Simulation-Based Optimization of a DC Microgrid: With Machine-Learning-Based Models and Hybrid Meta-Heuristic Algorithms

Tyler Van Deese

Follow this and additional works at: https://scholarcommons.sc.edu/etd

Part of the Electrical and Computer Engineering Commons

Recommended Citation

This Open Access Dissertation is brought to you by Scholar Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact digres@mailbox.sc.edu.
SIMULATION-BASED OPTIMIZATION OF A DC MICROGRID: WITH MACHINE-LEARNING-BASED MODELS AND HYBRID META-HEURISTIC ALGORITHMS

by

Tyler Deese

Bachelors of Science The Citadel 2017
Masters of Engineering University of South Carolina 2021

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Electrical Engineering
College of Engineering and Computing
University of South Carolina
2023

Accepted by:

Herbert Ginn, Major Professor
Kristen Booth, Committee Member
Austin Downey, Committee Member
Roger Dougal, Committee Member
Ann Vail, Dean of the Graduate School
I believe that achieving a doctorate requires a significant combination of hard work and some element of luck. I feel incredibly fortunate to find myself in this position, and it’s a culmination of efforts from many remarkable individuals who have contributed to my personal growth. Notably, my parents have played an instrumental role in affording me the opportunities to reach this point. My brother, John, inspired me to challenge my epistemology from a young age, and my sister, Tina, has been a profound influence on shaping the person I am today.

I must express my gratitude to my advisor, Herb, who has provided me with the guidance and support necessary to develop as both an engineer and a scientist. He consistently encourages experimentation and critical thinking. He is a good advisor, and a great man.
Abstract

The field of economic dispatch (ED) focuses on optimizing power flow in a power system to minimize costs. It has the potential to significantly enhance system effectiveness, and efficiency, and reduce operating costs. Various techniques have been employed to tackle this problem, each with its own strengths and weaknesses. One promising approach is simulation-based optimization (SBO), which allows for accurate modeling of system interactions and improved representation of expected results. However, SBO requires running numerous simulations to identify an optimal solution, and there is a possibility of not achieving the global optimum. This work aims to address these challenges using machine learning. The first contribution involves enhancing the computational efficiency of the SBO model by employing state-reduction techniques and neural network-based observers. This optimization reduces simulation time, thereby speeding up the search process. The second contribution involves developing a hybrid search algorithm by combining the genetic algorithm and the particle swarm method. Additionally, leveraging the cost-to-parameter correlation helps expedite the parameter search. This modified hybrid genetic algorithm reduces the number of simulations required to discover the optimum solution while providing increased confidence in the result. Finally, these two methods are applied to a system to demonstrate that, with their integration, a simulation-based optimizer can align economic dispatch parameters within minutes using standard computing devices. This significantly improves upon the traditional offline approach, which was necessitated by time constraints. This research focuses on enhancing the SBO technique for economic dispatch through machine learning. It includes improving computational efficiency
and developing a hybrid search algorithm, ultimately enabling real-time parameter alignment for economic dispatch on regular computing devices.
# Table of Contents

**Acknowledgments** ......................................................... iii

**Abstract** ........................................................................ iv

**List of Figures** ............................................................... ix

**Chapter 1 Introduction** ..................................................... 1

1.1 Introduction ................................................................. 1

1.2 Problem Statement ........................................................ 6

1.3 Machine Learning ........................................................... 7

1.4 RC Droop ..................................................................... 10

1.5 Simulation-based Optimization ........................................ 11

**Chapter 2 Efficient Neural Network-based Observer-aided Behavioral Modeling of Energy Storage Device SoC for Bus-tied Converters** ......................................................... 13

2.1 Introduction ................................................................. 14

2.2 Baseline Model DTB ....................................................... 15

2.3 Observer Feature Selection ............................................. 19

2.4 Neural Network Structure and Feature Selection ................. 21

2.5 Structure of FF-NN ........................................................ 22

2.6 Feature Exploration and Training ..................................... 24
| 2.7 | Observer-aided Behavioral Model | 27 |
| 2.8 | Computational Considerations | 29 |
| 2.9 | Results | 30 |
| 2.10 | Conclusion | 34 |

**Chapter 3**  Simulation-based Optimization of Virtual Impedance for Economic Dispatch: A Comparative Study of Meta-heuristic Algorithms  35

| 3.1 | Introduction | 36 |
| 3.2 | System Under Investigation | 39 |
| 3.3 | Cost Function | 40 |
| 3.4 | Model for Simulation Evaluation | 42 |
| 3.5 | Meta-heuristic Algorithms | 43 |
| 3.6 | Results | 49 |
| 3.7 | Conclusion | 54 |

**Chapter 4**  Alignment of Power and Energy System for Economic Dispatch Facing a Stochastic Load Forecast  56

| 4.1 | Introduction | 57 |
| 4.2 | System Under Investigation | 59 |
| 4.3 | Load Forecast Projections | 62 |
| 4.4 | Optimization Method | 63 |
| 4.5 | Problems with deterministic alignment | 68 |
| 4.6 | Comprehensive Training Load | 69 |
| 4.7 | Results | 71 |
4.8 Conclusion ................................................................. 74

CHAPTER 5 CONCLUSION .................................................... 77

5.1 Improvements in Modeling for Enhanced SBO Performance ........ 78
5.2 Performance of Metaheuristic Algorithm ............................... 79
5.3 Training Data Produced by Comprehensive Scenario ............... 81

BIBLIOGRAPHY ............................................................... 84
LIST OF FIGURES

Figure 1.1 Feed-forward Neural Network: A neural network with the feed-forward connection scheme with a single hidden layer. 8

Figure 1.2 Simulation Based Optimization Process Cycle 11

Figure 2.1 System level diagram showing two batteries, their interfacing converters, a DC bus, and the loads they supply. 16

Figure 2.2 Switching Average Boost Converter Model: with conduction losses and switching losses approximated through resistive elements. 17

Figure 2.3 The observer is made up of the FF-NN connected to the converter power and SoC, the output of \(dSoC\) is connected to an integrator and that is fed back into itself. 23

Figure 2.4 Battery SoC during Feature Space Exploration: This plot illustrates the battery’s SoC throughout the feature space exploration discharge. 25

Figure 2.5 Zoomed dSoC During Feature Space Exploration: The network is trained to output the dSoC, and the FF-NN is zoomed in over a short time horizon to show how close the agreement is to the baseline DTB. 27

Figure 2.6 The Feature Space Recreated Surface: The FF-NN was trained off of the data taken from the discharge and was used to recreate a feature space map. 27

Figure 2.7 Observer-aided Behavioral Model: The diagram uses virtual passives to represent the electrical contributions of each converter. 29

Figure 2.8 The Test Load Profile: The load profile simulates a residential scenario with high solar penetration and intermittent disruptions during midday. 31
Figure 2.9 Test Scenario Converter Currents: The currents from the converters that interface the batteries are shown from both models. 31

Figure 2.10 Test Scenario Bus Voltage Comparison: The simulation displays the common bus voltage for both models throughout the simulation. 32

Figure 2.11 SoC throughout Test Scenario Comparison 33

Figure 2.12 Runtime Comparison Histogram: On the left, the OABM histogram is shown, with a zoomed-in version in the middle. The OABM histogram is centered around 0.7 seconds, while the DTB histogram is centered around 50 seconds. 33

Figure 3.1 System Level Diagram: Converter A and Converter B are each connected to a source, and both are connected to the DC bus. The converters regulate the DC bus voltage and facilitate the sharing of the loads by means of EDC. 39

Figure 3.2 The Partial Derivatives of the First Objective Function: The plot is shown with respect to each converter. The partial with respect to $I_a$ is a fixed value of 4, and the partial derivative of $I_b$ is a line with a slope of 2 due to its cost being parabolic. 41

Figure 3.3 Partial Derivatives of Second Cost Function: The partial derivatives of the cost function are plotted to visualize the non-linear relationship. 42

Figure 3.4 Particle Swarm Optimization Algorithm: Diagram explaining the update process of a single particle in a PSO 45

Figure 3.5 Flowchart of Genetic Algorithms: The diagram shows GA with black, GAw/CD with black and blue, and GAw/CD&GI with black, red, and blue. 47

Figure 3.6 Discovered Electrical Optimum for First Cost Function: Converter Current Waveforms for the first cost function. 50

Figure 3.7 Discovered Electrical Optimum for Second Cost Function: The operating point found for the second objective function. 50

Figure 3.8 Algorithm Performance Spider-plot: The performance metrics are visualized to compare the four different meta-heuristic algorithms. 51
CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The optimization of Economic Dispatch (ED) in energy systems is crucial for ensuring a reliable and cost-effective operation. ED involves optimizing the operation of energy facilities to produce and distribute power at the lowest possible cost while maintaining grid reliability [29]. Simultaneously, optimizing the cost of energy production is economically beneficial and contributes to minimizing waste and extending the operational capabilities of energy systems. However, the formulation and solving ED as an optimization problem for a microgrid is challenging due to various factors, including non-linearities, randomness, and numerous constraints inherent in power systems [106]. Power systems comprise non-linear components such as power converters, generators, transmission losses, and energy storage devices, which pose challenges for linear optimization methods. Additionally, the unpredictable nature of loads and the randomness associated with renewable energy sources further complicate the optimization problem. Furthermore, power systems are subject to various operational constraints, including voltage tolerances, power limits, and ramp rates.

The formulation of the optimization problem for economic dispatch has historically simplified many aspects of the system because a utility-sized electrical grid is able to focus on instantaneous power generation, allowing for a significant simplification of the problem while providing useful information. Historically, response times and observation delays in the macro-grid have typically been on the order of sec-
onds. This is because the system’s transients were mainly controlled by mechanical power generation systems with significant inertia. Additionally, the aggregation of servicing loads enabled a slower and more deliberate investigation process [53]. In contrast to the macro grid, a microgrid typically lacks these large generators and aggregation causing the formulation of the optimization problem to require a timescale much faster, many systems are managed by power converters which typically operate in the micro-second timescale with loads that change in power ratings much more aggressively than what is seen in the macro grid. These approaches for economic dispatch on the macro grid often involve simplifying the problem through linearization or approximation methods to fit known optimization techniques. For instance, linear programming has been used to solve the instantaneous optimum in economic dispatch (ED), but it fails to consider the value associated with energy reserves [39]. Other methods include dynamic optimization through system discretization and scheduling [77]. The system’s numerous non-linear characteristics, including converter losses, battery charging and discharging efficiency, and energy reserves, exert significant influence on a smart grid, especially within a microgrid. In a macrogrid, the aggregation of many of these influences enables linearization. However, in a microgrid, due to the high relevance of these factors to the grid’s operation and the absence of aggregation of many systems, linearizing their influence becomes impractical. Microgrids usually have energy reserves that must be managed based on significant uncertainty, whether from uncertainty in energy generation from renewable energy sources or the stochastic nature of load consumption [131]. This problem then becomes even more complex with the notional idea of distributing loads between two or more different types of energy storage devices that adopt a policy to operate each other at the best operating positions for each one, such as frequency delegation between batteries and a supercapacitor [22]. The successful management of all these factors necessitates a control scheme capable of implementing policies that facilitate their integration. Even more
crucially, the formulation of the optimization problem must be based on the relationship between the control scheme parameters and the cost objective. This relationship is essential for optimizing a policy that effectively governs the system. Modeling the interactions of microgrids is a well-founded field of study, allowing for the investigation of a microgrid through scenario-based simulations. This approach enables a model to evaluate a non-linear and multi-objective, leading to the development of promising methods for optimization.

Simulation-Based Optimization (SBO) combines optimization techniques with modeling simulation to achieve accurate and flexible optimization of complex systems [57]. When the algebraic representation of the simulation is not easily obtainable, and there is a lack of derivative information, estimating derivatives through finite differences may not be appropriate due to the presence of noisy outputs and the high cost associated with running SBO is an effective tool [6]. SBO leverages accurate system models to evaluate objective functions and employs optimization algorithms to determine the optimal actions based on the simulation results. SBO offers advantages in terms of generality compared to traditional optimization methods. However, it may not guarantee finding the global optimum, and computational time can be a limitation. For online deployment of an SBO, the speed at which the solution is discovered may affect the value of that solution. An example is if the optimal power-flow policy of a day is decided, but implemented halfway through the day, then there is a lot of potential energy wasted by not quickly aligning the system to the new optimum. This creates a link between convergence speed, and the potential value brought to a system using an online SBO. To mitigate the issue with convergence speed, both the algorithm for the parameter selection and the model simulation computational time must be fast as compared to the response time of the system.

The models used in an SBO should represent the system accurately enough such that the simulated results are representative of the expected physical system results.
through the same scenario. The field of making a digital twin (DT) model is a quickly developing field focusing on developing models that serve as comprehensive digital representations of physical systems [59]. Modeling of microgrids has been successfully accomplished at many different levels of fidelity depending on the type of investigation being conducted. The primary function of the DT models used in an SBO is for evaluating the objective function over the simulated scenario based on the model inputs. In the SBO configuration, the model may run through thousands of simulations in order to converge on the near-optimum solution, this can take a lot of time to converge on a solution. As power systems grow larger and more complex, the computational challenges of online economic dispatch in real-time become increasingly significant. State reduction techniques, such as abstracted models and neural network-based state observers, can provide fast and accurate evaluations while maintaining computational efficiency. Other researchers have already presented methods for increasing a model’s computational efficiency through means such as parallel computing, latency-injection-method, state-space-nodal decomposition, and state reduction.

