymbol{\alpha}_{\mathrm{c}}^i - \tilde{\boldsymbol{\alpha}}_{\mathrm{c}}\big)^{\top}. \tag{4.19}$$

Algorithm 2: A-ABC-SMC algorithm for estimating the posterior distribution of parameters $\boldsymbol{\alpha}_{\mathrm{c}}$ using N particles, the prior distribution $\pi(\boldsymbol{\alpha}_{\mathrm{c}})$. $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}$ is the parameter set for particle i at iteration t.

Set maximum number of iterations N_T and set ϵ_1 by (4.13)
At iteration $t = 1$
for $1 \leq i \leq N$ **do**
  while $\rho(\boldsymbol{z}, \boldsymbol{z}^*) > \epsilon_1$ **do**
    Sample $\boldsymbol{\alpha}_{\mathrm{c}}^*$ from the prior: $\boldsymbol{\alpha}_{\mathrm{c}}^* \sim \pi(\boldsymbol{\alpha}_{\mathrm{c}})$
    Generate data $\boldsymbol{z}$ from $\boldsymbol{\alpha}_{\mathrm{c}}^*$: $\boldsymbol{z} \sim \mathrm{Model}(\boldsymbol{\alpha}_{\mathrm{c}}^*)$
    Calculate discrepancy $\rho(\boldsymbol{z}, \boldsymbol{z}^*)$ based on (4.12)
  end while
  Set $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,1)} \leftarrow \boldsymbol{\alpha}_{\mathrm{c}}^*$ and $w^{(i,1)} \leftarrow \frac{1}{N}$
end for
Generate Gaussian perturbation kernel $\kappa^2 = \mathcal{N}(\tilde{\boldsymbol{\alpha}}_{\mathrm{c}}^1, \Gamma^1)$ based on (4.19)
Determine ϵ_2 based on (4.10) and (4.14)
At iteration $t > 1$
for $2 \leq t \leq N_T$ **do**
  for $1 \leq i \leq N$ **do**
    while $\rho(\boldsymbol{z}, \boldsymbol{z}^*) > \epsilon_t$ **do**
      Sample $\boldsymbol{\alpha}_{\mathrm{c}}^*$ from the previous population $\{\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t-1)}\}_{\{1 \leq i \leq N\}}$ with probabilities $\{w^{(i,t-1)}\}_{\{1 \leq i \leq N\}}$ and perturb them to obtain $\boldsymbol{\alpha}_{\mathrm{c}}^{**} \sim \kappa^t(\tilde{\boldsymbol{\alpha}}_{\mathrm{c}}^{t-1}, \Gamma^{t-1})$
      Generate data $\boldsymbol{z}$ from $\boldsymbol{\alpha}_{\mathrm{c}}^{**}$: $\boldsymbol{z} \sim \mathrm{Model}(\boldsymbol{\alpha}_{\mathrm{c}}^{**})$
      Calculate discrepancy $\rho(\boldsymbol{z}, \boldsymbol{z}^*)$ based on (4.12)
    end while
    Set $\boldsymbol{\alpha}_{\mathrm{c}}^t \leftarrow \boldsymbol{\alpha}_{\mathrm{c}}^{**}$
    Set $w^{(i,t)} \leftarrow \frac{\pi\left(\alpha_{\mathrm{c}}^{(i,**)}\right)}{\sum_{j=1}^{N} w^{(j,t-1)} \kappa^t\left(\alpha_{\mathrm{c}}^{(i,**)} \middle| \tilde{\alpha}_{\mathrm{c}}^{(j,t-1)}\right)}$
  end for
  Generate Gaussian perturbation kernel $\kappa^{t+1} = \mathcal{N}(\tilde{\boldsymbol{\alpha}}_{\mathrm{c}}^t, \Gamma^t)$ based on (4.19)
  Determine ϵ_{t+1} based on (4.10) and (4.14)
end for

Proposed A-ABC-SMC Algorithm

The A-ABC-SMC algorithm with the proposed adaptive threshold sequence and perturbation kernel function is presented in Algorithm 2. When the threshold sequence is adaptive the algorithm will stop when a maximum number of N_T iterations has been performed [4, 5]. Alternatively, where the user defines a lowest threshold ($\epsilon_{\min}$), the A-ABC-SMC algorithm can be modified so that it will stop when the lowest threshold criteria is met.

The proposed A-ABC-SMC provides the posterior distribution of the parameters based on the N particles in the last iteration, not just one point estimation. The maximum-a-posteriori (MAP), the mean of the N particles in the last iteration for each parameter is chosen as the estimated parameter. Generator parameter calibration usually only deals with one generator and only a small number of its parameters with high identifiability will be calibrated. If many parameters have to be calibrated at the same time, the ABC SMC algorithm may face curse of dimensionality issue, in which case methods such as the one in [95] may be implemented in order to alleviate this issue and improve the scalability of the proposed approach.

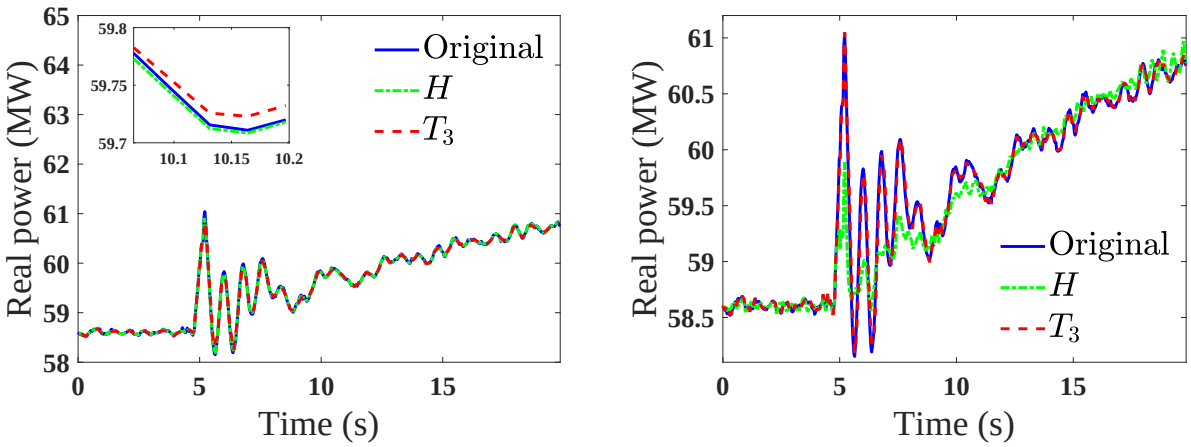
Simulation Results

The proposed method is implemented based on PSS/E and Python 2.7. We use the same hydro-generator model and PMU data in [75]. A PMU is installed at the 230-kV side of the substation with a sampling rate of 30 sample/s. All tests are performed on a desktop PC with Intel(R) Core(TM) i7-8700 and 8-GB RAM.

Critical Parameter Identification

For sensitivity analysis, a small perturbation $\Delta\alpha_i = 5\%|\alpha_i|$ is applied to each parameter. Table 4.1 lists the top fourteen critical parameters identified by the EPSG and sensitivity based approaches, in which the corresponding $\mathrm{tr}\big(\boldsymbol{W}(\alpha_i)\big)$ and $S(\alpha_i)$ are presented. It is seen that the EPSG based approach identifies H as a critical parameter while the sensitivity based approach does not, with $\mathrm{tr}\big(\boldsymbol{W}(H)\big) = 3210 \gg \mathrm{tr}\big(\boldsymbol{W}(T_3)\big) = 1470$. Meanwhile, the sensitivity based approach identifies T_3 as a critical parameter while the EPSG based approach does not, with $S(T_3) = 0.85$ slightly greater than $S(H) = 0.6$.

Figure 4.6 shows the real power under small perturbation (5%) and large perturbation (50%) of H or T_3. For small perturbation, real power is a little more sensitive to T_3 than to H. However, as shown in Figure 4.6b, under large perturbation, real power is much more sensitive to H, which is consistent with the result of the EPSG based approach.



(a) Small perturbation

(b) Large perturbation

Figure 4.6: Real power under small and large perturbations of H or T_3.

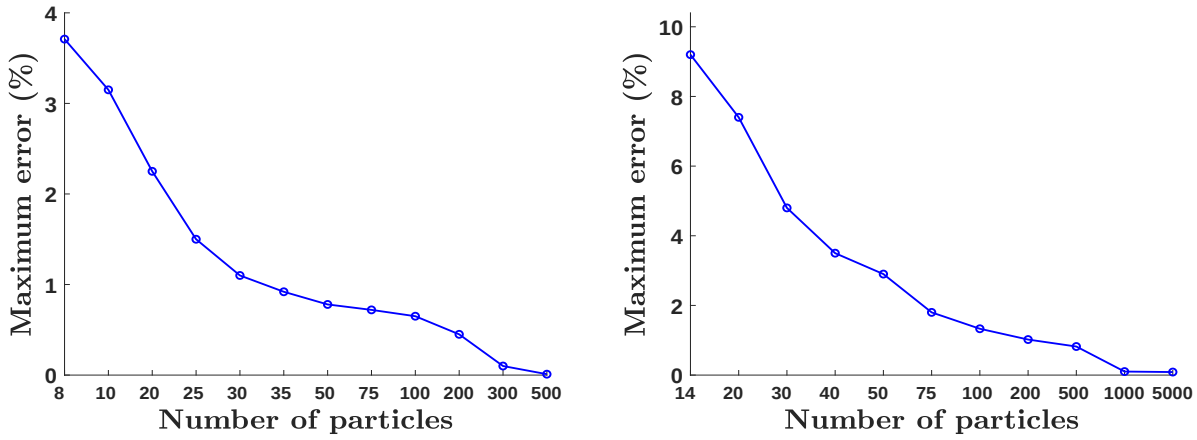
Table 4.1: Top Fourteen Critical Parameters Identified by Gramian-Based and Sensitivity-Based Approaches

EPSG		**Sensitivity**	
Parameter	$\mathrm{tr}\big(\boldsymbol{W}(\alpha_i)\big)$	Parameter	$S(\alpha_i)$
K_{S}	1270745	T_6	26.39
T_6	1270731	K_{S}	26.11
T_5	314231	T_5	22.23
K_{A}	4441	K_{A}	1.75
H	3210	T_{b}	1.35
a_{23}	2254	a_{23}	1.13
T'_{do}	2247	T'_{do}	1.11
T_{b}	2240	T_3	0.85
T_3	1269	T_1	0.83
T_1	1258	X_{d}	0.72
T_{c}	1044	T_{c}	0.67
X_{d}	988	H	0.61
X'_{d}	531	R_{T}	0.57
R_{T}	422	X'_{q}	0.52

Particle Population Size

A large N is desirable for high accuracy, but will increase the computational burden. It is reasonable to estimate the parameters of different dimensions using different numbers of particles. Figure 4.7 shows the maximum error for two and fourteen critical parameters under different particle numbers, in which the threshold sequence and kernel function in [5] are used. For two-parameter case,

we set acceptable error to be 1%. Based on Figure 4.7a $N = 30$ satisfies this error criterion. For fourteen parameter case, we set acceptable error to be 2%. Based on Figure 4.7b, $N = 75$ satisfies this error criterion.



(a) Two parameters

(b) Fourteen parameters

Figure 4.7: Maximum error of the estimated parameters for different number of particles.

Distance Function

We compare different distance functions including the L_1 distance in (4.12). The threshold sequence and kernel function are set as the same in [5]. Table 4.2 lists the average number of simulations in ten independent runs under different distance functions. It is seen that the L_1 distance function in (4.12) requires the smallest number of simulations.

Adaptive Threshold Sequence and Perturbation Kernel

We evaluate the performance of different threshold sequence schemes and perturbation kernel functions for the calibration of the moment of inertia H and the amplifier gain of the exciter K_{A} by the

Table 4.2: Average Number of Simulations for Different Distance Functions Over Ten Independent Runs

Euclidean [10]	chi-squared [92]	RMS [6]	(4.12)
1556	1486	1515	1448

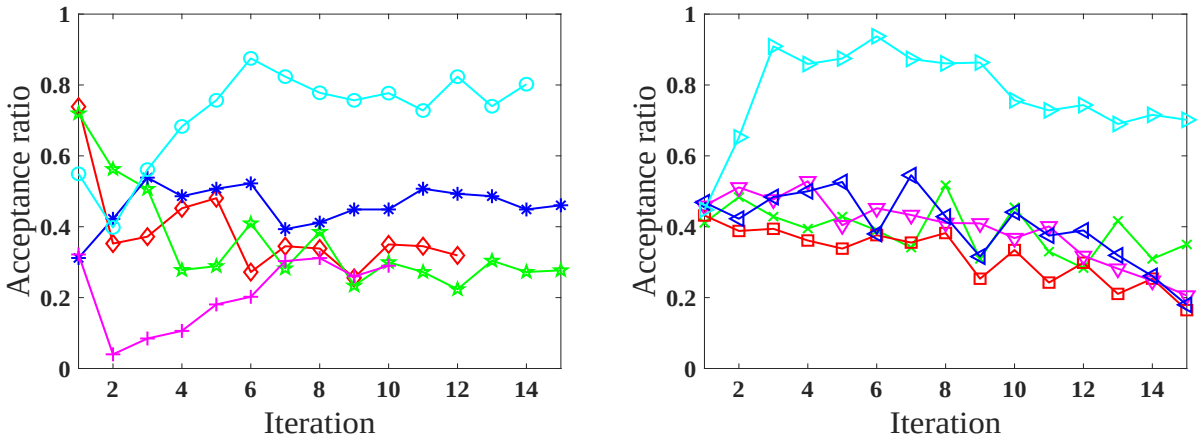
acceptance rate [10]. We consider the priors of the parameters as a Gaussian distribution whose mean has 10% deviation from the true values and whose standard deviation is 30% of the mean. For all simulations, we use $N = 30$ particles, the acceptable error is set as 1%, and the maximum number of iterations N_T is set to be 15.

In Figure 4.8, we show the acceptance rates for different threshold sequences and perturbation kernel functions. In Figure 4.8a, we set the perturbation kernel as the one in [5] and compare the acceptance rate for existing threshold sequence schemes and our proposed threshold sequence over ten independent runs. The proposed method has the highest acceptance rate. Similarly, we set the threshold sequence as the one in [5] and provide the acceptance rates in Figure 4.8b for existing perturbation kernel functions and our proposed kernel function. For the multivariate normal kernel with K_{n} nearest neighbors in [10], we choose $K_{\mathrm{n}} = 5$. It is seen that the proposed kernel has a significantly higher acceptance rate than the other methods.

We set the perturbation kernel as the one in [5] and show the threshold sequences and the number of simulations at each iteration in Figure 4.9. Although the proposed threshold sequence requires four more iterations than the methods in [4] and [7], the number of simulations in each iteration for the proposed threshold sequence is much smaller, leading to overall higher efficiency. Similarly in Figure 4.10, we set the threshold as the one in [5] and show the threshold sequence and the number of simulations in each iteration under different perturbation kernel functions. Since the same threshold sequence scheme is used, each case has very similar threshold sequences. As in

Figure 4.10b, the ABC SMC using the proposed perturbation kernel function has significantly smaller number of simulations compared with the existing kernels.

In Table 4.3, we compare the average number of simulations for different threshold sequences and perturbation kernels over ten independent runs. The proposed threshold sequence and kernel function require the least number of simulations to reach posterior distributions. When we consider the proposed threshold sequence and the perturbation kernel function at the same time, it only needs 320 simulations to reach the posterior distributions, achieving a speed-up of $1879/320 \approx 5.87$ compared with the case with the lowest efficiency.



(a) Threshold

(b) Transition kernel function

Figure 4.8: Average acceptance rate over ten independent runs for (a) different threshold sequences and (b) different perturbation kernel functions. In Figure 4.8a: $\diamond$ Method in [4], $\star$ Method in [5], [6], $+$ Method in [7], $*$ Method in [8], $\circ$ Proposed threshold sequence. In Figure 4.8b $\triangleleft$ Component-wise Uniform [9], $\square$ Component-wise Normal [5], $\times$ Multivariate normal [10], $\triangledown$ Multivariate normal with K_{n} nearest neighbors [10], $\triangleright$ Proposed kernel.

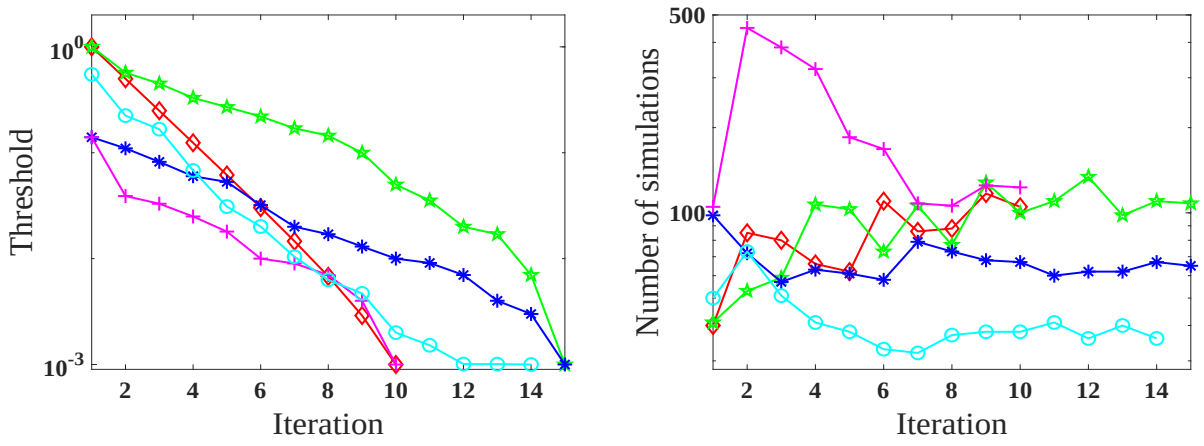


Figure 4.9: Threshold sequences and number of simulations in each iteration with fixed perturbation kernel: (a) Threshold sequences; (b) Number of simulations. $\diamond$ Method in [4], $\star$ Method in [5], [6], $+$ Method in [7], $*$ Method in [8], $\circ$ Proposed threshold sequence.

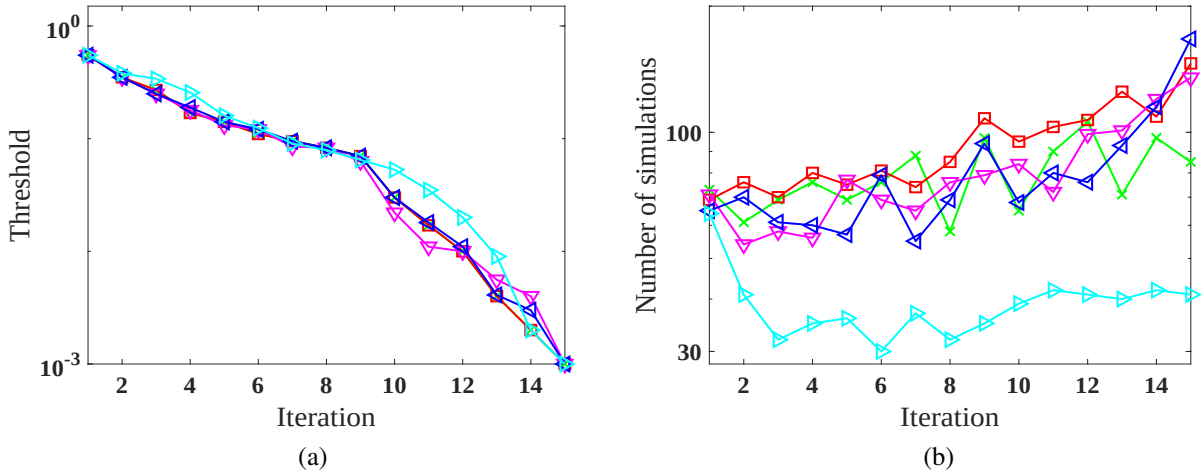


Figure 4.10: Threshold sequences and number of simulations in each iteration with fixed threshold sequence scheme and different perturbation kernels: (a) Threshold sequences; (b) Number of simulations. $\triangleleft$ Component-wise Uniform [9], $\square$ Component-wise Normal [5], $\times$ Multivariate normal [10], $\triangledown$ Multivariate normal with K_{n} nearest neighbors [10], $\triangleright$ Proposed kernel.

Table 4.3: Average Number of Simulations for Different Threshold Sequences and Perturbation Kernel Functions Over Ten Independent Runs

Threshold	**Simulations**	**Kernel**	**Simulations**
Method in [7]	1879	Component-wise Uniform [9]	1634
Methods in [5, 6]	1448	Component-wise Normal [5]	1448
Method in [4]	1260	K_{n} nearest neighbours [10]	1125
Method in [8]	1044	Multivariate normal [10]	1014
Proposed threshold	615	Proposed kernel	547

Calibration of Two Parameters

We calibrate H and K_{A} whose true values are $H^{\mathrm{t}} = 5.4$ and $K_{\mathrm{A}}^{\mathrm{t}} = 125$. The mean values of the prior distributions of these parameters are set as 10% greater than their true values to account for parameter uncertainties. We choose the lower/upper bounds of the uniform prior distributions as 50%, 70%, or 90% less/greater than the mean values. The standard deviations of the Gaussian prior distributions are set as 10%, 20%, or 30% of the mean value. The results for parameter calibration are given in Table 4.4. It is seen that the proposed method can accurately estimate the parameters under different prior distributions, and the largest errors for H and K_{A} are, respectively, 1.1% and 1.2%.

We then consider more extreme cases in which the prior is a Gaussian distribution whose mean has 20%, 30%, and 40% deviation from the true values and whose standard deviation is 10%, 30%, and 60% of the mean values. As in Table 4.5, even when the mean of the priors have very large deviations from the true values and the standard deviations of the priors are relatively small, the proposed approach still has high accuracy. For the highlighted first row in Table 4.5, the

Table 4.4: Parameter Calibration under Different Prior Distributions

Uniform prior			
$H(H^{\mathrm{t}} = 5.40)$		$K_{\mathrm{A}}(K_{\mathrm{A}}^{\mathrm{t}} = 125)$	
Prior	**Estimated** (% Error)	**Prior**	**Estimated** (% Error)
$\mathcal{U}$(2.9, 8.9)	5.39 (0.1)	$\mathcal{U}$(68.8, 206.3)	125.5 (0.4)
$\mathcal{U}$(1.8, 10.1)	5.43 (0.5)	$\mathcal{U}$(41.2, 233.8)	124.6 (0.3)
$\mathcal{U}$(0.6, 11.3)	5.47 (1.1)	$\mathcal{U}$(13.8, 261.2)	123.5 (1.2)
Gaussian prior			
Prior	**Estimated** (% Error)	**Prior**	**Estimated** (% Error)
$\mathcal{N}(5.94, 0.6^2)$	5.40 (0)	$\mathcal{N}(137.5, 13.8^2)$	125.1 (0.1)
$\mathcal{N}(5.94, 1.2^2)$	5.38 (0.3)	$\mathcal{N}(137.5, 27.5^2)$	125.2 (0.1)
$\mathcal{N}(5.94, 1.8^2)$	5.41 (0.2)	$\mathcal{N}(137.5, 41.3^2)$	124.1 (0.8)

prior/posterior distributions, and the estimated/true values of the parameters are shown in Figure 4.11. It is seen that even when the true values are at the tail of the assumed prior distributions the proposed algorithm can still very accurately estimate the parameters. In contrast, the proposed algorithm in [41] has an error greater than 25% for the same deviations and priors.

To compare our proposed method with that in [43], we consider a case similar to the test B in [43]. Specifically, we set the mean value of the prior distribution for H as 61% less than the true value and that for K_{A} as 225% greater than the true value, and choose the lower/upper bounds of the uniform prior distributions as 90% less/greater than the mean values. We show the performance of the proposed method in Table 4.6. It can be seen that our algorithm can find very accurate parameters. The estimation error of the proposed approach for H is 0.01% and for K_{A} is 0.05%, which are less than the corresponding errors, 0.08% and 0.5% respectively, in [43]. We also show the posterior distributions of the parameters in Figure 4.12. Note that the standard deviations of

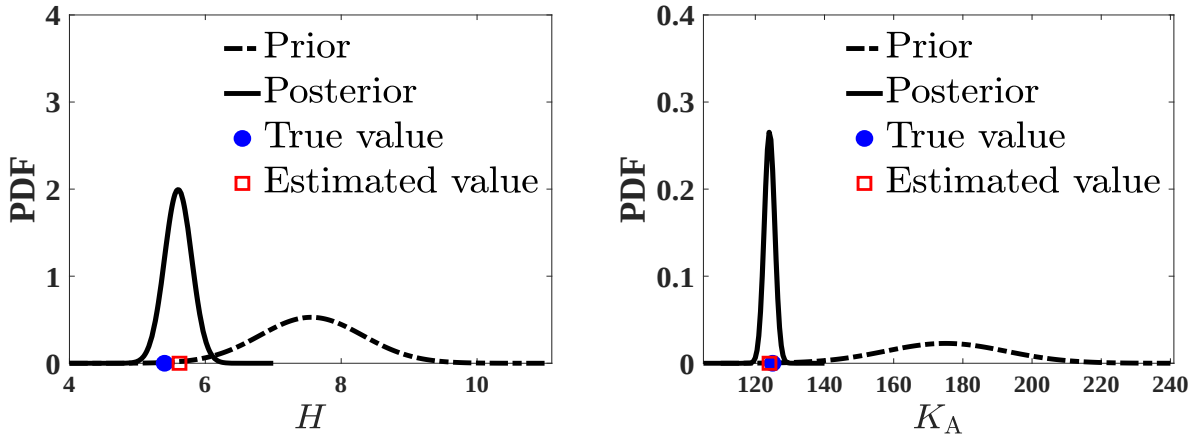
the posterior distributions obtained by our approach are much smaller than those in [43].

