8-2023

Achieving High Renewable Energy Integration in Smart Grids with Machine Learning

Yaze Li
University of Arkansas, Fayetteville

Achieving High Renewable Energy Integration in Smart Grids with Machine Learning

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering with a concentration in Electrical Engineering

by

Yaze Li
Tsinghua University
Bachelor of Science in Electronic Engineering, 2017

August 2023
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

ABSTRACT

The integration of high levels of renewable energy into smart grids is crucial for achieving a sustainable and efficient energy infrastructure. However, this integration presents significant technical and operational challenges due to the intermittent nature and inherent uncertainty of renewable energy sources (RES). Therefore, the energy storage system (ESS) has always been bound to renewable energy, and its charge and discharge control has become an important part of the integration. The addition of RES and ESS comes with their complex control, communication, and monitor capabilities, which also makes the grid more vulnerable to attacks, brings new challenges to the cybersecurity. A large number of works have been devoted to the optimization integration of the RES and ESS system to the traditional grid, along with combining the ESS scheduling control with the traditional Optimal Power Flow (OPF) control. Cybersecurity problem focusing on the RES integrated grid has also gradually aroused researchers' interest.

In recent years, machine learning techniques have emerged in different research field including optimizing renewable energy integration in smart grids. Reinforcement learning (RL), which trains agent to interact with the environment by making sequential decisions to maximize the expected future reward, is used as an optimization tool. This dissertation explores the application of RL algorithms and models to achieve high renewable energy integration in smart grids.

The research questions focus on the effectiveness, benefits of renewable energy integration to individual consumers and electricity utilities, applying machine learning techniques in optimizing the behaviors of the ESS and the generators and other components in the grid. The objectives of this research are to investigate the current algorithms of renewable energy integration in smart grids, explore RL algorithms, develop novel RL-based models and algorithms for optimization control and cybersecurity, evaluate their performance through simulations on real-world data set, and provide practical recommendations for implementation.

The research approach includes a comprehensive literature review to understand the challenges and opportunities associated with renewable energy integration. Various optimization algorithms, such as linear programming (LP), dynamic programming (DP) and various RL algorithms, such as Deep Q-Learning (DQN) and Deep Deterministic Policy Gradient (DDPG), are applied to solve problems during renewable energy integration in smart grids.

Simulation studies on real-world data, including different types of loads, solar and wind energy profiles, are used to evaluate the performance and effectiveness of the proposed machine learning techniques. The results provide insights into the capabilities and limitations of machine learning in solving the optimization problems in the power system. Compared with traditional optimization tools, the RL approach has the advantage of real-time implementation, with the cost being the training time and unguaranteed model performance. Recommendations and guidelines for practical implementation of RL algorithms on power systems are provided in the appendix.

ACKNOWLEDGEMENTS

I would like to express my gratefulness to my advisor, Dr. Jingxian Wu, for providing me the opportunity to finish this Ph.D. program. In many ways, I was not the excellent student he expected, but he always inspired and supported me to be a professional researcher and a mature person. I would like to also thank Dr. Yanjun Pan, for supporting me with my research during our weekly meetings.

I would like to thank Dr. Jeff Dix, Dr. Qinghua Li, and Dr. Roy McCann for serving in my dissertation committee. I appreciate their time and kindness during the busy schedules in summer.

I feel really honored to work with nice and talented people like Dr. Zuoen Wang, Dr. Israel Akingeneye, Dr. Samrat Nath, and Dr. Tanny Chavez Esparza. Without their help, suggestion, and previous hard work, this dissertation could not have been accomplished. I am grateful to Md Abul Hayat and Asif Emon who have accompanied me and helped me during my first year of Ph.D.

I would like to thank my Chinese friends Hui Wang, Nan Lin, Hao Chen, and Zhe Zhao, with whom I celebrated Chinese Festivals here. I would also thank Yujie Sude and Zachary Tipton for treating me as their little brother. I am grateful to Andong Zhou for her company and support during my depression and stressful time. I am fortunate to have Junning Fu as my soulmate and support me.

I would like to express my deepest appreciation to my mother: Xiuping Li, and my father: Yuejun Li for their love, care, support, and sacrifices in educating me and providing me a better life. I would like to also thank my wife, Yanchu Zhang, for her understanding, trust, tolerance and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

1.1 Background and Motivation

The increasing global concerns over climate change and the depletion of fossil fuel resources have propelled the rapid development and adoption of renewable energy sources. Electricity generated from renewable energy sources (RES) surpassed coal in the US for the first time in 2022 [3]. Solar and wind power, in particular, have witnessed significant growth in recent years.

1.1.1 ESS-assisted PV system

Solar power is a clean, inexpensive, and renewable energy source that is widely adopted around the world. One of the most efficient ways to harness solar power is through photo-voltaic (PV) cells, or solar panels, which convert light directly into electricity using photoelectric effects. The state of California in the United States mandates solar panels on the roofs of all new homes starting in 2020 [4]. The growing popularity and constantly increasing demands of solar energy necessitate optimum designs of PV system that can be seamlessly integrated in power grids and achieve maximum savings in energy and overall system cost.

However, the variability, uncertainty, and non-synchronous generation of PV power sources impose numerous challenges on the large-scale and cost-effective deployment of PV systems [5]. The intermittent and stochastic nature of solar energy creates an imbalance between energy supplies and demands, which has to be compensated by integrating PV system with energy storage system (ESS).

In addition to environmental considerations and the balance of supply and demand, economic benefits are also a major reason for individual electricity consumers to purchase solar panels. By storing excessive solar energy during off-peak hours and discharging the stored energy during high demand hours, ESS-assisted PV system can save the electricity bill significantly. However, many factors such as system costs, battery and solar panel aging, electricity cost inflation and long-term returns need to be considered when design such system for a long-term use. The user needs to know in advance whether the installation of this system can obtain long-term economic returns for his electricity consumption, what are the optimal system parameters, and what is the specific economic return. Even if an optimum ESS-assisted PV system is designed and installed, in actual use, users still need to know what is the optimum real-time battery charging and discharging strategy under the system parameters and real-time power consumption, or the ESS provides such scheduling algorithm inside, so as to achieve the expected economic return before purchasing and installing this system.

1.1.2 Grid-Connected ESS-assisted RES

Like solar energy, wind energy is affected by wind speed and has a time-varying and stochastic nature. The integration of high levels of renewable energy into existing power grids poses substantial technical and operational challenges [6]. The intermittent nature of renewable energy sources, coupled with their inherent uncertainty, demands innovative solutions to ensure grid stability, reliability, and optimal utilization of these clean energy resources. Similar to the local ESS-PV system, the ESS-RES in the grid again can be used to store excess energy during periods of high production and release energy during periods of high demand, which can stablize the grid and improve the reliability of the RES [7].

Traditional economic dispatch (ED) and optimal power flow (OPF) controllers are designed to minimize the grid operation cost by optimizing power generation schedules at each generator over a single time period [8]. However, the integration of ESS asks the grid controller to consider the storage behavior over multiple time periods [9]. In addition, it is shown that for power system with ESS, multi-period OPF provides a more economical control solution [10].

Another important requirement of the grid control algorithms is to make real-time decisions on the generation and the ESS schedule given the real-time load profile and uncertain RES profile. Therefore, real-time control that extends the traditional OPF to multi-period OPF and schedule the ESS is needed.