The optimization of Economic Dispatch in microgrids using SBO and metaheuristic algorithms offers several benefits, a multi-objective goal can easily be implemented into these types of optimization schemes, balancing computational complexity and optimization performance. However, these algorithms have limitations in their application because of stagnation and convergence time. Heuristic algorithms play a significant role in simulation-based optimization, particularly when the formulation of the non-linear problem is unknown or computationally demanding [57]. Heuristic algorithms, such as genetic algorithms, particle swarm optimization, and simulated annealing, provide efficient and effective solutions for complex optimization problems [38]. These algorithms use iterative search processes to explore the solution space and converge toward optimal or near-optimal solutions. Heuristic algorithms offer flexibility, scalability, and robustness, making them suitable for solving the Economic
Dispatch optimization problem in microgrids. The cost objectives can use models developed and modeled such as in [1, 123], allowing for the integration of considerations such as battery degradation, carbon emissions, system maintenance, etc. Ghasemi at el. [54] uses a model of a wind farm and a meta-heuristic algorithm to optimize the system online. Firstly, it enables the integration of complex system dynamics, operational constraints, and uncertainties into the optimization process, leading to more accurate and realistic results. Torken at el. [139] uses a microgrid model to optimize the energy management of a residential microgrid. Secondly, it provides flexibility in considering different objectives, such as cost minimization, emission reduction, and system reliability enhancement. Raghavan at el. [114] uses both a particle swarm optimization and genetic algorithm to optimize a microgrid around a 24-hour forecast, this paper addresses one of the main problems with these algorithms being that they can stagnate away from the optimum, or they can converge slowly. Vergara at el. [141] attempts to deploy a real-time online system optimization, but is limited by the convergence time to the optimum which forces the implementation for a scheduler to reoptimize the system every 15 minutes. Additionally, this approach can be applied to various microgrid applications, including renewable energy integration, smart grid operation, and microgrid planning and design.

The optimization of Economic Dispatch (ED) in energy systems is essential for reliable and cost-effective operation. ED involves minimizing power production costs while maintaining grid reliability. Power systems face challenges such as non-linearities, randomness, and operational constraints. Various methods have been proposed, including linear programming and dynamic optimization, but these methods can be inaccurate. Simulation-Based Optimization (SBO) combines optimization techniques with simulation models to achieve accurate and flexible optimization. SBO leverages system models and employs meta-heuristic algorithms to balance computational complexity. Digital Twin models provide accurate representations for simulation-based
optimization. Heuristic algorithms offer effective solutions for complex optimization problems. Integrating SBO and heuristic algorithms for microgrids allows accurate evaluations and efficient exploration of the solution space, but the problem of convergence time and stagnation must be addressed. This approach enables the consideration of system dynamics, constraints, and uncertainties, providing flexibility and applicability to various microgrid applications. Computational challenges in online economic dispatch have been addressed through methods like model simplification, parallel computing, and decomposition techniques, ensuring accurate and efficient solutions in real-time.

1.2 Problem Statement

To achieve reliable, efficient, and adaptive optimization for microgrids, an accurate, fast, and adaptive approach to ED is essential. A reliable system must consider multiple contingencies, while an accurate optimum solution can significantly reduce costs. The speed at which a system can identify and adapt to changes is crucial for online operations. SBO can meet these requirements by incorporating a few key enhancements. Firstly, it requires an accurate and computationally efficient system model that can adapt to changes. To address this, machine learning techniques, such as supervised learning, are used to train neural-network-based state observers for a fast and accurate representation of system considerations allowing for the construction of a fast and accurate behavioral model of a microgrid, this methodology is expressed in the Chapter 2. The SBO model utilizes this reduced state model to simulate stochastic scenarios and evaluate the objective function. Secondly, meta-heuristic algorithms must be employed for model input selection that ensures the SBO reliably converges to a near-optimum solution in a few iterations. Commonly proposed algorithms are compared and modified to create a hybrid algorithm tailored to this specific problem, this is explored in Chapter 3. The hybrid algorithm demonstrates reliable conver-
gence and is validated through testing on a system. The integration of this SBO framework with a digital twin enables accurate, fast, and adaptive optimization, resulting in a more reliable, efficient, and adaptive microgrid system. Lastly, the energy flow algorithms are built such that they may be interfaced with the optimization for the optimization of a digital twin to reduce fuel consumption, this is reported on in Chapter 4. This comprehensive approach to SBOs for microgrid optimization has not yet been reported in the literature standing apart from existing methods by the improvements and tailoring of the system controls to meet the specific optimization problem.

1.3 Machine Learning

Machine learning is a field within artificial intelligence that focuses on building and understanding methods for machines to learn by using data to improve performance for a task [98]. An important tool for achieving many desirable aspects of learning uses neural networks.

1.3.1 Neural Networks

Neural Networks are non-linear function that is constructed from the connection of neurons that allow for complex relationships to be learned [56]. Artificial neural networks are inspired by how the brain is perceived to work, as first explained in McCulloch and Pitts’ logical calculus of ideas immanent in nervous activity [93]. They are constructed from neurons that represent an activation function, and the neurons are connected which adds the influence from the inputs weights and biases of each connection. The activation function is used to add non-linearity to the network, which allows the network to approximate complex relationships. The connection scheme of the neural network is how each neuron is connected, and the most popular scheme is a feed-forward network [62] which is shown in Fig. 1.1. The input is three
inputs that get wired to the input layer, each having its own neuron. The input layer is then wired to the hidden layer, which contains four neurons. The hidden layer then connects its four neurons to the two output neurons to make the network outputs. Figure from [48].

![Feed-forward Neural Network](image)

**Figure 1.1 Feed-forward Neural Network:** A neural network with the feed-forward connection scheme with a single hidden layer.

For a neural network wired in feed-forward with a single hidden layer with linear activation for inputs and outputs, the input-to-output function is mathematically expressed in eq. (1.1).

\[
Y_{out} = W_o \sigma(W_i * X + B_i) + B_o
\]  

(1.1)

\[W_o\] is the hidden layer to output weights
\[B_o\] is the hidden layer to output biases
\[\sigma\] is the hidden layer activation function
\[W_i\] is the input to hidden layer weight
\[B_i\] is the input to hidden layer biases

The weights and biases in neural networks are trained using three different methods. Unsupervised learning allows the input data to train without a target output, causing the system to cluster inputs into groups with commonalities, which is useful
for data analysis [63]. Reinforced learning involves training without explicit output data but with a critic that provides feedback to maximize the reward obtained by the system [133]. Supervised learning, which is the focus of this study, involves training the neural network to approximate the relationship between input and output data using a known training set [20].

1.3.2 Supervised-learning

Supervised learning techniques manipulate the parameters of a neural network to optimize its performance by fitting input-output data during the training process. These techniques involve adjusting the weights and biases of the neural network through backward propagation and employing non-linear optimization methods, such as gradient descent [56]. In supervised learning, the training data consists of inputs and corresponding target outputs, and the network initializes its weights and biases, often using random initialization [61]. During the forward pass, input data is processed through the network to generate output predictions. The network’s output is then compared to the target output using a cost function, typically involving the squared error [104]. By calculating the gradient of the cost function, gradient descent is applied to iteratively update the network’s weights and biases, aiming to minimize the discrepancy between the predicted output and the target output [20]. During the training process, the network undergoes multiple iterations, with each iteration consisting of a forward pass, error calculation, and parameter updates. The objective is to iteratively improve the network’s performance and reduce the training error. Through this iterative process, the neural network gradually learns to map the inputs to the desired outputs, leading to improved accuracy and predictive capabilities [69].
1.4 RC Droop

RC Droop is a bus voltage regulation scheme that delegates steady-state and frequency load responses to different participants using bus voltage as a form of communication. It extends the concept of droop control by incorporating a virtual capacitance to introduce a filtering effect [152]. This additional parameter allows for a more sophisticated allocation of loads to multiple participants compared to traditional droop control. The Droop configuration consists of three key set points: a voltage reference, a virtual resistance, and a virtual capacitance. The voltage reference represents the desired bus voltage at no load, while the virtual resistance determines the voltage droop as the converter supplies current. The virtual capacitance acts as a filtering capacitor, enabling the converter to respond to specific frequency changes in voltage with a corresponding current response. Unlike explicit communication-based approaches, such as DC droop control, RC Droop utilizes the bus voltage as a means of communication to avoid converter interference and facilitate load delegation. An advantage of droop control schemes, including EDC, is their ability to dynamically adjust parameters based on internal calculations or external inputs without the need for explicit communication. This makes them suitable for asynchronous communication environments, where occasional packet loss or communication failures may occur without disrupting power flow. Additionally, RC Droop is capable of seamlessly transitioning to islanded operation in situations where participants are electrically disconnected, allowing them to continue fulfilling loads independently until reconnection is established. However, it is important to properly interface the droop parameters with the optimization technique to prevent unintended local optimum traps inherent in the RC droop scheme.
1.5 Simulation-based Optimization

Simulation-based Optimization (SBO) or Simulation Optimization is defined as finding the best input variables for a model for a simulated scenario without explicitly evaluating all possibilities while minimizing computational resources and maximizing information gained from the exploration [24]. The process is a two-step loop until the break criteria are met. First, the inputs to the model are decided, second the model inputs are evaluated through a simulation. The first process of deciding on inputs is a quickly developing field that is integrating different aspects of AI. This input decision is commonly decided by meta-heuristic algorithms because they produce near-optimal solutions quickly and have been bolstering their performance over time. The modeling of the system is application specific, for a microgrid, modeling of a digital twin is a developing but well-founded field. The SBO will iterate between testing model inputs and evaluating them through simulations until it converges on an optimum solution as seen in Fig. 1.2. This process is popular because it allows a lot of information about the process to be discovered much faster than other exploratory methods. The desire for speed comes from what was described earlier, the discovery of the optimum must be discovered in time for it to maintain value. In order to do this the model must be fast and accurate, and the meta-heuristic algorithm must
converge quickly and reliably.

To achieve reliable, efficient, and adaptive optimization for microgrids, an accurate, fast, and adaptive approach to ED is essential. A reliable system must consider multiple contingencies, while an accurate optimum solution can significantly reduce costs. The speed at which a system can identify and adapt to changes is crucial for online operations. SBO can meet these requirements by incorporating a few key enhancements. Firstly, it requires an accurate and computationally efficient system model that can adapt to changes. To address this, machine learning techniques, such as supervised learning, are used to train neural-network-based state observers for a fast and accurate representation of system considerations allowing for the construction of a fast and accurate behavioral model of a microgrid, this methodology is expressed in Chapter 2. The SBO model utilizes this reduced state model to simulate stochastic scenarios and evaluate the objective function. Secondly, meta-heuristic algorithms must be employed for model input selection that ensures the SBO reliably converges to a near-optimum solution in a few iterations. Commonly proposed algorithms are compared and modified to create a hybrid algorithm tailored to this specific problem, this is explored in Chapter 3. The hybrid algorithm demonstrates reliable convergence and is validated through testing on a system. The integration of this SBO framework with a digital twin enables accurate, fast, and adaptive optimization, resulting in a more reliable, efficient, and adaptive microgrid system. Lastly, the energy flow algorithms are built such that they may be interfaced with the optimization for the optimization of a digital twin to reduce fuel consumption, this is reported on in Chapter 4. This comprehensive approach to SBOs for microgrid optimization has not yet been reported in the literature standing apart from existing methods by the improvements and tailoring of the system controls to meet the specific optimization problem.
Chapter 2

Efficient Neural Network-based Observer-aided Behavioral Modeling of Energy Storage Device SoC for Bus-tied Converters

Power system investigations heavily rely on models and simulations, particularly for advanced investigations involving Monte Carlo or meta-heuristic algorithms. However, the use of faster-than-real-time system models for online iterative-based optimization in power systems faces challenges due to long investigation times and computational costs. In this chapter, a computationally efficient behavioral model is presented. It represents the electrical interactions between power converters operating in RC droop with state observation to approximate battery SoC. The model incorporates supervised-learned neural networks as state observers, integrating traditional modeling methods with machine learning. By modeling the behavior of the highest level of control for the converters, the model computes bus voltage, and converter currents, and estimates the state of charge for energy storage devices with neural network-based state observers. The approach utilizes a baseline physics-based model for feature selection and training. Comparisons between the proposed model called the Observer-aided behavioral model (OABM), and a baseline model are performed to evaluate its effectiveness. The results show that OABM achieves a significant speed improvement, 50 times faster than the baseline model while maintaining a similar level of fidelity.
2.1 Introduction

Digital twins (DT) are comprehensive digital representations of products that properties, conditions, and behavior of real-life objects through models and data [59]. Digital twins offer flexibility in investigating physical systems, including faster-than-real-time simulations for forecasting multiple scenarios. However, the need for numerous simulation evaluations requires balancing model fidelity against computational time [8]. Monte Carlo simulations are commonly used to handle system responses to stochastic inputs [41], while Simulation-Based Optimizers (SBO) and model-based reinforced learning require multiple iterations [117, 9, 154, 57].

Computational capabilities of processing hardware, such as CPUs, have been increasing dramatically [thompson2022importance, 124], but some applications remain computationally infeasible due to complex models [134, 121, 23]. Power system models, for example, face a trade-off between accuracy and computational efficiency. High-fidelity models may be too slow, while low-fidelity models may be too inaccurate to capture relevant dynamics [5, 34, 107, 70]. The trade-off between speed and fidelity is particularly apparent when modeling switching power converters.

Switching power converters use semiconductors to achieve regulated output responses. Modeling switching events incurs high computational costs, but system-level studies often use switching average models for computational efficiency [47, 3, 31, 76]. These models significantly decrease simulation costs by reducing the number of operations per time step and/or allowing larger time steps.