Table 4.5: Calibration of H and K_A for Different Deviations and Prior Distributions

Standard Deviation (10%)						
Deviation	H			K_A		
	Prior	Estimated	Error (%)	Prior	Estimated	Error (%)
40 %	**7.56**	**5.61**	**3.8**	**175**	**124.1**	**0.8**
30 %	7.02	5.31	1.5	162.5	124.1	0.7
20 %	6.48	5.38	0.4	150	125.2	0.2
Standard Deviation (30 %)						
Deviation	H			K_A		
	Prior	Estimated	Error (%)	Prior	Estimated	Error (%)
40 %	7.56	5.31	1.6	175	123.6	1.1
30 %	7.02	5.34	1	162.5	125.7	0.6
20 %	6.48	5.39	0.2	150	125.3	0.2
Standard Deviation (60 %)						
Deviation	H			K_A		
	Prior	Estimated	Error (%)	Prior	Estimated	Error (%)
40 %	7.56	5.4	0	175	125.3	0.3
30 %	7.02	5.41	0.2	162.5	125.2	0.2
20 %	6.48	5.40	0	150	125.2	0.2

Calibration of Fourteen Parameters

We calibrate the fourteen critical parameters identified from the Gramian based approach using the proposed ABC-SMC approach. The priors of these parameters are assumed to follow Gaussian



(a) H (b) K_{A}

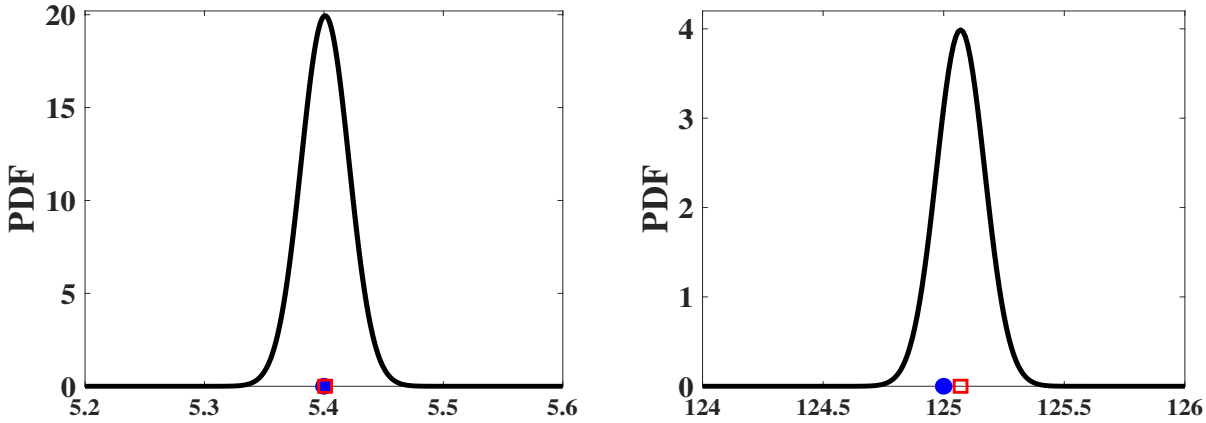
Figure 4.11: Prior and posterior distributions, and the true and estimated values for (a) H and (b) K_{A}.

Table 4.6: Parameter Calibration Under Bad Prior Distribution

$H(H^{\mathrm{t}} = 5.40)$		$K_{\mathrm{A}}(K_{\mathrm{A}}^{\mathrm{t}} = 125)$	
Prior	**Estimated** (% Error)	**Prior**	**Estimated** (% Error)
$\mathcal{U}(0.9, 17.2)$	5.401 (0.01)	$\mathcal{U}(28.1, 534.4)$	125.07 (0.05)

distributions whose means are 10% greater than their true values and whose standard deviations are 20% of the mean values. Table 4.7 lists the estimated values. It can be seen that the algorithm has good accuracy under high dimension.

In real application, true model parameters are never known. It is possible that for different events the algorithm may provide different parameters. In addition to engineering judgment and experience that can help choose reasonable parameters [19, 48], techniques can also be developed based on the available multiple events to help select the best parameters. For example, one can consider one event to estimate the parameters and use the other events to cross-validate the estimated pa-



(a) H (b) K_{A}

Figure 4.12: Posterior distributions, and the true and estimated values for (a) H and (b) K_{A}. □ Estimated values, • True values, – posterior distribution.

rameters. Here we assume PMU measurements for three events are available. We estimate the parameters with each one of the three events and calculate the 1-norm error defined in (4.12) for all three events. The parameter set with the smallest average 1-norm error is selected. Figure 4.13 shows the model validation results under three events with the original parameters and the estimated best parameters. It is seen that the mismatch between model outputs and PMU measurements is significant under original parameters while the model outputs from the estimated parameters can match the PMU measurements very well for all three events.

Time Efficiency

The generator model has 60 parameters. Dynamic simulation is solved by PSS/E which is called from Python 2.7. The calibration of two and fourteen critical parameters takes 1 and 10 minutes, respectively. By contrast, it takes at least 20 minutes for the two-parameter case and more than 2 hours for the fourteen-parameter case if using the existing threshold sequence schemes and

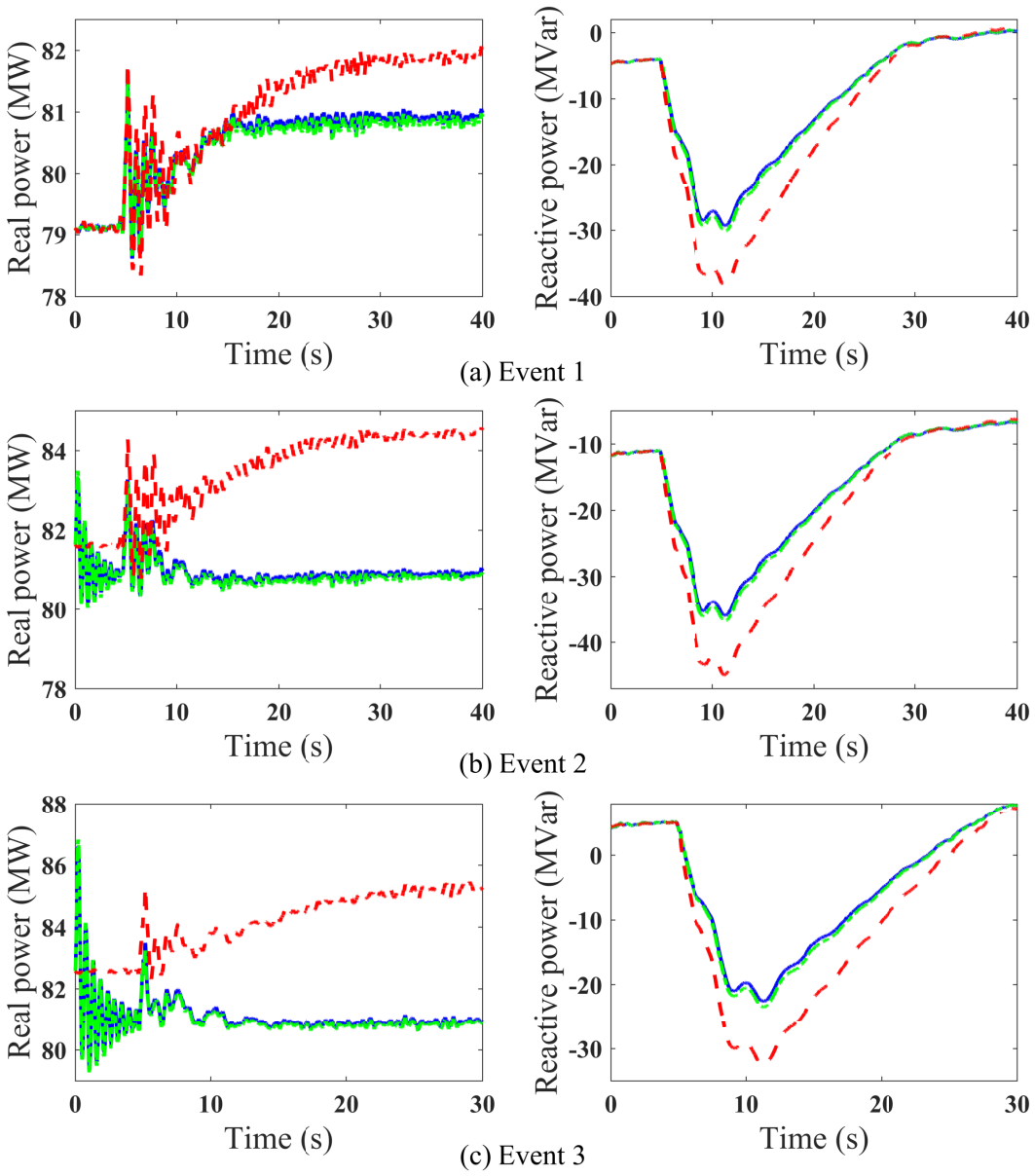


Figure 4.13: Model outputs before and after parameter calibration: (a) Event 1; (b) Event 2; (c) Event 3. — PMU measurements; - - model ouputs before calibration; . model outputs after calibration.

Table 4.7: Calibration of Fourteen Key Parameters

Parameter	**True value**	**Estimated value**	**Error (%)**
K_{S}	20	20.176	0.9
T_6	10	10.081	0.8
T_5	10	10.124	1.24
K_{A}	125	123.12	1.5
H	5.40	5.332	1.3
a_{23}	1.102	1.101	0.1
T'_{do}	5.40	5.352	0.9
T_{b}	3.86	3.970	2.9
T_3	0.15	0.150	0
T_1	0.15	0.151	0.6
T_{c}	0.90	0.909	1
X_{d}	0.57	0.566	0.7
X'_{d}	0.25	0.24	4
R_{T}	0.42	0.428	1.9

perturbation kernel functions.

Conclusion

This CHAPTER proposes a novel method for generator parameter calibration using PMU measurements. The critical parameters are identified by an empirical Gramian based method and are estimated by an adaptive ABC SMC approach. For speeding up the algorithm, we propose adaptive threshold sequence and perturbation kernel. The proposed approach has been successfully applied

to calibrate the parameters of a generator. The simulation results reveal that the proposed method can accurately and efficiently estimate the parameters.

CHAPTER 5: GENERATOR PARAMETER CALIBRATION USING A DEEP LEARNING MEASUREMENTS

Khazeiynasab, S. R., Zhao, J., Batarseh. I. (2021). A Parallel Multi-Modal LSTM for Power System Model Calibration. International Journal of Electrical Power & Energy Systems (under review).

Introduction

Renewable energies, smart loads, energy storage, and new market behavior are adding new sources of uncertainty to power systems. Therefore, planning in real-time and developing high-quality models is crucial to adapt to uncertainties. Model validation based on actual measurements is necessary for obtaining accurate representations of power systems dynamics with system uncertainties. This CHAPTER presents a new measurement-based method to calibrate the parameters of a synchronous generator by deep learning method based on the long-short term memory (LSTM) network. First, critical parameters are determined regarding the active/reactive behavior of the generator. Then, a parallel multimodal LSTM (PM-LSTM) is designed with flexible input time steps to capture important features of temporal patterns from time-series measurements. The extracted features are then fed into a dense layer to capture the joint representation of inputs. The simulations conducted for a hydro generator under different events show that the proposed method can estimate the model parameters accurately and efficiently even in the presence of gross errors in the prior distributions of the parameters.

With the rapid development of the learning methods, it is possible to use learning techniques to represent the complicated nonlinear relationship of the outputs of the generator with its parameters

[96]. Among these recent developments, long-short term memory (LSTM)-based methods are powerful techniques for representing complex nonlinear relationships. To calculate the generator model parameters more efficiently and more accurately, the LSTM technique is developed. The PMU measurement time series are captured with powerful temporal patterns by means of an LSTM network with a flexible number of temporal states. Further, the extracted features are fed into a linear regression model to estimate the full probability distributions of the generator parameters.

This CHAPTER proposes a generator parameter calibration approach based on LSTM and present a parallel multimodal LSTM (PM-LSTM) to further improve the efficiency. Compared to the existing methods, the advantages and improvements of the proposed method are as follows.

1. It can provide the full distributions of the parameters even when the generator model has significant model discrepancies due to gross errors in the parameters.
2. It can estimate the high dimensional cases accurately even when the parameter space is large enough.
3. The implementation is more straightforward and reliable compared to existing reinforcement and deep learning methods.
4. The proposed method is simulation-based and does not require a likelihood function or state-space model of the generator. Since commercial software already has stability models, its implementation is much easier and can be used very easily in real-world applications.

The contributions are summarized as follows.

1. To better understand the effects of the parameters on the outputs of the generator (active/reactive power), a trajectory-sensitivity analysis is performed and critical parameters are identified. Then, a generator parameter calibration was conducted with LSTM, a deep

learning-based method, that estimated the full probability distribution of parameters of a power system with numerous parameters.

2. To enhance computational efficiency and accuracy of a single LSTM (S-LSTM) model, a PM-LSTM approach has been proposed to find the optimal values of the parameters.

3. The PM-LSTM method has been applied to the different case studies with different prior distributions to find the optimal time steps for the time-series active/reactive power of the generator to estimate the parameters.

Generator Dynamic Model

The model under study consist of a synchronous generator with its controller. The block diagram of the models are shown in Fig. 4.1–4.4.

Proposed Framework

Fig. 5.1 illustrates the main flowchart for the proposed framework, which consists of two key modules, model validation, and model calibration. Multiple events are used to check the performance of the system under study. If any model deficiency is identified, the parameter calibration module is used to identify and update inaccurate parameter values. The calibration module includes two main steps: (1) identify the critical parameters (2) automatic parameter calibration based on the PM-LSTM model. If a model deficiency is detected for every additional event, the calibration process will be repeated until it is verified that the model matched the observed data well. More details of each module in Fig. 5.1 are provided in the following sections.

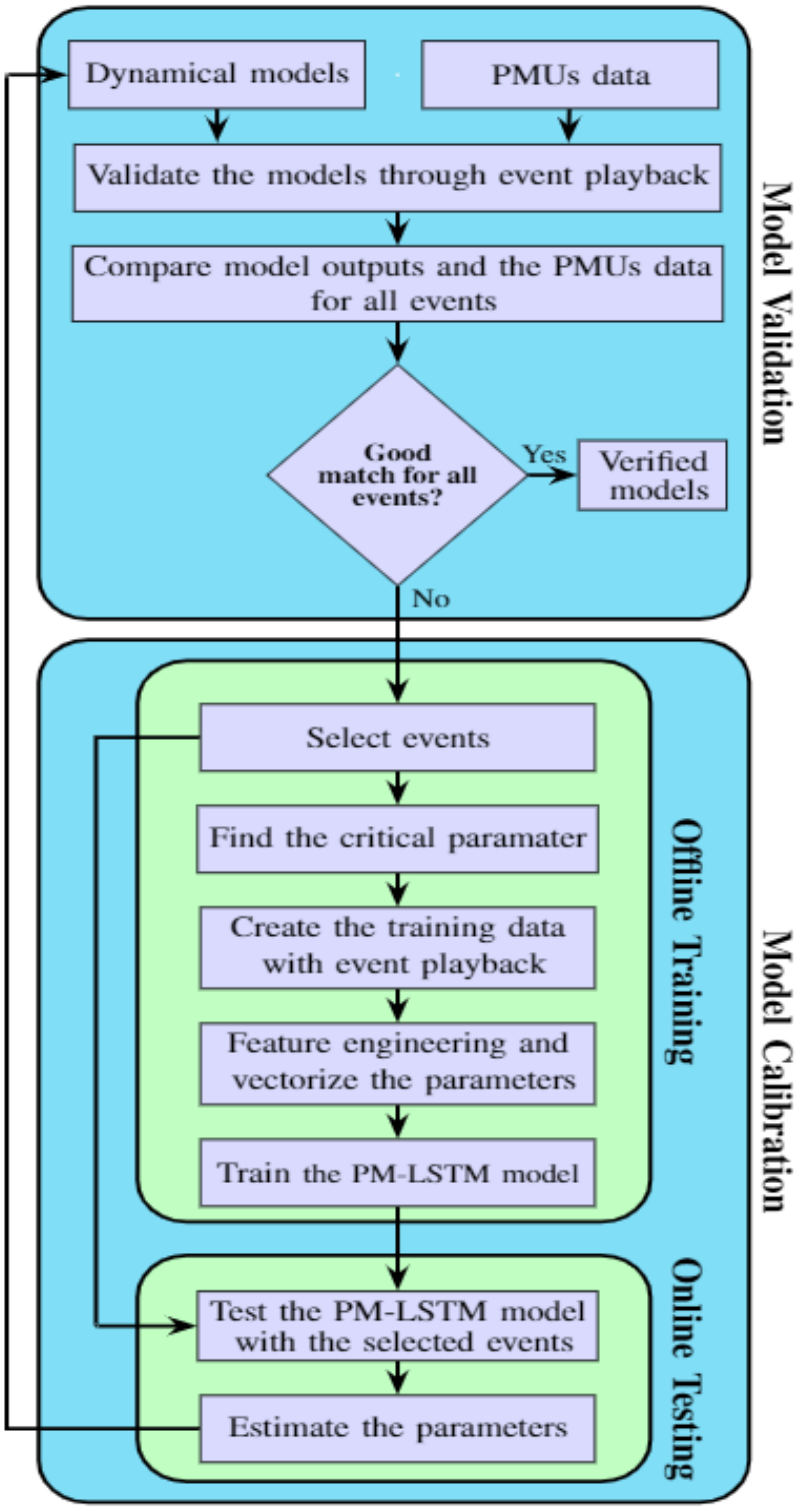


Figure 5.1: Proposed framework for model validation and parameter calibration.

Identifying Critical Parameters

The sensitivity of the outputs P_{m} and Q_{m} with regard to parameter α_i can be calculated as [19]:

$$S_P(\alpha_i) = \sum_{k=1}^{K} \frac{|P_{\mathrm{m},k}(\alpha_i^+) - P_{\mathrm{m},k}(\alpha_i^-)|}{K(\alpha_i^+ - \alpha_i^-)/\alpha_i} \tag{5.1}$$

$$S_Q(\alpha_i) = \sum_{k=1}^{K} \frac{|Q_{\mathrm{m},k}(\alpha_i^+) - Q_{\mathrm{m},k}(\alpha_i^-)|}{K(\alpha_i^+ - \alpha_i^-)/\alpha_i}, \tag{5.2}$$

where K is the number of time steps, $\alpha_i^+ = \alpha_i + \Delta\alpha_i$ and $\alpha_i^- = \alpha_i - \Delta\alpha_i$, and $\Delta\alpha_i$ is a small perturbation of α_i. After the sensitivity analysis, the parameters selected to be estimated are those with a large sensitivity [19, 39].

LSTM Based Parameter Estimation

Generator parameter estimation based on PMUs is a mathematical representation of the highly nonlinear relationship between parameters and outputs. Based on the generator model in Section 3, one can define the complex nonlinear relationship of the generator model parameters and the measurements using the generalized recursion computed by

$$\boldsymbol{\alpha}_{\mathrm{c}}^t = \mathcal{F}\big(\mathcal{F}_{\mathbf{P}}(\rho^{t-1}, \cdots, \rho^{t-t_{\mathrm{p}}}), \mathcal{F}_{\mathbf{Q}}(\sigma^{t-1}, \cdots, \sigma^{t-t_{\mathrm{q}}})\big), \tag{5.3}$$

where, ρ is the $t_{\mathrm{p}} \times \eta$-dimensional vector of real power features, and σ is the $t_{\mathrm{q}} \times \zeta$-dimensional vector of reactive power features. As shown in (5.3), the generator parameters at time step t is a nonlinear function $\mathcal{F}$ of the temporal features extracted by $\mathcal{F}_{\mathrm{P}}$ from active power and $\mathcal{F}_{\mathrm{Q}}$ from reactive power. Generator parameter estimation is to estimate the $\boldsymbol{\alpha}_{\mathrm{c}}^t$ by learning the optimal $\mathcal{F}$ [97, 98]. However, this function can not be formulated analytically, and the generator param-

eters can not be estimated. By incorporating a hidden layer of memory cells into the recurrent neural network, LSTM can establish the temporal correlation between previous information and the current conditions [99]. This property can help estimate the parameters of the generator from its captured dynamics.

LSTM has three special multiplicative computational units named as the input, output, and forget gate. The input gate is responsible for controlling the portion of the current state information, and the current input should be saved to the memory cells [100]. The forget gate controls the value of the last time cell, which will be saved, and the output gate controls the output of the cell. The structure of the LSTM network is shown in Fig. 5.2. The formulations of the different gates in an LSTM will be discussed below. The forget gate is as follow:

$$\varrho_t = \kappa(W_f * [h_{t-1},\ x_t] + b_f),$$

where h_{t-1} is the short-memory state transmitted by the previous LSTM unit, x_t is the input, ϱ_t is the forget gate, and $\kappa(\cdot)$ is a nonlinear smooth and differentiable activation function and usually considered as a *sigmoid* function, whose output is a value between zero and one. Which, zero means "let nothing pass" and one means "let everything pass" [101]. The forget gate output tells the cell state which information to forget by multiplying zero or one. The input gate controlled by the following equation

$$I_t = \kappa(W_i * [h_{t-1},\ x_t] + b_i),$$

where W_i and b_i are the weight and the bias of the input gate, respectively. The new candidate

value $\tilde{c}_t$ and the future cell state c_t are obtained by the following equations

$$\tilde{c}_t = \tilde{\kappa}(W_c * [h_{t-1},\ x_t] + b_c),$$
$$c_t = \varrho_t \circ c_{t-1} + i_t \circ \tilde{c}_t,$$

where W_c, and b_c are the weight and the bias of the candidate layer, and $\tilde{\kappa}(\cdot)$ is a *tanh* function for the memory content update value. The output gate unit is updated as following

$$o_t = \kappa(W_o * [h_{t-1},\ x_t] + b_o),$$

where W_o and b_o are the weight and bias of the output layer. The final output of short term memory h_t is obtained by the following equation

$$h_t = o_t \circ \tilde{\kappa}(c_t).$$

Where $[\cdot, \cdot]$ is the concatenation operator that merges two tensors, while $*$ is the matrix multiplication operator and $\circ$ is the Hadamard (element-wise) product [99]. The latent variable vector obtained from the LSTM at the tth round of network update is denoted by h_t, which is the tth element of the parameter temporal feature tensor h. The full set of the LSTM parameters is updated by performing stochastic gradient descent at each time step t.

Let real and reactive power of the model for $\boldsymbol{\alpha}_{\mathrm{c},i}$ be $\boldsymbol{P}_{\mathrm{m},i} = [p^1,\ p^2,\cdots, p^K]^\top$, and $\boldsymbol{Q}_{\mathrm{m},i} = [q^1,\ q^2,\cdots, q^K]^\top$. To use the LSTM, the inputs and outputs should be the time series data. Then, in parameter estimation, a vector of parameters should be provided. For this reason, the following vector is constructed $\boldsymbol{\alpha}_{\mathrm{c},i}$, with a size of K. Since the parameters of the generator are constant during the estimation, in this CHAPTER , it is assume that for $\boldsymbol{z}_i(\boldsymbol{\alpha}_{\mathrm{c},i})$, the parameter vectors is $\boldsymbol{\alpha}_{\mathrm{c},i} = [\boldsymbol{\alpha}^1_{\mathrm{c},i},\ \boldsymbol{\alpha}^2_{\mathrm{c},i},\cdots,\ \boldsymbol{\alpha}^K_{\mathrm{c},i}]^\top$, and the values of that are constant, i.e, $\boldsymbol{\alpha}^1_{\mathrm{c},i} = \boldsymbol{\alpha}^2_{\mathrm{c},i} = \cdots = \boldsymbol{\alpha}^K_{\mathrm{c},i}$. A

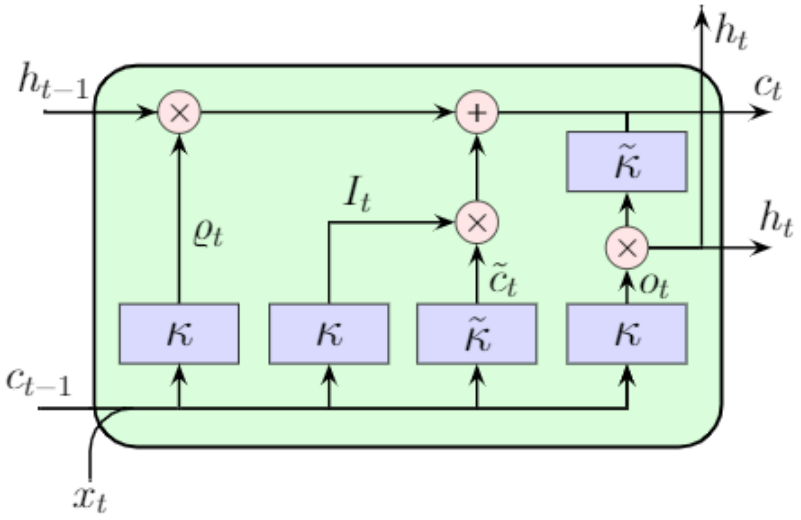


Figure 5.2: Structure of the LSTM block.

time-series parameter vector for the model under study, in addition to the time series of real power and reactive power, can be used to estimate its parameters.

Based on (5.1) and (5.2), some parameters have more impacts on the real power, denoted by $\boldsymbol{\alpha}_{\mathrm{c}}^{\mathrm{p}}$, and some of them have more effects on the reactive power, denoted $\boldsymbol{\alpha}_{\mathrm{c}}^{\mathrm{q}}$. With having N dynamic simulations, let define $\boldsymbol{\mathcal{P}}_{\mathrm{m}}$ which contains N model real power, i.e $\boldsymbol{\mathcal{P}}_{\mathrm{m}} = [\boldsymbol{P}_{\mathrm{m},1}, \boldsymbol{P}_{\mathrm{m},2}, \cdots, \boldsymbol{P}_{\mathrm{m},N}]^{\top}$, and $\boldsymbol{\mathcal{Q}}_{\mathrm{m}}$ which contains N model reactive power, i.e $\boldsymbol{\mathcal{Q}}_{\mathrm{m}} = [\boldsymbol{Q}_{\mathrm{m},1}, \boldsymbol{Q}_{\mathrm{m},2}, \cdots, \boldsymbol{Q}_{\mathrm{m},N}]^{\top}$, and and the parameter as $\boldsymbol{\Theta}_{\mathrm{c}} = [\boldsymbol{\alpha}_{\mathrm{c},1}, \boldsymbol{\alpha}_{\mathrm{c},2}, \cdots, \boldsymbol{\alpha}_{\mathrm{c},N}]^{\top}$. Based on the engineering experience [102], the steady-state mismatch can be determined strongly by the turbine governor model. Some parameters such as T_6 and T_5 only shift the total waveform of the generator outputs. Then, in this project, another feature extracted from the PMU measurements is used, which only reflects the impacts of these parameters to help the algorithm identify the parameters more accurately. This feature is derived as follows. Let define $\boldsymbol{P}_{\mathrm{md},i} = [p^1, p^2 - p^1,\ p^3 - p^2, \cdots, p^K - p^{K-1}]^{\top}$, and $\boldsymbol{Q}_{\mathrm{md},i} = [q^1, q^2 - q^1,\ q^3 - q^2, \cdots, q^K - q^{K-1}]^{\top}$, then two additional features, $\boldsymbol{\mathcal{P}}_{\mathrm{md}}$ and $\boldsymbol{\mathcal{Q}}_{\mathrm{md}}$ are used as the inputs for training the model which are defined

as follows. $\boldsymbol{\mathcal{P}}_{\mathrm{md}} = [\boldsymbol{P}_{\mathrm{md},1}, \boldsymbol{P}_{\mathrm{md},2}, \cdots, \boldsymbol{P}_{\mathrm{md},N}]^\top, \boldsymbol{\mathcal{Q}}_{\mathrm{md}} = [\boldsymbol{Q}_{\mathrm{md},1}, \boldsymbol{Q}_{\mathrm{md},2}, \cdots, \boldsymbol{Q}_{\mathrm{md},N}]^\top$. Two parallel multimodal LSTMs (M-LSTM) were built to build a comprehensive model. One M-LSTM is for active power, LSTM-A, and one M-LSTM for reactive power, LSTM-R. The outputs of the LSTM-A are the parameters of the system that have more effects on the active power, whereas those of the LSTM-R are the parameters that have more effects on the reactive power.