There is a growing interest in incorporating mobility into ESS to achieve mobile energy storage, which offers numerous advantages compared to stationary energy storage systems (SESS). These advantages include enhanced flexibility, rapid deployment, redundancy, reduced infrastructure costs, and environmental benefits. Mobile energy storage systems can be easily transported to different locations as needed, providing greater flexibility compared to their static counterparts [11]. They can also be swiftly deployed in response to emergencies or to support temporary events. In situations where static energy storage systems have failed or are unavailable, mobile energy storage ensures a continuous supply of power, serving as backup energy [12]. Moreover, mobile energy storage has the potential to reduce the necessity for costly grid infrastructure upgrades and can be utilized to power electric vehicles, contributing to the reduction of greenhouse gas emissions and air pollution [13].

The control and scheduling of a distribution grid with mobile energy storage are complex and challenging due to the dynamic and uncertain nature of both the transportation networks and the RES, as well as the need to minimize operational costs while ensuring efficient and reliable energy distribution over the entire grid. This requires a comprehensive system model that combines the scheduling of both transportation network and power flow. Again, such control algorithm needs to perform in real-time.

1.1.3 Cybersecurity of Grid with RES

Smart grids, equipped with advanced sensing, communication, and control technologies, have emerged as a promising framework for addressing these challenges [14]. The supervisory control and data acquisition (SCADA) system, an integral part of a smart grid, plays a crucial role in monitoring and controlling power grid operations through remote terminal units (RTUs). However, the vulnerability of SCADA systems to cyberattacks is a significant concern. A notable example is the cyberattack that occurred on December 23, 2015, targeting the SCADA system in the power grid of Kiev, Ukraine, resulting in a widespread blackout [15].

One prevalent form of attack is the false data injection (FDI) attack, which targets different layers and systems within the smart grid [16]. The primary objective of an FDI attack is to manipulate SCADA measurements, thereby undermining the accuracy of state estimation (SE) and leading to unreliable power system operations [17]. Additionally, denial-of-service (DoS) attacks pose another significant risk. A DoS attack disrupts SCADA system operations by obstructing communication links between devices or rendering certain measurement devices inaccessible [18].

Achieving high levels of renewable energy integration in smart grids necessitates interoperable distributed energy resource (DER) grid-support functions with advanced control and communication capabilities. However, the introduction of these complex capabilities also increases the vulnerability of renewable energy systems (RES) and exposes them to potential cyberattacks [19]. Such cyberattacks have the potential to disrupt normal grid operations and induce system instabilities, including line overloads, frequency and voltage violations, reverse power flow, and voltage collapse, particularly during periods of heavy load [20, 21]. Consequently, there is a pressing need to develop cybersecurity technologies capable of detecting and mitigating the adverse impacts of these cyber threats.

While numerous studies have focused on the cybersecurity of energy systems, their attention primarily centers around grid operations utilizing measurements obtained from supervisory control and data acquisition (SCADA) systems, remote terminal units (RTUs), and the underlying communication network of the grid [2, 22]. While these measurements serve as crucial indicators for grid operations, they prove inadequate as attacks can also target local measurements derived from RES sensors and actuators or the local control policies governing RES operations. Therefore, a comprehensive approach is required to address the cybersecurity challenges posed by potential attacks on both grid-level and local RES components.

Moreover, machine learning techniques, capable of analyzing vast amounts of data and making intelligent decisions, offer great potential in optimizing renewable energy integration in smart grids.

1.2 Research Objectives

This dissertation aims to explore the application of machine learning algorithms and models for achieving high renewable energy integration in smart grids. In specific, to develop machine learning approaches to model the RES and ESS behaviors in power systems at individual consumer's end and, most importantly the controller of the power system at the utility's end, to provide optimum design of integration of RES and make real-time decisions to control the behavior of the components in the system, detect the cyberattacks in the system, in order to save the operation cost and increase the reliability of the RES.

1.3 Dissertation outline

This dissertation is organized as follows: Chapter 2 gives an literature review on topics that are discussed in this dissertation. Chapter 3 and 4 focus on the optimal design of ESS-assisted PV system and the optimal control . A Dynamic Programming solution for ESS-assisted PV system design has been proposed in Chapter 3. The DP algorithm starts as an suboptimal approach for long-horizon optimizations, and then the idea of Bellman equations and action-value functions are moved to a Markov Decision Process (MDP) formulation of the real-time ESS-assisted PV scheduling problem in Chapter 4.

Chapter 5 and 6 focus on the optimal control of the generators and ESS in the grid integrated with renewable energy. We develop a model to extend traditional OPF to multi-period with wind energy to minimize the operation cost of the new system in Chapter 5. The model has combines the OPF with ESS schedule by reinforcement learning techniques that follows the MDP framework. The agent trained by the environment, thus is able to control the generations of the grid and the ESS schedule in real-time without any prediction model. The model is extended in Chapter 6 by replacing the stationary ESS into mobile energy carriers. The transition network and traffic model are combined with the original model, the agent trained by this new environment is able to control the MEC scheduling and assignment, along with the generations of the grid in real-time.

Finally, to face the cyberattacks in the smart grid with RES, a dynamic watermark algorithm is used to detect attacks on measurements of local PV farms in Chapter 7, and a reinforcement learning based detection algorithm is proposed to detect attacks on measurements from the SCADA in Chapter 8.

Chapter 9 provides a concluding remark and major contributions of this dissertation and propose possible future work for continued research.

2 Literature Review

2.1 Research on optimum ESS-assisted PV system design and scheduling

There have been growing interests in the optimum designs and scheduling of energy systems with energy storage devices (ESS). Most designs aim at minimizing the energy cost or operating cost through optimum scheduling of energy generation and/or storage. In [23], the optimum scheduling of a pump-storage hydro power station is formulated as a mixed integer linear programming (MILP) problem, where the non-linearity of power generation in hydro-turbines is approximated by a piecewise linear function. In [24], the optimum scheduling of distributed energy resources (DER) is formulated as a linear programming (LP) problem to reduce operation cost and to shave peak demands. In [25], the optimum scheduling of behind-the-meter ESS is formulated as a mixed integer non-linear programming (MINLP) problem, and the problem is equivalently transformed into a LP by introducing auxiliary variables. The complexity of LP-based approaches scales in general in polynomial time with respect to the number of decision variables and constraints. The complexity could be prohibitively high when the number of decision variables is large.

Optimum ESS scheduling can also be solved by using dynamic programming (DP), which relies on Bellman's principle of optimality to decompose the original problem into a sequence of simpler subproblems in a recursive manner [26], [27], [28], [29], [30], [31]. In the optimum designs of ESS systems, the state variables are usually states of charge of the ESS, which often need to be discretized for efficient solution [26]. For utility bill minimization, the demand charge introduces a supremum term in the objective function, which violates Bellman's principal of optimality necessary for DP. This problem is solved by using the concept of forward separable function with augmented state variables in [31], or multi-objective DP in [29]. The integration of ESS system with renewable energy sources adds new degrees-of-freedom that can further improve the efficiency of energy usage. The optimum design of power systems with both ESS and renewable energy sources are discussed in [28, 32], which consider the interactions of a variety of renewable energy sources and ESS devices, including PV panels, wind turbines, hydroelectric plants, pumping stations, etc.