The computational time of a one-step method-based simulation depends on various factors, including model complexity, size, ODE solver, relative tolerance, and large state matrices [130, 58]. Model complexity, interactions between systems, and high-frequency components affect the time step requirements. The size of the model depends on the number of states to be solved. Model solvers trade accuracy, speed, and application [79, 68, 32]. Loosening the relative tolerance allows for larger time
steps. Matrix inversion and multiplication, with a computational cost of N cubed (N: number of states), can be reduced by minimizing linearly dependent states or decomposition of the model [60, 42, 132, 11].

State observers estimate unmeasured or unsimulated states [146]. Shallow Feed-Forward Neural Networks (FF-NNs), implemented as part of state observers, provide empirical data approximations [2]. However, "black-box" aspects of neural networks introduce unpredictability [16]. To mitigate risks, inputs can be restricted, the number of inputs and neurons can be limited, and hidden layers can be minimized [125]. Maintaining a simple structure allows for quick computation and avoids dimensionality issues [15, 12, 113].

FF-NN state observers require training data for supervised learning [92]. The Digital Twin Block (DTB) characterizes a physical system within a specific investigation realm. The baseline DTB, consisting of batteries, converters, controls, filtering passives, and loads, produces training data for the FF-NN state observer to approximate the State of Charge (SoC) of the battery. Another model, the observer-aided behavioral model (OABM), combines the SoC FF-NN state observers with a simplified model of electrical characteristics to approximate large-signal interactions between power converters.

2.2 Baseline Model DTB

The baseline model DTB represents an islanded DC microgrid consisting of a common DC bus connected to loads and two power converters that interface two separate identical batteries. Both power converters operate in voltage control mode, working together using an RC droop power-sharing scheme that functions as a virtual capacitor and virtual resistor. RC droop utilizes an RC filter created by virtual impedances to divide the load current by frequency and allocate it to each of the converters connected to the bus, as shown in Figure 2.1.
The battery model is a zero-order empirical model that is fitted using a lookup table, matching the characteristics of the battery it was derived from. The power converter is a switching average model that operates in voltage control mode, employing a voltage control loop and an inner current control loop.

The topology of this islanded microgrid has been simplified in order to demonstrate the usefulness of the abstraction technique. While this specific topology is unlikely to be encountered in practical applications, the abstraction method presented in this chapter can be valuable for model abstraction in many other instances. For the purposes of this chapter, we assume that the baseline DTB has been properly characterized to represent the physical hardware it corresponds to. The hardware-DTB correspondence is not important for the methods shown in this chapter that create the observers for OABM. The observer is trained to represent states from the baseline DTB, so the models need to be compared to each other.

2.2.1 Converter Model

In this model, the converter losses are approximated using resistive elements. The model is depicted in Figure 2.2. The boost converter’s control input is the duty cycle $D$, the duty cycle is generated using two nested PI loops, which enable the operation of the buck converter in voltage control mode, incorporating an inner current control loop. This is a simplified version of the model presented in [47].
Figure 2.2 Switching Average Boost Converter Model: with conduction losses and switching losses approximated through resistive elements.

Converter Model Variables

- $V_{\text{in}}$ - input voltage (low side)
- $L_s$ - series inductance
- $R_s$ - series resistance
- $I_L$ - Inductor Current
- $D$ - Duty cycle
- $V_o$ - Converter output voltage (high side)
- $C$ - Output filtering capacitor
- $R_{\text{loss}}$ - Switching losses approximation resistor

2.2.2 Converter Control Scheme: RC Droop

RC droop is a bus voltage regulation scheme that delegates steady-state and frequency load responses to different participants using bus voltage as a form of communication [152]. The RC droop configuration consists of three key set points: a voltage reference, a virtual resistance, and a virtual capacitance. The voltage reference represents the desired bus voltage at no load, while the virtual resistance determines the voltage droop as the converter supplies current. The virtual capacitance acts as a filtering
capacitor, enabling the converter to respond to specific frequency changes in voltage with a corresponding current response. An advantage of droop control schemes, including RC droop, is their ability to dynamically adjust parameters based on internal calculations or external inputs without the need for explicit communication. This makes them suitable for asynchronous communication environments, where occasional packet loss or communication failures may occur without disrupting power flow. Additionally, RC droop is capable of seamlessly transitioning to islanded operation in situations where participants are electrically disconnected, allowing them to continue fulfilling loads independently until reconnection is established. However, it is important to properly interface the RC droop parameters with the optimization technique to prevent unintended local optimum traps inherent in the RC droop scheme.

2.2.3 Battery Model

The battery model is a zero-order model made up of an internal voltage $E_M$ in series with an internal resistance $R_0$ derived from [36]. The values of $E_M$ and $R_0$ change based on the SoC, and these values are characterized from the battery that would be modeled and stored in a lookup table. In Simulink, this is realized by two controlled voltage sources to represent $E_M$ and $R_0$. These voltage sources use the component values from the look-up table and calculate the voltage that would be across each source using the necessary transfer functions.

2.2.4 Bus Model and Loads

The bus model is a two-port system that connects the two battery-interfacing converters and loads. This is a lossless connection that has no additional passive filters beyond what is used in the converters. The loads are implemented as a controlled current source that derives its current from a current reference.
2.3 Observer Feature Selection

The observer is designed by selecting a desired output state to observe, along with inputs that provide a strong prediction of the output. It is preferable to choose inputs that allow for model decomposition and simplification. The output selection can be any desired state, but it must be deterministic based on the inputs it is associated with. The inputs of the observer fall into the realm of feature selection [71], with the additional consideration that the input selection should be a state that enables model reduction, decomposition, or simplification.

Several common techniques for feature selection are available, such as the drop-in/drop-out method, Pearson correlation coefficient selection, and a priori selection. In the drop-in or drop-out method, inputs are added or removed to determine the minimum number of inputs required to reliably train a network. This approach is effective but time-consuming, as it involves a brute-force search for a solution. Another technique is the use of Pearson correlation coefficient, which measures the linear correlation between inputs and outputs. However, this method may miss inputs with strong nonlinear impacts that are not well represented by correlation coefficient values. A powerful method is to leverage system knowledge and construct a nonlinear function that approximates the system based on a priori information. In this chapter, the observer is designed to approximate the State of Charge (SoC) of the batteries and the a priori method is employed to select inputs by using physics-based equations to justify the non-linear relationship between SoC and the power-into-the converter. Additionally, computational decomposition is considered in feature selection, as selecting certain inputs can render other states unnecessary, allowing for further model simplification.
2.3.1 Observer Output

For this work, the goal of the observer is to provide an approximation of the SoC of the batteries. This information is critical for almost all applications requiring a battery, in particular any sort of economic dispatch of energy within a micro-grid. An accurate approximation of SoC is required for any sort of SoC regulator that may be added to energy-balancing controllers.

2.3.2 Observer Input

The input of the observer is determined as the power supplied to the bus by the interface converter. This choice is made because the converter-to-bus interface provides an advantageous position for model decomposition while maintaining a strong non-linear relationship with the output, SoC. The strength of this relationship is demonstrated through a set of equations that establish the relationship between State of Charge (SoC) and the power output of the converter. One commonly used method for approximating SoC is Coulomb counting [105], which proves to be a simple yet effective approach. This method involves integrating the current into the battery and normalizing it relative to the battery’s maximum Amp-Hour capacity. The relationship is represented in Equation (2.1).

\[ \text{SoC} \approx \int (I_{\text{Battery}}) \, dt \]  

\[ (2.1) \]

In this topology, the current into the battery is related to the current from the bus. This relationship may be approximated by treating the power converter like a DC transformer. The power into the converter and out of the converter is shown in Fig. 2.2, and the power into the battery is shown in Equation (2.3).

\[ P_{\text{In}} = P_{\text{Out}} \cdot \eta - P_{\text{Losses}} \]  

\[ (2.2) \]

\[ V_{\text{Battery}} \cdot I_{\text{Battery}} = P_{\text{Out}} \cdot \eta - P_{\text{Losses}} \]  

\[ (2.3) \]
Using (2.3), it is possible to show the relationship between \( \text{SoC} \) and the \( P_{\text{out}} \) of the converter, as shown in Equation (2.4).

\[
\text{SoC} \approx \int \left( \frac{P_{\text{out}} \cdot \eta - P_{\text{losses}}}{V_{\text{Battery}}} \right) dt
\]

(2.4)

2.4 Neural Network Structure and Feature Selection

The neural network builds the observer, but it may not be the only part. This is due to the fact that there may be situations where the inputs and outputs of the neural network may be different than the inputs and the outputs of the observer, this is shown in the FF-NN used in this chapter. The neural network needs to have an output that allows for the observer output to be calculated and have inputs such that the neural network output may be trained. The number of inputs should also be reduced to as few as possible to allow for validation.

2.4.1 Memory Considerations

The desired output of the observer is the State of Charge (SoC) of the battery. However, SoC inherently contains memory, as the next iteration of SoC depends on previous iterations of SoC, as shown in Equation (2.4). This poses a challenge because the FF-NN system needs to incorporate the feedback of SoC with memory into its structure, either internally or externally through feedback. The memory of this observer will be achieved externally through the feedback of an integrator.

2.4.2 Neural Network Output

The FF-NN encounters a challenge when directly outputting the battery State of Charge (SoC). This is primarily due to the training data being obtained from variable
time steps. Consequently, the network becomes oblivious to the passage of time unless it is explicitly added as an input. Instead of introducing an additional input or encoding a time step within the data, the target output of the FF-NN was defined as the derivative of SoC with respect to time, \( dSoC \). By doing so, the FF-NN can be trained using time-independent inputs. The output of the FF-NN is then integrated to approximate SoC. This implementation offers the advantage of handling variable time steps while also reducing the number of required inputs. Furthermore, it enables the utilization of a memory-less neural structure, which is beneficial as structures with memory introduces numerous unnecessary states.

2.4.3 Neural Network Input

The input of the FF-NN consists of two components: the converter power and the State of Charge (SoC). The inclusion of converter power is justified similarly to the observer’s input selection, leveraging a priori knowledge of the system. The non-linear relationship between converter power and SoC is supported by Equation (2.4). The SoC is fed back into the FF-NN as it influences the battery characteristics that correspond to the next change in SoC, denoted as \( dSoC \). This relationship is explained in the previous subsection regarding memory considerations and can also be observed in Equation (2.5) through the Battery Voltage. A diagram illustrating how the FF-NN is utilized to construct the observer, including its inputs and outputs, is presented in Figure 2.3.

\[
dSoC \approx \left( P_{Out} \cdot \eta - P_{Losses} \right) / V_{Battery}
\]  

(2.5)

2.5 Structure of FF-NN

The structure consideration for this design is to make the network as simple as possible with smooth activation functions. A simple structure would be one that has
Figure 2.3 The observer is made up of the FF-NN connected to the converter power and SoC, the output of $dSoC$ is connected to an integrator and that is fed back into itself.

few neurons and few connections. The feed-forward connection scheme may not be optimum for this deployment, but the number of neurons is small enough to allow for a full connection between layers. The number of neurons also is minimized in the hidden layer. The rule of thumb is as few neurons as it takes to obtain the desired results. Knowing how many are needed will depend on how well the network is able to agree with the feature space, which is explained in subsection 2.6.1. Lastly, having a smooth activation function allows for the non-linearity of the data to be captured, while allowing for the solver to take large time steps.

The FF-NN for this network consists of two neurons in the input layer, six neurons in the hidden layer, and one neuron in the output layer. These neurons are connected in a fully connected feed-forward scheme, where each neuron in the input layer is connected to every neuron in the hidden layer, and all the hidden layer neurons are connected to the output neuron.

The input and output layers employ a linear activation function, or no activation function, while the hidden layer neurons use a Tansig activation function. This specific network architecture is designed to be as compact as possible, reducing computational requirements, while still providing enough neurons to effectively train the desired function.

The Tansig activation function is used in the hidden layer to obtain the required non-linearity for the FF-NN. The selection of activation was picked based upon agree-
ment of the feature space, which is further explained in subsection 2.6.1. A diagram of the FF-NN neural structure is shown in Fig. 2.3.

2.6 Feature Exploration and Training

In order to train the FF-NN effectively using supervised learning, training data must be provided to cover all possible inputs. The collection of all possible combinations of inputs and outputs is referred to as the feature space. To thoroughly explore this feature space, the DTB baseline needs to simulate scenarios that generate data points for each input combination at a reasonable resolution. The collected data is then used to train the FF-NN, enabling it to replicate the feature space and allowing for comparison against the original data. If the FF-NN has one or two inputs with a single output, the feature space can be visually verified. However, if there are more inputs, an error-based algorithm must be employed for validation.

2.6.1 Feature Space Exploration

For the FF-NN used in the SoC observer, the output is the dSoC, and the inputs are the converter power and SoC. This means that to thoroughly explore the feature space, the converter must source and sink power at various levels of SoC. To accomplish this, the converter is loaded with two cycles of a sinusoidal function, followed by a slight battery discharge. This process is repeated until the battery is fully discharged. In this discharge pattern, training data is generated that properly explores both converter power and SoC. The load profile is depicted in Fig. 2.4. The battery transitions from full charge to complete discharge while periodically sourcing and sinking maximum power to the bus. The effects of this periodic sourcing and sinking can be observed in the zoomed-in subplot.
2.6.2 Supervised Learning

The FF-NN is trained using supervised learning, a method where labeled training data is utilized to teach the system how to respond effectively [80]. The training data is obtained from the original DTB. During the training process, a cost function is computed, and a non-linear optimization technique is employed to minimize this cost. The commonly used cost function is the squared error between the target value \( T \) and the output of the network \( Y \), as illustrated in Equation (2.6).

\[
Cost = \frac{1}{2} \cdot (T - Y)^2
\]  

(2.6)

The output denoted as \( Y \) in Equation (2.6), is computed by the FF-NN. It is represented as a function that involves the weights \( W \), biases \( B \), and activation function. The activation function used is the Tan-sig function, as shown in Equation (2.7). The weights and biases in the output layer are indicated by the subscript ‘\( o \)’, while those in the hidden layer are denoted by ‘\( h \)’. The resulting output is expressed in Equation (2.8).
\[ \sigma(u) = \text{tansig}(u) \]  
\[ Y_{output} = W_o \cdot \sigma(W_h \cdot X_{inputs} + B_h) + B_o \]  

The Cost, as described in Equation ((2.6), is minimized through a two-step process: the forward-backward pass and a non-linear optimization technique [120]. During the forward pass, the output \( Y_{output} \) in Equation ((2.8) is calculated using a training dataset of inputs. The Cost is then computed by comparing this output to the target value in the training data. To update the weights and biases, a direction to adjust them is determined using a non-linear optimization method, such as gradient descent. This process is performed for each data point in the training dataset. The forward and backward pass, together known as an epoch, is repeated until the error is sufficiently minimized or until a maximum number of epochs is reached.