The inputs tensors for LSTM-A considered as $\boldsymbol{\mathcal{P}}_{\mathbf{A}} = [\boldsymbol{\mathcal{P}}_{\mathbf{m}}, \boldsymbol{\mathcal{P}}_{\mathbf{md}}]^\top \in \mathbb{R}^{N*K}$, and $\boldsymbol{\mathcal{Q}}_{\mathbf{R}} = [\boldsymbol{\mathcal{Q}}_{\mathbf{m}}, \boldsymbol{\mathcal{Q}}_{\mathbf{md}}]^\top \in \mathbb{R}^{N*K}$ as inputs for LSTM-R. The maximum lag values for the time series active and reactive power are considered as t_{p} and t_{q}. At each training time step t, LSTM-A observes $\boldsymbol{\rho}_{t_{\mathrm{p}} \times \eta}$ which is from $\boldsymbol{\mathcal{P}}_{\mathbf{A}}$ and includes $\boldsymbol{\rho}_{t_{\mathrm{p}}}$ to $\boldsymbol{\rho}_1$; and ith LSTM input is a tensor with size $1 * \eta$ and the corresponding output is h_i^{ρ}. Similarly, LSTM-R observes $\boldsymbol{\sigma}_{t_{\mathrm{q}} \times \zeta}$ from $\boldsymbol{\mathcal{Q}}_{\mathbf{R}}$ that includes $\boldsymbol{\sigma}_{t_{\mathrm{q}}}$ to $\boldsymbol{\sigma}_1$ and ith LSTM input is a tensor with size $1 * \zeta$ and the corresponding output is h_i^{σ}. When all the input vectors in LSTM-A and LSTM-R are observed, the average temporal latent variables at time step t is computed as follows [97]:

$$\psi_t^{\boldsymbol{\rho}} = \frac{1}{t_{\mathrm{p}}} \sum_{k=0}^{t_{\mathrm{p}}} h_k^{\boldsymbol{\rho}}, \tag{5.4}$$

$$\psi_t^{\boldsymbol{\sigma}} = \frac{1}{t_{\mathrm{q}}} \sum_{k=0}^{t_{\mathrm{q}}} h_k^{\boldsymbol{\sigma}}, \tag{5.5}$$

where $h_k^{\boldsymbol{\rho}}$ and $h_k^{\boldsymbol{\sigma}}$ are the latent variable vectors for active/reactive power obtained from the LSTM-A and LSTM-R at the kth round of network update. At each time step t, the temporal averages of the parameters set are fed into the linear regression models, $\phi_\rho(\cdot)$ and $\phi_\zeta(\cdot)$, and the parameters at t are estimated. The structure of PM-LSTM is shown in Fig. 5.3. Note that in the PM-LSTM model, the number of LSTM layers in LSTM-A is t_{p}, and the number of LSTM layers in LSTM-R is t_{q}. With having the t_{p} and t_{q} and defining the maximum number of the epoch $T_{\max}$ and the number of training step T_{s} to tune the proposed model, the algorithm of the PM-LSTM can be written as

Algorithm 4.

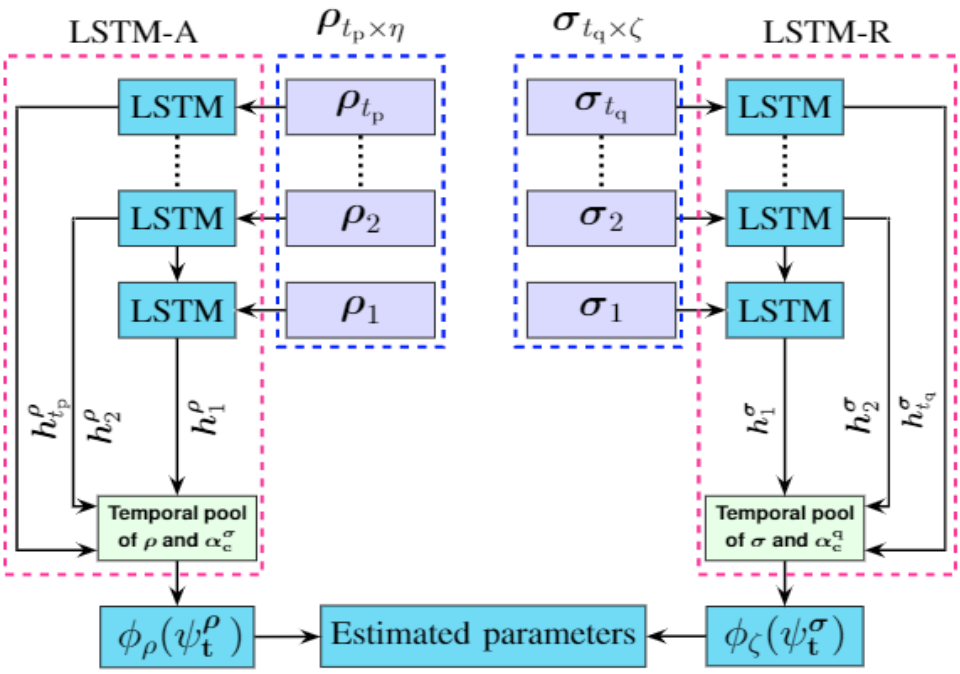


Figure 5.3: The proposed PM-LSTM for generator parameter estimation.

Simulation Results

The proposed method is tested based on power system models, which are implemented in PSS/E software. The same hydro-generator model and PMU data in [75] are used. A PMU is installed at the 230-kV side of the substation with a sampling rate of 30 samples/s. Two metrics are used to evaluate the performance of different methods, including: mean absolute error (MAE), mean squared error (MSE).

Algorithm 3: Proposed PM-LSTM model for generator parameter estimation.

```
Set the hyper parameters; number of epoch = T_max, number of training step = T_s
Find α_c^p and α_c^q based on (5.1) and (5.2)
Vectorize the training data
Extract the input features
Set the time steps t_p and t_q for the real and reactive power based on (27)
Set the training step t = 1, round steps k, k' = 1, and training epoch i = 1
while i ≤ T_max do
    while t ≤ T_s do
        do in parallel
            while k ≤ t_p do
                Feed ρ_k to the LSTM-A block
                Obtain the h_k^ρ
                k ← k + 1
            end while
            while k' ≤ t_q do
                Feed σ_k' to the LSTM-R block
                Obtain the h_k'^σ
                k' ← k' + 1
            end while
        end
        Compute the ψ_t^ρ, ψ_t^σ based on (5.4) and (5.5)
        Compute the estimated parameter with linear regression, φ(ψ_t^ρ) and φ(ψ_t^σ)
        Update the weight matrices
        t ← t + 1
    end while
    i ← i + 1
end while
```

Critical Parameter Identification

For sensitivity analysis, a small perturbation $\Delta\alpha_i = 5\,\%|\alpha_i|$ is applied to each parameter. From the trajectory sensitivity, it is found that only the parameters shown in Table 5.1 are the problematic parameters for the system among all the synchronous generator parameters. The top eighteen critical parameters and their active and reactive power sensitivities are listed in Table 5.1. Based on Table 5.1, for the two parameters case, H is the output of LSTM-A, and K_{A} is the output of LSTM-R. For the eighteen parameters case, the parameter set $\{T_6, K_{\mathrm{s}}, T_5, K_{\mathrm{A}}, T_{\mathrm{b}}, T'_{\mathrm{do}}, T_3, T_1, X_{\mathrm{d}}\}$

is the output of LSTM-R, and $\{a_{23}, H, T_{\rm c}, R_{\rm T}, X'_{\rm d}, X'_{\rm q}, T_{\rm R}, R_{\rm P}, A_1\}$ set is estimated with LSTM-A.

Table 5.1: Top Eighteen Critical Parameters of the Model

Parameter	$\mathbf{S_P}(\alpha_\mathbf{i})$	$\mathbf{S_Q}(\alpha_\mathbf{i})$	**Parameter**	$\mathbf{S_P}(\alpha_\mathbf{i})$	$\mathbf{S_Q}(\alpha_\mathbf{i})$
T_6	0.13	26.26	$X_{\rm d}$	0.02	0.70
$K_{\rm s}$	0.11	26.25	$T_{\rm c}$	0.60	0.20
T_5	0.12	22.11	H	0.58	0.05
$K_{\rm A}$	0.1	1.68	$R_{\rm T}$	0.53	0.02
$T_{\rm b}$	0.07	1.23	$X'_{\rm d}$	0.21	0.45
a_{23}	1.01	0.05	$X'_{\rm q}$	0.41	0.13
$T'_{\rm do}$	0.14	0.94	$T_{\rm R}$	0.40	0.02
T_3	0.12	0.77	$R_{\rm P}$	0.30	0.01
$T1$	0.1	0.77	A_1	0.04	0.03

Dataset

To collect the training and test data for the two-parameter case, a case similar to the test B in [42,43] is considered. Specifically, the mean values, $\mu_\pi(\boldsymbol{\alpha}_{\rm c})$, of the prior distributions, $\pi(\boldsymbol{\alpha}_{\rm c})$, is set as 10 % greater than the true values and choose the lower/upper bounds of the uniform prior distributions as 90 % less/greater than the $\mu_\pi(\boldsymbol{\alpha}_{\rm c})$ to account for the parameter uncertainties. For eighteen parameter case, the $\mu_\pi(\boldsymbol{\alpha}_{\rm c})$ is set as 10 % greater than the true values and the lower/upper bounds of the $\pi(\boldsymbol{\alpha}_{\rm c})$ are set as 50 % less/greater than the $\mu_\pi(\boldsymbol{\alpha}_{\rm c})$. Multiple parameters are independent, identically distributed random chosen, and with event playback, dynamic generator data generated automatically respective to the critical parameters. Based on the test cases, different simulations number for training the network are used. In this CHAPTER , for two parameters case, 3×10^3, and for eighteen parameters, 3×10^4 dynamic simulations are collected to train, validate, and test

the model. 90 % of the available data is used for training and cross-validation, and 10 % is used for testing. For each simulation, the time horizon is set as 20 s. The sampling time is set as $0.033\,\mathrm{s}$, since the reporting rate of PMU measurement is assumed to be 30 samples/s.

Hyper-parameter Setting

In this project, a grid search optimization method via scikit-learn [103] is used to find the network hyper-parameters such as batch size, hidden neurons, learning rate, etc. The learning rate is set to 10^{-3}. Besides, to avoid over-fitting, L_2 regularization with the regularization coefficient $\lambda = 2$ is used for the dense layers. All the LSTM layers have 100 neurons with the *tanh* and *linear* activation function. The values of the η and ζ are equal to 2. The proposed model is implemented in Python 3.8 and all case studies using the deep learning technique are carried out using a high-level neural networks API, Keras [104]. In this CHAPTER , a supervised learning algorithm is used to find the time lags' optimal values t_{p} and t_{q} which will be explained in the following section.

Time Steps for the Input Features

In each time step, the PM-LSTM uses predefined values for t_{p} and t_{q}. Generally, time lags can not be too large since the value of the current input have strong relationships with those in a short-term due to the time series characteristics. The model is evaluated on the testing dataset to assess the performance of model with the chosen t_{p} and t_{q} in terms of MAE and MSE metrics. Fig. 5.4 shows the MAE and MSE values of PM-LSTM for the model with two parameters, as a function of the number of the time steps t_{p} and t_{q}, respectively. The deeper colors mean, smaller MAE and MSE values and better performance of the time steps accordingly. As it can be seen, for two parameters case, the optimal values for t_{p} and t_{q} are 6 and 4.

Fig. 5.5 shows the MAE and MSE values of PM-LSTM for the model with eighteen key parameters as a function of the number of the time steps t_{p} and t_{q}. As it can be seen, for this case, the optimal values for t_{p} and t_{q} is 8. In this project, these values for the time lag of real and reactive power are considered.

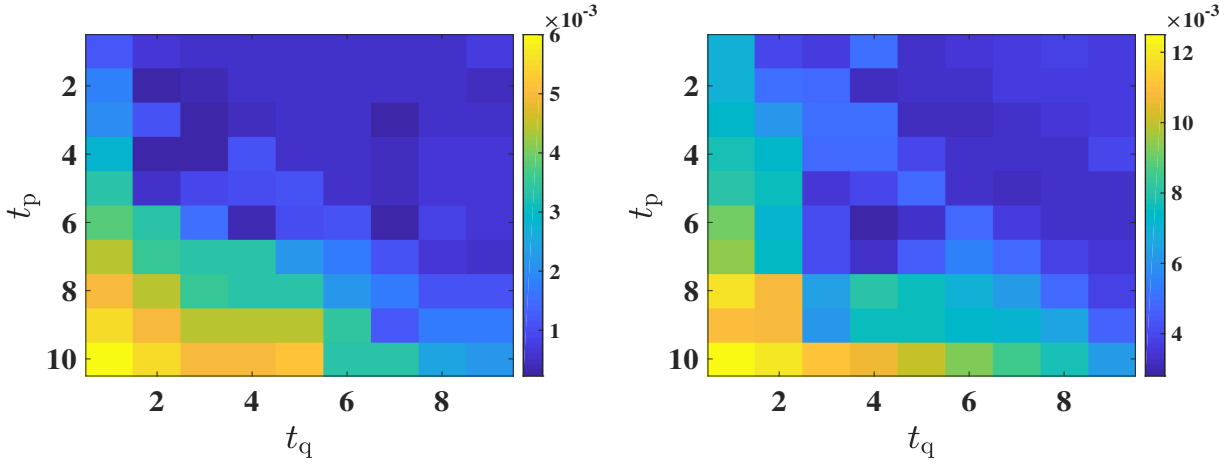


Figure 5.4: Validation of metrics for two parameters for different time lags; (a) MSE; (b) MAE.

Calibration of Two Parameters

The moment of inertia H and the amplifier gain of the exciter K_{A} are chosen to validate the proposed methods, same as [41, 42]. It is evident from Table 5.1 that K_{A} has more impacts on the reactive power, while H influences more the active power. Table 5.2 shows the prior distributions, true values, and mean of the posterior distributions as the estimated values for a two parameters case with the true values, $H^{\mathrm{t}} = 5.4$ and $K_{\mathrm{A}}^{\mathrm{t}} = 125$. As it can be seen, the well-trained model can effectively estimate the parameters. For the sake of clarity, ten different samples for time series active/reactive power with the estimated and the true values of the parameters for one specific event

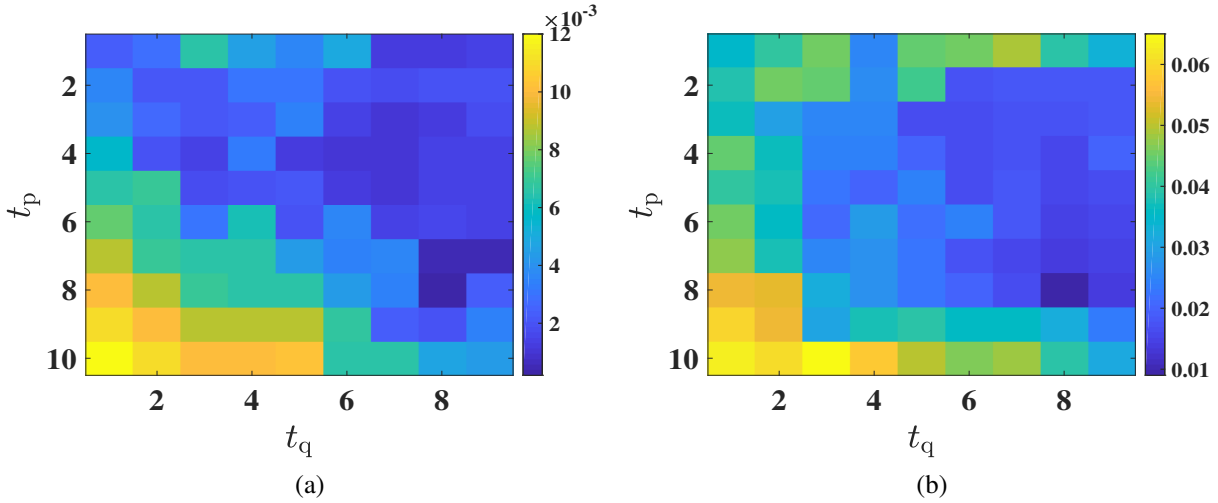


Figure 5.5: Validation of metrics for eighteen key parameters for different time lags; (a) MSE; (b) MAE.

are provided in Fig. 5.6.

Table 5.2: Comparison of the Estimated Values with True Values for Two Parameters Case

H ($H^{\mathrm{t}} = 5.40$)		K_{A} ($K_{\mathrm{A}}^{\mathrm{t}} = 125$)	
$\pi(H)$	**Estimated (Error (%))**	$\pi(K_{\mathrm{A}})$	**Estimated (Error (%))**
$\mathcal{U}(0.59,\ 11.28)$	5.41 (0.1)	$\mathcal{U}(13.75, 261.25)$	124.9 (0.08)

Higher-Dimensional Case

This case study demonstrates the performance of the trained model in a higher-dimensional case with eighteen critical parameters. The prior distributions for these parameters for training the model are assumed to have a $\mu_\pi(\alpha_{\mathrm{c}})$ with 10 % deviation from the true values and follow a uniform distribution as $\mathcal{U}\Big(\boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_{\mathrm{c}}) - \boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_{\mathrm{c}}) \cdot 50\,\%,\ \ \boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_{\mathrm{c}}) + \boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_{\mathrm{c}}) \cdot 50\,\%\Big)$. Table 5.3 lists the true and

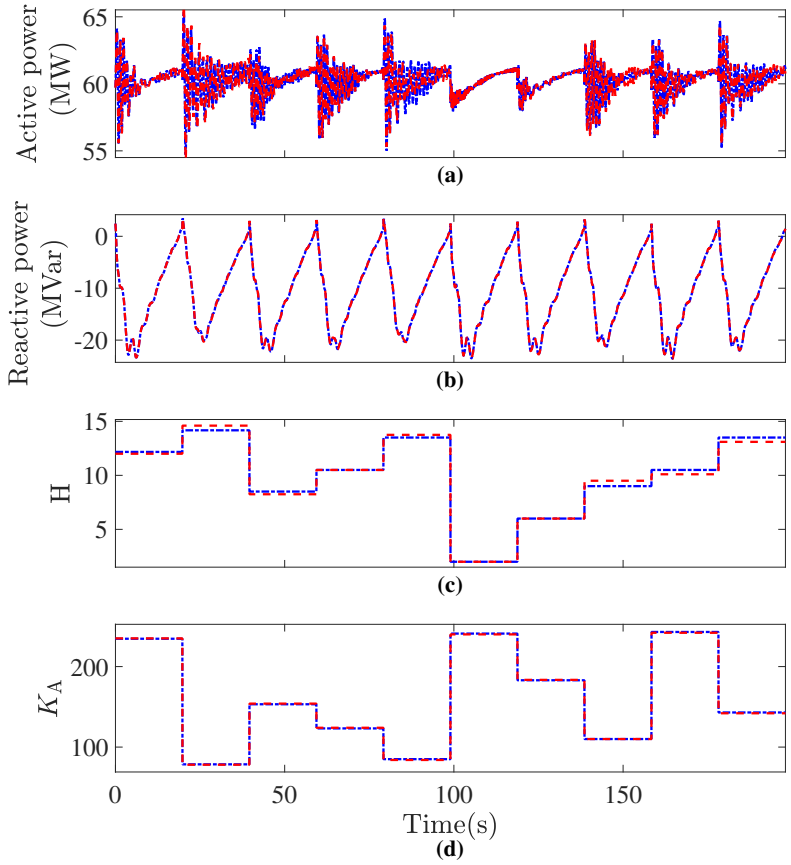


Figure 5.6: Time domain curves of (a) active power; (b) reactive power; (c) H; and (d) K_{A}.

estimated values and the estimation errors verifies the well-trained model has acceptable accuracy for high dimensional parameters.

In real application, true model parameters are never known. It is possible that for different events, the algorithm may bring different parameter packs. In addition to engineering judgment and experience that can help choose reasonable parameters [19, 105], techniques can also be developed based on the available multiple events to help select the best parameters. For example, one can consider one event to estimate the parameters and use the other events to cross-validate the esti-

Table 5.3: Calibration of Eighteen Key Parameters

Parameter	**True value**	**Estimated** (% Error)	**Parameter**	**True value**	**Estimated** (% Error)
T_6	10	9.99 (0.1)	X_{d}	0.57	0.57 (0)
K_{s}	20	20.09 (0.4)	T_{c}	0.90	0.89 (0.6)
T_5	10	10.03 (0.3)	H	5.4	5.41 (0.1)
K_{A}	125	124.83 (0.1)	R_{T}	0.42	0.42 (0)
T_{b}	3.86	3.86 (0)	X'_{d}	0.25	0.25 (0)
a_{23}	1.102	1.101 (0.2)	X'_{q}	0.32	0.32 (0)
T'_{do}	5.4	5.41 (0.1)	T_{R}	1	1 (0)
T_3	0.15	0.15 (0)	R_{P}	0.01	0.01 (0)
$T1$	0.15	0.154 (2.6)	A_1	0.035	0.0345 (1.4)

mated parameters. In this project, it assumes that measurements for four events are available and trained the model with two different events. The remaining two events, were used to estimate the system parameters. The parameters are estimated with each one of the two events and calculate the MSE metric for the other two events. The parameter set with the smallest MSE is selected as the estimated parameters. Fig. 5.7 shows the model validation results under two events with the original parameters and the estimated best parameters and $\boldsymbol{\mu}_{\pi}(\boldsymbol{\alpha}_{\mathrm{c}})$. It is seen that the mismatch between model outputs and PMU measurements is significant with $\boldsymbol{\mu}_{\pi}(\boldsymbol{\alpha}_{\mathrm{c}})$, while the model outputs with the selected estimated parameters can match the PMU measurements very well for the two events.

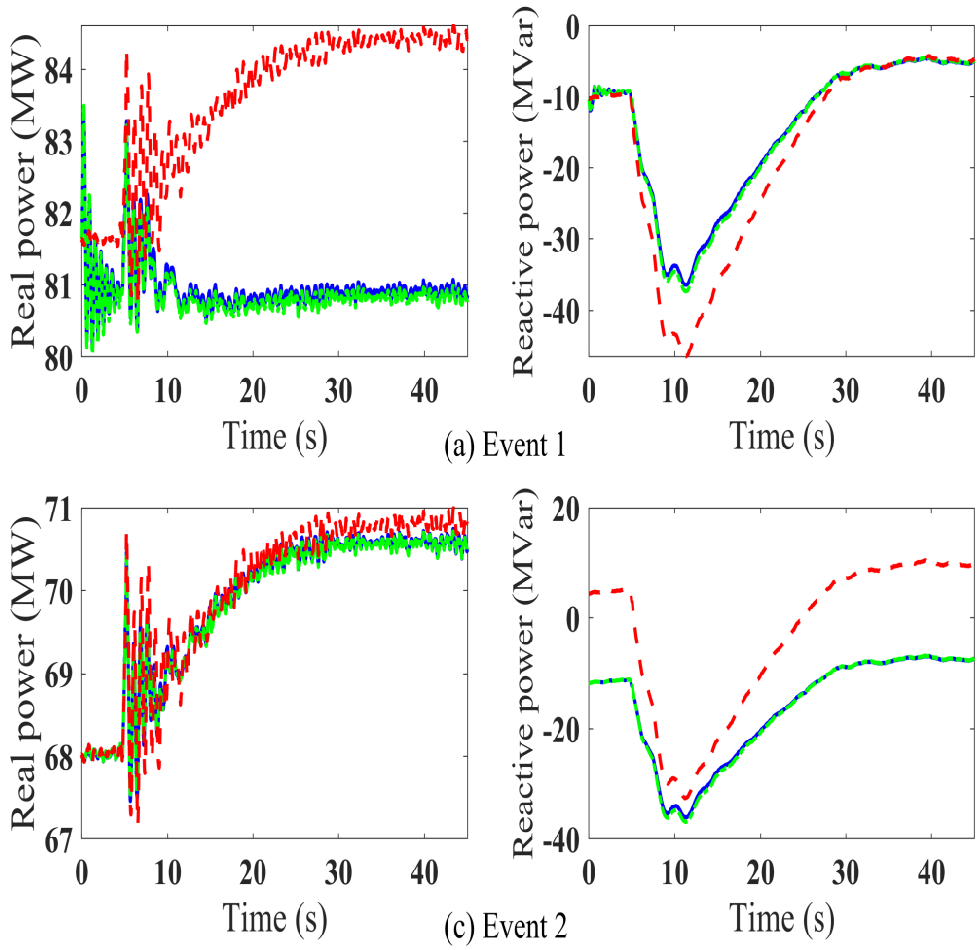
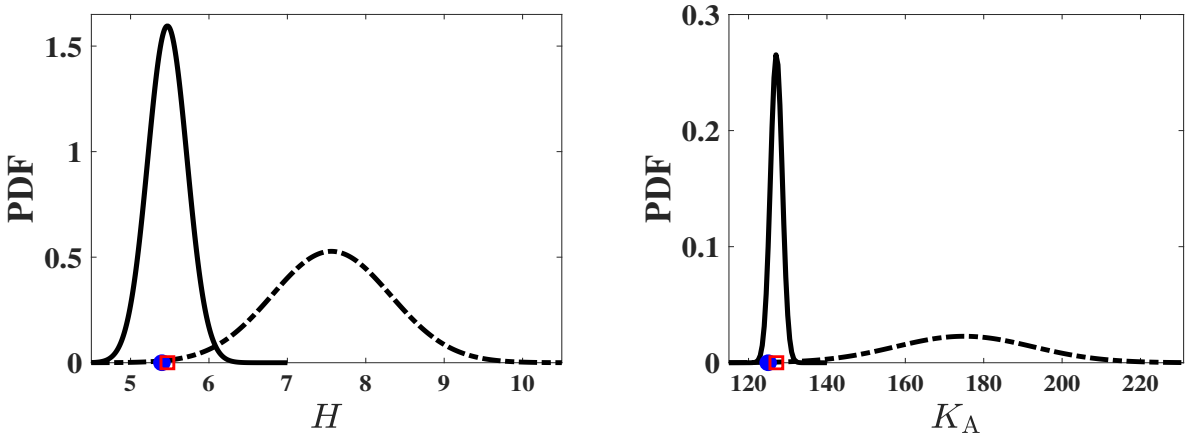


Figure 5.7: Model outputs before and after parameter calibration: (a) Event 1; (b) Event 2. — PMU measurements; - - model ouputs before calibration; . model outputs after calibration.