The operation cost and wear-out cost of the system are included in the formulations in [24, 25, 28]. However, the initial procurement and installation costs are not considered in those works.

In addition to system cost, the optimum design should also consider aging effects of the devices, where the efficiencies and/or capacities of both solar panels and batteries degrade gradually over time [33]. To accurately model the aging effects, the optimization needs to be performed over a time horizon over the entire life cycles of batteries and solar panels. However, the time horizon in many existing works are one day [34], and multi-day costs are obtained by multiplying the daily cost by the number of days [32, 35]. A 24-month dataset is used in [36], but with the assumption that the battery is fully charged at the beginning of each day. Thus the optimization horizon is still within one day. In [23] and [25], the time horizon is extended to one month. A one-year optimization horizon is considered in [37] without considering the aging effects. In [38], the design is performed over a 3-year optimization period, which is still shorter than the device life cycles, and it only considers the battery cycling aging effects.

2.2 Research on real-time ESS-assisted PV system scheduling

Online energy management for systems with renewable energy sources depends critically on load and renewable energy generation forecasting. The auto-regressive moving average (ARMA) model is a powerful tool for short-term load forecasting [39]. In [40], the load and PV generation forecast is performed by using a recurrent neural network (RNN) with long short term memory (LSTM). The deep learning based LSTM method is used in combination with auto-regressive integrated moving average (ARIMA) model for load forecasting in [41], where ARIMA model captures the stationary pattern of load segments, and LSTM extracts the non-linear relations of load segments.

Load forecasting and energy management can be combined into an integral process under the framework of reinforcement learning (RL) [42]. RL-based methods learn the operation environment of the energy system and model it as a Markov decision process (MDP) without explicitly identifying environment parameters such as transition probabilities. Energy management can then be optimized by applying deep neural network (DNN) to learn the inherent patterns of renewable energy generations and grid operations [43]. Value-based RL algorithms such as Q-learning [44] and policy-based actor-critic approach [45] have been successfully used to solve battery scheduling problems. A deep Q-learning (DQN) method is proposed in [46] to optimize battery scheduling for a microgrid with PV sources. DQN-based solutions require discretizations of the originally continuous action and/or state spaces. Discretization causes inevitable performance losses, and the complexity grows exponentially with the increase of the discretization levels.

Deep policy gradient (DPG) based RL methods can be applied to operation environment with continuous action/state spaces [47] and [48]. Deep deterministic policy gradient (DDPG) is an off-policy version of DPG [49]. DDPG has been applied to various resource allocation problems in a wide range of applications, such as wireless communications [50] and [51], energy management for electrical vehicles [52], battery scheduling in smart homes [53], and scheduling in battery swapping stations [54], etc.

2.3 Research on OPF for system with RES and ESS

In order to cope with the increased integration of DER in power systems, a large number of OPF algorithms have been developed to achieve real-time responses, mainly through the tradeoff between complexity and performance. In [55], wind power forecast is performed by considering multiple possible scenarios, and individual OPFs are solved in paralle for each possible scenario. The results are provided in the form of a lookup table as a reference for grid operations. In [56], the single-period OPF is transformed into a multi-period OPF algorithm by separating a longer time horizon into sub-intervals. An online gradient projection algorithm is developed to solve real-time OPF (RT-OPF) for a power grid without RES or ESS [57]. An iterated single-period OPF approach is developed to solve RT-OTF by using a quasi-Newton method [58]. Linear relaxation of the AC OPF is often used in these works, and the optimization can be solved by algorithms such as alternating direction method of multipliers (ADMM) [59] or stochastic dual dynamic programming (SDDP) [60].

In addition to the efforts made to bring real-time response to single-period OPF, another notable solution for multi-period OPF can be achieved through model predictive control (MPC), which adopts predictive models to represent the dynamic behaviors of RES [61]. In [62], the multi-period OPF for a power grid with RES and ESS is formulated as a mixed-integer linear programming (MILP) by using a novel sparse formulation of the affinely adjustable robust counterpart (AARC). Heuristic tools such as particle swarm optimization (PSO) and genetic algorithm (GA) are often used when ESS scheduling is optimized along with power flow [63–65].

Recently a large number of works have resorted to machine learning (ML) algorithms to solve RT-OPF due to its fast response and lower complexity. Fully connected neuron (FCN) network is trained as energy manage system (EMS) for RT-OPF in [66]. Deep reinforcement learning (DRL) algorithms such as Deep Q-Networks (DQN) [67] and Double DQN (DDQN) [68] have been employed to optimize power dispatching problems in power grids. A Lagrangian based deep deterministic policy gradient (DDPG) agent is trained to solve RT-OPF for a power grid without RES [69].

2.4 Research on OPF for system with RES and MEC

Various design and scheduling techniques have been proposed to optimize power grids with mobile energy storage systems. In [70], a mixed integer linear programming (MILP) algorithm is proposed to solve the mobile energy storage scheduling problem, which is formulated to minimize overall system cost by considering energy demand, energy production, and transportation constraints. The results in [70] are extended into a stochastic programming framework that incorporates uncertainties in renewable energy production and various loads in the grid [71]. In [72], the system is designed by minimizing the total cost associated to the energy loss in the power and transportation networks with MILP. A suboptimal two-stage solution is given in [73], where the power flow is solved on an instantaneous transit system by solving a mixed-integer second-order cone programming (SOCP), and transportation problem of mobile energy storage is solved by using the particle swarm optimization (PSO) algorithm. In addition to the financial benefits, the employment of MEC can improve the flexibility and resilience of power systems. For example, the worst-case weighted sum of survived loads is used as the design objective to improve the power grid resilience in disaster recovery [74], and the total load supported by mobile energy storage is maximized to improve system flexibility in [75].

Recently, reinforcement learning (RL) has been applied to optimize the decision-making process in both energy and transportation systems [47] and [76]. RL is a promising machine learning technique that involves learning through trial and error by maximizing a reward function, and it has shown great potential in solving complex decision-making problems. Moreover, deep reinforcement learning (DRL) algorithms have been proposed to make sequential decisions in complex systems. DRL is a sub-field of RL that leverages deep neural networks (DNN) to approximate the optimum policy. One of the most widely used DRL algorithms is the deep deterministic policy gradient (DDPG) algorithm, which has shown remarkable performance in solving complex continuous control problems [49]. DDPG is a model-free, actor-critic RL algorithm that uses two DNNs, an actor network and a critic network, to learn an optimal policy. The optimal electrical vehicle (EV) charging in urban transportation networks considering uncertainties in wind power output and traffic demands is solved by DDPG in [77].

2.5 Research on Cybersecurity for PV farms

The rapid advancement of machine learning (ML) during the past decade has driven the development of ML-based cyberattack detection methods. Most of the ML-based methods are data-driven, and they do not require physical models of the system. Various data-driven ML algorithms, such as one-class support vector machines (OCSVMs), random forests (RFs), and principal component analysis (PCA) were applied to multiple sources of time-series data for distributed anomaly detection on a single solar panel [78]. Deep neural networks (DNN) with long short-term memory (LSTM) were applied to detect data integrity attacks by using the Northeast Solar Energy Research Center (NSERC) solar farm dataset [79]. Raw data collected from micro PMU (μPMU) were used for the detection of cyberattacks on photovoltaic (PV) farms by using data-driven methods such as decision tree (DT) and K-nearest neighbor (KNN) [80]. Most ML approaches require a large amount of data during the offline training stage, and sometimes it might be difficult to obtain a sufficient amount of training data from cyber-physical systems (CPS).