Once the training has been completed, the FF-NN needs to be validated. This can be done by multiple methods, one of which is by overlaying the training scenario response over the FF-NN estimated response, this is shown in Fig. 2.5.

2.6.3 Feature Space Verification

Once the FF-NN has had data collected to explore the feature space, the network can be trained. The network is trained using supervised learning and used MATLAB’s deep learning toolbox to train using gradient descent. Once the network has been trained, the feature space can be approximated by the FF-NN, and is shown in Fig. 2.6. The surface is the FF-NN’s recreation of the input pairs to outputs and some of the original data points from the baseline DTB are used as a visual verification of the feature map agreement.
Figure 2.5 Zoomed dSoC During Feature Space Exploration: The network is trained to output the dSoC, and the FF-NN is zoomed in over a short time horizon to show how close the agreement is to the baseline DTB.

Figure 2.6 The Feature Space Recreated Surface: The FF-NN was trained off of the data taken from the discharge and was used to recreate a feature space map.

2.7 Observer-aided Behavioral Model

One of the powerful advantages of constructing these observers is their ability to accurately approximate states using reduced-order models. These models employ supervised-learned FF-NN-based observers and simplified abstracted models to repre-
sent the electrical domain of the system and the states to be observed. This combined model is referred to as the observer-aided behavioral model (OBAM). OABM represents the same two-battery system described in 2.2 and modeled in the baseline DTB. It consists of two components: the extended droop converter portion, which models the electrical interactions of the converter, and the SoC observers for both batteries. The aim is to create a model that requires significantly fewer computations while still providing highly accurate information for energy-balancing equations. The combination of the RC droop converter model and SoC observers are used to construct the OABM and a diagram of the model is shown in Fig. 2.7. This enables the calculation of bus voltage, current contribution, and power output for each converter. The converter’s power output serves as input to a trained neural network-based observer, approximating the State of Charge (SoC) for each battery.

2.7.1 RC droop Converter Model

When operating in RC droop, the converters are designed to act as if they have virtual passives, such as a virtual resistor that causes a voltage drop for some proportion of the current provided. Due to this design, they may be represented as these virtual passives in a simplified circuit with a voltage source at the value of the reference voltage in series with a passive at the value of the virtual passives [152]. This implementation neglects the bandwidth of the converter, so transients are not represented in the converter model, as well as there is no small signal represented in this model. This modeling technique assumes that the bandwidth of the voltage control loop is sufficiently slower than the current control loop and that the converter will be operating within its operating bandwidth and current limits. For some investigations, enforcement of limitations of particular bandwidth and limits may be required. This may be implemented into the model, but adding enforced limits to the model adds more states and thus computation and is not further investigated in this chapter.
Figure 2.7 Observer-aided Behavioral Model: The diagram uses virtual passives to represent the electrical contributions of each converter.

2.7.2 SoC Observers

The SoC observers have their integrator initialized with the initial SoC and require the power from the converter to approximate the SoC. The power is calculated from the electrical domain RC droop portion of the model and passed into the observer as an input. This allows for the SoC of the battery to be approximated off of the simplified electrical domain.

2.8 Computational Considerations

To simulate a scenario, the model uses mathematical operations to calculate successive time steps. The simulation speed depends on the operations per time step. The total computation time is influenced by the number of time steps; a single time step’s computation time is multiplied by the total steps. This calculation doesn’t consider variable time step determination time. MATLAB’s computational overhead affects computation time, though not directly explored here; it’s a key factor.
2.9 Results

The results are the comparisons between the baseline DTB and the OABM model. The two models will be introduced to identical load profiles and will have their electrical domain, SoC measurements, and computation time compared. The test scenario is designed to show multiple frequencies to see any disagreement between the two models. The electrical domain results will show the bus voltage and output current and will be comparing the simplified model used in the OABM to the switching average models used in the baseline DTB. The SoC will demonstrate how well the FF-NN is at approximating the SoC over a long-term scenario. Lastly, the computational time will be compared between the two.

2.9.1 Test Scenario

To compare the baseline DTB to the OABM, both systems will be set to the same droop set points, and both will be simulated through the load profile shown in Fig. 2.8. The droop configuration is set such that converter one has a virtual resistance of 1 Ω or one volt per amp, and converter two is set to 2 Ω. In this configuration, we expect converter one to sink and source two-thirds of the current, while converter two sinks and sources the remaining third. The load profile generated is designed to approximate a scaled version of a 24-hour load profile that experiences high solar penetration as well as periodic peaks over the course of the day. The load has a random perturbation that occurs every 10 minutes to add high-frequency transients. These transients are added to show how the OABM reacts to them despite the fact that the OABM is not tailored to handle them. Lastly, the SoC of battery one begins at 70%, while the SoC of battery two begins at 80%.
Figure 2.8  The Test Load Profile: The load profile simulates a residential scenario with high solar penetration and intermittent disruptions during midday.

Figure 2.9  Test Scenario Converter Currents: The currents from the converters that interface the batteries are shown from both models.

2.9.2 Electrical Domain

The comparison of the Electrical domain will verify the validity of the implementation of the simplified RC droop model in the OABM compared to the baseline DTB. It is expected that the models are in agreement except for the converter transients because the OABM is not designed to include these transients. The comparison between the baseline DTB and the OABM simulated converter currents is shown in Fig. 2.9, and bus voltages are shown in Fig. 2.10 where the OABM and DTB exhibit strong agreement, except for a few high-frequency spikes.
2.9.3 SoC Comparisons

The SoC and the dSoC are the primary focus of the FF-NN-based observer. The dSoC comparison will demonstrate how well the FF-NN is at approximating the instantaneous dSoC. Since the FF-NN approximates the dSoC and then is integrated, any error from the FF-NN approximation will cause the error to be integrated, meaning that the longer the simulation time and the greater the dSoC error is, the greater the SoC error could be. The comparison between the baseline DTB and the OABM battery SoCs is shown in Fig. 2.11.

2.9.4 Computation Time

The computational time of a simulation has many variable factors when deployed on a Windows machine, how the operating system manages memory, the physical hardware, background processes, as well as which CPU core gets assigned to run the threads [17]. While a lot of these processes hold an effect on the overall speed, many of these things can be neglected if the computational speed of the simulation is consistently faster. In order to reduce variance from these factors, the simulations
Figure 2.11 SoC throughout Test Scenario Comparison

Figure 2.12 Runtime Comparison Histogram: On the left, the OABM histogram is shown, with a zoomed-in version in the middle. The OABM histogram is centered around 0.7 seconds, while the DTB histogram is centered around 50 seconds.

will be run in a large batch. This is based on the idea that if the simulation is consistently and significantly faster on one machine, then it would translate to other machines based on the initial arguments for why the OABM model should be faster. The results of the batch of simulations are shown in the histogram in Fig. 2.12.
2.10 Conclusion

A feed-forward neural network-based observer was used to estimate the state of charge of a battery. The observer approximated the state of charge by analyzing the power supplied to the load through the battery’s converter. To train the network, various scenarios were explored by generating data through the simulation of a digital twin block model with a specific load. The network was trained to predict the derivative of the battery’s state of charge, taking into account the converter’s power and the current state of charge.

By integrating the feed-forward network, the observer was able to estimate the current state of charge, providing an abstracted electrical representation of the system. The observer considered the converter’s response to the bus in order to approximate the state of charge. This approach was utilized to develop the behavioral model known as the observer-aided behavioral model. To evaluate the performance of the SLAM, a day-long load profile was used to compare it with the baseline digital twin block model. The electrical characteristics of both models were well-aligned, with the exception of high-frequency components. Furthermore, the reported state of charge of the two batteries from both models agreed.

To measure the performance, the baseline digital twin block model and the SLAM were executed together and their run-time was recorded. A histogram was employed to illustrate the significant reduction in simulation run-time achieved by the SLAM, which demonstrated a more than 50-fold decrease in computational time. This substantial reduction in simulation run-time opens up possibilities for deploying the SLAM in online systems, particularly in iterative-based systems like simulation-based optimizers.
Chapter 3

Simulation-based Optimization of Virtual Impedance for Economic Dispatch: A Comparative Study of Meta-heuristic Algorithms

This study explores the application of a simulation-based optimizer for dynamic economic dispatch optimization of a microgrid using different meta-heuristic algorithms. A converter-based power system operating with a virtual resistance and capacitance control is used as the system for demonstration. Four meta-heuristic algorithms, including particle swarm optimization (PSO), genetic algorithm, modified genetic algorithm with correlative descent, and further modified genetic algorithm with correlative descent and global influence (GAw/CD&GI), are used to optimize the system with two distinct objective functions. The EDC control scheme allows for advanced power-sharing capabilities compared to voltage droop control. The system is modeled using neural network-based state observers and an abstracted behavioral model. The algorithms determine the EDC set points, and the system is simulated to evaluate the cost function and iteratively approach the near-optimum solution. The convergence metrics of each algorithm are recorded over 50 attempts. While all algorithms align the system near the optimum, PSO exhibits inconsistent reliability. The genetic algorithm performs reliably but is outperformed by GAw/CD. GAw/CD&GI consistently converges faster and more reliably, indicating the potential for optimizing EDC set
points using a simulation-based optimizer.

3.1 Introduction

With the implementation of economic dispatch [136], a microgrid can intelligently balance the supply and demand of electricity, taking into account factors such as energy costs, availability of renewable energy, battery storage capacity, and local consumption patterns. This optimization process minimizes operational expenses, reduces reliance on external power sources, and enhances the overall efficiency and resilience of the microgrid. Economic dispatch in a microgrid promotes the seamless integration of renewable energy resources, facilitates demand response initiatives, and improves energy management capabilities at the local level, thereby fostering sustainability, cost savings, and energy independence [147]. Microgrids fulfill economic dispatch by deliberate energy/power flow usage.

The power/energy flow usage is facilitated by an energy/power flow control scheme such as power control references [44, 26, 43], energy transfers/transactions [138, 86, 33, 158], or voltage droop control [96, 83, 118, 111, 49, 119]. Each power flow scheme has set points that define the behavior of its participants. In this study, voltage droop control is used in conjunction with a virtual capacitance referred to as extended droop control (EDC). This control scheme delegates the magnitude and frequency of power flow based on the voltage it is regulating, enabling a complex power-sharing scheme that may operate from discrete and asynchronous communication [152, 89, 155, 129]. This gives each participant four set points that change the behavior of the power flow and their interaction with other local participants. In this study, a DC bus has two voltage-regulating EDC participating converters that regulate a shared bus that supplies the load to make an isolated microgrid. These EDC set points can be used to facilitate near-optimal power flow around a cost function, but the optimal EDC set points must be selected to minimize cost.
An objective function, cost function, and fitness function are functions that are used to evaluate the performance of a particular scenario or state. These functions are made with the intention of giving an accurate metric to help achieve the desired goal of the optimizer. The function needs to account for everything that needs to be evaluated, which can lead to highly complicated functions that become difficult to explore. A few aspects of the cost function that affect how easily the function may be explored are the number of extremes, how rugged the function is with respect to its parameters, and how wide the domain to search is. Metrics on how difficult it is to explore a function are possible to calculate using techniques like the Welsch transformation [72]. Understanding the properties of the cost function is important when choosing which optimization algorithm should be deployed, as some are capable of handling different types better. The cost functions used in this chapter are two mathematically defined relationships between the cost and current supplied by each converter to compare the meta-heuristic algorithms. The system cost function then can be optimized to determine the optimal set points.

Linearizing the optimization problem to solve for the instantaneous optimum is a popular method [64, 157, 39, 156, 115]. This approach fails to consider considerations of energy reserves, thermal considerations, non-linear relationships, and a policy that would defer from a policy that is optimum over a scenario. Another method is to discretize the system into multiple state representations or time windows, allowing for the problem to be addressed by Integer programming [77, 150, 94, 66]. However, the problem with the discretization implementation is that the number of states and the resolution of their states increases the number of investigation instances. This can create a situation where the computational requirements are infeasible. A popular approach is the usage of meta-heuristic algorithms, such as particle swarm optimization (PSO) or genetic algorithm (GA) [37, 135, 140, 28, 51, 46, 74, 4, 108, 10]. Since the parameter set of EDC includes both frequency domain and time domain factors
mixed with system factors, including non-linearities, randomness, and numerous constraints inherent in power systems, the four optimization techniques investigated are particle swarm optimization (PSO), genetic algorithm (GA), the genetic algorithm with correlative descent (GAw/CD), and a further modified genetic algorithm with correlative descent and global influence (GAw/CD&GI), as well as modified variants of GA.