Tolerance Test and Performance Comparison

In this section, the proposed method will be tested under different deviations in the parameters, and its performance will be compared with the benchmark methods. For this purpose, more extreme cases are considered in which the prior distribution is a Gaussian distribution whose mean has a ϑ % deviation from the true values and whose standard deviation is 10 %, 30 %, and 60 % of the

mean values. All the other settings are the same for all simulations. Table 5.4 shows the results for ϑ equal to 20 %, 30 %, and 40 %, in which the mean of the prior distribution is shown by $\mu_\pi(\alpha_{\rm c})$. This table shows that, even when the mean of the priors have considerable deviations from the true values and the standard deviations of the priors are relatively small, the proposed approach still has high accuracy. For the first highlighted row in Table 5.4, the prior/posterior distributions, and the estimated/true values of the parameters are shown in Fig. 5.8. It is seen that even when the true values are at the tail of the assumed prior distributions, the proposed method can still accurately estimate the parameters. In contrast, the proposed algorithm in [41] has an error greater than 25 % for the same deviations and priors. Moreover, the proposed method in [41] for the case where the standard deviation is large and the parameters are in support of the prior distribution has an error greater than 18 %. In contrast, the proposed method has an error less than 1 % as shown in the second highlighted row.



(a) H (b) $K_{\rm A}$

Figure 5.8: Prior and posterior distributions, and the true and estimated values for (a) H and (b) $K_{\rm A}$. **- -** Prior distribution, **–** posterior distribution, □ estimated values, • true values.

To compare the proposed method with the method in [43], which considers the gross error in the prior distributions, a case similar to the test B in [43] is considered. Specifically, the mean value

Table 5.4: Calibration of H and K_A for Different ϑ and $\mu_\pi(\cdot)$

Standard Deviation (10%)						
ϑ	H			K_A		
	$\mu_\pi(H)$	Estimated	Error (%)	$\mu_\pi(K_\mathrm{A})$	Estimated	Error (%)
40 %	**7.56**	**5.42**	**0.4**	**175**	**126.1**	**0.8**
30 %	7.02	5.39	0.2	162.5	126.1	0.4
20 %	6.48	5.39	0.2	150	125.4	0.3
Standard Deviation (30 %)						
ϑ	H			K_A		
	$\mu_\pi(H)$	Estimated	Error (%)	$\mu_\pi(K_\mathrm{A})$	Estimated	Error (%)
40 %	**7.56**	**5.38**	**0.3**	**175**	**124.1**	**0.7**
30 %	7.02	5.38	0.3	162.5	124.6	0.3
20 %	6.48	5.41	0.2	150	124.7	0.2
Standard Deviation (60 %)						
ϑ	H			K_A		
	$\mu_\pi(H)$	Estimated	Error (%)	$\mu_\pi(K_\mathrm{A})$	Estimated	Error (%)
40 %	7.56	5.42	0.4	175	125.3	0.2
30 %	7.02	5.39	0.2	162.5	124.8	0.2
20 %	6.48	5.42	0.3	150	124.7	0.2

of the prior distribution for H is considered as 61 % less than the true value and that for K_A as 225 % greater than the true value, and the lower/upper bounds of the uniform prior distributions is considered as 90 % less/greater than the mean values. The performance of the proposed method in Table 4.6. The performance of the proposed method in Table 4.6 shows that the proposed model can estimate the parameters with high accuracy. The estimation error of the proposed approach for H is $0.05\,\%$ and for K_A is $0.3\,\%$, which are less than the corresponding errors, $0.08\,\%$ and $0.5\,\%$

respectively, in [43].

Table 5.5: Parameter Calibration Under Bad Prior Distribution

$H(H^{\mathrm{t}} = 5.40)$		$K_{\mathrm{A}}(K_{\mathrm{A}}^{\mathrm{t}} = 125)$	
$\pi(H)$	**Estimated** (% Error)	$\pi(K_{\mathrm{A}})$	**Estimated** (% Error)
$\mathcal{U}$(0.9, 17.2)	5.403 (0.05)	$\mathcal{U}$(28.1, 534.4)	125.44 (0.3)

To compare the proposed method with the Q-learning method in [47], the following settings are considered. The mean value of the prior distribution is set as 10 percent greater than the true values, and the lower/upper bounds of the uniform prior distributions are set as 70% less/greater than the mean values. Table 5.6 shows the performance of the proposed method. It can be seen that the proposed method can estimate parameters with high accuracy. The estimation error of the proposed approach for H is 0% and for K_{A} is 0.2%, which are less than the corresponding errors, 0.6% and 0.9% reported in [47].

Table 5.6: Parameter Calibration under Uniform prior Distribution

$H(H^{\mathrm{t}} = 5.40)$		$K_{\mathrm{A}}(K_{\mathrm{A}}^{\mathrm{t}} = 125)$	
$\boldsymbol{\pi}(\mathbf{H})$	**Estimated** (% Error)	$\boldsymbol{\pi}(\mathbf{K}_{\mathrm{A}})$	**Estimated** (% Error)
$\mathcal{U}$(1.8, 10.1)	5.40 (0)	$\mathcal{U}$(41.2, 233.8)	125.3 (0.2)

A similar case to [48] is used here to compare the proposed method with the deep Q-learning method. The same parameters with the same prior distributions are considered as well. Table 5.7 shows that maximum estimation error of the proposed method is 0.4%, which is much less than the 8% error value reported for the deep Q-learning method [48].

Table 5.7: Calibration of Five Parameters under Uniform prior Distribution

Parameter	**True value**	**Prior distribution**	**Estimated** (% Error)
H	5.4	$\mathcal{U}(0.86, 8.58)$	5.40 (0)
X'_{d}	0.25	$\mathcal{U}(0, 0.71)$	0.249 (0.8)
X'_{q}	0.32	$\mathcal{U}(0, 0.58)$	0.318 (0)
K_{A}	125	$\mathcal{U}(75, 175)$	124.5 (0.4)
T_{A}	0.02	$\mathcal{U}(0, 1)$	0.02 (0)

To compare the proposed method with the A-ABC-SMC in [11], a high dimensional case with eighteen critical parameters is considered. For the prior distribution, the mean value is set as 10 percent greater than true values and the lower/upper bounds of the uniform prior distributions are set as 50 % less/greater than the mean values. The real and reactive power for the parameter estimated by the A-ABC-SMC approach and the outputs for the true values of the parameters are shown in Fig. 5.9. It can be seen that the outputs of the proposed approach have smaller discrepancies than the outputs of the A-ABC-SMC approach [11].

The performance and accuracy of the proposed PM-LSTM model is compared with different methods. Table 5.8 shows the normalized MSE and MAE metrics for gated recurrent unit (GRU) [106], single LSTM (S-LSTM), multimodal LSTM (M-LSTM), and parallel single LSTM (P-LSTM) methods. The dataset and the hyper-parameters for all the methods are the same. The estimation error shows that the proposed PM-LSTM has the smallest metrics between all the methods and can significantly increase the estimation procedure accuracy. Fig. 5.10 shows the model validation results for the estimated parameters with different methods and the original parameters for

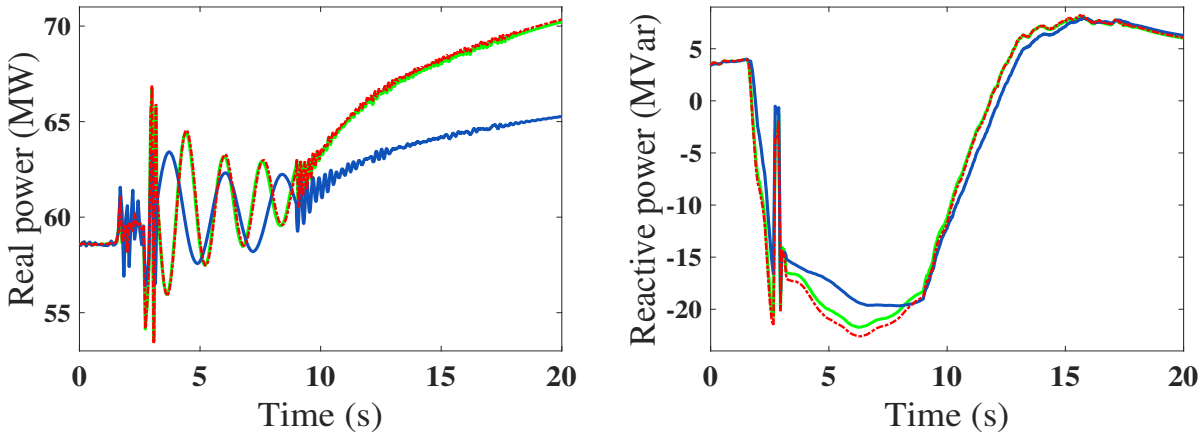


Figure 5.9: Model outputs from [11] method and the proposed method; (a) Real power and (b) reactive power. -. Measurements, – Calibration with method in [11], — Calibration with the proposed method.

one specific event. It is seen that the mismatch between model outputs and PMU measurements significantly decreases with the proposed model. Moreover, due to the high sensitivity values of the reactive power parameters, the mismatches between the reactive power curves are much more apparent than the active power curves in Fig. 5.10.

Table 5.8: Error Metrics of the Different Methods for High-Dimensional Case

Metric	**GRU**	**S-LSTM**	**M-LSTM**	**P-LSTM**	**PM-LSTM**
MAE (%)	4.52	5.13	4.55	4.73	1.11
MSE (%)	5.82	7.65	5.24	5.46	1.93

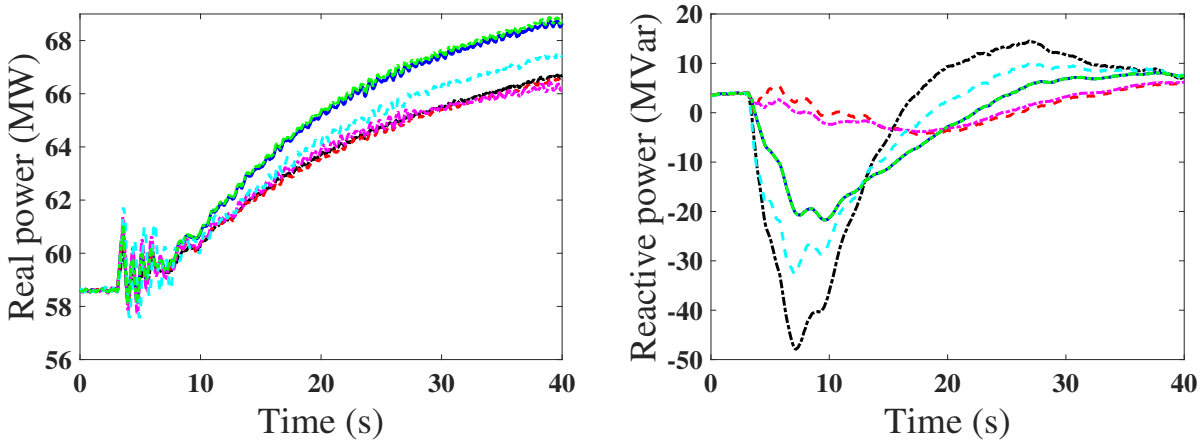


Figure 5.10: Model outputs for the different methods: (a) Real power; (b) Reactive power. — PMU Measurement; —. GRU — — S-LSTM; —. P-LSTM; — — M-LSTM; and —. PM-LSTM.

Time Efficiency

A Windows server with Intel(R) Core(TM) i7-8700 and 16 GB memory was used to generate the simulation data and training the model. Offline training of the PM-LSTM model took 2 hours for two parameters and around 18 hours for eighteen parameters. Online testing of the well-trained model with the new measurements take less than 5 seconds for each test case.

Conclusions

This CHAPTER presents a method to validate stability models, identify critical model parameters, and calibrate them using PMU measurements. With a sensitivity approach, critical parameters of the generator are identified. A PM-LSTM model is trained based on the event playback using different dynamic simulations. The robustness of the proposed model is tested on two and eighteen critical parameter cases. Simulation results indicate that the proposed method can accurately and

efficiently estimate the full distributions of the parameters.

CHAPTER 6: POWER PLANT MODEL PARAMETER CALIBRATION USING CONDITIONAL VARIATIONAL AUTOENCODER

S. R. Khazeiynasab, J. Zhao, I. Batarseh and B. Tan, "Power Plant Model Parameter Calibration Using Conditional Variational Autoencoder," in IEEE Transactions on Power Systems, doi: 10.1109/TPWRS.2021.3107515.

Introduction

Accurate models of power plants play an important role in maintaining the reliable and secure grid operations. In this CHAPTER , we propose a synchrophasor measurement-based generator parameter calibration method by a novel deep learning method with high computational efficiency. An elementary effects-based approach is developed to identify the critical parameters from a nonlinear system with much better performance than the widely used trajectory sensitivity-based method. Then, synchrophasor measurement-based conditional variational autoencoder is developed to estimate the parameter posterior distributions even in the presence of a high-dimensional case with eighteen critical parameters to be calibrated. The effectiveness of the proposed method is validated for a hydro generator with very detailed model. The results show that the proposed approach can accurately and efficiently estimate the generator parameter posterior distributions even when the parameters true values are not in support of the prior distribution.

This CHAPTER proposes a conditional variational autoencoder (CVAE) framework for power system model calibration. CVAE defines a set of nonlinear latent variables to describe the probability distribution function (PDF) of the model parameters using an encoding-decoding architecture. According to the real/reactive power measurements through Artificial Neural Networks (ANNs), the

encoder estimates the PDF of latent variables. The decoder maps the latent variables PDFs into actual model parameter PDFs using a Rectified Linear Unit (ReLU) ANN.

Compared to the existing methods, the characteristics of the proposed method are as follows. 1) Only a small number of samples is needed and the computational efficiency is high; 2) It can provide accurate parameters when the generator model has significant model discrepancies due to gross errors in the parameters, and even when prior distribution does not support the true values; 3) It can estimate the high dimensional cases accurately even when the parameter space is large enough; 4) The implementation is more straightforward and reliable without many complications as compared to the existing reinforcement and deep learning methods; 5) The proposed method is simulation-based and does not need a likelihood function or state-space model of the generator. As commercial software already has stability models, implementation is much easier and it can be used in real-world applications. The contributions are summarized as follows:

1. Instead of using a sensitivity-based approach, an elementary effects approach is proposed directly for the nonlinear power system models to more accurately identify the critical parameters with the highest identifiability.

2. We formulated a CVAE inference framework to estimate the full posterior distributions of a decentralized generator model parameters.

3. Insightful case studies are presented and discussed in detail. The impact of the different prior distributions, different lower/upper bounds on the prior distributions, and the error in the parameters initial values are analyzed. A specific case, where the true values are not in support of the prior distributions is studied. Finally, a high-dimensional case with eighteen critical parameters under multiple events is presented.

Proposed Framework

Fig.6.1 illustrates the main flowchart for the proposed framework, which consists of two key modules, model validation/verification, and model calibration. Multiple events are used to check the performance of the system under study. If any model deficiency is identified, the parameter calibration module is used to identify and update inaccurate parameter values. The calibration module includes two main steps: (1) identify the critical parameters (2) automatic parameter calibration based on the CVAE model. In a real application, true model parameters are never known. In addition to engineering judgment and experience that can help choose reasonable parameters, techniques can also be developed based on the available multiple events to help select the best parameters. If a model deficiency is detected again for every additional event, the calibration process will be repeated until it is verified that the model matched the observed data well. More details of each module in Fig. 6.1 are provided in the following sections.

Identifiability of Parameters

A synchronous generator has many parameters. Calibrating all parameters could be computationally challenging, and also not every parameter is identifiable. Therefore, the parameters with the highest identifiability should be first identified [19]. To find the critical parameters, some researchers use the traditional trajectory sensitivity of each parameter (e.g., [19,39]) which is defined as follows:

Sensitivity Based Approach

The sensitivity of P_{m} and Q_{m} with regard to parameter α_i can be calculated as follows:

$$S(\alpha_i)=\sum_{k=1}^{K}\frac{|P_{\mathrm{m},k}(\alpha_i^+)-P_{\mathrm{m},k}(\alpha_i^-)|+|Q_{\mathrm{m},k}(\alpha_i^+)-Q_{\mathrm{m},k}(\alpha_i^-)|}{K(\alpha_i^+-\alpha_i^-)/\alpha_i},$$

where $\alpha_i^+ = \alpha_i + \Delta\alpha_i$ and $\alpha_i^- = \alpha_i - \Delta\alpha_i$, and $\Delta\alpha_i$ is a small perturbation of α_i. The parameters with the largest sensitivities will be considered as having the highest identifiability [19].

Elementary Effects Based Approach

The sensitivity analysis is only locally defined for one operating point, and the nonlinear behavior of the generator model is inevitably lost. The variance-based method requires a high number of model simulations [107]. Here we adapt the elementary effects (EE) method [107, 108] to analyze the sensitivity of the outputs to parameters. The following sets are defined for EE:

$$\begin{aligned}
&\mathcal{T} = \{\boldsymbol{\mathcal{T}}_1,\cdots,\boldsymbol{\mathcal{T}}_r;\boldsymbol{\mathcal{T}}_j \in \mathbb{R}^{v\times v},\ {\boldsymbol{\mathcal{T}}_j}^{\top}\boldsymbol{\mathcal{T}}_j=\boldsymbol{I}_v,\ j=1,\ldots,r\},\\
&M = \{c_1,\cdots,c_G;\ c_m \in \mathbb{R},\ c_m > 0,\ m = 1,\ldots,G\},\\
&O = \{\boldsymbol{o}_1,\cdots,\boldsymbol{o}_v;\boldsymbol{o}_i \in \mathbb{R}^v\},
\end{aligned}$$

where $\mathcal{T}$ defines the initial parameter perturbation direction, r is the number of matrices for perturbation directions, M defines the perturbation sizes, l is the number of perturbation sizes for each direction, $\boldsymbol{I}_v$ is an identity matrix with dimension v, O defines the parameter to be perturbed, and o_i is a zeros but with a unit as ith component. For the ith parameter α_i with fixed initial states $\boldsymbol{x}_0$,

EE can be defined as

$$\text{EE}(\alpha_i) = \sum_{k=1}^{K} \frac{|P_{\text{m},k}(\boldsymbol{\alpha}') - P_{\text{m},k}(\boldsymbol{\alpha}_0)| + |Q_{\text{m},k}(\boldsymbol{\alpha}') - Q_{\text{m},k}(\boldsymbol{\alpha}_0)|}{K c_m},$$

where c_m is a value which is defined based on l-level, $\boldsymbol{\alpha}_0$ is the selected paramater from the prior parameter space distribution, such that the new value $\boldsymbol{\alpha}' = \boldsymbol{\alpha}_0(1 + \boldsymbol{\mathcal{T}}_j o_i c_m)$ is still in the parameter space, and $\mathcal{T}$ is chosen as

$$\mathcal{T} = \{\boldsymbol{I}_v, -\boldsymbol{I}_v\},$$

where I_v is an identity matrix with dimension v and based on $\mathcal{T}$, we perturb each parameter in both positive and negative directions.

The EE method is based on the construction of finite paths in the prior parameter space. The design is based on generating random starting points for each trajectory, then moving one parameter at a time in random order. Then, the distribution of EE corresponding to the ith parameter, $\mathcal{F}_i$, is determined by randomly sampling different $\boldsymbol{\alpha}$ from the parameter space. In this method, we use the Latin Hypercube sampling method [73] to determine the starting points.

Two sensitivity indices are used to determine the critical parameter based on the EE method; the mean and standard deviation of the distribution $\mathcal{F}_i$ are denoted by $\widehat{\mu}$, and $\widehat{\sigma}$ respectively. The $\widehat{\mu}$ indicates the overall influence of the parameter on the output, and $\widehat{\sigma}$ estimates the ensemble of effects of the factor [109]. In this method, based on our numerical experiments, the mean index $\widehat{\mu}$ defined in [107] is used as a metric to measure the overall identifiability of a parameter.

Based on engineering experience, some parameters do not change much during the lifetime of the generator. If that is the case, we can either use this information to assign prior distributions to those parameters properly or avoid calibrating them even if they are identified as critical parameters.

Proposed CVAE Method for Parameter Estimation

Assuming the prior distribution for parameter $\boldsymbol{\alpha}_{\mathrm{c}}$ as $\pi(\boldsymbol{\alpha}_{\mathrm{c}})$. Bayes's theorem allows us to write its posterior distribution $p(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z})$ in terms of the prior distribution and the observation distribution $p(\boldsymbol{z}|\boldsymbol{\alpha}_{\mathrm{c}})$ [85]. The form of this posterior is

$$p(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z}) = \frac{l(\boldsymbol{z}|\boldsymbol{\alpha}_{\mathrm{c}})\pi(\boldsymbol{\alpha}_{\mathrm{c}})}{p(\boldsymbol{z})}, \tag{6.1}$$

where $l(\boldsymbol{z}|\boldsymbol{\alpha}_{\mathrm{c}})$ is the data likelihood, which describes the observation process. $p(\boldsymbol{z})$ is the distribution of all possible measurements and is $p(\boldsymbol{z}) = \int \pi(\boldsymbol{\alpha}_{\mathrm{c}}) l(\boldsymbol{z}|\boldsymbol{\alpha}_{\mathrm{c}}) d\alpha_{\mathrm{c}}$. Note that a closed-form likelihood function is typically unavailable or computationally prohibitively expensive. For the parameter calibration problem, the computational cost of having an explicit likelihood function is a major challenge.

A more straightforward approach is to perform simulations for the model using different parameters, compare the simulated results with the observed data, and estimate the likelihood of a given parameter set to generate outputs that match the observed data. Therefore, some approximate inference approaches have been proposed. Inference through MCMC processes has been proposed as a popular method [110], but these methods are expensive and suffer from slow convergence. Recently, the variational inference methods have been proposed. These techniques are more efficient and have been explored using optimization techniques in various fields [111]. In this CHAPTER , we use the variational approach whose goal is to learn an approximation to the posterior distribution of (6.1) for a new measurement. We call this parametric distribution as $p_{\mathrm{ap}}(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z})$.

Variational Autoencoder

Variational autoencoders (VAE) are directed models with latent variables [112]. The generative process in variational autoencoder is as follows: first, a latent variable $\mathcal{G}$ is generated from the prior distribution $\pi(\boldsymbol{\alpha}_{\mathrm{c}})$, and then the data $\boldsymbol{\alpha}_{\mathrm{c}}$ are generated from the generative distribution $p_{\boldsymbol{\Theta}_{\mathbf{vae}}}(\boldsymbol{\alpha}_{\mathrm{c}}|\mathcal{G})$. In VAE, the $p_{\boldsymbol{\Theta}_{\mathbf{vae}}}(\boldsymbol{\alpha}_{\mathrm{c}}|\mathcal{G})$ modeled by a neural network with parameters Θ_{vae}. A variational lower bound can be optimized efficiently using backpropagation and stochastic gradient descent, and Θ_{vae} is tuned by maximizing the likelihood of the training data points [113]. We do not have control over the process of VAE data generation. Unlike VAE, CVAE models both latent variables and data directly, both conditioned to some random variables.

CVAE for Parameter Estimation

Generator parameter estimation based on measurements is a mathematical representation of the highly nonlinear relationship between parameters and outputs. However, this function cannot be formulated analytically, and the generator parameters cannot be estimated. The CVAE-based method is a powerful technique for representing complex nonlinear relationships. To this end, the CVAE technique is developed to characterize the parameter estimation. Let us define the simulation data as $\Xi = \{\boldsymbol{z}_i\}_{i=1}^{N}$, $\Pi = \{\boldsymbol{\alpha}_{\mathrm{c},i}\}_{i=1}^{N}$, and $\Upsilon = [\Xi, \Pi]$. The goal of CVAE is to fit a model of the conditional probability distribution $p_{\mathrm{ap}}(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z})$ [114].