In contrast to pure data-driven methods, model-based methods utilize the underlying physical models of CPS to monitor system operations. The knowledge of the physical model can help improve detection accuracy and reduce the amount of training data. For example, measurement results can be compared to state estimations in a smart grid, and the residues can then be used for anomaly detection [81, 82]. The detection can be performed by using either a single measurement or a sequence of historical measurements, such as the windowed χ^2 detector [83]. All these detection methods can be classified as passive methods. One of the limitations of the passive methods is that they might not be able to detect cyberattacks designed by using full knowledge of the system model, as the adversary can use the knowledge of the physical model to match the attacked data with state estimation results.

Dynamic watermarking is an active defense method that adds a small random signal, i.e., “watermark", to the input of the controller [84]. The power of the random signal is small such that it does not disturb normal system operations, and the detector can utilize the statistical distributions of the watermark signal to test the operation conditions of the CPS. Dynamic watermarking was first proposed to improve the performance of χ^2 detector [84,85]. However, it is unable to detect attacks with post-attack distributions fitting historical measurements, such as the replay attack. This problem can be solved by using two actuator tests with respect to the covariance of the residuals and the correlation between the residuals and watermarks [86]. The two-test dynamic watermarking scheme is used as the active defense method for the automatic generation control (AGC) of a power system, and to detect attacks applied to voltage and current measurements of a grid-connected PV system [87]. The two-test dynamic watermarking algorithm is later extended to general linear time invariant (LTI) systems with a single statistical test based on the Wishart distribution [88], and to linear time varying systems and nonlinear systems in [89].

2.6 Research on Cybersecurity for Smart Grids

A plethora of algorithms have been developed for cyberattack detection by using power system state estimations, where the measurement results can be compared to estimation results to identify the presence of anomalies. Many algorithms are developed by using static state estimation (SSE) with a simplified DC system model due to its low computational complexity [90–92]. However, SSE cannot capture the dynamic state transitions in power systems, and it is in general not suitable for real-time monitoring of power system operations. Dynamic state estimation (DSE) with Kalman filter (KF) and its derived algorithms are widely used for power system estimation and intrusion detection [93]. A KF-based DSE algorithm is used to detect FDI in automatic generation control (AGC) system in [94]. Distributed Kalman filter (DKF) was used in [95] to reduce the computation complexity for attack detection, and extended Kalman filter (EKF) was used in [96, 97] to model the nonlinear measurement function of AC power system model for FDI attack detection. A robust Cubature Kalman Filter (RCKF) based approach is proposed for systems with power generators under cyberattacks [98]. In [99], a correlation-based DSE is proposed to detect DoS attacks.

With the recent rapid advances in artificial intelligence (AI) and machine learning (ML), there have been growing interests in applying ML algorithms for intrusion or cyberattack detection in smart grids. In [100], a real time FDI detection algorithm is developed by performing robust principal component analysis (PCA). Various deep learning algorithms, such as deep belief network (DBN) [101, 102], recurrent neural network (RNN) [103], and deep neural network (DNN) [104], are developed for FDI detection. Reservoir computing (RC) is an extension framework of NN. It consists of three parts: a feed-forward NN as the input layer, an RNN as the middle layer, and a weighted adder as the output layer [105]. A delayed feedback RC is used for detecting dynamic attacks in smart grids [106, 107]. These algorithms utilize a data-driven approach, that is, the input to the neural networks are measurements from the power system, and the output are detection results. Preprocessing of the measurements can be done to make the training more efficient, such as the wavelet transform in [108]. A state–action–reward–state–action (SARSA) algorithm based on reinforcement learning (RL) is developed for FDI detection in [109]. A Q-Learning method with nearest sequence memory is adopted to detect FDI attack to automatic voltage control (AVC), where the values of the Q-function are discretized and stored in a lookup table [110]. In [111], a deep Q-network (DQN) is designed to defend FDI in power grids, and it utilizes a DNN with discrete states to approximate the Q-function required for Q-learning.

2.7 Research Gaps

Although researchers adopt many techniques for high renewable energy integration in smart grids, there are some fundamental gaps in the existing literature. The gaps are described below:

- For the ESS-assisted PV system design, the optimizations in most existing works are performed by assuming a fixed-sized ESS. In the design of battery-assisted PV systems, the optimum capacity of BESS and the number of solar panels are important decision parameters, and their optimum values are in general not readily available before hand. The costs of solar panels and BESS accounts for a large amount of initial investment. The optimum designs need to ensure that the cost saving due to ESS and renewable energy sources can outweigh the system cost in the long run, thus the system cost should be a critical parameter in the system design. The optimization horizon in most existing works are too short.

- For the OPF control problem, the popular MPC approach requires muti-period prediction and multi-period optimization in real-time, which leads to a high computing complexity and long responding time.

- For the OPF control with MEC, the transportation models used for dispatching the MEC are very complex. One approach to dispatching problem is the transit delay model (TDM), which records the delay and location of each MESS at each sample time [73]. The transit delay depends on both the distance and the traffic congestion delay. The latter depends on not only the location but also the time because of the traffic flow. Most research assumes a static traffic demand based on historical data and stores the transit delay in a location-location-time tensor. The other is the Time-space Network (TSN), which requires less number of decision variables and constraints. However, a MINLP is still unavoidable [?].

- Most existing detection methods focus on detection accuracy, with no or little attention to detection delay. Detection delay is critical to the cybersecurity of energy systems as a shorter delay means a timely response that can minimize the negative impacts of the attacks. Quickest change detection (QCD) aims at minimizing the detection delay subject to constraints on detection accuracy. QCD is usually implemented by means of sequential analysis such as the sequential probability ratio test (SPRT) [112], the cumulative sum (CUSUM) [92, 113], generalized likelihood ratio (GLR) testing [114], etc. Most algorithms require perfect knowledge of the post-change distribution [115], which is usually difficult, if not impossible, to obtain [116].

3 Optimum Integration of Solar Energy With Battery Energy Storage Systems

The intermittent and stochastic nature of solar energy creates an imbalance between energy supplies and demands. Such an imbalance can be partly compensated by integrating the PV system with ESS. The ESS can store excessive solar energy during off-peak hours and discharge the stored energy during high-demand hours, such that both energy usage and peak demands can be significantly reduced.

In this chapter, we propose to perform optimum designs of battery-assisted PV systems by including system costs, aging effects of batteries and solar panels, inflation of electricity costs, and discounted long-term returns as design parameters. The whole system design and optimization will stand at the consumers' (commercial or residential users) point of view as a starting point, therefore the power grid is not modeled in this section.

3.1 Problem formulation

Consider a user who installs the PV system with ESS as shown in Fig. 3.1.