In order to evaluate the cost-to-parameter relationship, a digital twin model can be used to simulate through a simulation using these set points and providing a cost associated with them and the meta-heuristic algorithm can then iterate to search for near-optimum set points, known as simulation-based optimization (SBO) [57]. The concept of a digital twin encompasses a comprehensive digital representation of a system, including its properties, conditions, and behavior. This simulation approach allows for the evaluation to include randomness and non-linear relationships within the digital twin. By simulating hypothetical scenarios, a digital twin allows for realistic behavior to be replicated in a deployed environment [59]. With the assumption of an accurate digital twin representation, SBO struggle with the problems inherent to the meta-heuristic algorithms they employ, such as stagnation and convergence. The number of iterations required to find the optimum depends on the system’s objective, hyper-parameters, and algorithm. In this investigation, the four different algorithms are tested against each other by giving each algorithm a parallel batch size of 16 instances to iterate through 25 search epochs. Metrics were taken to compare the convergence speed and quality to the optimum, and the metrics show that the best to worst performer for this case was GAw/CD&GI, GAw/CD, GA, then PSO. For dynamic implementation, the best-performing algorithm should be deployed to increase the time to align set-points to a new forecast.
3.2 System Under Investigation

The system used in this investigation is an isolated DC microgrid that consists of two bus regulating converters, a common bus DC, and loads shown in Fig 3.1. The two converters are referred to as Converter A and Converter B, both are operating in EDC to regulate the bus voltage and delegate loads between the two. The common DC bus is what connects the converters and the loads together, and this bus voltage is the voltage that is regulated. The load is a fixed two Amp load with a pulsed load with an amplitude of two Amps with a period of three seconds, and a duty cycle of 0.5.

EDC allows the converters to share the power flow between the two using the bus voltage droop as communication. The EDC is an extension to a voltage control loop that allows for the bus voltage to be regulated within a designed tolerance while delegating loads based on their magnitude and frequency. This concept is described in [152]. The converters operating in EDC have four set points each, each set point is used as a parameter to optimize the system for the objective. The four EDC parameters are the voltage reference, the virtual resistance, the virtual capacitance, and the initial voltage drop across the capacitor. The voltage reference sets the expected voltage of the bus under no load. The virtual resistor accounts for how many Volts per amp the bus voltage will drop. The virtual capacitor allows for frequency delegation between the two converters. The voltage drop across the capacitor is used to initialize the slow time constants of the steady-state response of the RC interactions.

Figure 3.1 System Level Diagram: Converter A and Converter B are each connected to a source, and both are connected to the DC bus. The converters regulate the DC bus voltage and facilitate the sharing of the loads by means of EDC.
between the converters. With two converters, the system will have eight set points that act as parameters for the simulation-based optimizer to optimize the power flow.

3.3 Cost Function

Two cost functions are used to demonstrate how the meta-heuristic algorithms perform for optimization of differing complexity.

3.3.1 Cost Function One: Peak Shedding

This cost function is designed to investigate the efficacy of the simulation-based optimizer and is able to find the global minimum for a cost function with a linear derivative using the EDC parameters. At the global minimum, the electrical response is identifiable from a time-series waveform of the converter currents by the peak shedding behavior encouraged by the converters. The cost function is built such that it evaluates the cost of current from the two different bus-tied converters. The cost of Converter A is a linear relationship, and the cost of Converter B is parabolic, this is seen in eq. (3.1). \( I_a \) and \( I_b \) are the current contributions from Converter A and Converter B, respectively.

\[
\text{Cost}_1(I_{a,b}) = 4I_a + I_b^2
\]  \hspace{1cm} (3.1)

The partial derivatives of the cost function with respect to \( I_a \) and \( I_b \) are shown in eq. (3.2) and eq. (3.3) respectively. The partial derivative of cost with respect to \( I_a \) becomes greater than the partial derivative of cost with respect to \( I_b \) at 2 Amps, this is graphically shown in Fig. 3.2. This means that the optimum is to use converter B to source current for any load 2 amps or less and to use converter A for all loads greater than 2 amps. This behavior will appear as peak shedding from the perspective of Converter B. The intersection of the two is at 2 amps, suggesting that the optimum
for this cost function would be to use $I_b$ until 2 amps, then use $I_a$ for any higher load current.

$$\frac{\partial Cost_1(I_{a,b})}{\partial I_a} = 4$$  \hspace{1cm} (3.2)$$

$$\frac{\partial Cost_1(I_{a,b})}{\partial I_b} = 2I_b$$  \hspace{1cm} (3.3)

### 3.3.2 Cost Function Two: Non-linear Cost

One of the known problems of any optimization problem is the possibility of the algorithm finding a local optimum instead of the global optimum. This cost function is designed to investigate the susceptibility of the proposed algorithms to becoming trapped in other local minimums. This cost function is described mathematically in eq. (3.4). The equation was designed to have more optimums by using a fourth-order polynomial.

$$Cost_2(I_{a,b}) = I_a + \frac{1}{4}I_b^4 - I_b^3 + I_b^2 + I_b$$  \hspace{1cm} (3.4)

The partial derivative of the second cost function with respect to $I_a$ and $I_b$ is shown in eq. (3.5) and eq. (3.6) respectively and is graphically shown in Fig. 3.3. This cost function provides a more complicated objective for the simulation-based optimizer.

![Figure 3.2 The Partial Derivatives of the First Objective Function: The plot is shown with respect to each converter. The partial with respect to $I_a$ is a fixed value of 4, and the partial derivative of $I_b$ is a line with a slope of 2 due to its cost being parabolic.](image-url)
to optimize around, with the expectation of Converter B operating between 1 to 2 amps, and Converter A supplying the rest of the load current.

\[
\frac{\partial \text{Cost}_2(I_{a,b})}{\partial I_a} = 1
\]  

(3.5)

\[
\frac{\partial \text{Cost}_2(I_{a,b})}{\partial I_b} = I_b^3 - 3I_b^2 + 2I_b + 1
\]  

(3.6)

3.4 Model for Simulation Evaluation

Simulation is the core of simulation-based optimization because the simulation reports back the cost of system operation for a particular set of parameters. A model must encapsulate everything required to accurately evaluate the cost function through a simulation. The computational time of the simulation is also relevant because each epoch is a batch of multiple simulations, making the number of total simulations run the product of the epochs by the batch size. The batches can be paralleled, but the epochs cannot, and the simulation-based optimizer will have to run the product of the batch as many epochs as it takes to converge on an optimum, often resulting in many simulations. There are many different approaches to modeling power systems, the one deployed is an abstracted model that simplifies the electrical domain and uses supervised learning for state observers.

![Graph](image)

Figure 3.3 Partial Derivatives of Second Cost Function: The partial derivatives of the cost function are plotted to visualize the non-linear relationship.
The observer-aided behavioral model is designed to model the converter interactions between multiple bus-tied converters operating in EDC while using machine learning to approximate the states of relevant metrics such as battery state of charge. This model maintains a similar level of fidelity to a switching average model but is optimized around computational speed, allowing for a moderate level of accuracy with the capability to iterate through the simulation-based optimizer much faster than similar accuracy models.

3.5 Meta-heuristic Algorithms

3.5.1 Particle Swarm

Particle Swarm Optimization (PSO) is a popular stochastic optimization technique that utilizes simple mathematical formulas to tackle complex problems. Inspired by the social behavior of bird flocks searching for food [126]. In this swarm-based algorithm, particles move throughout the search space seeking the optimum position that minimizes a given cost function. The particles move each iteration according to a calculated velocity. There are 3 components that dictate the velocity of each particle. The first component is Inertia, a particle’s velocity is inherited from the previous iteration and added to the particle’s new position. The second component is the social component, which updates the particle’s velocity based on the global best position of the entire swarm. The final component is the cognitive component, updating a particle’s velocity based on its own personal best position [74]. These 3 components help each particle iteratively move to its best position and the global best position in the search space. While this algorithm is a simple solution to many problems, it fails to deliver in some respects.

The position of each particle in the swarm is updated based on its current position, velocity, and the best position found by the particle itself (personal best) and the best position found by any particle in the swarm (global best). The position update
equation for each particle is given by eq. (3.7).

\[ x_i(t+1) = x_i(t) + v_i(t+1) \]  \hspace{1cm} (3.7)

The velocity of each particle is updated based on its current velocity, the cognitive component (personal best), and the social component (global best). The velocity update equation for each particle is given by eq. (3.8).

\[
v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_{\text{best}_i}(t) - x_i(t)) + c_2 \cdot r_2 \cdot (g_{\text{best}}(t) - x_i(t)) \] \hspace{1cm} (3.8)

The initial positions and velocities of the particles are randomly generated within the search space. The initialization equation for the position and velocity of each particle is given by eq. (3.9).

\[
x_i(0) = \text{randomly generated position} \]
\[
v_i(0) = \text{randomly generated velocity} \hspace{1cm} (3.9)

Variables:

- \( x_i(t) \): Position of particle \( i \) at time \( t \)
- \( v_i(t) \): Velocity of particle \( i \) at time \( t \)
- \( p_{\text{best}_i}(t) \): Personal best position of particle \( i \) at time \( t \)
- \( g_{\text{best}}(t) \): Global best position found by any particle in the swarm at time \( t \)
- \( w \): Inertia weight, controlling the impact of the previous velocity
- \( c_1 \): Cognitive coefficient, controlling the influence of the personal best position
• \( c_2 \): Social coefficient, controlling the influence of the global best position

• \( r_1, r_2 \): Random values in the range \([0,1]\) used to introduce stochasticity in the velocity update equation

As problems increase in complexity and the size of the search space increases dramatically. The search space now has plenty of local minimums that can trap particles and reduce the convergence speed [143]. Running a PSO algorithm twice on the same problem may result in two different solutions. This is because PSO is dependent on a lot of randomnesses when initialized and when calculating velocity. To combat the accuracy and convergence issues, this chapter proposes that the cognitive component is replaced with a non-random correlative descent. The particle swarm algorithm process is shown in Fig. 3.4.

3.5.2 Genetic Algorithm

Stochastic Optimization is a category of methods for finding extremas of an objective function using randomness [51]. There are many different types of stochastic optimization techniques, making it difficult to draw generalizations. How the randomness is applied and how the optimizer uses information changes the effects it has on the optimizer. The stochastic search deployed in this chapter is a type of GA [50]. This
is named after the analogy of how it operates, the fittest candidate produces offspring that are slight variations of itself, and this generational process repeats until it discovers the fittest candidate possible. The fittest is used to refer to the fittest in the generation. This algorithm is a simple and effective local search algorithm with the only information being shared between generations being the origin of its ‘parent.’ A diagram of the genetic algorithm is shown in Fig. 3.5 by the black portion of the diagram.

This algorithm is a fast and effective local extrema search algorithm. This makes it a powerful tool for single-objective optimization goals, where there is only one optimum extrema. Otherwise, this algorithm is highly susceptible to getting ‘trapped’ in local extrema and not being able to discover the global extrema. What allows this trapping is a mixture of the search algorithm’s hyper-parameters and the size, roughness, and shape of the traps. There are also additional methods that may be added to the algorithm to avoid this problem such as multiple particle starting and particle reincarnation [128].

3.5.3 Genetic Algorithm with Correlative Descent

A common technique for optimizing a non-linear differentiable function around a particular cost is using gradient descent. This is a popular method of training neural networks, and is used in the forward-pass back-propagation method of updating the learnables in supervised learning, learnables being the weights and biases that are the parameters that will change based upon the training [73]. This leads to the idea of running a simulation and using the resulting cost similarly to the forward pass, and the optimizable set points of the system as the learnables that must be adjusted to optimize the system. The problem is that a power system has constraints, randomness, and non-linear interactions which makes deriving a cost-to-parameter gradient difficult to the point that it not have been achieved without significant
linearization or gradient approximation [102]. This method then requires another means of determining the direction by which to update the parameters.

The Pearson correlation coefficient is a normalized covariance between two sets of data. It is a scalar between -1 and 1 that represents the strength of the correlation, with 1 being a perfect linear correlation, 0 being no correlation, and -1 being a perfect inverse linear correlation. This is mathematically described in eq. (3.10) from [30]. Both $X$ and $Y$ are arrays of data. The $\text{cov}$ function in eq. (3.10) is the covariance between $X$ and $Y$, $\rho_X$ is the standard deviation of $X$, and $\rho_Y$ is the standard deviation of $Y$. 

Figure 3.5 Flowchart of Genetic Algorithms: The diagram shows GA with black, GAw/CD with black and blue, and GAw/CD&GI with black, red, and blue.
\[
\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y}
\]  

(3.10)

The process starts with a batch of simulations that explores a perturbance of the parameters intended to search a local region of the potential possible parameters. The batch reports how each simulation performed with regard to the parameters it simulated and the cost it accumulated. The Pearson correlation is calculated between the cost and the parameters, this is to approximate the linear correlation between the cost and each parameter under investigation. Then the cost-to-parameter correlation coefficient is compared to a threshold that is set as a hyper-parameter. If the coefficient is lower than this threshold, then the parameter value is not altered, but if it is higher than this threshold, then the value of the parameter is changed to follow the correlation for better cost. This is shown in Fig. 3.5 by the black and blue portions of the diagram.

3.5.4 **Genetic Algorithm with Correlative Descent and Global Influence**

The way this algorithm works is by having multiple particles that act as a neighborhood for the GA to search. The GA searches an area with a handful of mutations of the set points, and these set points return their fitness after their simulations. This neighborhood search information is then used to calculate the cost to parameter Pearson correlation, and then the influence it should have on the neighborhoods. The fittest in the neighborhood shares information with the other neighborhoods and moves in the direction proposed by the correlation descent, then the global influence from the other neighborhoods is used to determine the next search location of the neighborhood. The flowchart of a single neighborhood interaction is shown in Fig. 3.5 by the black, blue, and red of the diagram. Multiple neighborhoods may be used to search the exploration space, and are initialized randomly in the exploration space. This allows a different method of exploration of the parameters and objective.
3.6 Results

The test scenarios are designed to compare the strengths and weaknesses of the four algorithms. Each simulation-based optimizer will have a batch size of 16 simulations, and 25 epochs to converge on a result. Because the simulation-based optimizer is not deterministic in its convergence, 50 iterations of each simulation optimizing the EDC set points will be run for each cost function.