CVAE contains three important modules, an encoder, latent space, and a decoder. An encoder with parameter θ_{en} gets the input $\boldsymbol{z}$ and produces a Gaussian distribution $p_{\theta_{\mathrm{en}}}(\mathcal{G}|\boldsymbol{z})$. Before generating the posterior distributions, CVAE first projects the inputs into a lower dimensional space $\mathcal{G}$, called the latent space [115]. The latent space acts as a bottleneck to encourage the model to uncover the salient features [116]. A decoder, parameterized with θ_{de} uses $\boldsymbol{z}$ and samples from latent space to

produce $p_{\theta_{\rm de}}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z},\mathcal{G})$. The distribution of $p(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z})$ is given as follow

$$p(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z}) \approx p_{\rm ap}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z}) = \int_{\mathcal{G}} p_{\theta_{\rm en}}(\mathcal{G}|\boldsymbol{z}) p_{\theta_{\rm de}}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z},\mathcal{G}) d\mathcal{G}. \tag{6.2}$$

To produce $p_{\rm ap}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z})$, the samples from the $\mathcal{G}$ only should produce $\boldsymbol{\alpha}_{\rm c}$, i.e $p_{\theta_{\rm en}}(\mathcal{G}|\boldsymbol{z}) \approx 0$. However, as shown in [115], this is an intractable problem due to the integral over the latent space variable $\mathcal{G}$ and a network cannot be trained directly to do this [116]. To this end, we introduce a recognition network $\kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z},\boldsymbol{\alpha}_{\rm c})$, modelled by an additional neural network and governed by the trainable network parameters $\theta_{\rm re}$, which is a proposal distribution. This function can help select values of $\mathcal{G}$, which are likely producing $\boldsymbol{\alpha}_{\rm c}$. Then (6.2) can be written as follows:

$$p_{\rm ap}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z}) = \int_{\mathcal{G}} \frac{p_{\theta_{\rm en}}(\mathcal{G}|\boldsymbol{z}) p_{\theta_{\rm de}}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z},\mathcal{G})}{\kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z},\boldsymbol{\alpha}_{\rm c})} \kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z},\boldsymbol{\alpha}_{\rm c}) d\mathcal{G} = \mathbb{E}_{\kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z},\boldsymbol{\alpha}_{\rm c})}\Big(\frac{p_{\theta_{\rm en}}(\mathcal{G}|\boldsymbol{z}) p_{\theta_{\rm de}}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z},\mathcal{G})}{\kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z},\boldsymbol{\alpha}_{\rm c})}\Big).$$

The goal is to find the parameters of the network to maximize the log-likelihood of $p_{\rm ap}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z})$ over the dataset, i.e

$$\underset{\boldsymbol{\Theta}_{\rm cvae}}{\operatorname{argmax}} \ \log p_{\rm ap}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z}) = \underset{\boldsymbol{\Theta}_{\rm cvae}}{\operatorname{argmax}} \ \log \mathbb{E}_{\kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z},\boldsymbol{\alpha}_{\rm c})}\Big(\frac{p_{\theta_{\rm en}}(\mathcal{G}|\boldsymbol{z}) p_{\theta_{\rm de}}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z},\mathcal{G})}{\kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z},\boldsymbol{\alpha}_{\rm c})}\Big).$$

where $\boldsymbol{\Theta}_{\rm cvae}$ is the CVAE model parameter, which is $\boldsymbol{\Theta}_{\rm cvae} = \theta_{\rm en} \cup \theta_{\rm de} \cup \theta_{\rm re}$. By using Jensen's inequality, the evidence lower bound (ELBO) is given as

$$\log p_{\rm ap}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z}) = -\mathbb{E}_{\kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z},\boldsymbol{\alpha}_{\rm c})}\Big[p_{\theta_{\rm de}}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z},\mathcal{G})\Big] + \mathfrak{C}(\kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z},\boldsymbol{\alpha}_{\rm c}) \parallel p_{\theta_{\rm en}}(\mathcal{G}|\boldsymbol{z})),$$

where $\mathfrak{C}(\cdot \parallel \cdot)$ is the Kullback-Leibler divergence between two distributions and cannot be negative. Then, we get

$$\log p_{\rm ap}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z}) \geq \text{ELBO}.$$

Now, the ELBO can be optimized via stochastic gradient descent algorithm [114, 117]. The loss function for the network is composed of two terms, the "reconstruction" loss that is a measure of how well the decoder network with distribution $p_{\theta_{\rm de}}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z},\mathcal{G})$ predicts the true parameters $\boldsymbol{\alpha}_{\rm c}$. The Kullback-Leibler distance that measures the similarity between the two distributions, $p_{\theta_{\rm en}}(\mathcal{G}|\boldsymbol{z})$ from the encoder and $\kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z},\boldsymbol{\alpha}_{\rm c})$ from recognition network. The total loss for sample i is as follows:

$$L(\boldsymbol{\alpha}_{{\rm c},i},\boldsymbol{z}_i) = -\mathbb{E}_{\kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z_i},\boldsymbol{\alpha}_{\rm c})}\Big[p_{\theta_{\rm de}}(\boldsymbol{\alpha}_{{\rm c},i}|\boldsymbol{z}_i,\mathcal{G})\Big] + \mathfrak{C}\big(\kappa_{\theta_{\rm re}}(\mathcal{G}|\boldsymbol{z}_i,\boldsymbol{\alpha}_{{\rm c},i}) \parallel p_{\theta_{\rm en}}(\mathcal{G}|\boldsymbol{z}_i)\big). \tag{6.3}$$

Fig. 6.2 illustrates the graphical model of a CVAE during the training and test phases. In Fig. 6.2b, $\Theta_{\rm c}$ are the posterior distributions of the estimated parameters which are drawn from the distribution $p_{\rm ap}(\boldsymbol{\alpha}_{\rm c}|\boldsymbol{z})$. The proposed algorithm for training the CVAE model is presented in Algorithm 5.

Simulation Results

The proposed method is tested using PSS/E software. We use the same hydro-generator model, and PMU data in [75]. A PMU is installed at the 230-kV side of the substation with a sampling rate of 30 samples/s.

Critical Parameter Identification

For sensitivity analysis, a small perturbation $\Delta\alpha_i = 5\,\%|\alpha_i|$ is applied to each parameter. For the EE method, the uncertainty in the parameters is modelled by assuming that each α_i has a uniform distribution as $\mathcal{U}\big(\alpha_i - \alpha_i\cdot 50\,\%, \alpha_i + \alpha_i\cdot 50\,\%\big)$. We set the number of paths as 100 and $G = 5$. Table 6.1 lists the top eighteen critical parameters identified by EE and sensitivity based approaches for the same event, in which the normalized corresponding $\mathrm{EE}(\alpha_i)$ and $S(\alpha_i)$ are presented.

Algorithm 4: Training the proposed CVAE model for generator parameter estimation.

Set the hyper parameters; number of iteration = $T_{\max}$, learning rate, latent space dimension, batch size ($\mathrm{B_s}$)
Find $\boldsymbol{\alpha}_{\mathrm{c}}$ based on Section (6)
Generate training data with the dynamic model regarding the $\boldsymbol{\alpha}_{\mathrm{c}}$
Initialised Θ_{cvae}
while $k \leq T_{\max}$ **do**
    while $t \leq B_{\mathrm{s}}$ **do**
        do in parallel
            Feed the encoder with batch t of Ξ, and compute $p_{\theta^k_{\mathrm{en}}}(\mathcal{G}|\Xi_t)$
            Feed the reconstruction encoder with batch t of Υ, and compute $\kappa_{\theta^k_{\mathrm{re}}}(\mathcal{G}|\Xi_t, \Pi_t)$
        end
        do in parallel
            Encode the output of the encoder into a multivariate Gaussian distribution in the latent space as $\mathcal{N}(\mu_{\mathrm{en}}, \sum_{\mathrm{en}})$
            Encode the output of the reconstruction encoder into a multivariate Gaussian distribution in the latent space as $\mathcal{N}(\mu_{\mathrm{re}}, \sum_{\mathrm{re}})$
            Calculate the Kullback-Leibler distance between two distributions, $\mathcal{N}(\mu_{\mathrm{re}}, \sum_{\mathrm{re}})$ and $\mathcal{N}(\mu_{\mathrm{en}}, \sum_{\mathrm{en}})$
        end
        do in parallel
            Draw a sample from the distribution $\mathcal{N}(\mu_{\mathrm{en}}, \sum_{\mathrm{en}})$
            Feed the decoder with a batch t from Ξ, and the chosen sample and compute $p_{\theta^k_{\mathrm{de}}}(\Pi_t|\Xi_t, \mathcal{G})$
            Calculate the reconstruction loss
        end
        $t \leftarrow t + 1$
    end while
    Calculate the total loss based on (6.3)
    Update the Θ_{cvae}
    $k \leftarrow k + 1$
end while

According to the sensitivity analysis results, X'_{d} is more critical than X_{d}, T_{c}, and H. Table 6.2 shows the total change of the real/reactive power for these parameters under small (5 %) and large (50 %) perturbations. It is observed that the total changes regarding the parameter X'_{d} under the large perturbation are less than others, while the corresponding change for small perturbation is greater than other parameters. This means that this parameter is not more critical than others. The EE method also confirms this, and it is clear from the Table 6.1 that X'_{d} is less critical than the

mentioned parameters.

Table 6.1: Top Eighteen Critical Parameters Identified by Elementary Effects and Sensitivity-Based Approaches

No.	**Elementary Effects**		**Sensitivity**	
	Parameter	$\mathrm{EE}(\alpha_i)$	Parameter	$S(\alpha_i)$
01	K_{S}	1	T_5	1
02	T_5	0.99	K_{S}	0.99
03	T_6	0.91	T_6	0.71
04	K_{A}	0.59	a_{23}	0.48
05	T'_{do}	0.48	K_{A}	0.45
06	H	0.41	T_{b}	0.43
07	T_{b}	0.37	T'_{do}	0.41
08	a_{23}	0.33	X'_{d}	0.38
09	T_c	0.32	H	0.37
10	X_{d}	0.31	T_{c}	0.33
11	T_1	0.18	A_1	0.33
12	T_3	0.17	X'_{q}	0.33
13	A_1	0.08	T_3	0.32
14	X'_{d}	0.07	T_1	0.31
15	X'_{q}	0.07	T_2	0.30
16	R_{T}	0.04	X''_{d}	0.29
17	R_{P}	0.01	X_{d}	0.29
18	T_{R}	0.01	T''_{d0}	0.28

Table 6.2: Four Chosen Parameters Effect on the Outputs Under the Small and Large Disturbances

Parameter	**Total change of real and reactive power**	
	Small perturbation	Large perturbation
X'_{d}	383.5	493.8
X_{d}	377.5	505.8
T_{c}	373.5	515.5
H	380.1	587.6

Choice of Number of Simulations

It is desirable to have as many simulations as possible for high accuracy, however, it will impose a greater computational burden. It is reasonable to estimate the parameters of different dimensions with different numbers of simulations. Fig. 6.3 shows the maximum error for two and eighteen critical parameters under different numbers of simulations. For the two-parameter case, we set the maximum acceptable error to be 0.3%. Based on Fig. 6.3a when the number of simulations is equal to 500, this error criterion will be satisfactory. For eighteen parameters case, we set the maximum acceptable error to be 4%. From Fig. 6.3b, a number of simulations equal to 2×10^3 satisfies this error criterion.

Generation of Dataset and Parameter Settings

Based on the test cases, we use different prior distributions and different simulations for training the network. Multiple critical parameters are randomly chosen and we automatically generate dynamic generator data with respect to them by injecting the voltage and frequency to the system under study. In this CHAPTER , for two parameters case, we generate 500 dynamic simulations,

and for eighteen parameters, we generate 2×10^3 dynamic simulations for training the model. We choose 90 % of the available data for training and cross-validation, and 10 % for testing. The sampling time is set as 0.033 s since the reporting rate of PMU measurement is assumed to be 30 samples/s. For each simulation, the time horizon for getting the whole dynamic responses is set as 25 s. We calculate the loss value in relation to the different time series lengths for choosing the optimal values of the time series length for real and reactive power, which will be explained in the next section.

The selection of appropriate Hyper-parameters is often necessary for satisfactory performance in the deep learning-based methods. In this CHAPTER , we use the Bayesian process method [118] via scikit-learn [103] to find the network hyper-parameters such as max-pool size, stride length, the size of the latent space, and etc. The learning rate is set to 3×10^{-3}. We set the dimension of the latent space as 20. We use the different activation functions for our model, such as Sigmoid and ReLU. For the fully connected layers, we use a dropout of 0.2. The max-pooling layers have a pool size of 20 and stride length of 20. All the case studies using Python 3.8 and deep learning were carried out using a neural networks API, Pytorch [119].

Since some parameters do not have the same impact on the dynamic behavior of the real and reactive power, we choose different time series lengths for the inputs. For training the CVAE model, the time series P_{m} with length t_{P} and Q_{m} with length t_{Q} are feed into the encoder and reconstruction network. Fig. 6.4 shows the normalized loss function for two and eighteen critical parameters under different time series lengths. For two parameters case, the optimal value for t_{P} and t_{Q} are 4 s and 4 s. For the eighteen parameters case, the optimal values for t_{P} and t_{Q} are 4 s, and 7 s. Then, we will consider these values for the t_{P}, and t_{Q}.

Calibration of Two Parameters

For the two parameters case, we consider a case similar to the test B in [42,43]. We calibrate H and K_{A}, whose true values are $H^{\mathrm{t}} = 5.4$ and $K_{\mathrm{A}}^{\mathrm{t}} = 125$. The mean values of these parameters' prior distributions are set as 10 % greater than their true values to account for parameter uncertainties. We choose the lower/upper bound of the uniform prior distributions as 90 % and 200 % less/greater than the $\mu_{\pi}(\boldsymbol{\alpha}_{\mathrm{c}})$ to account for the parameter uncertainties. The standard deviations of the Gaussian prior distributions are set as 30 % of the mean value. The results for uniform, $\mathcal{U}(\cdot)$, and Gaussian, $\mathcal{N}(\cdot)$, prior distributions are given in Table 6.3. It is seen that the proposed method can accurately estimate the parameters under different prior distributions, and the largest errors for H and K_{A} are, respectively, 0.2 %, and 0.3 %.

Table 6.3: Parameter Calibration under Different Prior Distributions

$H(H^{\mathrm{t}} = 5.40)$		$K_{\mathrm{A}}(K_{\mathrm{A}}^{\mathrm{t}} = 125)$	
$\boldsymbol{\pi}(\mathbf{H})$	**Estimated** (% Error)	$\boldsymbol{\pi}(\mathbf{K}_{\mathrm{A}})$	**Estimated** (% Error)
$\mathcal{U}$(0.60, 12)	5.401 (0)	$\mathcal{U}$(13.8, 275)	125.4 (0.3)
$\mathcal{N}(6, 1.8^2)$	5.41 (0.2)	$\mathcal{N}(137.5, 41.3^2)$	124.7 (0.2)

Tolerance Test

In this section, we evaluate the performance of the proposed model under different parameter deviations. For this purpose, we consider more extreme cases in which the prior distribution is a Gaussian distribution whose mean has a ϑ % deviation from the true values and whose standard deviation is 10 %, 30 %, and 60 % of the mean values. All other settings are the same for all simulations. Table 6.4 shows the results for ϑ equal to 20 %, 30 %, and 40 %, in which the mean

of the prior distribution is shown by $\mu_\pi(\alpha_c)$. It can be seen that even when the means of the priors have considerable deviations from the true values and the standard deviations of the priors are relatively small, the proposed approach still has high accuracy. For the first highlighted row in Table 6.4, the maximum error is 0.4 %, which by contrast, the proposed algorithm in [41] has an error greater than 25 % for the same deviations and priors. Moreover, the proposed method in [41] for the case, where the standard deviation is large, and the parameters are in support of the prior distribution has an error greater than 18 %. By contrast, the proposed method has an error less than 0.4 %, as shown in the second highlighted row.

To compare our proposed method with the method in [43], which considers the gross error in the prior distributions, we consider a case similar to the test B in [43]. Specifically, we set the mean value of the prior distribution for H as 61 % less than the true value and that for K_A as 225 % greater than the true value, and choose the lower/upper bounds of the uniform prior distributions as 90 % less/greater than the mean values. We show the performance of the proposed method in Table 6.5. It shows that the proposed model can estimate the parameters with high accuracy. The estimation error of the proposed approach for H is $0.05\,\%$ and for K_A is $0.2\,\%$, which are less than the corresponding errors, $0.08\,\%$ and $0.5\,\%$ respectively, in [43].

Calibration of Eighteen Parameters

This case study demonstrates the performance of the proposed model in a higher-dimensional case with eighteen critical parameters. These parameters are the same are identified by the EE method and listed in Table 6.1. The prior distributions of these parameters for training the model are assumed to have a $\mu_\pi(\alpha_c)$ with 10 % deviation from the true values and follow a uniform distribution as $\mathcal{U}\big(\boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_c) - \boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_c)\cdot 50\,\%,\ \boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_c) + \boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_c)\cdot 50\,\%\big)$. Table 6.7 lists the true and estimated values. The estimation errors verifies the well-trained model has acceptable accuracy for

Table 6.4: Calibration of H and K_A for Different ϑ and $\mu_\pi(\cdot)$

Standard Deviation (10%)						
ϑ	H			K_A		
	$\mu_\pi(H)$	Estimated	Error (%)	$\mu_\pi(K_\mathrm{A})$	Estimated	Error (%)
40 %	**7.56**	**5.41**	**0.1**	**175**	**125.5**	**0.4**
30 %	7.02	5.39	0.1	162.5	125.5	0.4
20 %	6.48	5.39	0.2	150	125.3	0.2
Standard Deviation (30 %)						
ϑ	H			K_A		
	$\mu_\pi(H)$	Estimated	Error (%)	$\mu_\pi(K_\mathrm{A})$	Estimated	Error (%)
40 %	**7.56**	**5.39**	**0.2**	**175**	**124.6**	**0.3**
30 %	7.02	5.40	0.3	162.5	124.6	0.3
20 %	6.48	5.41	0.2	150	124.8	0.1
Standard Deviation (60 %)						
ϑ	H			K_A		
	$\mu_\pi(H)$	Estimated	Error (%)	$\mu_\pi(K_\mathrm{A})$	Estimated	Error (%)
40 %	7.56	5.38	0.4	175	124.7	0.2
30 %	7.02	5.39	0.2	162.5	125.2	0.2
20 %	6.48	5.42	0.3	150	125.3	0.2

high dimensional parameters.

We also compare the performance of the proposed method with that in [11]. We set the prior distribution for the two methods the same. Specifically, we use 10% deviation in the parameters and use a uniform distribution as $\mathcal{U}\big(\boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_\mathrm{c}) - \boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_\mathrm{c}) \cdot 50\,\%,\ \boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_\mathrm{c}) + \boldsymbol{\mu}_\pi(\boldsymbol{\alpha}_\mathrm{c}) \cdot 50\,\%\big)$. The performances of the proposed method and A-ABC-SMC are shown in Fig .6.5. As it can be seen,

Table 6.5: Parameter Calibration Under Bad Prior Distribution

$H(H^{\mathrm{t}} = 5.40)$		$K_{\mathrm{A}}(K_{\mathrm{A}}^{\mathrm{t}} = 125)$	
$\pi(H)$	**Estimated** (% Error)	$\pi(K_{\mathrm{A}})$	**Estimated** (% Error)
$\mathcal{U}$(0.9, 17.2)	5.403 (0.05)	$\mathcal{U}$(28.1, 534.4)	125.3 (0.2)

Table 6.6: Parameter Calibration under Misspecified Prior Distributions

$H(H^{\mathrm{t}} = 5.40)$		$K_{\mathrm{A}}(K_{\mathrm{A}}^{\mathrm{t}} = 125)$	
$\boldsymbol{\pi}(\mathbf{H})$	**Estimated** (% Error)	$\boldsymbol{\pi}(\mathbf{K}_{\mathrm{A}})$	**Estimated** (% Error)
$\mathcal{U}$(6.5, 12)	5.45 (1.1)	$\mathcal{U}$(70, 100)	126.5 (1)
$\boldsymbol{\mathcal{N}(7, 0.3^2)}$	**5.41 (0.2)**	$\boldsymbol{\mathcal{N}(91, 5.3^2)}$	**124.1 (0.8)**

the proposed method achieves better performance than the A-ABC-SMC.

Different events may enable the calibrations of different parameters, and there is no closed-form solution to the best event for generator model calibration. However, we can identify the key parameters that should be estimated based on each event. If engineering judgment can help select reasonable parameters, multiple events can also be utilized in the development of techniques to help select the best parameters for calibration [19, 48]. Using one event for parameter estimation and the other for cross-validation is one example. In this CHAPTER , we train the model using two different events and use two events for cross-validation. Based on EE analysis, the top eighteen key parameters are the same in all the events. Table 6.8 shows the root mean square errors (RMSE) of real and reactive power for two events calculated by

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{K}} \sum_{\mathrm{t}=1}^{\mathrm{K}} (\boldsymbol{y}_{\mathrm{m}}(\mathrm{t}) - \boldsymbol{y}_{\mathrm{meas}}(\mathrm{t}))^2},$$

Table 6.7: Calibration of Eighteen Key Parameters

Parameter	**True value**	**Estimated** (% Error)	**Parameter**	**True value**	**Estimated** (% Error)
K_{s}	20	19.75 (1.1)	X_{d}	0.57	0.57 (0)
T_5	10	9.97 (0.3)	T_1	0.15	0.14 (4.1)
T_6	10	9.63 (0.3)	T_3	0.15	0.14 (4.3)
K_{A}	125	122.8 (1.7)	A_1	0.035	0.035 (0)
T'_{do}	5.4	5.34 (1)	X'_{d}	0.25	0.25 (0)
H	5.4	5.4 (0)	X'_{q}	0.32	0.32 (0)
T_{b}	3.86	3.81 (1)	R_{T}	0.42	0.42 (0)
a_{23}	1.102	1.107 (0.4)	R_{P}	0.01	0.01 (0)
T_{c}	0.9	0.87 (2.2)	T_{R}	1	0.99 (1)

where $\boldsymbol{y}_{\mathrm{m}}$ and $\boldsymbol{y}_{\mathrm{meas}}$ are the model response and its corresponding measurement value. From Table 6.8, it can be seen that the proposed method significantly reduces the RMSE for two events. The parameter set with the lowest RMSE is chosen as the estimated parameters. Fig. 6.6 shows the model validation results under two events with the original parameters and the estimated best parameters and $\boldsymbol{\mu}_{\pi}(\boldsymbol{\alpha}_{\mathrm{c}})$. For both events, there is a significant mismatch between model outputs and PMU measurements when original parameters, $\boldsymbol{\mu}_{\pi}(\boldsymbol{\alpha}_{\mathrm{c}})$, are considered. By contrast, for the selected estimated parameters, the model outputs and PMU measurements match very well.

Table 6.8: RMSE Values of the Real and Reactive Power for Two Events

RMSE	**Active power**		**Reactive power**	
Events	Original Parameters	Estimated Parameters	Original Parameters	Estimated Parameters
1	2.9	0.21	3.2	0.33
2	1.8	0.14	4.3	0.16

Time Efficiency

A Windows server with Intel(R) Core(TM) i7-8700 and 8GB memory was used to generate the simulation data. It took around 5 minutes to generate the dynamic simulation data for two parameters and 25 minutes for eighteen parameters. The CVAE model's training took less than 2 minutes for two parameters and around 3 minutes for eighteen parameters. Testing the well-trained model with the new measurements takes less than 1 s for two parameters case and eighteen parameters.

Conclusions

This CHAPTER presents a framework to systematically validate stability models, identify problematic model parameters, and calibrate them using online PMU measurements. With Elementary Effects (EE) analysis, we first identify the critical parameters of the generator. Using the event playback, we generate dynamic simulations for training a CVAE model, based on which, we estimate the model parameters. We tested the proposed model on two and eighteen critical parameter cases to evaluate its robustness. Simulation results indicate that the proposed method can accurately and efficiently estimate the full distributions of the parameters even when the actual values

of the parameters are out of the prior distributions. The proposed framework can be extended for more complex power system models, such as aggregate renewable power plants and composite load models. This will be investigated in our future works.

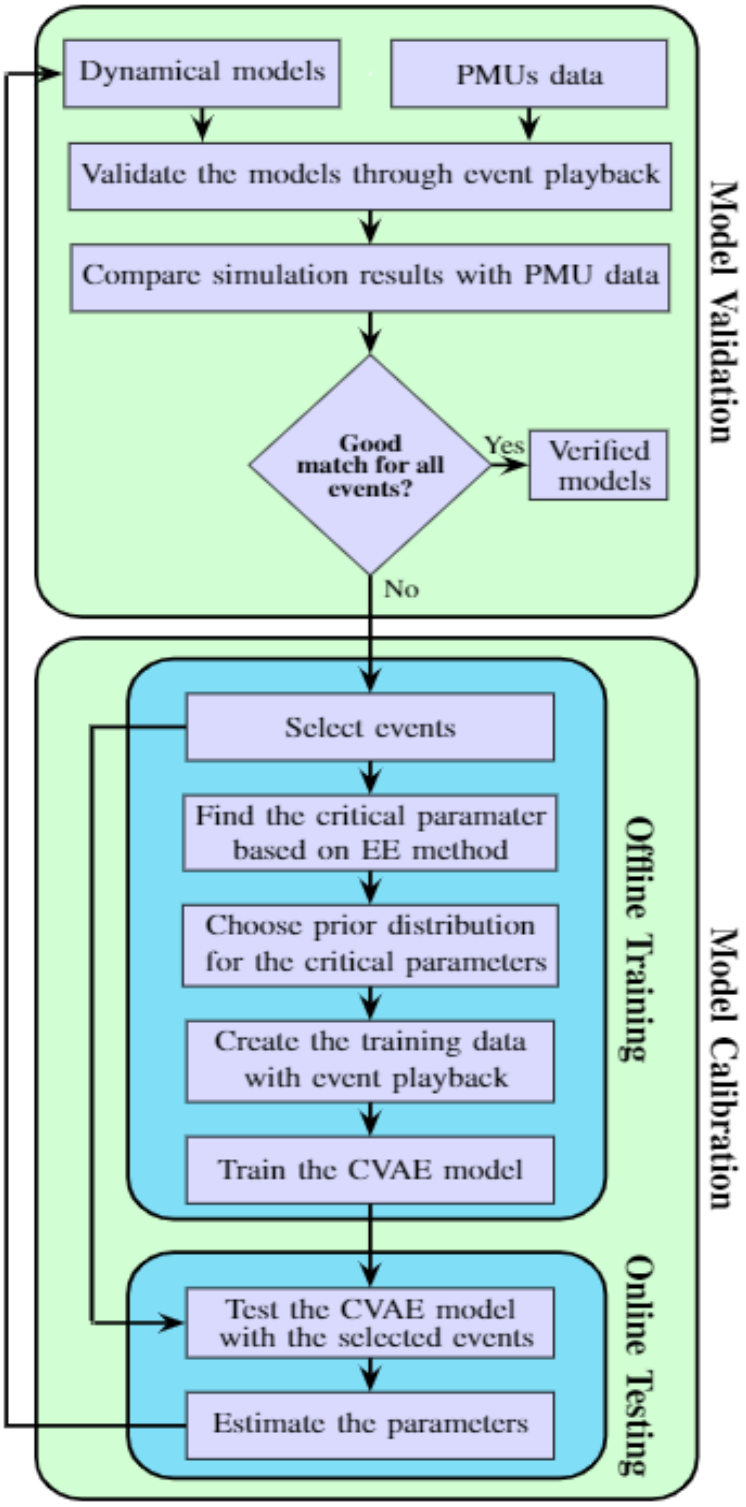


Figure 6.1: Proposed framework for model validation and parameter calibration.

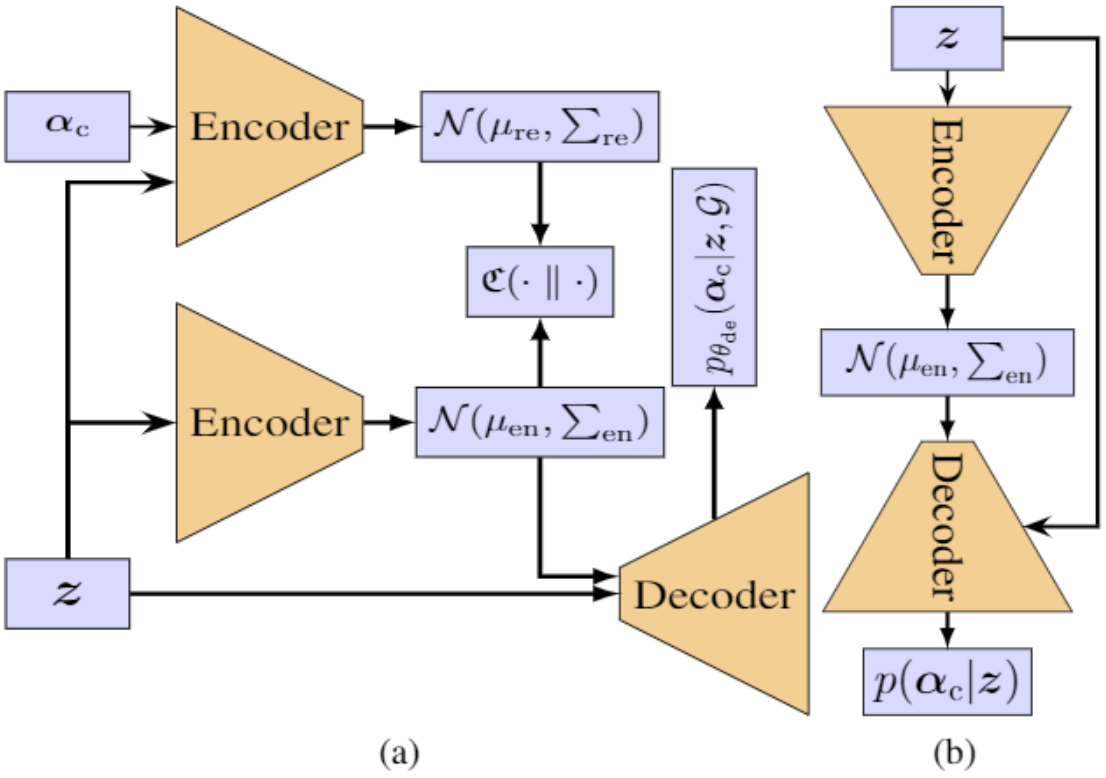


Figure 6.2: The framework of the proposed CVAE: (a) during training; (b) during test.