3.1.1 Battery Model

The time is divided into short non-overlapping windows with duration Δt each, e.g., $\Delta t = 1$ hour. The state of charge (SOC), or the energy stored in the battery, can be described by the following difference equation:

$$c_{t+1} = c_t + q_t^c \gamma_e - \frac{q_t^d}{\gamma_e} \tag{3.1}$$

where t is the time window index, c_t denotes the energy stored in the battery at the beginning of the t-th time window, q_t^c and q_t^d are the energy charged to and discharged from the battery at time window t, respectively, and γ_e denotes the energy efficiency, which represents the energy loss in the battery. We have $\gamma_e = \gamma_{\text{inv}}\sqrt{\gamma_{\text{batt}}}$, where γ_{inv} is the inverter efficiency and γ_{batt} is the battery round-trip efficiency [117].

Due to the physical limits of the battery, the average charging and discharging rates at any time window are limited by the number of batteries and the physical limit of each

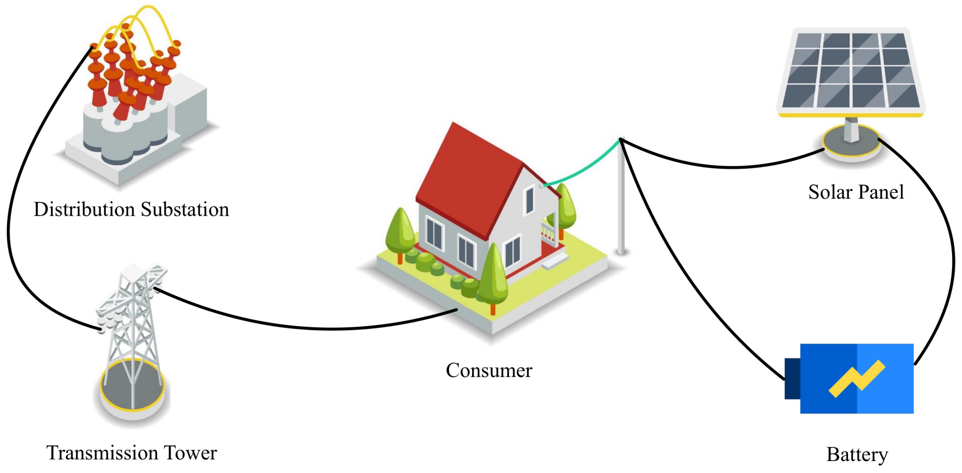


Figure 3.1: ESS assisted PV system

battery, as

$$0 \leq q_t^c \leq n_b q_c^{\max} \Delta t, \quad \forall t \in \mathcal{T} \tag{3.2}$$

$$0 \leq q_t^d \leq n_b q_d^{\max} \Delta t, \quad \forall t \in \mathcal{T} \tag{3.3}$$

where n_b is the number of batteries, $q_c^{\max}$ and $q_d^{\max}$ are the maximum charging and discharging rate of a single battery, respectively, $\mathcal{T} = \{1, 2, \cdots, T\}$ is the time horizon under consideration, with T being the total number of time windows.

In addition to the charging and discharging limits, the SOC at any time is limited by the total capacities of the battery as

$$0 \leq c_t \leq n_b c^{\max}[1 - \alpha \cdot (m-1)^{0.75} - \beta\sqrt{m-1}] \tag{3.4}$$

where $c^{\max}$ is the initial battery capacity, or the maximum energy, of a single battery, α and β are the calendar aging and cycling aging coefficients of the battery described in months, respectively, and m is the age of the battery in months [118]. The aging coefficient models the phenomenon that the battery capacity becomes smaller over a long period of time.

3.1.2 Power from the grid

Denote q_t^{net} as the energy bought from the utility at time window t. Then we have

$$q_t^{\text{net}} = q_t^{\text{ld}} - q_t^{\text{sol}} + q_t^c - q_t^d \tag{3.5}$$

where q_t^{ld} denotes the actual load at time window t, and q_t^{sol} denotes the PV energy collected from the solar panels at time window t. The PV energy can be modeled as

$$q_t^{\text{sol}} = n_s q_t^0 \gamma_s^{m-1} \tag{3.6}$$

where n_s is the number of solar panels, q_t^0 is the PV energy collected by a single panel at time window t, γ_s is the efficiency of the solar panel described in months, m is the age of the solar panel in months. The solar panel efficiency describes the aging effect of the solar panel over time.

3.1.3 Objective function

The total cost of the system consists three parts: energy charge, power or demand charge, and system cost.

Energy charge

Time-of-Use (TOU) is a rate plan that is determined by both the amount of energy bought from the utility and the time when the energy is consumed. Despite slight differences in rate between utilities, most TOU plans divide days into peak hours, part-peak hours, and off-peak hours. Similarly, weeks are divided into weekdays and weekends; years are divided into summer months and winter months.

Define the set of peak hours, part-peak hours, and off-peak hours as $\mathcal{H}_{\text{pk}}$, $\mathcal{H}_{\text{pp}}$, and $\mathcal{H}_{\text{op}}$, respectively. Denote the electricity cost during these three sets of hours as P_{pk}, P_{pp}, and P_{op}, respectively. Considering the effects of inflation, the unit price ($ per kWh) at time window t is given by:

$$p_t = \begin{cases} P_{\text{pk}}(1 + r_{\text{infl}}^{\lfloor \frac{t-1}{W} \rfloor}), & t \in \mathcal{H}_{\text{pk}} \\ P_{\text{pp}}(1 + r_{\text{infl}}^{\lfloor \frac{t-1}{W} \rfloor}), & t \in \mathcal{H}_{\text{pp}} \\ P_{\text{op}}(1 + r_{\text{infl}}^{\lfloor \frac{t-1}{W} \rfloor}), & t \in \mathcal{H}_{\text{op}} \end{cases} \tag{3.7}$$

where r_{infl} is the annual inflation rate of electricity cost, W is the total number of windows in one year, and the floor operator $\lfloor a \rfloor$ returns the largest integer that is less than or equal to a.

Define $\mathcal{H}_m$ as the set of time window indices that belong to the m-th month, and assume there are M months in the time horizon $\mathcal{T}$, that is, $\mathcal{T} = \bigcup_{m \in \mathcal{M}} \mathcal{H}_m$. The energy charge with the given TOU is:

$$C_{\text{E}} = \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{H}_m} p_t \max\left(q_t^{\text{net}}, 0\right). \tag{3.8}$$

where $\mathcal{M} = \{m | \mathcal{H}_m \subseteq \mathcal{T}\}$ is the set of indices of months in the optimization time horizon.

Demand charge

The demand charge is proportional to the highest average power in each month. Considering the effects of inflation, denote the demand charge at the m-th month

$$D_{\max,m} = D_0 \cdot (\,1 + r_{\text{infl}}^{\lfloor \frac{m-1}{12} \rfloor}) \tag{3.9}$$

where D_0 is the initial demand charge in the unit of \$ per kW.

The total demand charge can be calculated as

$$C_{\text{D}} = \sum_{m=1}^{M} D_{\max,m} \max_{t \in \mathcal{T}_m} \frac{q_t^{\text{net}}}{\Delta t} \tag{3.10}$$

System cost

The costs of solar panels and batteries include the costs for product procurement, installation, and maintenance. It is assumed that the total cost is proportional to the number of solar panels and batteries, as

$$C_{\text{S}} = P_s n_s + P_b n_b \tag{3.11}$$

where P_s and P_b are the unit costs (including procurement and installation) of solar panels and batteries, respectively, and n_s and n_b are the number of solar panels and batteries, respectively.