The results are broken into a cost domain and an electrical domain. The cost domain is used to compare results between how the simulation-based optimizer converged to the optimum across the two different cost functions. The electrical domain is used to verify that the EDC set points operate electrically as expected while at the optimum. All of the search algorithms implemented store their global best and the fittest per batch of 16 simulations. The cost domain analysis will be looking at the best cost per batch, this is to determine how the algorithm converges instead of how quickly the simulation-based optimizer gets a random highly fit candidate early in the epoch cycle. Due to a lot of randomness in the simulation-based optimizer, the 50 instances of each algorithm iterated through 25 search epochs for each of the cost functions. Due to the multitude of outputs from the algorithms, the electrical domain will be expressed in the current waveform produced by the best candidate across all of the algorithms found, which is a good representation of how all of the successfully converging set points performed.

3.6.1 Cost Domain: Cost Function One

The different meta-heuristic algorithms optimize the EDC set points around the first cost function, and all have their fittest cost per batch compared to the epoch plotted for all 50 iterations. This visualization allows a visual comparison of how each of the different algorithms performs, converges, and settles. Each algorithm has a ‘cooling’ effect that decreases the entropy of the system over time. The search trend of the
3.6.2 **Cost Domain: Cost Function Two**

The simulation-based optimizer is used to optimize the EDC set points for the second cost objective. The best for batch compared to epoch is used again to show the behavior of the simulation-based optimizer. The search trend of the algorithms for the second cost function is shown in Fig 3.10.

3.6.3 **Electrical Domain**

The electrical domain results are the global best-set point results discovered for each objective function. This is because all of the EDC set points that were discovered
Figure 3.8  Algorithm Performance Spider-plot: The performance metrics are visualized to compare the four different meta-heuristic algorithms.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Cost Func</th>
<th>Epoch</th>
<th>GAw/CD</th>
<th>GA</th>
<th>GAw/CD&amp;GI</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1</td>
<td>15</td>
<td>60.66</td>
<td>63.41</td>
<td>62.61</td>
<td>71.07</td>
</tr>
<tr>
<td>σ</td>
<td>1</td>
<td>15</td>
<td>2.13</td>
<td>4.01</td>
<td>2.66</td>
<td>23.43</td>
</tr>
<tr>
<td>Mean</td>
<td>1</td>
<td>25</td>
<td>58.85</td>
<td>62.05</td>
<td>61.53</td>
<td>70.43</td>
</tr>
<tr>
<td>σ</td>
<td>1</td>
<td>25</td>
<td>0.43</td>
<td>3.66</td>
<td>2.09</td>
<td>25.49</td>
</tr>
<tr>
<td>Mean</td>
<td>2</td>
<td>15</td>
<td>46.41</td>
<td>46.53</td>
<td>46.39</td>
<td>51.36</td>
</tr>
<tr>
<td>σ</td>
<td>2</td>
<td>15</td>
<td>0.27</td>
<td>0.62</td>
<td>0.28</td>
<td>31.11</td>
</tr>
<tr>
<td>Mean</td>
<td>2</td>
<td>25</td>
<td>46.24</td>
<td>46.33</td>
<td>46.29</td>
<td>52.52</td>
</tr>
<tr>
<td>σ</td>
<td>2</td>
<td>25</td>
<td>0.046</td>
<td>0.26</td>
<td>0.18</td>
<td>31.31</td>
</tr>
</tbody>
</table>

Table 1. Convergence Metrics of Algorithms

by the different simulation-based optimizer and iterations acted nearly identically to the other optimum points discovered. The electrical domain is used to verify the response that EDC set points allow. The electrical optimum for the first objective function is seen in Fig. 3.6. The results are as expected, the base loads are handled by Converter B, and the 2 amp pulse is handled by Converter A.

The discovered electrical optimum for the second objective function is seen in Fig. 3.7. The results are as expected and are justified by the cost function, the electrical domain is used to verify the response that EDC set points allow. Converter
Figure 3.9 Comparison of Search Trends for First Cost Function

A operates within the region where its partial derivative is less, outside the bounds of [1-2amps]. Converter B current then operates within the region within [1-2 amps] to supply the load.
3.6.4 Comparison Metrics

To ensure a meta-heuristic algorithm is an effective tool for real-time optimization in economic dispatch, it is desirable to minimize the number of epochs required to find the best option. We assess this behavior by comparing the mean and standard deviation at 15 and 25 epochs for both cost functions. The following metrics are
provided along with an explanation of their relevance, with the metric values shown in Table 1, and visualized in Fig. 3.8. The PSO performed the worst by a large margin in all metrics taken. The genetic algorithm placed third in all metrics taken, however, performed significantly better than the PSO. The GA w/ CD performed slightly better than just the GA in all metrics. Finally, the GAw/CD&GI performed best in all metrics taken.

**Performance metrics of the algorithms for each cost function:**

- **Mean at Epoch 25**: This metric indicates the effectiveness of the algorithm’s convergence by evaluating the degree of settlement based on the mean value.

- **Standard Deviation at Epoch 25**: This metric indicates the effectiveness of the algorithm’s convergence by evaluating the degree of confidence in the settlement based on the standard deviation.

- **Mean at Epoch 15**: This metric indicates the effectiveness of the algorithm’s speed of convergence by evaluating the degree of settlement based on the mean value at an intermediate value.

- **Standard Deviation at Epoch 15**: This metric indicates the effectiveness of the algorithm’s speed of convergence by evaluating the degree of confidence in the settlement based on the standard deviation at an intermediate value.

3.7 **Conclusion**

The study utilized four meta-heuristic algorithms in a simulation-based optimizer to optimize parameters for a model of a simple example microgrid with two converters operating in extended droop control (EDC). The model represents power systems operating in EDC and is a behavioral model with similar fidelity to a switching average model. Two different objective functions were tested, and each meta-heuristic
algorithm had 25 epochs to find the optimal solution. To address stochasticity in
the simulation-based optimizer, 50 iterations of each combination of objective func-
tion and optimizer were executed. The four algorithms examined were GA, PSO,
GAw/CD, and GAw/CD&GI. The evaluation of the results focused on two key fac-
tors: the algorithms' ability to find the optimum and the speed of convergence. The
mean and standard deviation at epoch 15 and epoch 25 were used as metrics to as-
sess these criteria. In terms of performance, the PSO performed the worst across
all metrics for this application. The GA ranked third in all metrics but still per-
formed well enough to consistently find near-optimal set points. The modified GA
with Correlative Descent showed better performance than the standard GA. Finally,
the GAw/CD&GI exhibited the best performance across the metrics considered. For
offline usage of the simulation-based optimizer without strict convergence time con-
straints, running multiple iterations of any meta-heuristic algorithm and storing the
best solution from all iterations could be employed. However, for real-time deploy-
ment, where faster convergence is desired and confidence in optimization is crucial,
the GAw/CD&GI is better suited for such scenarios.
CHAPTER 4
ALIGNMENT OF POWER AND ENERGY SYSTEM FOR ECONOMIC DISPATCH FACING A STOCHASTIC LOAD FORECAST

This chapter addresses the complexities of the pre-alignment of a power and energy system (PES) for optimal power flow when facing a stochastic load forecast projection. In this chapter, a demonstrative PES using virtual impedance to facilitate bus regulation and load delegation between energy sources is investigated. This developing configuration is popular in hybrid energy systems as it allows a complex energy management strategy that is distributed and requires few tunable parameters. Since there is no consensus on how this type of system should operate this chapter looks into three options for battery charge management. The system then uses simulation-based optimization with the employment of a metaheuristic algorithm to optimize the system around different scenarios, also referred to as aligning the system to that load profile. The chaotic relationship between the load and energy management policy adopted by a system that has been aligned to a specific load profile is shown to demonstrate the required care needed when aligning these systems. Lastly, a comprehensive training load that is derived from the load forecast projection is proposed to answer the problems of state appraisal, chaotic behavior, and sustainable energy management policies. This comprehensive load profile is used to align a simple SoC management strategy and a neural-network-based strategy to demonstrate the requirement of tunable parameters to achieve a dynamic sustainable policy. Lastly,
the neural-network-based system is compared to a popular solution to this problem, which is a PI feedback loop for SoC charge management. This comparison suggests that the neural-network-based system has adopted a policy that may yield an average expected reduction in fuel consumption of 1.91%.

4.1 Introduction

Properly managing energy is crucial for the efficient operation of a power and energy system. This is typically achieved by implementing intelligent systems to oversee energy management. However, achieving optimal performance in energy management systems is challenging due to their multi-objective, cross-domain, stochastic, and chaotic nature [91, 149, 112, 77, 45]. This is coupled with the complexities of the pre-alignment of a power system to adjust energy policies based upon a set of new information, forecasted load projections, or changes in energy values [21, 100, 40, 55, 148].

The objectives of the power and energy system for this investigation serve as a foundation for demonstrating the complexity of the formulation of these problems. The three objectives are to maintain bus voltage within tolerance, regulate the battery state of charge (SoC) within set rules, and minimize the fuel consumption for power generation. The incorporation of energy storage devices may require an investigation into the frequency domain. This is because certain energy generators lack the bandwidth or have a rate limit imposed on them that makes bus regulation impossible on their own [14, 110, 142, 103, 151]. Also, the composition of energy storage may make it more suitable for specific frequency ranges of operation [87, 144, 152, 84, 85]. In our investigation, we deliberately restrict the output of two generators to create a bandwidth deficit, necessitating the battery’s involvement in maintaining bus voltage tolerance when serving a load.

Another common challenge faced by power and energy systems is the unpre-
dictable nature of the loads they serve, which can vary stochastically [7, 116, 101, 88, 95]. This uncertainty means that the system cannot operate optimally but should strive for the best possible solution while remaining prepared for changes in the load experienced by the system. For our investigation, we confine stochastic inputs within a forecasted load projection.

Furthermore, optimizing power and energy systems is complicated by the chaotically deterministic relationship between system inputs and outputs [25, 27, 75, 153]. This means the boundary conditions of input do not necessarily correspond to those of the system’s output boundary conditions, making statistical analysis challenging. In our investigation, we treat the load as the system input and the SoC as the output, illustrating their chaotic behavior before proposing a solution.

A popular approach to optimize such systems is simulation-based optimization (SBO), which often employs metaheuristic algorithms to aid in decision-making and parameter optimization [81, 99, 35, 52, 18, 109]. However, determining which scenario the system should optimize around is problematic due to the stochastic input and chaotic output. This problem is demonstrated by showing the system’s energy response when the system experiences an alignment mismatch with the load. This alignment challenge is addressed by providing the system with sufficient training data through the design of a comprehensive load profile, inspired by machine learning techniques of feature exploration and emergent policy encouragement [67, 145, 13, 78, 97]. These emergent policies require not only ample training data but also a sufficient number of tunable parameters to facilitate the adoption of a dynamic near-optimal policy.

This study investigates three different SoC control strategies, two in their policy adoption capabilities and one as a baseline. Battery-tied converters participate in bus voltage regulation by operating as virtual capacitance and resistance, managing their average response through a reference voltage. The first strategy employs a fixed ref-
erence, and the second utilizes a feed-forward neural network as a feedback controller, with its weights and biases optimized through SBO. Lastly, a hand-designed PI SoC feedback controller serves as a baseline, as it is a common solution to SoC regulation [82, 19, 122, 65, 90]. We introduce a “no compensator” case to emphasize the need for tunable parameters, showcase the flexibility of the neural network, and compare its efficacy against the PI controller to validate the adopted policy’s performance.

In summary, our study explores the complexities of optimizing power and energy systems to a forecast load projection, addressing challenges related to stochastic inputs, chaotic outputs, and the adoption of policies. We propose a comprehensive load design inspired by machine learning techniques and investigate three distinct SoC control strategies to assess their policy adoption capabilities.

4.2 System Under Investigation

![System Under Investigation One-Line Diagram](image)

Figure 4.1 System Under Investigation One-Line Diagram

The system being investigated consists of two generators, a battery, loads, and bus interfacing power converters. All converters operate using virtual impedance for droop control to regulate the bus voltage and facilitate power sharing between them. A one-line diagram of this configuration is shown in Fig. 4.1. However, the converters of the generator have certain limitations, such as ramp rate, AVR response, maximum current, and single current unidirectional. These restrictions are imposed on the
generator-tied converters to explore solutions to a power generation source operating in a bandwidth deficit for bus voltage regulation. This type of deficit is normally caused by the avoidance of mechanical stress or electro-mechanical restrictions for the power output limits of the generator.

On the other hand, the converter of the battery utilizes both virtual capacitance and virtual resistance without any additional limitations. This control implementation allows the bus-tied converters to participate together to maintain voltage within the desired tolerance. The battery’s converter operates by charging and discharging the battery to manage the higher frequency loads that are unable to be serviced by the generators which is realized through the virtual capacitance in the control loop. The battery’s interfacing converter’s control loop is shown in Fig. 4.2, the input to the outermost control loop is the voltage reference, which dictates the lower frequency behavior of the battery’s power flow. A primary focus of this chapter is on how that voltage reference should be managed in order to allow for a stable SoC regulation such that the battery can maintain its rule in bus regulation while charging when it is most efficient for the generators to charge. The SoC regulation options explored are shown in section 4.2.1.