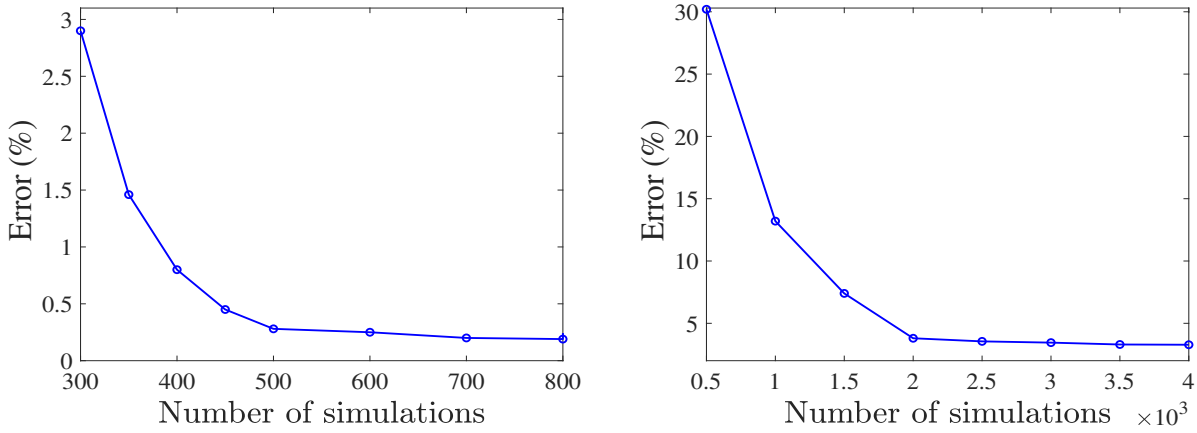


Figure 6.3: Maximum error of the estimated parameters for different number of simulations. (a) Two parameters; (b) Eighteen parameters.

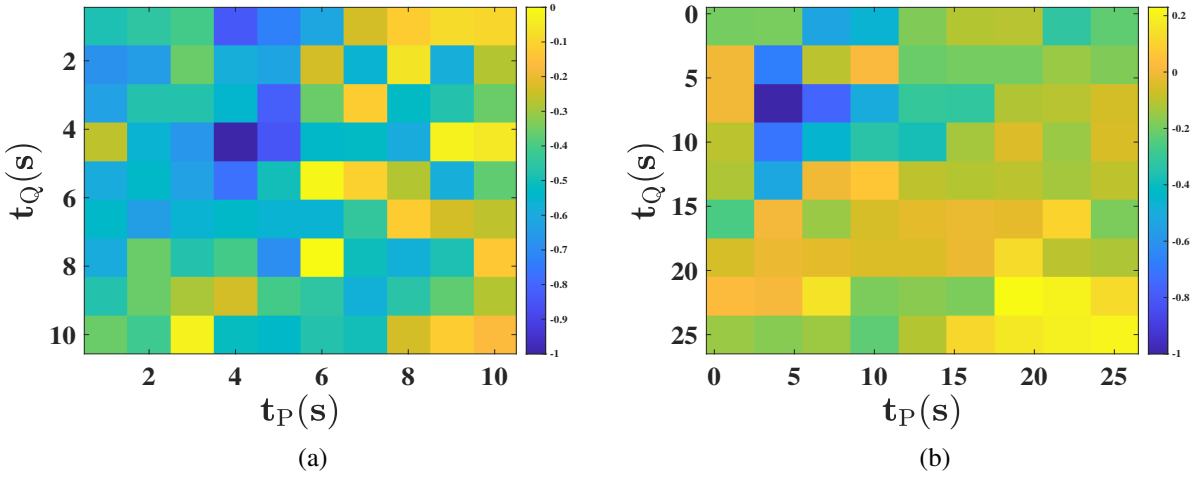


Figure 6.4: Validation of loss function for two and eighteen key parameters for different time series lenght; (a) two parameters; (b) eighteen parameters.

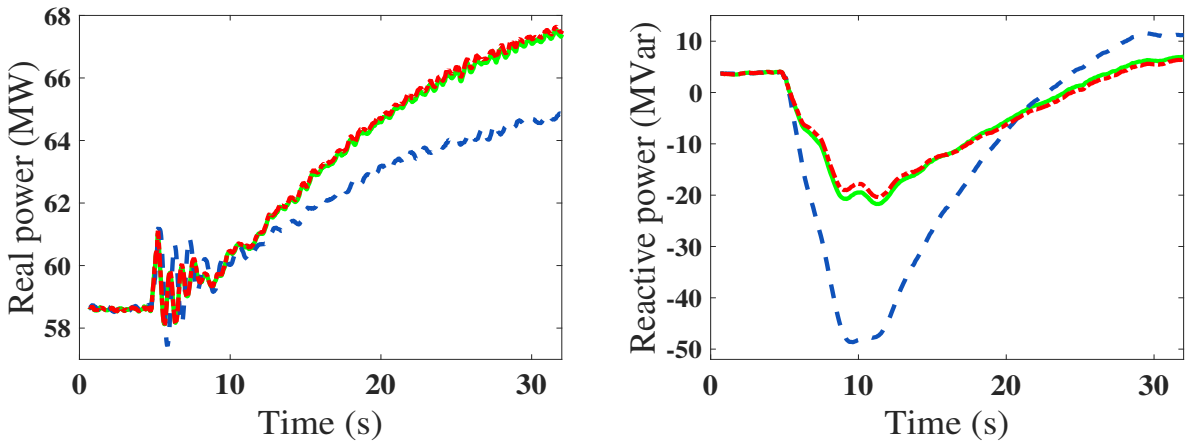


Figure 6.5: Model outputs with A-ABC-SMC method and the proposed method; (a) Real power and (b) reactive power. - - Calibration with A-ABC-SMC method, -. Measurements, — calibration with the proposed method.

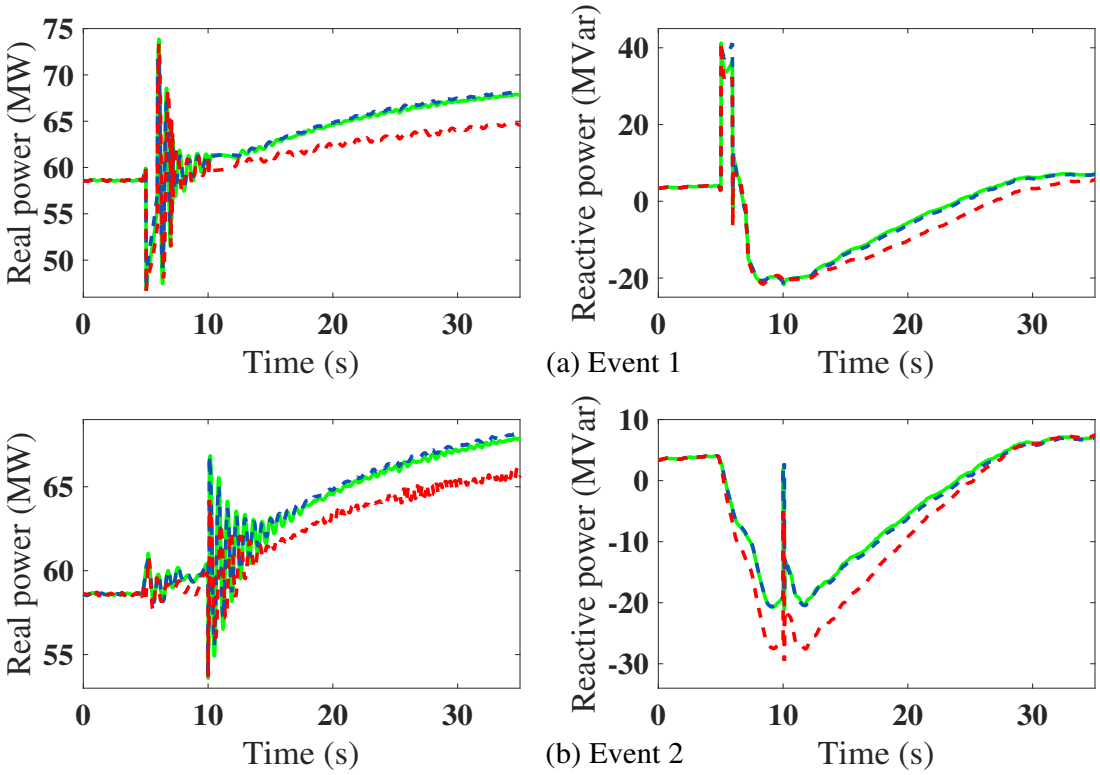


Figure 6.6: Model outputs for estimated parameters: (a) Event 1; (b) Event 2. — PMU measurements; - - model outputs before calibration; . model outputs after calibration.

CHAPTER 7: MEASUREMENT-BASED PARAMETER IDENTIFICATION OF DC-DC CONVERTERS WITH A-ABC-SMC

Khazeiynasab, S. R., & Batarseh, I. Measurement-Based Parameter Identification of DC-DC Converters with Adaptive Approximate Bayesian Computation. Australasian Universities Power Engineering Conference (AUPEC 2021)

Introduction

Switch-mode power converters (SMPC) are broadly used in different power electronics applications, including motor drives, computers, portable electronics, domestic appliances, or in power conversion systems for renewable generation, among others [50–52]. Parameter identification can extract the parameters of the converters and generate accurate discrete simulation models. In this CHAPTER , we propose a measurement-based converter parameter calibration method by A-ABC-SMC method, which estimates the parameters related to passive and parasitic components. At first, we propose to find suitable prior distribution for the parameter which we do not know the prior information about them. With having prior distributions, we can use the ABC-SMC to find the exact values of the parameters of the converter. We chose the distance function carefully and based on the simulations we assigned the best method for the threshold sequencing. For improving the computationally of the algorithm, we propose an adaptive weight that helps the algorithm to find the optimal values with fewer simulations. The effectiveness of the proposed method is validated for a DC-DC buck converter. The results show that the proposed approach can accurately and efficiently estimate the posterior distributions of the buck parameters subject to gross errors in the prior distributions of the parameters. The proposed algorithm can also be applied to other parameter identifications and optimization applications such as rectifiers, filters, or power supplies, among

others. The contributions are summarized as follows.

1. For the parameter which we do not know the initial values, like the impedance of the DC power supply, we introduce a method to find their prior distributions.

2. We perform DC-DC converter parameter calibration by adaptive ABC SMC, which estimates the posterior distributions of the parameters by a simulation-based procedure.

3. We improve the computational efficiency of ABC SMC based parameter calibration by developing adaptive weights of the particles at each iteration. This weighting scheme, helps the algorithm not to be stuck in the local optimal, and also the algorithm needs less number of simulation to find the posterior distributions of the parameters.

DC-DC Converters Parameter Calibration by A-ABC-SMC

Mathematical models have become powerful tools for model analysis. However, as the models become more complex, the computational challenges of parameter inference and model validation are increasingly vast. Let $\boldsymbol{z}^* = [\boldsymbol{V}_{\text{meas}}^\top \ \boldsymbol{I}_{\text{meas}}^\top]^\top$ be the measurements with the actual value of the parameters, and $\boldsymbol{z} = [\boldsymbol{V}_{\text{out}}^\top \ \boldsymbol{I}_{\text{out}}^\top]^\top$ be the outputs of model and $\boldsymbol{\alpha}_{\text{c}}$ is the parameter vector which we want to estimate. Assuming the prior distribution for $\boldsymbol{\alpha}_{\text{c}}$ as $\pi(\boldsymbol{\alpha}_{\text{c}})$. Fig. 8.2 shows the framework for DC-DC converter parameter estimation.

ABC-based methods use systematic comparisons between real and simulated data in order to obtain a good approximation to the true (but unobtainable) posterior distribution

$$p(\boldsymbol{\alpha}_{\text{c}}|\boldsymbol{z}^*) = \frac{l(\boldsymbol{z}^*|\boldsymbol{\alpha}_{\text{c}})\pi(\boldsymbol{\alpha}_{\text{c}})}{\int l(\boldsymbol{z}^*|\boldsymbol{\alpha}_{\text{c}})\pi(\boldsymbol{\alpha}_{\text{c}})d\boldsymbol{\alpha}_{\text{c}}}, \tag{7.1}$$

where the denominator is referred to as the Bayesian Evidence; and the integral runs over all

possible parameter values. $p(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z}^*)$ is posterior distribution, and $l(\boldsymbol{z}^*|\boldsymbol{\alpha}_{\mathrm{c}})$ is the likelihood of $\boldsymbol{\alpha}_{\mathrm{c}}$ given data $\boldsymbol{z}^*$. Instead of evaluating the likelihood, ABC-based approaches use systematic comparisons between real and simulated data. If ϵ is sufficiently small, the distribution $p_\epsilon(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z}^*)$ will be a good approximation of the posterior distribution. Recently, algorithms using SMC with particle filtering have gained growing attention [5, 12, 78]. ABC SMC samples from a sequence of distributions that increasingly resemble the target posterior. They are constructed by estimating the intermediate distributions $p_{\epsilon_t}(\boldsymbol{\alpha}_{\mathrm{c}}|\boldsymbol{z})$ for a decreasing sequence of $\{\epsilon_t\}_{1\leq t\leq N_T}$ where T is the maximum number of iterations [87]. The algorithm first generates an initial pool of N particles that satisfy $\rho(\boldsymbol{z},\boldsymbol{z}^*)\leq\epsilon_1$ by randomly sampling from the prior $\pi(\boldsymbol{\alpha}_{\mathrm{c}})$. In the following iterations, successive distributions are randomly constructed by sampling from the previous population with probabilities $\{w^{(i,t-1)}\}_{\{1\leq i\leq N\}}$ where $w^{(i,t-1)}$ is the weight for the ith particle in iteration $t-1$. To filter and perturb the particles, we need a transition kernel. A transition kernel κ^t is used to perturb the particles and find $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}$'s. The new particle $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}$ is used to simulate $\boldsymbol{z}$ and if $\rho(\boldsymbol{z},\boldsymbol{z}^*)\leq\epsilon_t$ is satisfied, the particle is accepted. The process is repeated until N particles are accepted. At iteration t, the ABC SMC algorithm proposes parameters from the following distribution [88]

$$q^{\mathrm{t}}=\begin{cases}\pi(\boldsymbol{\alpha}_{\mathrm{c}}), & t=1\\ \sum_{j=1}^{N}w^{(j,t-1)}\kappa^t\left(\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}\middle|\boldsymbol{\alpha}_{\mathrm{c}}^{(j,t-1)}\right), & t>1,\end{cases}\tag{7.2}$$

At each iteration, new weights are assigned to the particles, and in the next iteration the particles with larger weights become better represented in the population. The importance weights associated with an accepted population $\{\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}\}_{\{1\leq i\leq N\}}$ are calculated as [5]:

$$w^{(i,t)}=\begin{cases}\frac{1}{N}, & t=1\\ \frac{\pi\left(\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}\right)}{\sum_{j=1}^{N}w^{(j,t-1)}\kappa^t\left(\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}\middle|\boldsymbol{\alpha}_{\mathrm{c}}^{(j,t-1)}\right)}. & t>1\end{cases}\tag{7.3}$$

The efficiency of ABC SMC heavily relies on a proper choice of the perturbation kernel function $\kappa^t(\cdot|\cdot)$, the distance function, $\rho(\boldsymbol{z}, \boldsymbol{z}^*)$, having a good prior distributions for the parameters, the threshold sequence $\{\epsilon_t\}_{1\leq t\leq T}$, and the weights of the particles in each iterations [10]. In this CHAPTER , we carefully assigned a distance function and focused on the adaptive weight and how to find the good prior distributions for the parameters which we do not know their initial values. These factors will be discussed below.

Distance Function

Choosing a summary statistic and distance metric which are sensitive to the parameters of interest is a crucial step in parameter inference with ABC SMC [88]. In this CHAPTER , we choose the following L_2 distance function based on numerical experiments:

$$\rho(\boldsymbol{z}, \boldsymbol{z}^*) = \frac{1}{2K}\|\boldsymbol{z} - \boldsymbol{z}^*\|_2, \tag{7.4}$$

where $\|\cdot\|_2$ is the 2-norm of a vector.

Probability Weight

The probability weights of the particles allow the algorithm to search in the regions with high-probability and to reject particles from low-probability regions of the parameter space [5, 110]. In [5, 10, 88], the weights for the all particles at iteration $t = 1$ are equal to $1/N$. In [4], the weight considered based on the prior distribution for the parameters. However, since the particles sampled randomly in the first iteration, the distance of the particles is different, and assign an equal weight causes the algorithm to search around the particles which may be far from the optimal value. In this CHAPTER , we consider the weights based on the discrepancy of the particles.

Let consider the discrepancy vector at the first iteration as $\boldsymbol{\rho}^1 = [\rho_1^1, \rho_2^1, \cdots, \rho_N^1]$. Based on the discrepancy, for the particle which has a smaller discrepancy there is more probability being close to the optimal, then the particle with smaller discrepancy should have a greater weight. Then, in this CHAPTER , we assigned the weight for the particle i at the first iteration as follows:

$$w^{(i,1)} = \frac{1}{\rho_i^1}. \tag{7.5}$$

For iteration $2 \leq t \leq T$, [4] used the prior distribution and a forward and a backward kernels to assign the weights for the particles, [5] improved the weights and used (7.3) to calculate the weights of the particles. But, in the cases where the prior distributions are not well known for the parameters, using the weight based on (7.3) is not a good choice. In this CHAPTER , for particle i at iteration t we use the prior information of the parameter, $\pi\big(\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}\big)$, and the information of how much the particles are close to the optimal value, ρ_i^t. We defined a constant β to make a trade-off between the prior information of the particles and their distances. This constant can help the algorithm not to be stuck in search around the local optima. Then, the weight for $2 \leq t \leq T$ for particle i is calculated as follow:

$$w^{(i,t)} = \beta\, \pi\big(\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}\big) + (1-\beta)\, \frac{1}{\rho_i^t}. \tag{7.6}$$

Adaptive Threshold Sequence

To balance the computational efficiency and the accuracy of the posterior distribution, we define a threshold sequence:

$$\mathcal{E} = \{\epsilon_1, \epsilon_2, \cdots, \epsilon_T\}, \tag{7.7}$$

where $\epsilon_1 > \epsilon_2 > \cdots > \epsilon_T$. If the threshold is too large, too many proposed particles are accepted; if it is too small, the ABC algorithm is not efficient since many proposed particles will be rejected [88]. Selecting it adaptively based on some quantile of the threshold in the previous iteration has better performance [89]. In this CHAPTER , we use the following threshold sequence scheme:

- We choose ϵ_1 as the acceptance rate in the first iteration is equal to 0.5. We run the simulation for $K_{\text{ini}} = 2\,N$, and chose the median of the all discrepancy of K_{ini} simulations.
- For $\epsilon_{2:T-1}$, ϵ_{t+1} is calculated based on the $q\text{th}$-percentile of the distribution of particle distances in iteration t.

Prior Distribution Correction

An interesting and inexpensive feature of the proposed approach is based on the first step, in which we can estimate the parameters of the system, even we do not know the prior distribution of the parameters. For these cases, we add the prior correction at the first step of the algorithm. For the parameters that we do not know good prior distributions for them, we consider a uniform distribution with very small lower and very large upper bounds. This step makes the proposed method robust to such deviation and makes it suitable for use in the case in which one lacks appropriate prior knowledge about the true parameters. For instance, if the prior $\pi\big(\boldsymbol{\alpha}_{\text{c}}\big)$ is misspecified, it means that the true parameter is not contained in the support of $\pi\big(\boldsymbol{\alpha}_{\text{c}}\big)$. In the case of the power electronics application, for example, we do not know the impedance of the power supply. In this approach, we model the distribution $p(\rho(\cdot)|\boldsymbol{\alpha}_{\text{c}})$ based on the parameter, i. e for any value of input we calculate the discrepancy for N_0 simulation. Let consider the discrepancy for parameter i as $\boldsymbol{\rho_0} = \{\rho_0^1, \rho_0^2, \cdots, \rho_0^{N_0}\}$. We consider the N_{p} smallest distance of the N_0 distances, and based on the N_{p} distances, we consider a Gaussian distribution for the prior distribution with the following

mean and variance.

$$\mu = \min(\boldsymbol{\rho_0}),$$
$$\sigma^2 = \frac{\sum_{i=1}^{N_\mathrm{p}} (\rho_0^i - \mu)^2}{N_\mathrm{p}}. \tag{7.8}$$

Proposed ABC SMC Algorithm

The proposed ABC SMC algorithm is presented in Algorithm 5. The ABC SMC algorithm will stop when the lowest threshold in the threshold sequence is less than the predefined smallest threshold or when a maximum number of T iterations has been performed [4, 5]

Simulation Results

The model of the DC-DC buck converter is based on TPS40200EVM-002 model and is built in the Matlab/Simulink and the proposed algorithm is implemented in Python. All tests are performed on a desktop PC with Intel(R) Core(TM) i7-8700 and 8-GB RAM.

Parameter setting

In this CHAPTER , in all simulations, we consider $q = 0.75$ for choosing the thresholds. Based on the simulations results, we chose the $\beta = 0.4$. We consider a Gaussian distribution kernel same as [5]. We set the maximum number of iteration as $T = 10$.

Algorithm 5: Adaptive ABC SMC algorithm for estimating the posterior distribution of parameters $\boldsymbol{\alpha}_{\mathrm{c}}$ using N particles, the prior distribution $\pi(\boldsymbol{\alpha}_{\mathrm{c}})$, given data $\boldsymbol{z}^*$. $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t)}$ is the parameter set for particle i at iteration t.

Find the prior distributions for the parameter based on (7.8)
Set maximum number of iterations T and set ϵ_1 by section (4)
At iteration $t = 1$
for $1 \leq i \leq N$ **do**
  while $\rho(\boldsymbol{z}, \boldsymbol{z}^*) > \epsilon_1$ **do**
    Sample $\boldsymbol{\alpha}_{\mathrm{c}}^*$ from the prior: $\boldsymbol{\alpha}_{\mathrm{c}}^* \sim \pi(\boldsymbol{\alpha}_{\mathrm{c}})$
    Generate data $\boldsymbol{z}$ from $\boldsymbol{\alpha}_{\mathrm{c}}^*$: $\boldsymbol{z} \sim \mathrm{Model}(\boldsymbol{\alpha}_{\mathrm{c}}^*)$
    Calculate discrepancy $\rho(\boldsymbol{z}, \boldsymbol{z}^*)$ based on (7.4)
  end while
  Set $\boldsymbol{\alpha}_{\mathrm{c}}^{(i,1)} \leftarrow \boldsymbol{\alpha}_{\mathrm{c}}^*$
  Set $w^{(i,1)}$ based on (7.5)
end for
Generate Gaussian perturbation kernel $\kappa^2 = \mathcal{N}(\tilde{\boldsymbol{\alpha}}_{\mathrm{c}}^1, \Gamma^1)$
Determine ϵ_2 based on section (4)
At iteration $t > 1$
for $2 \leq t \leq T$ **do**
  for $1 \leq i \leq N$ **do**
    while $\rho(\boldsymbol{z}, \boldsymbol{z}^*) > \epsilon_t$ **do**
      Sample $\boldsymbol{\alpha}_{\mathrm{c}}^*$ from the previous population $\{\boldsymbol{\alpha}_{\mathrm{c}}^{(i,t-1)}\}_{\{1 \leq i \leq N\}}$ with probabilities $\{w^{(i,t-1)}\}_{\{1 \leq i \leq N\}}$ and perturb them to obtain $\boldsymbol{\alpha}_{\mathrm{c}}^{**} \sim \kappa^t(\boldsymbol{\alpha}_{\mathrm{c}}^t, 2\,\Gamma^{t-1})$
      Generate data $\boldsymbol{z}$ from $\boldsymbol{\alpha}_{\mathrm{c}}^{**}$: $\boldsymbol{z} \sim \mathrm{Model}(\boldsymbol{\alpha}_{\mathrm{c}}^{**})$
      Calculate discrepancy $\rho(\boldsymbol{z}, \boldsymbol{z}^*)$ based on (7.4)
    end while
    Set $\boldsymbol{\alpha}_{\mathrm{c}}^t \leftarrow \boldsymbol{\alpha}_{\mathrm{c}}^{**}$
    Calculate $w^{(i,t)}$ based on (7.6)
  end for
  Generate Gaussian perturbation kernel $\kappa^{t+1} = \mathcal{N}(\boldsymbol{\alpha}_{\mathrm{c}}^t, 2\,\Gamma^{t-1})$
  Determine ϵ_{t+1} based on section (4)
end for

Adaptive Weight

For comparing the different methods, we use the acceptance rate which is defined as follow:

$$\mathrm{acc} = \frac{N}{N_{\mathrm{s}}}, \tag{7.9}$$

where N is the number of the particles which are used in the algorithm, and N_{s} is the total number of simulation during each iteration. At first, we compare the acc at iteration 2 with the w_{e} which is used in [5, 12, 120]. Fig. 7.1a shows the acceptance rate for fifty independent simulations with the proposed weight and the w_{e}. As it can be seen, the proposed weight has greater acceptance rate for all the simulations.