The objective of the problem is to minimize the long term total cost of the system by identifying the optimum number of solar panels and batteries required for the system. The optimum identification of the solar panels and batteries depends on the charging and discharging schedule, thus the charging and discharging rates, q_t^c and q_t^d, for all $t \in \mathcal{T}$, will also be considered as optimization variables.

Since the optimum design is performed off-line before system installation, the load information, q_t^{ld}, and the solar energy, q_t^{sol}, are unknown during the design phase. However, the optimum design can be performed by using known historical data given that the load and weather for a given location do not change dramatically from year to year [25]. Thus the known historical data of q_t^{ld} and q_t^{sol} are used in the optimum design as in [25].

Based on the above models and analysis, the optimization problems can be formulated as

$$\begin{aligned}
\min.\quad & C_{\mathrm{E}} + C_{\mathrm{D}} + C_{\mathrm{S}} && \text{(P1)}\\
\text{s.t.}\quad & (6.12) - (3.6), &&\\
& n_s, n_b \in \mathbb{Z}_+, && (3.12)\\
& n_s \le N_s, && (3.13)\\
& n_b \le N_b, && (3.14)
\end{aligned}$$

where $\mathbb{Z}_+$ is the set of non-negative integers, N_s and N_b are the maximum number of solar panels and batteries allowed in the system, respectively, and the optimization is performed with respect to the following variables: $\{q_t^c, q_t^d\}_{t\in\mathcal{H}}, n_s$, and n_b. The values of N_s and N_b are usually determined by the area of installation.

The above problem is a MINLP problem. The non-linearity comes from the maximum term in both the energy charge C_{E} in (4.5) and the demand charge C_{D} in (4.6). The problem is in general NP-hard.

3.2 Different approaches

3.2.1 Mixed Integer Linear Programming

In this section, we transform (P1) into an equivalent MILP problem [33], which can be optimally solved by using the branch-and-bound (B&B) algorithm [119].

Since the non-linearity in (P1) comes from the maximum term in the objective function, we can equivalently convert it to a linear objective function by introducing new variables [120]. Define two new variables, $q_{+,t}^{\text{net}}$ and $q_{\max,m}^{\text{net}}$, with the following new constraints

$$q_t^{\text{net}} \leq q_{+,t}^{\text{net}}, \quad \forall t \in \mathcal{T} \tag{3.15}$$

$$q_t^{\text{net}} \leq q_{\max,m}^{\text{net}}, \quad \forall t \in \mathcal{H}_m \tag{3.16}$$

$$q_{+,t}^{\text{net}} \geq 0, \quad \forall t \in \mathcal{T} \tag{3.17}$$

$$q_{\max,m}^{\text{net}} \geq 0, \quad \forall m \in \mathcal{M} \tag{3.18}$$

Based on the above definition and constraints, the energy charge and demand charge can be upper bounded, respectively, by

$$\bar{C}_{\text{E}} = \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{H}_m} p_t q_{+,t}^{\text{net}} \tag{3.19}$$

$$\bar{C}_{\text{D}} = \frac{1}{\Delta t} \sum_{m \in \mathcal{M}} D_{\max,m} q_{\max,m}^{\text{net}} \tag{3.20}$$

The MINLP problem (P1) can now be equivalently converted to a new problem as

$$\begin{aligned} \min. \quad & \bar{C}_{\text{E}} + \bar{C}_{\text{D}} + C_{\text{S}} \\ \text{s.t.} \quad & (6.12) - (3.6), (3.12) - (3.18) \end{aligned} \tag{P2}$$

where the optimization variables are: $\{q_t^c, q_t^d, q_{+,t}^{\text{net}}\}_{t \in \mathcal{T}}$, $\{q_{\max,m}^{\text{net}}\}_{m \in \mathcal{M}}$, n_s, and n_b. Compared to (P1), (P2) has two new groups of optimization variables, $\{q_{+,t}^{\text{net}}\}_{t \in \mathcal{T}}$ and $\{q_{\max,m}^{\text{net}}\}_{m \in \mathcal{M}}$. The energy and demand charges in (P1) are replaced in (P2) with their respective upper bounds, which are linear functions of the optimization variables. As a result, the non-linearity in (P1) is removed and (P2) is an MILP.

The equivalence between (P1) and (P2) is established in the following lemma.

Lemma 1. *The optimum solution to (P2) is also the optimum solution to (P1).*

Proof. Denote the optimum cost functions from (P1) and (P2) as $C_{\text{E}}^* + C_{\text{D}}^* + C_{\text{S}}^*$ and $\bar{C}_{\text{E}}^\dagger + \bar{C}_{\text{D}}^\dagger + C_{\text{S}}^\dagger$, respectively. Since $C_{\text{E}} \leq \bar{C}_{\text{E}}$ and $C_{\text{D}} \leq \bar{C}_{\text{D}}$ by definition, we have $C_{\text{E}}^* \leq \bar{C}_{\text{E}}^\dagger$ and $C_{\text{D}}^* \leq \bar{C}_{\text{D}}^\dagger$.

Next we will show that $C_{\text{E}}^* = \bar{C}_{\text{E}}^\dagger$ and $C_{\text{D}}^* = \bar{C}_{\text{D}}^\dagger$ by using contradiction. Assume $C_{\text{E}}^* < \bar{C}_{\text{E}}^\dagger$ and $C_{\text{D}}^* < \bar{C}_{\text{D}}^\dagger$, then we can always make $\bar{C}_{\text{E}}^\dagger$ and $\bar{C}_{\text{D}}^\dagger$ smaller by letting

$$q_{+,t}^{\text{net}} = \max(0, q_t^{\text{net}}) \tag{3.21}$$

$$q_{\max,m}^{\text{net}} = \max_{t \in \mathcal{H}_m} q_t^{\text{net}} \tag{3.22}$$

while keeping all other variables unchanged to satisfy all constraints. Thus we must have $C_{\mathrm{E}}^{*} = \bar{C}_{\mathrm{E}}^{\dagger}$ and $C_{\mathrm{D}}^{*} = \bar{C}_{\mathrm{D}}^{\dagger}$, and the equality is achieved when (3.21) and (3.22) are satisfied. That is, (3.21) and (3.22) give the optimum values of $q_{+,t}^{\mathrm{net}}$ and $q_{\max,m}^{\mathrm{net}}$ in (P2)

Substituting the optimum values of $q_{+,t}^{\mathrm{net}}$ and $q_{\max,m}^{\mathrm{net}}$ given in (3.21) and (3.22) into (P2), we can see that (P2) is exactly the same as (P1). Thus the optimum solutions to the two problems are equivalent. This completes the proof. □

The MILP problem in (P2) is still non-convex due to the integer constraints. The MILP can be optimally solved by using the B&B algorithm [119], which performs systematic enumeration of subsets (branches) of the feasible region by iteratively dividing the current branch into two branches based on solutions of relaxed integer linear program in the current branch. The solution in each branch are compared to estimated upper and lower bounds of the optimal value, and a branch is discarded if it cannot outperform the best result found so far by the algorithm.

The B&B algorithm can obtain the globally optimum solution to (P2) with a complexity that is much lower than exhaustive search. The calculation in each branch requires solving a LP problem.