The strategy used to charge the battery can significantly impact the overall fuel efficiency of the system. This is because the efficiency of the generator is affected by the amount of power it provides, as well as the losses incurred during the battery’s charge and discharge process.

4.2.1 Battery SoC Control Scheme

The battery-tied converter has a fixed virtual resistance, while the virtual capacitance \( V_c \) and the initial voltage across the virtual capacitor \( V_{c0} \) are parameters to be optimized. The reference voltage is manipulated to manage the power flow using two separate methods. The first is a fixed reference that does not change throughout the
scenario and is to be optimized. The second is a neural network-based regulator that
dynamically changes the reference voltage whose weights and biases are optimized
for. Lastly, there is a PI Regulator for SoC that is not tuned by the optimizer. The
PI regulator is used as the baseline as it is a popular and reliable solution to this type
of system.

**EDC with no SoC Regulation**

The first method involves a fixed voltage reference chosen by the optimizer. This
method is the simplest implementation but lacks adaptability to deviations from the
optimized load scenario. Learnable parameters: $V_c, V_{c0}, K_{V_{ref}}$.

$$V_{Reference} = K_{V_{ref}} \quad (4.1)$$

**EDC with Neural Network SoC Regulation**

A neural network SoC regulation is achieved using a 2-10-1 feed-forward neural net-
work. Neural networks have demonstrated potential in feedback controls for SoC
regulation [hussein2015derivation]. The inputs are the $SoC$ and the bus voltage
$V_{bus}$, representing the ‘2’ in the 2-10-1 structure. The hidden layer consists of 10
neurons with a Tansig activation function. The output, representing the ‘1’, is the
reference voltage. The calculation of the voltage reference is given by Eq. 4.2, where
\( \tilde{V} \) and \( \tilde{W} \) denote weights and biases. Learnable parameters: \( V_c, V_0, \tilde{V}, \tilde{W} \). The neural network has a total of 41 learnable weights and biases.

\[
V_{\text{Reference}} = \tilde{V} \cdot \text{tansig}(\tilde{W} \cdot \tilde{X})
\]
\[
\tilde{X} = \text{transpose}[1, \text{SoC}, V_{\text{Bus}}]
\]  

4.2.2 EDC with PI SoC Regulation

A Proportional-Integral (PI) feedback loop is implemented and is to be viewed as a baseline battery charge regulation to compare. The input to the loop is the negated difference between the measured \( \text{SoC} \) and the desired \( \text{SoC}_{\text{SetPoint}} \); and the output is the \( V_{\text{Reference}} \).

The negation is applied due to the inverse relationship between the reference voltage and the battery current. The calculation of \( V_{\text{Reference}} \) is defined by Eq. 4.3. This implementation is considered to be more traditional, understandable, and robust. Learnable parameters: \( V_c, V_0, \text{SoC}_{\text{SetPoint}}, K_i, K_p \).

\[
V_{\text{Reference}} = (\text{SoC}_{\text{Measured}} - \text{SoC}_{\text{SetPoint}}) \cdot (K_i \cdot s + K_p)
\]  

4.2.3 Generator Control Scheme

For the generator’s converter, the voltage reference and virtual resistance are fixed, with no virtual capacitance. The bandwidth restrictions are enforced in the inner current loop. The bandwidth controls are shown in Fig. 4.3 to create the deficit to demonstrate the battery participation.

4.3 Load Forecast Projections

The load forecast projections for this investigation are fabricated and are not specific to any hardware and are used as a specific to help describe the general process and problems with stochastic load optimization. The load forecast projection is shown in
Fig. 4.4, where the **Worst Case** is the most energy-intensive instance possible for the system to come across, and the **Most Probable Case** is the instance that is the most likely load the system would experience. Potential instances are generated to demonstrate the potential other instances the system could experience.

### 4.4 Optimization Method

The optimization objectives represent the desired outcomes for the system, and these values are mathematically translated into a cost function. This type of optimization problem does not have a known numerical solution, so the approximation of the solution is achieved by simulation-based optimization (SBO) with the employment of a meta-heuristic algorithm. The objective of the optimizer is to minimize the cost function, taking into account all of these considerations.
4.4.1 Cost Function

This relationship is expressed in Eq. 4.4. The cost function incorporates three different considerations: the fuel usage of the system, the bus voltage tolerance, and the battery SoC tolerance.

\[
Cost_{Total} = \int_0^{T_{end}} (Cost_{Fuel} + P_{Vbus} + P_{SoC}), dt \tag{4.4}
\]

Generator Fuel Cost

The generator fuel cost is determined by approximating the relationship between the power output of the generator’s converter and the fuel consumed by the generator. This approximation involves mapping the fuel flow rate to the generator and the power output of the generator, and then mapping the power output of the generator to the power output of the converter. In this case, a relationship similar to that of a scaled gas turbine engine is utilized to better align with the available power capabilities for laboratory verification at UofSC in the future.

The relationship between the fuel flow rate and power is measured and fitted using the function shown in Eq. 4.5. This function is also depicted graphically in Fig. 4.5, which illustrates the nearly square root relationship of the function. It is important to note that the function is scaled in power units, and the fuel flow rate represents the expected flow rate from the generator at a given power rating of the bus-tied converter.

\[
Cost_{Fuel}(P_{Gen}) = 929.9 \cdot P_{Gen} + 66.9 \tag{4.5}
\]

\[
P_{bus} = P_{gen} - P_{conduction} - P_{switching} \tag{4.6}
\]
DC Bus Voltage Tolerance Penalty

The DC bus is regulated by two converters operating in Extended Droop Control (EDC), which means that the bus voltage changes as the load changes. However, the extent of this voltage change depends on the EDC set points. It is crucial for a DC system to maintain the voltage within a specific range to ensure the proper operation of devices connected to the bus. Therefore, the objective function includes a penalty for voltage deviations from the desired tolerance.

Under-voltage is undesirable because it can lead to issues when the generator is connected to the bus through an active rectifier. The voltage must remain sufficiently high to operate within the boost region and avoid entering the passive rectification region. Failure to do so would result in improper bus voltage regulation and a failure of the EDC scheme to function as intended. Conversely, over-voltage is also problematic. The converters have a maximum tolerated voltage, mainly determined by the switching devices or passive components used in their construction. Exceeding this maximum voltage threshold could cause a failure in switches, capacitors, or other...
The consideration for DC voltage tolerance is mathematically expressed in Eq. 4.7. The values of $A_v$ and $B_v$ are chosen based on the desired level of strictness for the penalty.

$$V_d = |V_{bus} - \frac{(V_{BusMax} + V_{BusMin})}{2}|$$

$$P_{V_{Bus}}(V_{Bus}) = \begin{cases} 
A_v * V_d & V_{Bus} \geq V_{BusMax} \\
0 & V_{BusMin} \leq V_{Bus} \leq V_{BusMax} \\
B_v * V_d & V_{Bus} \leq V_{BusMin} 
\end{cases} \quad (4.7)$$

Here, $V_{bus}$ represents the actual bus voltage, $V_{upper}$ denotes the upper voltage limit, and $V_{lower}$ represents the lower voltage limit. The constants $A_v$ and $B_v$ are selected based on the desired penalty severity.

**SoC Tolerance Penalty**

The battery State of Charge (SoC) tolerance is incorporated for three reasons: to preserve emergency energy reserves, to ensure the battery operates within feasible limits, and to limit the depth of charge/discharge to extend the battery’s lifespan. These objectives are achieved by setting minimum and maximum SoC values and enforcing them through penalty functions. The minimum SoC guarantees the presence of emergency energy reserves and prevents the battery from fully discharging. The maximum SoC ensures that the battery is never overcharged. Although not included in this study, a cost consideration for depth of discharge could be added.

The penalty for SoC tolerance is mathematically expressed in Eq. 4.8. The constants $A_{SoC}$ and $B_{SoC}$ are selected based on the desired level of penalty severity.

$$SoC_d = |SoC - \frac{(SoC_{Max} + SoC_{Min})}{2}|$$
Figure 4.7  Diagram of Simulation-Based Optimization Cycle

\[
P_{SoC} = \begin{cases} 
A_{SoC} \cdot SoC_d & SoC \geq SoC_{Max} \\
0 & SoC_{Min} \leq SoC \leq SoC_{Max} \\
B_{SoC} \cdot SoC_d & SoC \leq SoC_{Min} 
\end{cases}
\]  \hspace{1cm} (4.8)

Here, \(SoC\) represents the measured battery State of Charge, \(SoC_{Max}\) is the maximum allowable SoC, and \(SoC_{Min}\) is the minimum allowable SoC. The term \(SoC_d\) measures the deviation of the current SoC from the midpoint between the maximum and minimum SoC values. The penalties are applied based on the condition of the SoC with respect to the specified limits.

4.4.2 Simulation-based Optimization

The system is optimized by a simulation-based optimization (SBO), which iteratively evaluates the system fitness through simulation and decides the next model parameters using a metaheuristic algorithm. The SBO utilizes a fast and accurate behavioral modeling technique to simulate the system through a scenario with a set of parameters and assigns it a cost. The SBO uses a meta-heuristic algorithm that is a modified genetic algorithm to both use each generation’s cost to correlation and global discovered influence to decide the next generation’s search location. Through this iterative process, the SBO is able to decide the virtual capacitor values and the reference volt-
Figure 4.8 Flowchart of Metaheuristic Algorithm: Genetic Algorithm with Correlative Descent and Global Influence

age or the neural network learnables that dictate the reference voltage. The algorithm diagram is shown in Fig. 4.8. The algorithm starts at an initial search location with its initial set points. The set points mutate and the batch is simulated. The fittest of the batch is then selected. The cost to set-point Pearson correlation coefficient is then the influence calculated. Then the influence from other neighborhoods is used to produce global influence.

4.5 PROBLEMS WITH DETERMINISTIC ALIGNMENT

The statistical alignment of this system will not work for this system because it is chaotically deterministic with regard to the load it experiences and the output of the energy reserves, lacks state appraisal, and hunts the optimum between forecast
updates. Chaotically deterministic means that if the system is optimized for one
particular load profile and experiences another, the system’s energy reserves will react
in a way that is difficult to predict. The system also lacks state appraisal to value the
energy in reserves, making the discovered optimum to deplete energy reserves entirely.
State appraisal is a deceptively difficult task, as the value of energy is dynamic and it
is difficult to value without changing the emergent policy that the system will discover.
For example, an over-appraisal will encourage the battery to stay fully charged, and
an under-appraisal will deplete the battery, with a spectrum of behaviors in between.
Lastly, the deployment of this type of optimization requires the system to periodically
update the policy based on the next forecast or the system could face failure. This is
also tied to the fact that if the system encounters a load that it did not align itself to,
it is most likely far from the optimum policy. A demonstration of these problems is
shown in Fig. 4.9. Notice the distinction between the alignment and the encountered
loads, when the alignment and the encountered are the same, we see the problems
of state appraisal through the energy reserves being depleted. When the alignments
are mismatched, the system responds in a manner that is far from optimum and
potentially causes system failure. These are the primary problems this study faces
to offer a solution for by means of changing the load profile the system aligns to a
comprehensive load.

4.6 Comprehensive Training Load

The comprehensive training load is designed with consideration to the expected emer-
genent behavior, sustainable energy strategy, avoidance of state appraisal, training data
generation, and reduced simulation time. The comprehensive load is generated from
a subset of the potential loads and fabricated loads, which are scheduled in time
blocks. The types of loads seen and the order of the blocks are decided based on the
considerations stated above.
The design consideration of emergent behavior is difficult to foresee but may be discovered through trial and error. The discovered best practice was to start the system with the no-load block to allow the simulation to initialize and develop a steady-state strategy for no load. The next block is the full load block, this encourages the system to retain sufficient energy reserve in the battery to handle a step to the worst-case scenario. The remaining blocks are placed for sufficient training data generation.

The design consideration towards sufficient training data is based on the attempt to explore the inputs to the NN SoC regulator. The inputs to the regulator are the bus voltage Vbus and battery SoC. The system is operating in a droop configuration,
suggesting that a magnitude sweep in the load will encourage the system’s bus voltage to also conduct a magnitude sweep. The verbiage of ‘encourage’ and ‘attempts’ is intentionally chosen as the emergent optimal policy could satisfy the system cost goals, but still prohibit the system from operating within the entirety of its feature space. This lack of exploration is considered to be sufficient as long as the load seen is within the scope of the forecast projections the system trained around.

The duration of the blocks is designed with the consideration of a sustainable energy strategy. The longer the block, the more the emergent policy would be encouraged to develop a sustainable energy strategy. This is not feasible though because of computational restrictions. Therefore, the duration of the block is based on a time scale of how quickly the interfacing converter could discharge the battery. This is then multiplied by an order of magnitude and then trimmed to fit into a rounded 5-hour block. The block duration and was verified by trial and error.

Lastly, the total run time reduces the influence state appraisal would have if it were estimated. This can be justified by looking at Eq. 4.4 and adding a constant $k$ to represent the appraised value of energy stored in the final cost. The limit of the integral approaches is infinite as $T_{end}$ increases. This suggests, that for any value of $k$, the influence on the cost function is reduced as $T_{end}$ increases. This puts the total run time at odds with increasing the simulation duration to negate the influence of state appraisal, and computational restrictions. The total run-time duration was verified by trial and error.

Different blocks could be included, such as one held at the most probable load to encourage a developed strategy at the most probable scenario.

4.7 Results

Results are broken into two metrics. First how NN and no comp responded to the training when investigated over a Monte Carlo investigation to demonstrate how the
adoption of a policy requires enough tuneable parameters to be realized. Second, the expected improvement of the NN compared to a more traditional system PI controller for SoC compensation.