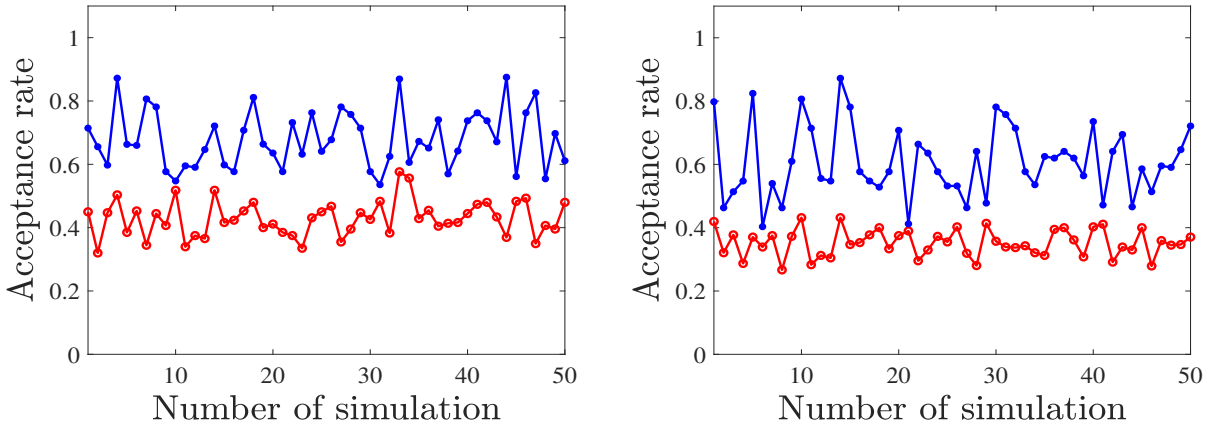


Figure 7.1: Acceptance rates for fifty different simulations (a) $t = 2$ and (b) whole iterations. $\circ$ Weight in [5, 12] $*$ Proposed weight.

Fig. 7.1b shows the acceptance rate for whole iterations of fifty different simulations. It can be seen that the algorithm can find the posterior distributions with less number of simulations with compare to the other methods.

Calibration of Buck Converter

In this CHAPTER , we consider the non-isolated buck converter. Its topology is shown in Fig. 7.2. Since the frequency of switching is high, then the parasitic elements of the converter component should be considered. We consider R_{M} as the the parasitic resistance for the MOSFET, R_{L} for

inductor, $R_{\rm c}$ for the capacitor. We also model the input capacitor as a $C_{\rm in}$ series with a parasitic resistance as $R_{\rm cin}$ [50]. The parasite element of the voltage source is also very important, we consider inductor as $L_{\rm s}$ series with resistance as $R_{\rm s}$. At first, we get the outputs of the Buck converter with a set of parameters, $\boldsymbol{\alpha}_{\mathbf{c}}^{\mathbf{True}}$, which we know their values. We consider the outputs of the model regarding the $\boldsymbol{\alpha}_{\mathbf{c}}^{\mathbf{True}}$ as $\boldsymbol{z}^*$. For the resistance of the power supply, we assume that we do not the prior distribution for it. Then, we consider a uniform distribution as $\mathcal{U}(0, 10000)$ to consider all uncertainties. Then, by the simulation based on section 7 we found that the prior distribution can be considered as a Gaussian distribution as $\mathcal{N}(0.5, 10)$. For the other parameters, we consider the uniform distribution as the prior distributions for the parameters and estimate their values. We consider the mean values of the parameter as 20% percent greater than the true value to consider the uncertainties. We choose the lower/upper bounds of the uniform prior distributions for the parameters as a very small number and very large number. Table 7.1 shows the prior distributions, the estimated values, and the estimation errors. It is seen that the proposed method can accurately estimate the parameters under a uniform prior distribution with a small/large for lower/upper bounds.

To analyze the performance of the converter with the estimated parameters under the transient and steady-state conditions, we change the load at the output. Fig. 7.3 shows the performance of the converter under the transient condition, and Fig. 7.4 shows the performance of the converter in the steady-state condition. As can be seen, the output of the converter with the estimated parameters is very close to the measurements.

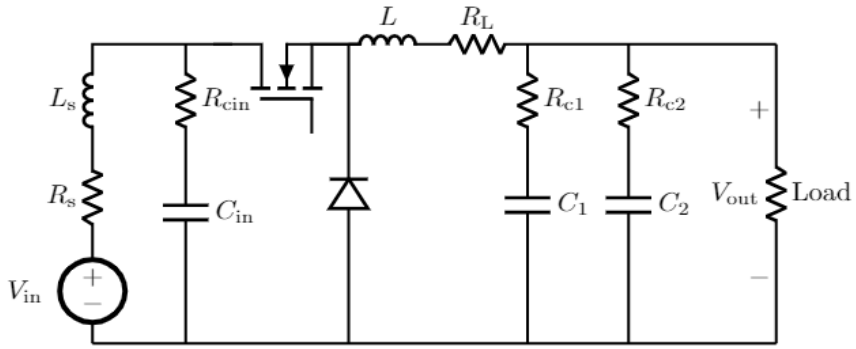


Figure 7.2: Circuit representation of the DC-DC buck converter.

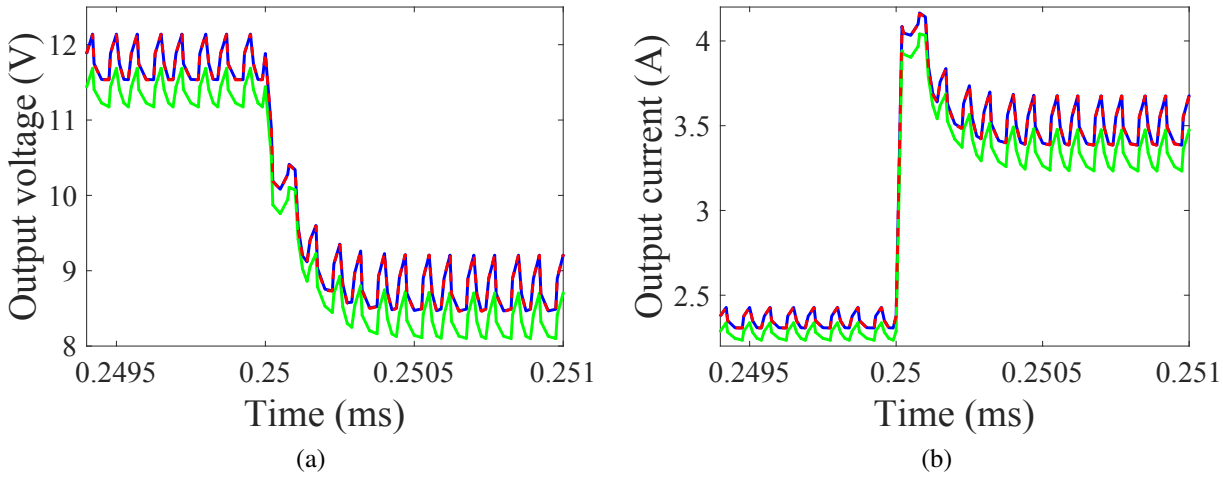


Figure 7.3: Buck converter operating under transient condition. (a) Output voltage; (b) Output current. — Measurements; .- converter before calibration; - - converter after calibration.

Conclusion

This CHAPTER has proposed a parameter calibration method for DC-DC buck power converter based on an A-ABC-SMC approach. We developing the ABC SMC algorithm by proposing a novel and straightforward weight scheme. The proposed algorithm tested on a DC-DC converter with

Table 7.1: Actual and Identified Values of the Parameters of the Buck Converter.

Parameter	**True value**	**Prior distribution**	**Estimated value**	**% Error**
R_{s}	$0.16\,\Omega$	$\mathcal{N}(0.5, 3)$	0.16	0
L_{s}	$0.40\,\mu\mathrm{H}$	$\mathcal{U}(0, 2)$	$0.40\,\mu\mathrm{H}$	0
R_{M}	$40\,\mathrm{m}\Omega$	$\mathcal{U}(0, 4)$	$39.5\,\mathrm{m}\Omega$	1
C_{in}	$100\,\mu\mathrm{F}$	$\mathcal{U}(0, 0.01)$	$99.2\,\mu\mathrm{F}$	1
R_{cin}	$75\,\mathrm{m}\Omega$	$\mathcal{U}(0, 1.5)$	$76.1\,\mathrm{m}\Omega$	1
L	$33\,\mu\mathrm{H}$	$\mathcal{U}(0, 3.3\,\mathrm{m})$	$32.56\,\mu\mathrm{H}$	1
R_{L}	$60\,\mathrm{m}\Omega$	$\mathcal{U}(0, 6)$	$60.8\,\mathrm{m}\Omega$	1
R_{c1}	$65\,\mathrm{m}\Omega$	$\mathcal{U}(0, 6.5)$	$64.8\,\mathrm{m}\Omega$	0
C_1	$100\,\mu\mathrm{F}$	$\mathcal{U}(0, 0.01)$	$100\,\mu\mathrm{F}$	0
R_{c2}	$300\,\mathrm{m}\Omega$	$\mathcal{U}(0, 30)$	$300.8\,\mathrm{m}\Omega$	0
C_2	$100\,\mu\mathrm{F}$	$\mathcal{U}(0, 0.01)$	$99.3\,\mu\mathrm{F}$	1

its parasite and passive elements of the converter. Test results show that the proposed approach can find the exact values of the parameters for a converter by considering the passive and parasite components. We also analyze the steady-state and transient performance of the converter with the estimated parameters, the results show the great performance of the algorithm.

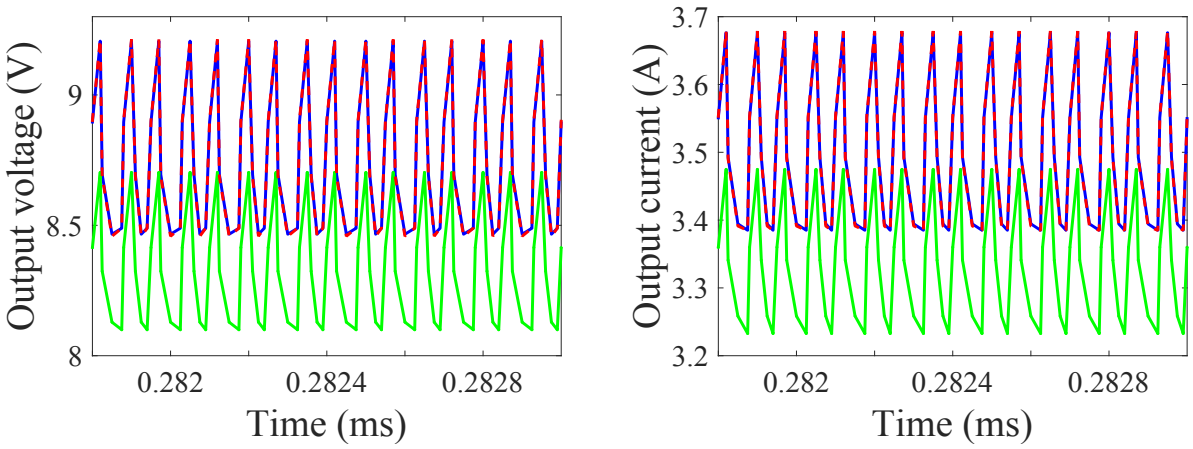


Figure 7.4: Buck converter operating under steady state condition. (a) Output voltage; (b) Output current. — Measurements; .- converter before calibration; - - converter after calibration..

CHAPTER 8: POWER ELECTRONIC PARAMETER CALIBRATION USING A DEEP LEARNING BASED METHOD

introduction

Battery technology has improved considerably in efficiency, cost, and reliability, while utility-scale PV deployment has accelerated rapidly. These developments pave the way for energy storage systems to be combined with PV and grid to provide low-cost energy, grid stability, and reliability for the grid. The integration of energy storage into the PV and grid system can enable multiple critical features for future grid needs, including grid support, load shifting, peak shaving, energy backup, etc. The transition from a grid-tied to a more distributed configuration will require capabilities far beyond those of today's grid-tied power electronics. Developing cost-effective, reliable, and simplified designs are required to expand the deployment of PV+storage hybrid systems. Parameter identification can estimate the parameters of the converters and besides, with having the exact parameters we can have a generate accurate simulation models.

In this CHAPTER , we consider a Three Port Multilevel Inverter (TPMI) proposed by [121] as shown in Fig. 8.1. We propose a measurement-based inverter parameter calibration method by a novel deep learning method. At first, with sensitivity analysis, the critical parameters are identified. Then, with having prior distributions of the parameters, we generate the simulations for training a CVAE model. The proposed algorithm can also be applied to other parameter identifications and optimization applications such as rectifiers, filters, or power supplies, among others.

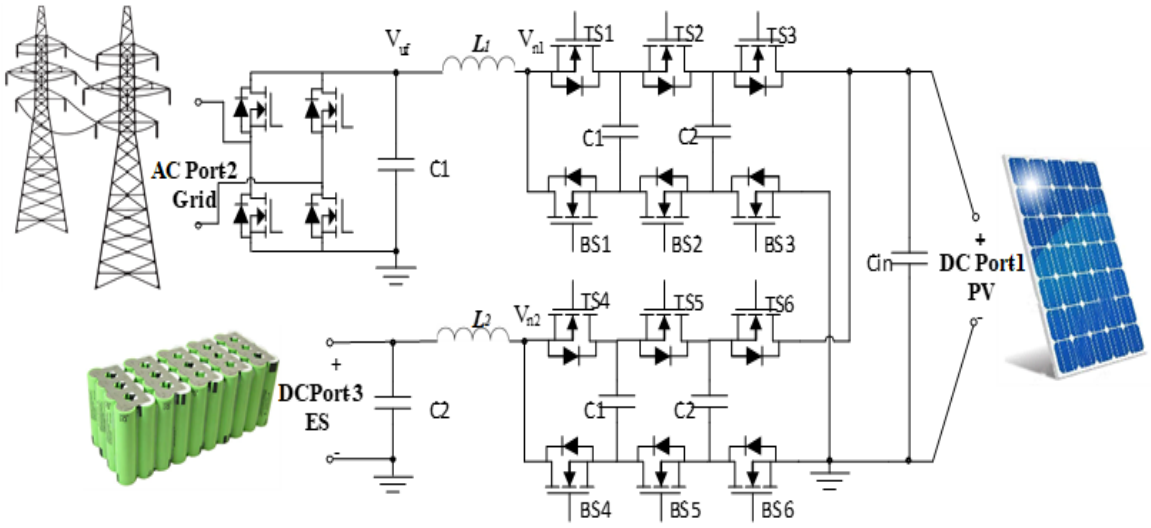


Figure 8.1: Schematic of TPMI.

Inverter Parameter Estimation Framework

Inverter mathematical models are powerful tools for analyzing models. As models become more complex, the computational challenges of parameter inference and model validation become increasingly complex. However, the likelihood function may not always be available because it is computationally too expensive to calculate or does not have a closed-form likelihood function. However, simulating the models is possible. Fig. 8.2 shows the framework for PEC parameter estimation.

Identifiability of Parameters

A PEC has many parameters. Calibrating all parameters could be computationally challenging. Therefore, the parameters with the highest identifiability should be first identified [47]. In this CHAPTER a sensitivity based approach is used to find the critical parameters of the case under

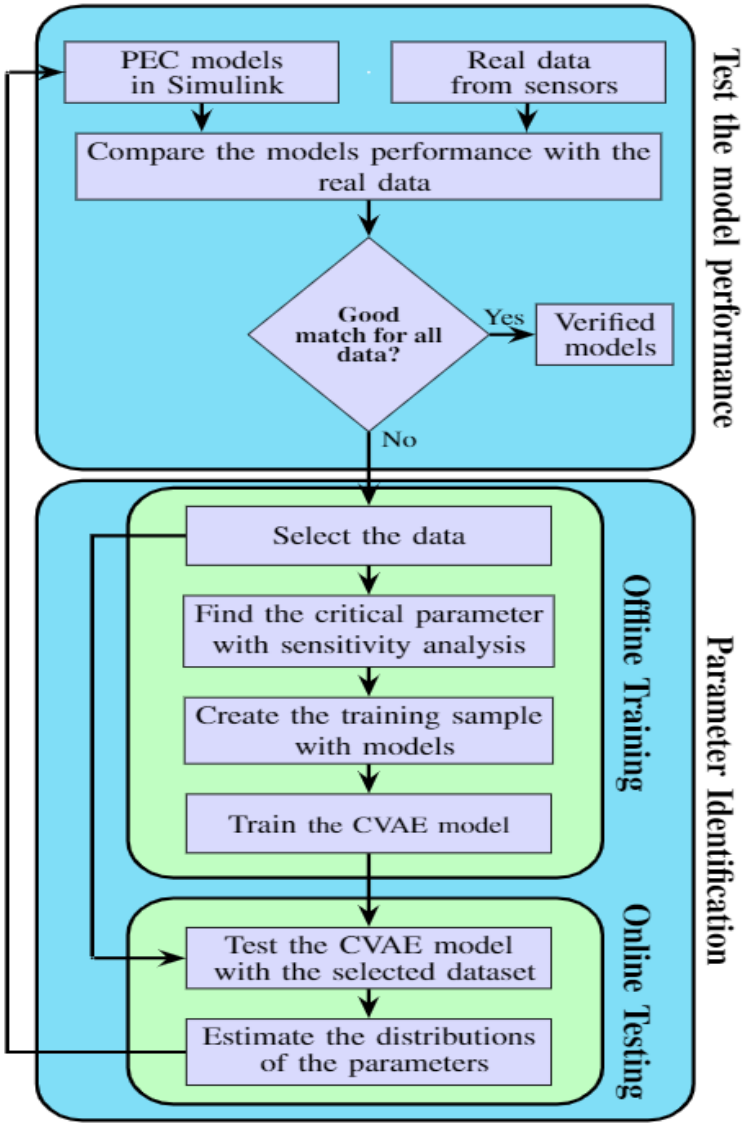


Figure 8.2: Proposed framework for test the PEC model performance and parameter identification.

study. The sensitivity of parameter α_i regarding the the ac and dc output voltage calculated as

follows:

$$S_{\mathrm{ac}}(\alpha_i) = \sum_{k=1}^{K} \frac{|V_{\mathrm{ac},k}(\alpha_i^+) - V_{\mathrm{ac},k}(\alpha_i^-)|}{K(\alpha_i^+ - \alpha_i^-)/\alpha_i},$$
$$S_{\mathrm{dc}}(\alpha_i) = \sum_{k=1}^{K} \frac{|V_{\mathrm{dc},k}(\alpha_i^+) - V_{\mathrm{dc},k}(\alpha_i^-)|}{K(\alpha_i^+ - \alpha_i^-)/\alpha_i},$$

where $\alpha_i^+ = \alpha_i + \Delta\alpha_i$ and $\alpha_i^- = \alpha_i - \Delta\alpha_i$, and $\Delta\alpha_i$ is a small perturbation of α_i. The parameters with the largest sensitivities will be considered as having the highest identifiability [122].

Simulation Results

The proposed model is implemented in MATLAB/Simulink software. By defining the suitable prior distribution, we generate the training data. All the case studies using Python 3.8 and deep learning were carried out using a neural networks API, Pytorch [119].

Critical Parameter Identification

For sensitivity analysis, a small perturbation $\Delta\alpha_i = 10\,\%|\alpha_i|$ is applied to each parameter. From the trajectory sensitivity, it is found that only the parameters shown in Table 8.1 are the problematic parameters for the PEC under study. The top fourteen critical parameters and their normalized sensitivities regarding the voltage of AC and DC ports are provided in Table 8.1.

Choice of Number of Simulations

As many simulations as possible are desirable for high accuracy, they will impose a greater computational burden. As a result, it is reasonable to estimate the parameters of different dimensions

Table 8.1: Top Fourteen Critical Parameters of the Model

Parameter	$\mathbf{S}_{\mathrm{ac}}(\alpha_{\mathbf{i}})$	**Parameter**	$\mathbf{S}_{\mathrm{dc}}(\alpha_{\mathbf{i}})$
L_{link1}	1	C_{out2}	1
C_{out1}	0.9	L_{link2}	0.97
C_1	0.3	R_{out1}	0.62
L_{link2}	0.02	C_{out1}	0.26
C_{out2}	0.02	L_{link2}	0.25
R_{out1}	0.01	C_2	0.049
R_{link}	0.01	R_{link2}	0.03

with different numbers of simulations.

Fig. 8.3 shows the maximum error for fourteen critical parameters under different numbers of simulations. We set the maximum acceptable error to be 1%. Based on Fig. 8.3 when the number of simulations is equal to 2000, this error criterion will be satisfactory.

Generation of Dataset and Parameter Settings

Multiple critical parameters are randomly chosen and we automatically generate dynamic TPMI data. The sampling time is set as 0.0001. For each simulation, the time horizon for getting the whole dynamic responses is set as 3 s.

The selection of appropriate Hyper-parameters is often necessary for satisfactory performance in the deep learning-based methods. In this CHAPTER , we use the Bayesian process method [118] via scikit-learn [103] to find the network hyper-parameters such as max-pool size, stride length, the size of the latent space, and etc. The learning rate is set to 1×10^{-3}. We set the dimension of

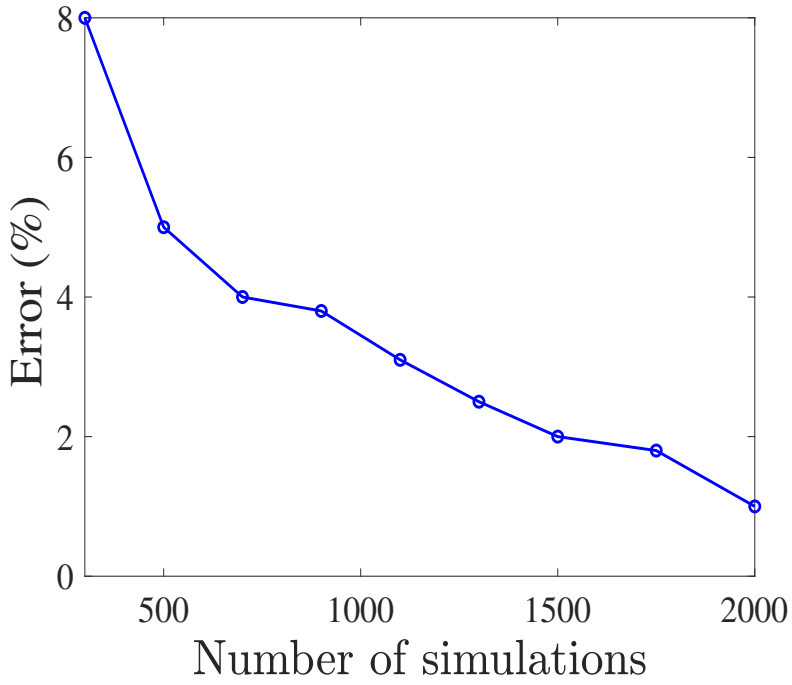


Figure 8.3: Maximum error of the estimated parameters for fourteen critical parameters.

the latent space as 10. We use the different activation functions for our model, such as Sigmoid and ReLU. For the fully connected layers, we use a dropout of 0.3. The max-pooling layers have a pool size of 10 and stride length of 10. All the case studies using Python 3.8 and deep learning were carried out using a neural networks API, Pytorch [119].

Calibration of Fourteen Parameters

This case study demonstrates the performance of the proposed model in a higher-dimensional case with fourteen critical parameters. These parameters are the same are identified by the sensitivity based method and listed in Table 8.1. The prior distributions of these parameters for training the model are assumed to have a $\mu_\pi(\alpha_c)$ with 30 % deviation from the true values and follow a uniform

distribution as $\mathcal{U}\big(\boldsymbol{\mu}_{\pi}(\boldsymbol{\alpha}_{\mathrm{c}}) - \boldsymbol{\mu}_{\pi}(\boldsymbol{\alpha}_{\mathrm{c}}) \cdot 90\,\%,\ \ \boldsymbol{\mu}_{\pi}(\boldsymbol{\alpha}_{\mathrm{c}}) + \boldsymbol{\mu}_{\pi}(\boldsymbol{\alpha}_{\mathrm{c}}) \cdot 90\,\%\big)$. Table 8.2 lists the true and estimated values. The accuracy of the estimation verifies the well-trained model has acceptable accuracy for high dimensional parameters. The dc and ac output voltages of the TPMI under a specific event are shown in Fig. 8.4. It can be seen that the model with the initial values of the parameters has discrepancy with the measurements, however the calibrated model has a very good match output with the measurements.

Table 8.2: Calibration of Fourteen Key Parameters

Parameter	**True value** ($\times 10^{-6}$)	**Estimated** ($\times 10^{-6}$)	**Parameter**	**True value** ($\times 10^{-6}$)	**Estimated** ($\times 10^{-6}$)
L_{link1}	33	33.4	C_{out2}	11	11
C_{out1}	11	10.8	L_{link2}	33	33.5
C_1	11	11.2	R_{out1}	19	18.9
L_{link2}	33	32.5	L_{link2}	33	32.8
C_{out2}	11	10.8	C_{out1}	11	11.2
R_{out1}	19	19.3	C_2	10	10.3
R_{link}	2×10^4	2×10^4	R_{link2}	2×10^4	2×10^4

Conclusions

This CHAPTER presents a framework to systematically validate stability models, identify problematic model parameters, and calibrate them using online PMU measurements. With Elementary Effects (EE) analysis, we first identify the critical parameters of the generator. Using the event

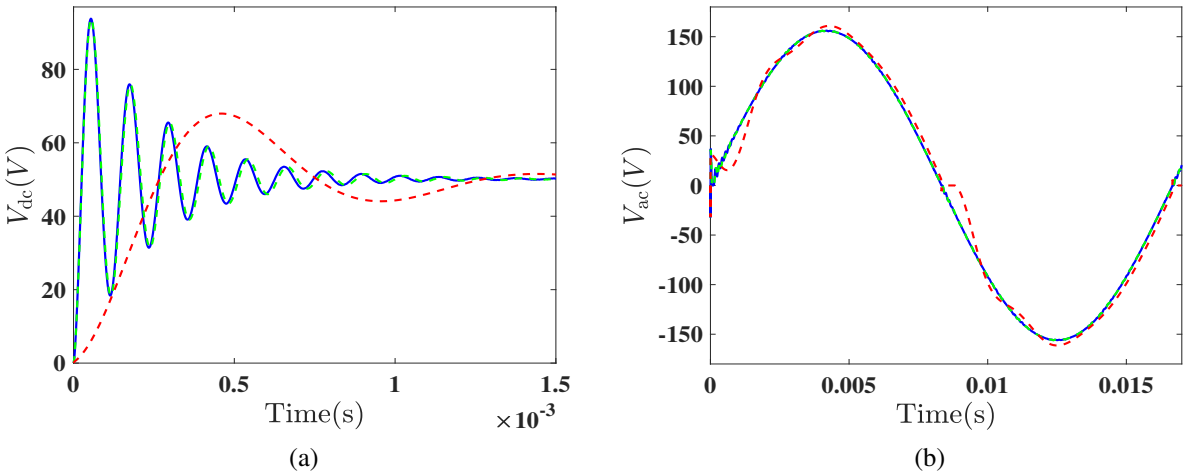


Figure 8.4: TPMI performance with calibrated and the true value parameters. (a) DC port output voltage (b) AC port output voltage; — Measurements; .- with the initial values; - - with calibrated values.

playback, we generate dynamic simulations for training a CVAE model, based on which, we estimate the model parameters. We tested the proposed model on two and eighteen critical parameter cases to evaluate its robustness. Simulation results indicate that the proposed method can accurately and efficiently estimate the full distributions of the parameters even when the actual values of the parameters are out of the prior distributions. The proposed framework can be extended for more complex power system models, such as aggregate renewable power plants and composite load models. This will be investigated in our future works.