To accurately account for the aging effects of the battery and solar panel, the optimization needs to be performed over the entire life cycle of the battery and/or solar panels. As a result, the optimization time horizon is in the order of years or longer. For example, if the expected life cycle of a battery is 10 years and we use 1-hour windows, then there are a total of $H = 87,648$ hours and $M = 120$ months if include two leap years. As a result, the total number of optimization variables are $3H + M + 2 = 263,066$. Even though LP can be solved in polynomial time, the large number of optimization variables makes the complexity extremely high and requires very long optimization time. In the B&B algorithm, relaxed LP needs to be performed in each branch, and this further improves the computation complexity.

3.2.2 Dynamic Programming

In this section, we propose to solve (P1) by developing a low complexity algorithm with the help of dynamic programming (DP).

The optimization variables in (P1) include the charging/discharging schedule, $\{q_t^c, q_t^d\}_{t\in\mathcal{T}}$, and the number of batteries and solar panels, n_s and n_b. In order to implement DP, (P1) is decomposed into two sub-problems by separating the dynamic variables, $\{q_t^c, q_t^d\}_{t\in\mathcal{T}}$, and static variables, $\{n_s, n_b\}$, as

$$\begin{aligned} \min_{\{q_t^c, q_t^d\}_{t\in\mathcal{T}}} \quad & C_{\text{ED}} \triangleq C_{\text{E}}(n_s, n_b) + C_{\text{D}}(n_s, n_b) && \text{(P1a)} \\ \text{s.t.} \quad & (6.12) - (3.6), \end{aligned}$$

$$\begin{aligned} \min_{\{n_s, n_b\}} \quad & C^*_{\text{ED}}(n_s, n_b) + C_{\text{S}}(n_s, n_b) && \text{(P1b)} \\ \text{s.t.} \quad & n_s, n_b \in \mathbb{Z}_+, \\ & n_s \leq N_s, \\ & n_b \leq N_b, \end{aligned}$$

where $C^*_{\text{ED}}(n_s, n_b)$ is the optimum solution to (P1a) given n_s and n_b.

In (P1a), we first fix the static variables n_s and n_b, and minimize the energy and demand charges by identifying the optimum dynamic variables $\{q_t^c, q_t^d\}_{t\in\mathcal{T}}$. Under a fixed n_s and n_b, the cost of battery and solar panels are fixed, so C_{S} is excluded from the objective function in (P1a). In the objective function, the energy and demand charges are expressed as explicit functions of n_s and n_b. With the optimum scheduling obtained from (P1a), we can then identify the optimum values of n_s and n_b in (P1b). (P1a) can be solved by using DP, and (P1b) can be solved through coordinate descent with binary search.

Solving (P1a) with DP

DP is based on Bellman's principle of optimality, which states that the state and decision at the current moment fully determine the optimum policy in the future. However, Bellman's principle of optimality cannot be readily applied to (P1a), mainly due to the form of the demand charge C_{D} in its objective function.

In the calculation of the objective function, the summation in C_{E} is performed on the time scale of small time windows (e.g. hours), yet the summation in C_{D} is performed on the time scale of months, and the maximum operator in C_{D} is performed over all the time windows within a month. Such maximum operations cannot be readily described by the Bellman equation. Thus we need to transform the calculation of C_{D} such that it has the same time scale as the calculation of C_{E}, and then transform (P1a) into an equivalent problem that satisfies Bellman's principle of optimality.

Denote the indices of the first and last time windows in $\mathcal{H}_m$ as t_{m1} and t_{m2}, respectively. Then we define a new state variable $\phi_t^{(m)}$ as

$$\phi_t^{(m)} = \begin{cases} 0, & t < t_{m1} \\ \max(\phi_{t-1}^{(m)}, q_t^{\text{net}}), & t \in \mathcal{H}_m \\ \max_{t \in \mathcal{H}_m} q_t^{\text{net}}, & t > t_{m2} \end{cases} \tag{3.23}$$

Based on the above definition, the demand charge in the m-th month can be calculated as

$$C_{\text{D}}^{(m)} = \frac{D_{\max,m}}{\Delta t} \sum_{t \in \mathcal{H}_m} \left(\phi_t^{(m)} - \phi_{t-1}^{(m)} \right) \tag{3.24}$$

The total demand charge can then be calculated as

$$C_{\text{D}} = \sum_{m \in \mathcal{M}} C_{\text{D}}^{(m)} = \frac{1}{\Delta t} \sum_{m \in \mathcal{M}} D_{\max,m} \sum_{t \in \mathcal{H}_m} \left(\phi_t^{(m)} - \phi_{t-1}^{(m)} \right) \tag{3.25}$$

From (3.23), we have

$$C_{\text{D}} = \frac{1}{\Delta t} \sum_{m \in \mathcal{M}} D_{\max,m} \sum_{t \in \mathcal{H}_m} \max(0, q_t^{\text{net}} - \phi_{t-1}^{(m)}) \tag{3.26}$$

The cost function in (P1a) can then be rewritten as

$$C_{\text{E}} + C_{\text{D}} = \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{H}_m} \left[p_t \max(0, q_t^{\text{net}}) + \frac{D_{\max,m}}{\Delta t} \max(0, q_t^{\text{net}} - \phi_{t-1}^{(m)}) \right] \tag{3.27}$$

The cost function in (3.27) can help us convert (P1a) into an equivalent problem that satisfies Bellman's principal of optimality. For the optimization problem, the action variable at $t \in \mathcal{H}_m$ is $a_t = \{q_t^c, q_t^d\}$, and the state variables are $\beta_t = \{S_t, \phi_{t-1}^{(m)}\}$. To facilitate analysis, for $t \in \mathcal{H}_m$, define

$$r(\beta_t, a_t) = p_t \max(0, q_t^{\text{net}}) + \frac{D_{\max,m}}{\Delta t} \max(0, q_t^{\text{net}} - \phi_{t-1}^{(m)}) \tag{3.28}$$

With the above definition, (P1a) can be equivalently converted to

$$\begin{aligned} \min_{\{q_t^c, q_t^d\}_{t \in \mathcal{T}}} \quad & \sum_{m \in \mathcal{M}} \sum_{t \in \mathcal{H}_m} r(\beta_t, a_t) \\ \text{s.t.} \quad & (6.12) - (3.6), (3.23) \end{aligned} \tag{P3}$$

In (P3), the cost function is decomposed as the summation of $r(\beta_t, a_t)$, which depends solely on the current state variable β_t and action variable a_t. Constraints (6.12) and (3.23) describes the evolution of the state variables; constraints (3.2)-(3.6) specify the boundaries of the state and action variables. Thus (P3) satisfies Bellman's principal of optimality. The Bellman equation of (P3) can be written as

$$\begin{aligned} V_0(\beta_0) &= \min_{\{q_0^c, q_0^d\}} r(\beta_0, a_0) = 0 \\ V_t(\beta_t) &= \min_{\{q_t^c, q_t^d\}} \left[r(\beta_t, a_t) + V_{t-1}(\beta_{t-1}) \right], \quad 1 \leq t \leq T, \end{aligned} \tag{3.29}$$

Specifically, the objective is to find at each time window i the optimum value function $V_t(\beta_t)$ that corresponds to the optimum cost in the time horizon between $[1, t]$. In the above notation, β_0 denotes the initial state at $t = 0$, $J = V_T(\beta_T)$ is the optimal cost from the initial state to the finial state.