4.7.1 Comprehensive Training Results

The results show how the two different SoC regulators performed after aligning with the comprehensive training load profile. The results show the energy response of the aligned system with the worst case, most probable case, and a set of random potential instances. The run time is over the course of an hour.

No Compensator Aligned with Comprehensive Load

The system with a fixed voltage reference, the no SoC compensator, energy response is shown in Fig. 4.11, where several problems persist with the emergent energy management policy. The majority of the potential instances show that within the forecast horizon window, the policy overcharges the battery. The energy trajectory of the policy also shows that the system will eventually violate the SoC rule. This shows that the emergent policy is incapable of operating continuously without violation of the
SoC rule. This violation would destabilize the operation of the battery role in the system.

![Diagram of SoC compensator energy response after comprehensive training](image)

Figure 4.11 No SoC Compensator Energy Response after Comprehensive Training

**Neural Network Aligned with Comprehensive Load**

The system with a neural network SoC regulator is shown in Fig. 4.12, where the emergent policy shows a sustainable energy policy for all tested load scenarios. A post-hawk justification of the emergent behavior of the NN suggests the system has learned a close-to-optimum solution where the system retains sufficient energy to both operate sustainably and handle the potential for the load to increase in demand. For all tested loads the system did not violate any of the voltage or SoC rules. The worst-case scenario shows how the emergent policy comes close to the minimum energy reserve rule, while still maintaining a safety margin as well as sustainable operation. The trajectories of all of the tested loads suggest that the system could continue to operate continuously without an update so long as the system experiences a load within the set of the forecast, and potentially some outside of it.
4.7.2 Expected Improvement

The comprehensively trained NN for the SoC system is compared to a more traditional PI SoC regulator by simulating both systems through a year of servicing identical loads that are randomly pulled from the pool of potential instances that change every hour of simulation. This test is conducted in this manner to see an average expected improvement across the different loads, and to negate the effect of remaining stored energy as much as possible to avoid state-appraisal. A PI compensation loop was used to regulate the SoC after the virtual capacitor was chosen by the metaheuristic algorithm.

4.8 Conclusion

This study has explored the intricacies of shared power and frequency delegation among grid-tied converters that utilize virtual impedance to achieve near-optimal power flow. It provides strong evidence for the adoption of a virtual capacitor in conjunction with an energy storage device to delegate specific frequency responses,
Table 1: Long-Term Performance of Regulators

<table>
<thead>
<tr>
<th>SoC Reg.</th>
<th>Annual Fuel Consumption (Units)</th>
<th>Normalized Fuel Consumption (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprehensively Trained NN SoC Regulator</td>
<td>3.56E10 (Units)</td>
<td>98.09% (-1.91%)</td>
</tr>
<tr>
<td>Optimized PI SoC Regulator</td>
<td>3.49E10 (Units)</td>
<td>100%</td>
</tr>
</tbody>
</table>

thereby ensuring the quality of bus voltage. The study also highlights the importance of simulation-based optimization for system alignment, emphasizing the need for careful consideration of tunable parameters, control architecture, and load configuration.

The problems inherent in a stochastic input and chaotically deterministic system are expanded upon and demonstrated through alignment mismatch to show how the system could lead to failure without proper consideration. In a chaotically deterministic system that uses an SBO to optimize itself for load forecast projections, it is imperative to provide ample training data and a sufficient number of optimizable parameters to facilitate the emergence of effective policies. The comprehensive load is used to generate sufficient information for the optimization of the system to answer the problems inherent in this type of system.

While the comprehensive load proves valuable for optimizing the system and addressing inherent problems, it is clear that this load alone is insufficient. The system aligning with the load must possess an adequate number of tunable parameters to dynamically implement an effective energy policy. This deficiency is evident when comparing it to a simplified system with fixed references, which struggles to maintain a reliable energy strategy across various scenarios. In contrast, the neural network SoC regulator system stands out for successfully meeting all design criteria. Notably, it is expected to outperform the commonly employed PI control by up to 1.91% in
terms of fuel efficiency.
Simulation-Based Optimization (SBO) is emerging as a potent tool for optimizing, designing, and evaluating power and energy systems. In this approach, a system is modeled through a scenario, utilizing a metaheuristic algorithm that iteratively selects inputs based on fitness. It’s poised to be a defining method for aligning systems in the years to come.

However, SBO faces several significant challenges. First and foremost are the computational restrictions. The computational intensity of the model being simulated, the duration of the simulation, and the number of iterations the metaheuristic algorithm must undergo to converge on a high-confidence solution are all intertwined. These factors can place considerable constraints on the effectiveness of SBO.

Another critical factor to consider is the confidence in the solutions generated through SBO. This confidence is at the crossroads of three key elements: the trustworthiness of the model used to align the system, the efficiency of the metaheuristic algorithm’s search within the parameter space, and the quality of the training scenarios that guide the alignment process. Achieving a high level of confidence in all these areas is crucial for the success of SBO.

Lastly, the selection of training data for SBO remains a challenging aspect. The field of machine learning is still grappling with numerous unanswered questions. Many techniques have been applied in SBO, but their justifications often rely on post-hoc analysis and trial-and-error methods. While such an approach is popular in machine learning, it represents a substantial area for potential improvement.
In conclusion, Simulation-Based Optimization holds immense promise for the optimization, design, and evaluation of power and energy systems. Overcoming the challenges related to computational constraints, solution confidence, and training data selection will be instrumental in realizing its full potential in the years ahead.

5.1 Improvements in Modeling for Enhanced SBO Performance

In the realm of Simulation-Based Optimization (SBO), the modeling of the systems under consideration is a cornerstone of the entire process. It plays a pivotal role in shaping the efficiency, accuracy, and overall effectiveness of the optimization and alignment procedures. As we look forward to the future of SBO, it becomes evident that improvements in modeling are not merely desirable but rather essential for realizing the full potential of this approach.

One of the most notable aspects of modeling in SBO is its influence on the accuracy of the predicted outcomes. These improvements in accuracy translate directly into more informed and reliable proposed optimal parameter sets chosen by the SBO. A precise model provides an accurate representation of the physical system, enabling the Simulation-Based Optimizer to explore a more relevant parameter space during its iterations. This leads to the identification of more optimal solutions, as the optimizer can distinguish subtle variations and complexities in the system’s response to different inputs. By achieving a higher degree of modeling accuracy, SBO can effectively pinpoint solutions that are closer to the true optimal settings, reducing the potential for costly trial-and-error in practical applications.

For the feasible deployment of this type of optimization to a power and energy system that detects events, and aligns itself to the projected forecast, there are time constraints to discovering and deploying the optimal solution. Faster model simulations offer several advantages in the context of SBO. First and foremost, they expedite the convergence process of the metaheuristic algorithm, allowing it to ex-
explore a broader range of potential solutions within the same time frame, or expedite its exploration to deliver a solution faster. This accelerated convergence can be particularly valuable in situations where time constraints are crucial, such as in dynamic or real-time decision-making processes. Furthermore, with the same computational resources, the SBO will be elevated in temporal performances when the model is made more efficient. This increased exploration capacity can uncover hidden, non-intuitive, or complex relationships within the system, ultimately leading to the identification of more refined and superior optimal solutions.

A compelling avenue for improving both accuracy and speed in modeling is the integration of machine learning techniques, such as the one proposed in this document. Machine learning, with its ability to adapt and refine models based on data, offers a promising compromise between accuracy and computational efficiency. By continuously learning from the system’s behavior, machine learning models can evolve to closely match the real-world system, thereby improving the quality of optimization outcomes.

The significance of modeling in Simulation-Based Optimization cannot be overstated. As we move forward, enhancements in modeling accuracy and simulation computational efficiency hold the key to unlocking the full potential of SBO. Through more accurate models and faster simulations, we can achieve higher-confidence solutions, reduced convergence times, and ultimately reliably use a SBO in optimizing and aligning complex power and energy systems within a time constraint.

5.2 Performance of Metaheuristic Algorithm

Within the framework of Simulation-Based Optimization (SBO), the choice and performance of the metaheuristic algorithm play a pivotal role in the confidence and efficiency of the optimization process. Metaheuristic algorithms are the intelligence of SBO, guiding the exploration of the parameter space and searching for optimal
solutions. As such, their performance directly impacts the effectiveness of SBO. In this section, we delve into the key ways in which the performance of metaheuristic algorithms influences SBO outcomes.

One of the primary indicators of a metaheuristic algorithm’s performance is its convergence speed. A well-performing algorithm can reach a near-optimal solution with fewer iterations. This directly translates into reduced computational resource requirements. When the algorithm converges rapidly, it minimizes the time and computational power needed to find a suitable solution. This aspect is particularly critical in time-sensitive applications, where quick decision-making is imperative. Efficient metaheuristic algorithms reduce the overall computational burden, making SBO more practical and cost-effective. By enabling quicker convergence, they enable the optimization process to be deployed in real-time or within tight deadlines, providing optimal solutions that can be used in many ways, but are particularly appealing to fast system alignment of power and energy systems.

Metaheuristic algorithms with superior performance exhibit the ability to explore a broader range of the parameter space. They can navigate through complex, multi-dimensional landscapes more effectively, allowing for a more comprehensive examination of potential solutions. This enhanced exploration capacity increases the likelihood of identifying optimal solutions, even in highly complex and nonlinear systems. Robust algorithms can avoid getting stuck in local optima. They are adept at escaping suboptimal regions of the solution space and progressing towards more globally optimal solutions. This property is essential when dealing with complex, multi-modal optimization problems, as it ensures that the optimizer does not prematurely settle for suboptimal outcomes. Robust and adaptable algorithms are vital in power and energy system applications where system dynamics and problem structures may vary and have highly complex relationships between operational cost and parameter sets. They ensure that SBO remains a reliable tool in the face of uncertainty and changing
conditions, making it adaptable to the evolving demands of power and energy systems optimization.

As SBO continues to evolve, the development and integration of advanced metaheuristic algorithms are of paramount importance. These algorithms can incorporate sophisticated techniques for better exploration and exploitation of the solution space. By harnessing the power of artificial intelligence, advanced metaheuristics can continuously learn and improve their performance over time, adapting to the complexities of modern power and energy systems. Faster convergence, enhanced exploration, robustness, and adaptability are key attributes of well-performing algorithms that directly impact the success of SBO. The ongoing development and integration of advanced metaheuristic algorithms are essential for future-proofing SBO and ensuring its effectiveness in addressing the ever-evolving challenges of power and energy systems optimization.

5.3 TRAINING DATA PRODUCED BY COMPREHENSIVE SCENARIO

The training data for the scenario is arguably the most critical component of this optimization scheme. This concept draws its inspiration from reinforcement learning, where it is imperative to provide agents with sufficient information for them to adopt emergent policies and act accordingly. We have adapted this concept and integrated it into the Simulation-Based Optimization (SBO) framework, effectively training system parameters using metaheuristic techniques to emulate the behavior of neural network-based agents. While the importance of this aspect cannot be overstated, it also stands on the cutting edge of current literature, with ongoing debates and emerging possibilities. Simply put, the comprehensive scenario aims to take forecasted load projections and transform them into a unified scenario that furnishes the system with the requisite information to explore its feature space, develop optimal policies, and promote sustainable practices. The comprehensive scenario serves as the linchpin of
the SBO process, providing a critical foundation for the optimizer to make informed
decisions. It bridges the gap between data and actionable insights, allowing the
system to effectively navigate the complex terrain of power and energy optimization.

Training data plays a pivotal role in enabling comprehensive feature space explo-
ration within the realm reinforced-learning and how this SBO is being used. It serves
as an informational fundamental which the optimization process is built around. By
supplying the system with diverse and representative data using the comprehensive
scenario, training data empowers the optimizer to traverse the vast landscape of fea-
ture space. It equips the system with the necessary insights to discern patterns,
interdependencies, and nuances within the data, allowing it to uncover hidden re-
lationships and exploit opportunities for optimization. In essence, training data is
the compass that guides the exploration of feature space, facilitating the discovery of
optimal solutions and the development of effective policies that can enhance the effi-
ciency and sustainability of power and energy systems. Without high-quality training
data, the system’s ability to explore and exploit the feature space would be severely
limited, making it a requirement of successful SBO alignment of power and energy
systems.

Training data serves as a catalyst for the emergence of policies within the context
of reinforced learning and this SBO. By supplying the system with a dataset with
specific events and temporal situations, the system learns vital policies from these
interactions. This observational learning process allows the emergent policies to sur-
face, as the system identifies patterns and behaviors that lead to favorable outcomes.
The more comprehensive and representative the training data, the greater the poten-
tial for the emergence of innovative and effective policies, so long as the system has
sufficient methods to dynamically adopt them.

This aspect of this study leaves the most unanswered questions with this aspect of
the SBO. It is clear that this technique can provide the system valuable information
to learn off of by carefully selecting training data. It is also clear that as the system changes and the number of features the tunable parameters are interfaced with, the scenario will also have to change. The field of reinforced-learning is a good field to mimic with this deployment, and there is a strong notion to train the system with as much data as possible. This concept does not seem sustainable in either reinforced-learning or in the SBO deployment as the amount of data being used in training does not seem to beat out a choice selection of quality data. This study shows that this concept of the comprehensive scenario worked well in the particular case, but it whether it is viable and translatable to other types of training is still undecided.
BIBLIOGRAPHY


[90] Hisham Mahmood, Dennis Michaelson, and Jin Jiang. “Decentralized power management of a PV/battery hybrid unit in a droop-controlled islanded mi-


[101] Carlos Moreira et al. “A stochastic model to simulate electric vehicles motion and quantify the energy required from the grid”. In: (2011).


[141] Pedro P Vergara, Ricardo Torquato, and Luiz CP Da Silva. “Towards a real-time Energy Management System for a Microgrid using a multi-objective ge-