Conclusions and Future Works

This CHAPTER presents a framework to systematically validate PEC models, identify problematic model parameters, and calibrate them using online measurements. With sensitivity analysis, we first identify the critical parameters of the model. Based on the prior distributions of the parameters

we create the training data. We tested the proposed model on a TMPI model. Simulation results indicate that the proposed method can accurately and efficiently estimate the full distributions of the parameters. The proposed framework can be extended for more complex power electronic models.

CHAPTER 9: CONCLUSIONS AND FUTURE WORKS

This thesis lies at the power system model validation and machine learning and engineering interface, focusing on power plants and power electronic components. Therefore, we use the event playback to use the PMU time-series measurements to validate the model and then calibrate the model.

We then use the sensitivity analysis to find the critical parameters and consider a system with low-dimensional parameters. With the black-box optimization and reinforcement learning-based methods, we calibrate the model with four and eight parameters.

We then consider a comprehensive hydro-power plant model with a synchronous generator and its controller, i.e., an exciter, a governor, a power system stabilizer. With an approximate Bayesian computation (ABC) based method, we consider the fourteen critical parameters, and then by developing the ABC with sequential Monte Carlo (SMC) sampler, we calibrate the model.

Since the generator in the industry has more parameters, we then consider a high-dimensional case with eighteen critical parameters. We then use a deep learning method. We proposed a Long Short-Term Memory (LSTM)-based method that improves the estimation accuracy.

We then focused on the conditional variational autoencoders (CVAE) to improve the computational cost of the model calibration. We showed that the CVAE-based method has an outstanding performance from the computational aspect and accuracy with the extensive results with the proposed method. We also applied the CVAE-based method to power electronic components.

This thesis can be extended in several following directions:

- Apply the proposed methods to estimate the parameters of the different power system compo-

nents such as composite load models.

- Apply the proposed methods to a power system with a high penetration of renewable energy and a high number of power electronics components.

LIST OF REFERENCES

[1] NERC (North America Electric Reliability Council), "Power system model validation, a white paper by the nerc model validation task force of the transmission issues subcommittee," Dec. 2010.

[2] P. Siemens, "PSS®E 32.0 model library," *PSS/E Manual*, 2009.

[3] J. Weber, "Description of machine models genrou, gensal, gentpf and gentpj," *PowerWorld Corporation*, Dec. 2015.

[4] S. A. Sisson, Y. Fan, and M. M. Tanaka, "Sequential monte carlo without likelihoods," *Pnas*, vol. 104, no. 6, pp. 1760–1765, Feb. 2007.

[5] M. A. Beaumont, J.-M. Cornuet, J.-M. Marin, and C. P. Robert, "Adaptive approximate Bayesian computation," *Biometrika*, vol. 96, no. 4, pp. 983–990, Oct. 2009.

[6] M. Lenormand, F. Jabot, and G. Deffuant, "Adaptive approximate Bayesian computation for complex models," *Computational Statistics*, vol. 28, no. 6, pp. 2777–2796, 2013.

[7] J. Cisewski-Kehe, G. Weller, C. Schafer *et al.*, "A preferential attachment model for the stellar initial mass function," *Electron. J. Stat.*, vol. 13, no. 1, pp. 1580–1607, Jul. 2019.

[8] U. Simola, J. Cisewski-Kehe, M. U. Gutmann, J. Corander *et al.*, "Adaptive approximate Bayesian computation tolerance selection," *Bayesian Analysis*, 2020.

[9] C. P. Barnes, D. Silk, X. Sheng, and M. P. Stumpf, "Bayesian design of synthetic biological systems," *PNAS*, vol. 108, no. 37, pp. 15 190–15 195, Sept. 2011.

[10] S. Filippi, C. P. Barnes, J. Cornebise, and M. P. Stumpf, "On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo," *Stat. Appl. Genet. Mol*, vol. 12, no. 1, pp. 87–107, Mar. 2013.

[11] S. R. Khazeiynasab and J. Qi, "Generator parameter calibration by adaptive approximate bayesian computation with sequential monte carlo sampler," *IEEE Trans. Smart Grid*, 2021.

[12] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf, "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems," *Journal of the Royal Society Interface*, vol. 6, no. 31, pp. 187–202, Feb. 2009.

[13] H.-M. Gutmann, "A radial basis function method for global optimization," *Journal of global optimization*, vol. 19, no. 3, pp. 201–227, Mar. 2001.

[14] NERC, "Standard mod-026-1—verification of models and data for generator excitation control system or plant volt/var control functions," 2014.

[15] S. Mohiuddin and J. Qi, "Maximum correntropy extended Kalman filtering for power system dynamic state estimation," in *IEEE Power and Energy Society General Meeting*. IEEE, 2019, pp. 1–5.

[16] S. A. Nugroho, A. F. Taha, and J. Qi, "Robust dynamic state estimation of synchronous machines with asymptotic state estimation error performance guarantees," *IEEE Trans. Power Syst.*, vol. 35, no. 3, pp. 1923–1935, May 2020.

[17] H. Alatrash, A. Mensah, E. Mark, G. Haddad, and J. Enslin, "Generator emulation controls for photovoltaic inverters," *IEEE Trans. Smart Grid,*, vol. 3, no. 2, pp. 996–1011, May 2012.

[18] S. R. Khazeiynasab and J. Qi, "Resilience analysis and cascading failure modeling of power systems under extreme temperatures," *arXiv preprint arXiv:2009.14155*, 2020.

[19] R. Huang, R. Diao, Y. Li, J. Sanchez-Gasca, Z. Huang, B. Thomas, P. Etingov, S. Kincic, S. Wang, R. Fan *et al.*, "Calibrating parameters of power system stability models using advanced ensemble kalman filter," *IEEE. Trans. Power Syst.*, vol. 33, no. 3, pp. 2895–2905, Oct. 2018.

[20] S. Nuthalapati, *Power System Grid Operation Using Synchrophasor Technology*. Springer, 2019.

[21] J. Zhao, A. Gomez-Exposito, M. Netto, L. Mili, A. Abur, V. Terzija, I. Kamwa, B. C. Pal, A. K. Singh, J. Qi *et al.*, "Power system dynamic state estimation: Motivations, definitions, methodologies and future work," *IEEE Trans. Power Syst.*, vol. 34, no. 4, pp. 3188–3198, Jul. 2019.

[22] S. Nugroho, A. F. Taha, and J. Qi, "Robust dynamic state estimation of synchronous machines with asymptotic state estimation error performance guarantees," *IEEE Trans. Power Syst.*, 2019.

[23] N. Zhou, D. Meng, Z. Huang, and G. Welch, "Dynamic state estimation of a synchronous machine using PMU data: A comparative study," *IEEE Trans. Smart Grid*, vol. 6, no. 1, pp. 450–460, Jan. 2015.

[24] J. S. JOHN, "San diego's 2011 blackout caused by poor planning, runaway errors," 2012. [Online]. Available: https://bit.ly/3r7SbsM

[25] M. Hasni, O. Touhami, R. Ibtiouen, M. Fadel, and S. Caux, "Estimation of synchronous machine parameters by standstill tests," *Math. Comput. Simulat.*, vol. 81, no. 2, pp. 277–289, Oct. 2010.

[26] E. Mouni, S. Tnani, and G. Champenois, "Synchronous generator modelling and parameters estimation using least squares method," *Simul. Model. Pract. Th.*, vol. 16, no. 6, pp. 678–689, Jul. 2008.

[27] R. Wamkeue, C. Jolette, A. B. M. Mabwe, and I. Kamwa, "Cross-identification of synchronous generator parameters from rtdr test time-domain analytical responses," *IEEE Trans. Energy Convers.*, vol. 26, no. 3, pp. 776–786, May 2011.

[28] B. Zaker, G. B. Gharehpetian, M. Karrari, and N. Moaddabi, "Simultaneous parameter identification of synchronous generator and excitation system using online measurements," *IEEE Trans. Smart Grid*, vol. 7, no. 3, pp. 1230–1238, Oct. 2015.

[29] P. M. Ashton, C. S. Saunders, G. A. Taylor, A. M. Carter, and M. E. Bradley, "Inertia estimation of the gb power system using synchrophasor measurements," *IEEE. Trans. Power Syst.*, vol. 30, no. 2, pp. 701–709, Jul. 2014.

[30] M. Ariff, B. Pal, and A. K. Singh, "Estimating dynamic model parameters for adaptive protection and control in power system," *IEEE. Trans. Power Syst.*, vol. 30, no. 2, pp. 829–839, Jul. 2014.

[31] L. Fan, Z. Miao, and Y. Wehbe, "Application of dynamic state and parameter estimation techniques on real-world data," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 1133–1141, Jan. 2013.

[32] G. Chavan, M. Weiss, A. Chakrabortty, S. Bhattacharya, A. Salazar, and F.-H. Ashrafi, "Identification and predictive analysis of a multi-area WECC power system model using synchrophasors," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1977–1986, Mar. 2016.

[33] MathWorks, "Model-based calibration toolbox," in *https://www.mathworks.com/products/mbc/*, 2017.

[34] P. Pourbeik, "Automated parameter derivation for power plant models based on staged tests," in *2009 IEEE/PES Power Systems Conference and Exposition*. IEEE, Mar. 2009, pp. 1–9.

[35] C.-C. Tsai, L.-R. Chang-Chien, I.-J. Chen, C.-J. Lin, W.-J. Lee, C.-C. Wu, and H.-W. Lan, "Practical considerations to calibrate generator model parameters using phasor measurements," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2228–2238, Feb. 2016.

[36] A. A. Hajnoroozi, F. Aminifar, and H. Ayoubzadeh, "Generating unit model validation and calibration through synchrophasor measurements," *IEEE Trans. Smart Grid*, vol. 6, no. 1, pp. 441–449, Nov. 2014.

[37] A. Mohamed, A. Hussain, S. M. Zali, and A. Ariffin, "A systematic approach in estimating the generator parameters," *Elec. Power Compon. Syst.*, vol. 30, no. 3, pp. 301–313, Mar. 2002.

[38] A. Boboń, A. Nocoń, S. Paszek, and P. Pruski, "Determination of synchronous generator nonlinear model parameters based on power rejection tests using a gradient optimization algorithm," *Bull. Pol.*, vol. 65, no. 4, 2017.

[39] Z. Huang, P. Du, D. Kosterev, and S. Yang, "Generator dynamic model validation and parameter calibration using phasor measurements at the point of connection," *IEEE. Trans. Power Syst.*, vol. 28, no. 2, pp. 1939–1949, Mar. 2013.

[40] L. Fan and Y. Wehbe, "Extended kalman filtering based real-time dynamic state and parameter estimation using PMU data," *Electric Power Systems Research*, vol. 103, pp. 168–177, Oct. 2013.

[41] Y. Xu, C. Huang, X. Chen, L. Mili, C. H. Tong, M. Korkali, and L. Min, "Response-surface-based Bayesian inference for power system dynamic parameter estimation," *IEEE Trans. Smart Grid*, Jan. 2019.

[42] Y. Xu, L. Mili, X. Chen, M. Korkali, and L. Min, "A Bayesian approach to real-time dynamic parameter estimation using PMU measurement," *IEEE. Trans. Power Syst.*, Sep. 2019.

[43] Y. Xu, L. Mili, M. Korkali, and X. Chen, "An adaptive Bayesian parameter estimation of a synchronous generator under gross errors," *IEEE Trans. Ind. Informat.*, Oct. 2019.

[44] A. Costa, E. Di Buccio, M. Melucci, and G. Nannicini, "Efficient parameter estimation for information retrieval using black-box optimization," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1240–1253, July 2017.

[45] A. Costa and G. Nannicini, "Rbfopt: an open-source library for black-box optimization with costly function evaluations," *MATH PROGRAM*, vol. 10, no. 4, pp. 597–629, Dec. 2018.

[46] R. Huang, R. Fan, T. Yin, S. Wang, and Z. Tan, "Parameters calibration for power grid stability models using deep learning methods," *arXiv preprint arXiv:1905.03172*, May 2019.

[47] S. R. Khazeiynasab, J. Qi, and I. Batarseh, "Generator parameter estimation by q-learning based on pmu measurements," in *Innovative Smart Grid Technologies (ISGT)*, Feb. 2021, pp. 01–05.

[48] S. Wang, R. Diao, T. Lan, Z. Wang, D. Shi, G. N. America, H. Li, and X. Lu, "A drl-aided multi-layer stability model calibration platform considering multiple events," *PESGM*, pp. 1–5, 2020.

[49] S. Wang, R. Diao, C. Xu, D. Shi, and Z. Wang, "On multi-event co-calibration of dynamic model parameters using soft actor-critic," *IEEE Trans. Power Syst.*, 2020.

[50] J.-R. Riba, M. Moreno-Eguilaz, S. Bogarra, and A. Garcia, "Parameter identification of dc-dc converters under steady-state and transient conditions based on white-box models," *Electronics*, vol. 7, no. 12, p. 393, Dec. 2018.

[51] I. Batarseh and A. Harb, *Power Electronics*. Springer, 2018.

[52] I. Batarseh and K. Siri, "Generalized approach to the small signal modelling of dc-to-dc resonant converters," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 29, no. 3, pp. 894–909, 1993.

[53] A. A. Hussein, A. Pise, X. Chen, and I. Batarseh, "Enhancement of li-ion battery performance at low temperatures by dc-dc converter duty-cycle autotuning," in *2018 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE, Sep. 2018, pp. 2115–2119.

[54] S. Yang, D. Xiang, A. Bryant, P. Mawby, L. Ran, and P. Tavner, "Condition monitoring for device reliability in power electronic converters: A review," *IEEE Trans. Power Electron*, vol. 25, no. 11, pp. 2734–2752, May 2010.

[55] H. Balakrishnan, M. Moreno-Ezuilaz, J.-R. Riba, S. Bogarra, and A. Garcia, "Dc-dc buck converter parameter identification based on a white-box approach," in *(PEMC)*. IEEE, Aug. 2018, pp. 242–247.

[56] B. H. Lin, J. T. Tsai, and K. L. Lian, "A non-invasive method for estimating circuit and control parameters of voltage source converters," *IEEE Trans Circuits Syst I Regul Pap*, vol. 66, no. 12, pp. 4911–4921, Aug. 2019.

[57] L. M. Moore and H. N. Post, "Five years of operating experience at a large, utility-scale photovoltaic generating plant," *Progress in Photovoltaics: Research and Applications*, vol. 16, no. 3, pp. 249–259, 2008.

[58] E. Wolfgang, "Examples for failures in power electronics systems," *ECPE tutorial on reliability of power electronic systems, Nuremberg, Germany*, pp. 19–20, 2007.

[59] A. Abuelnaga, M. Narimani, and A. S. Bahman, "Power electronic converter reliability and prognosis review focusing on power switch module failures," *Journal of Power Electronics*, pp. 1–16, 2021.

[60] G. E. Mejia-Ruiz, M. R. Paternina, J. M. Ramirez, A. Zamora-Mendez, G. Bolivar-O *et al.*, "A system identification-based modeling framework of bidirectional dc-dc converters for power grids," *Journal of Modern Power Systems and Clean Energy*, 2021.

[61] J. Abu-Qahouq and I. Batarseh, "Generalized analysis of soft-switching dc-dc converters," in *2000 IEEE 31st Annual Power Electronics Specialists Conference. Conference Proceedings (Cat. No. 00CH37018)*, vol. 1. IEEE, 2000, pp. 185–192.

[62] G. M. Buiatti, A. M. Amaral, and A. M. Cardoso, "An unified method for estimating the parameters of non-isolated dc/dc converters using continuous time models," in *INTELEC 07-29th International Telecommunications Energy Conference*. IEEE, Sep. 2007, pp. 334–341.

[63] G. Rojas-Dueñas, J.-R. Riba, and M. Moreno-Eguilaz, "Nonlinear least squares optimization for parametric identification of dc–dc converters," *IEEE Trans. Power Electron.*, vol. 36, no. 1, pp. 654–661, Jun. 2020.

[64] M. M. F. S. Algreer, "Microprocessor based signal processing techniques for system identification and adaptive control of dc-dc converters," Ph.D. dissertation, Newcastle University, 2012.

[65] F. Alonge, F. D'Ippolito, and T. Cangemi, "Identification and robust control of dc/dc converter hammerstein model," *IEEE Trans. Power Electron.*, vol. 23, no. 6, pp. 2990–3003, Dec. 2008.

[66] V. Valdivia, A. Barrado, A. LÁzaro, P. Zumel, C. Raga, and C. FernÁndez, "Simple modeling and identification procedures for "black-box" behavioral modeling of power converters based on transient response analysis," *IEEE Trans. Power Electron.*, vol. 24, no. 12, pp. 2776–2790, Oct. 2009.

[67] S. Rahmani, S. M. Mousavi, and M. J. Kamali, "Modeling of road-traffic noise with the use of genetic algorithm," *Applied Soft Computing*, vol. 11, no. 1, pp. 1008–1013, Jan. 2011.

[68] E. Aarts and J. Korst, "Simulated annealing and boltzmann machines," *INFORMS Journal on Computing*, 1988.

[69] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*. Siam, 2009, vol. 8.

[70] R. G. Regis and C. A. Shoemaker, "A stochastic radial basis function method for the global optimization of expensive functions," *INFORMS Journal on Computing*, vol. 19, no. 4, pp. 497–509, Nov. 2007.

[71] K. Elsayed and C. Lacor, "Robust parameter design optimization using kriging, rbf and rbfnn with gradient-based and evolutionary optimization techniques," *APPL MATH COMPUT*, vol. 236, pp. 325–344, Jun. 2014.

[72] K. Holmström, "An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization," *J GLOBAL OPTIM*, vol. 41, no. 3, pp. 447–464, Jul. 2008.

[73] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, May 1979.

[74] G. I. Diaz, A. Fokoue-Nkoutche, G. Nannicini, and H. Samulowitz, "An effective algorithm for hyperparameter optimization of neural networks," *IBM J RES DEV*, vol. 61, no. 4/5, pp. 9–1, Sep. 2017.

[75] P. Etingov, F. Tuffner, J. Follum, X. Li, H. Wang, R. Diao, Y. Zhang, Z. Hou, Y. Liu, D. Kosterev *et al.*, "Open-source suite for advanced synchrophasor analysis," in *2018 IEEE/PES(T&D)*. IEEE, Apr. 2018, pp. 1–5.

[76] J. Cussens, "Approximate bayesian computation for the parameters of prism programs," in *International Conference on Inductive Logic Programming*. Springer, Jun. 2010, pp. 38–46.

[77] J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman, "Population growth of human y chromosomes: a study of y chromosome microsatellites." *Mol. Biol. Evol*, vol. 16, no. 12, pp. 1791–1798, Dec. 1999.

[78] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré, "Markov chain Monte Carlo without likelihoods," *PNAS*, vol. 100, no. 26, pp. 15 324–15 328, 2003.

[79] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf, "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems," *J. R. Soc. Interface*, vol. 6, no. 31, pp. 187–202, Jul. 2008.

[80] S. Lall, J. E. Marsden, and S. Glavaški, "A subspace approach to balanced truncation for model reduction of nonlinear control systems," *Int. J. Robust Nonlin.*, vol. 12, no. 6, pp. 519–535, May 2002.

[81] J. Qi, J. Wang, H. Liu, and A. D. Dimitrovski, "Nonlinear model reduction in power systems by balancing of empirical controllability and observability covariances," *IEEE Trans. Power Syst.*, vol. 32, no. 1, pp. 114–126, May 2016.

[82] C. Himpe and M. Ohlberger, "A unified software framework for empirical gramians," *Journal of Mathematics*, vol. 2013, 2013.

[83] A. K. Singh and J. Hahn, "Determining optimal sensor locations for state and parameter estimation for stable nonlinear systems," *Industrial & engineering chemistry research*, vol. 44, no. 15, pp. 5645–5659, Jul. 2005.

[84] A. K. Singhh and J. Hahn, "Sensor location for stable nonlinear dynamic systems: Multiple sensor case," *Industrial & engineering chemistry research*, vol. 45, no. 10, pp. 3615–3623, May 2006.

[85] S. A. Sisson, Y. Fan, and M. Beaumont, *Handbook of approximate Bayesian computation*. Chapman and Hall/CRC, Aug. 2018.

[86] Y. Schaelte and J. Hasenauer, "Efficient exact inference for dynamical systems with noisy measurements using sequential approximate bayesian computation," *BioRxiv*, 2020.

[87] D. Silk, S. Filippi, and M. P. Stumpf, "Optimizing threshold-schedules for sequential approximate Bayesian computation: applications to molecular systems," *Stat. Appl. Genet. Mol*, vol. 12, no. 5, pp. 603–618, Oct. 2013.

[88] E. Jennings and M. Madigan, "astroabc: an approximate Bayesian computation sequential Monte Carlo sampler for cosmological parameter estimation," *Astron. Comput*, vol. 19, pp. 16–22, Apr. 2017.

[89] C. C. Drovandi, A. N. Pettitt, and M. J. Faddy, "Approximate Bayesian computation using indirect inference," *J. R. Stat. Soc C-appl*, vol. 60, no. 3, pp. 317–337, May 2011.

[90] S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, and T. Kanamori, "Statistical outlier detection using direct density ratio estimation," *Knowl Inf Syst*, vol. 26, no. 2, pp. 309–336, Feb. 2011.

[91] D. Prangle *et al.*, "Adapting the ABC distance function," *Bayesian Analysis*, vol. 12, no. 1, pp. 289–309, 2017.

[92] T. McKinley, A. R. Cook, and R. Deardon, "Inference in epidemic models without likelihoods," *Int. J. Biostat.*, vol. 5, no. 1, Jul. 2009.

[93] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist*, vol. 22, no. 1, pp. 79–86, Mar. 1951.

[94] O. Cappé, R. Douc, A. Guillin, J.-M. Marin, and C. P. Robert, "Adaptive importance sampling in general mixture classes," *Stat. Comput*, vol. 18, no. 4, pp. 447–459, Dec. 2008.

[95] D. Prangle, R. G. Everitt, and T. Kypraios, "A rare event approach to high-dimensional approximate bayesian computation," *Statistics and Computing*, vol. 28, no. 4, pp. 819–834, Jul. 2018.

[96] H. Jahangir, H. Tayarani, S. Baghali, A. Ahmadian, A. Elkamel, M. A. Golkar, and M. Castilla, "A novel electricity price forecasting approach based on dimension reduction strategy and rough artificial neural networks," *IEEE T IND INFORM*, vol. 16, no. 4, pp. 2369–2381, Aug. 2019.

[97] M. Cui, M. Khodayar, C. Chen, X. Wang, Y. Zhang, and M. E. Khodayar, "Deep learning-based time-varying parameter identification for system-wide load modeling," *IEEE Trans. Smart Grid*, vol. 10, no. 6, pp. 6102–6114, Jan. 2019.

[98] M. Khodayar and J. Wang, "Probabilistic time-varying parameter identification for load modeling: A deep generative approach," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 1625–1636, Feb. 2020.

[99] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[100] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on lstm recurrent neural network," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Sep. 2017.

[101] M. Dabbaghjamanesh, A. Moeini, N. D. Hatziargyriou, and J. Zhang, "Deep learning-based real-time switching of hybrid ac/dc transmission networks," *IEEE Trans. Smart Grid*, Dec. 2020.

[102] Y. Li, R. Diao, R. Huang, P. Etingov, X. Li, Z. Huang, S. Wang, J. Sanchez-Gasca, B. Thomas, M. Parashar *et al.*, "An innovative software tool suite for power plant model validation and parameter calibration using pmu measurements," in *2017 PESGM*. IEEE, 2017, pp. 1–5.

[103] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, Nov. 2011.

[104] F. Chollet *et al.*, "keras," 2015. [Online]. Available: https://github.com/fchollet/keras

[105] S. Wang, R. Diao, T. Lan, Z. Wang, D. Shi, H. Li, and X. Lu, "A drl-aided multi-layer stability model calibration platform considering multiple events," in *PESGM*. IEEE, 2020, pp. 1–5.

[106] R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, and J. Wang, "Machine health monitoring using local feature-based gated recurrent unit networks," *IEEE Trans. Ind. Electron.*, vol. 65, no. 2, pp. 1539–1548, Jul. 2017.

[107] F. Campolongo, J. Cariboni, and A. Saltelli, "An effective screening design for sensitivity analysis of large models," *Environmental modelling & software*, vol. 22, no. 10, pp. 1509–1518, Oct. 2007.

[108] M. D. Morris, "Factorial sampling plans for preliminary computational experiments," *Technometrics*, vol. 33, no. 2, pp. 161–174, May 1991.

[109] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[110] A. Jasra, S. S. Singh, J. S. Martin, and E. McCoy, "Filtering via approximate Bayesian computation," *Stat. Comput*, vol. 22, no. 6, pp. 1223–1237, Nov. 2012.

[111] F. Tonolini, J. Radford, A. Turpin, D. Faccio, and R. Murray-Smith, "Variational inference for computational imaging inverse problems," *J Mach Learn Res*, vol. 21, no. 179, pp. 1–46, 2020.

[112] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, Dec. 2013.

[113] O. Ivanov, M. Figurnov, and D. Vetrov, "Variational autoencoder with arbitrary conditioning," *arXiv preprint arXiv:1806.02382*, Jun. 2018.

[114] B. Ivanovic, K. Leung, E. Schmerling, and M. Pavone, "Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach," *IEEE Robot. Autom. Lett*, vol. 6, no. 2, pp. 295–302, Dec. 2020.

[115] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *NeurIPS*, vol. 28, pp. 3483–3491, 2015.

[116] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, "Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy," *arXiv preprint arXiv:1909.06296*, 2019.

[117] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, Nov. 2016.

[118] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *26th Annual Conference on Neural Information Processing Systems 2012, NIPS 2012*, 2012, pp. 2951–2959.

[119] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*. Curran Associates, Inc., 2019, pp. 8024–8035.

[120] M. A. Beaumont, W. Zhang, and D. J. Balding, "Approximate Bayesian computation in population genetics," *Genetics*, vol. 162, no. 4, pp. 2025–2035, Dec. 2002.

[121] M. T. Elrais, "A novel gan-basd three-port flying capacitor multilevel inverter."

[122] S. R. Khazeiynasab and J. Qi, "Pmu measurement based generator parameter calibration by black-box optimization with a stochastic radial basis function surrogate model," *2020 North American Power Symposium (NAPS)*, 2020.