The optimum schedule q_t^c and q_t^d can be obtained by solving the Bellman equation in (5.40) through forward recursion based on following equations:

$$(s_{\beta_t}, a_{\beta_t}) = \arg\min_{(s,a)\in\Omega_{\beta_t}} \left[r(\beta_t, a) + V_{t-1}(s) \right], \tag{3.30}$$

where Ω_{β_t} is the space that contains all state and action pairs, (s, a), such that we can reach state β_t from $\beta_{t-1} = s$ by taking action a. At time t, a_{β_t} gives the optimum action to reach state β_t by minimizing the accumulated cost from the initial state to the current state β_t; correspondingly, s_{β_t} is the state preceding β_t with the action a_{β_t}. Then the value function $V_t(\beta_t)$ can be updated by

$$V_t(\beta_t) = r(\beta_t, a_{\beta_t}) + V_{t-1}(s_{\beta_t}) \tag{3.31}$$

However, there is in general no closed-form solution to the Bellman equation for arbitrary state β_t. We can numerically solve the Bellman equation by using algorithms such as relative value iteration algorithm (RVIA) [121], the Viterbi algorithm [122], or the Bellman-Ford algorithm [123]. The implementation of these algorithms requires the discretization of the continuous state variables and action variables. A larger discritization step size can reduce complexity at the cost of a lower precision, and vice versa. The tradeoffs between complexity and accuracy have been studied in [124], [125], [126], [127]. The impacts of discretization step are studied with numerical examples in the next section, and it will be shown that appropriate discretization step can be chosen to achieve negligible approximation errors while maintaining low complexity of the algorithm.

The action space contains two variables, q_t^c and q_t^d. Thus the size of the action space grows quadratically with the number of discretization levels of the charging/discharging action. Since the complexity of the DP algorithm is directly proportional to the size of the action and state spaces, we propose to reduce the dimension of the action space by replacing the two action variables q_t^c and q_t^d in (P3) with a single variable, q_t^{net}, as

$$\begin{aligned} \min_{\{q_t^{\text{net}}\}_{t\in\mathcal{T}}} \quad & \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{H}_m} r(\beta_t, q_t^{\text{net}}) \\ \text{s.t.} \quad & (6.12) - (3.6), (3.23) \end{aligned} \tag{P4}$$

The equivalence between (P3) and (P4) is established in the following lemma.

Lemma 2. *The optimum solution to (P4) is also the optimum solution to (P3).*

Proof. For a given pair $\{q_t^c, q_t^d\}$, q_t^{net} is uniquely determined according to (4.4). On the other hand, for a given q_t^{net}, we may have multiple $\{q_t^c, q_t^d\}$. To prove the equivalence between (P3) and (P4), it is sufficient to show that the optimum $\{q_t^c, q_t^d\}$ for (P3) can be uniquely determined by the optimum $\{q_t^{\text{net}}\}$ for (P4), that is, there is a bijective relationship between the optimum solutions between (P3) and (P4).

Assume there are two different pairs of action variables, $a_t^1 = (q_t^{c1}, q_t^{d1})$ and $a_t^2 = (q_t^{c2}, q_t^{d2})$, which provide the same transition from $\beta(i-1)$ to β_t in the discretized state space. By definition:

$$q_t^{c1}\gamma_e - \frac{q_t^{d1}}{\gamma_e} = q_t^{c2}\gamma_e - \frac{q_t^{d2}}{\gamma_e} = c_t - c_{t-1} \tag{3.32}$$

then we have:

$$(q_t^{c1} - q_t^{c2})\gamma_e = \frac{q_t^{d1} - q_t^{d2}}{\gamma_e} \tag{3.33}$$

Without loss of generality, assume $q_t^{c1} > q_t^{c2}$. Given the fact that $0 < \gamma_e < 1$, we have,

$$\begin{aligned} q_t^{c1} - q_t^{c2} &> q_t^{d1} - q_t^{d2} \\ q_t^{c1} - q_t^{d1} &> q_t^{c2} - q_t^{d2} \\ q^{\text{net}1}{}_t &> q^{\text{net}2}{}_t \end{aligned}$$

Based on (3.28), we have

$$r(\beta_t, a_t^1) > r(\beta_t, a_t^2) \tag{3.34}$$

Then according to (3.30), a_t^1 cannot be the optimum solution. Thus an optimum q_t^{net} corresponds to one unique pairs of (q_t^c, q_t^d). This above analysis proves the bijective relationship between the optimum q_t^{net} and (q_t^c, q_t^d). This completes the proof. □

Based on the equivalence between (P3) and (P4), we can solve (P4) by using the scalar variable q_t^{net} as the action variable during the numerical solution of the Bellman equation with (3.30) and (3.31). The Bellman equation is solved with a modified Bellman-Ford algorithm, and details are given in Algorithm 1.

Algorithm 1 Modified Bellman-Ford algorithm

```
Require: Step size $q$, discretized state space $\mathcal{S}$, discretized action space $\mathcal{A}$, cost function
    $r(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}$.
1: Initialization: set the initial state $\beta_0 = 0$, the initial value function $V_0(\beta_0) = 0$.
2: for $t = 1$ to $T$ do
3:   for $\beta_t \in \mathcal{S}$ do
4:     Find the optimal (previous state, action) pair that can reach $\beta_t$:
           $$(s_{\beta_t}, a_{\beta_t}) = \arg\min_{(s,a)\in\Omega_{\beta_t}} [r(\beta_t, a) + V_{t-1}(s)]$$
5:     Compute the value function
           $$V_t(\beta_t) = r(\beta_t, a_{\beta_t}) + V_{t-1}(s_{\beta_t})$$
6:   end for
7: end for
8: Identify the optimum ending state $\beta_T^* = \arg\min_{\beta_T \in \mathcal{S}} V_T(\beta_T)$
9: Identify the optimum actions and states at each iteration by tracing back from the the
   optimum ending state
       $$a_{t-1}^* = a_{\beta_t^*}, \quad \beta_{t-1}^* = s_{\beta_t^*}, \text{ for } t \in \{T, T-1, \cdots, 2\}$$
Ensure: Optimum policy $a_t^*$, for $t \in \{1, 2, \cdots, T-1\}$.
```

Solving (P1b) with Coordinate Descent

The coordinate descent (CD) algorithm is used to solve the optimum values of n_b and n_s by using the results of the DP algorithm in the previous subsection. The CD algorithm is implemented by using relaxed integer programming (RIP) of (P1), where we remove the integer constraint $n_b, n_s \in \mathbb{Z}_+$. The integer-relaxed version of (P1) forms a convex problem, because it contains only linear and maximum operators of the optimization variables: n_s, n_b, and $\{q_t^c, q_t^d\}_{t\in\mathcal{T}}$.

The CD algorithm is summarized in Algorithm 2. In the algorithm, $C^*_{\text{ED}}(n_s, n_b)$ is the optimum solution to (P1a) according to Algorithm 1.

Algorithm 2 CD algorithm to compute n_b and n_s

```
Require: Step size $q$.
1: Initialization: Iteration counter $k \leftarrow 0$, and $n_b^0 \leftarrow N_b$, $n_s^0 \leftarrow N_s$.
2: repeat
3:     Identify the value of $n_s^{(k+1)}$: using binary search to solve
           $$n_s^{(k+1)} = \arg\min_{n_s} \left\{ C^*_{\text{ED}}\left(n_s, n_b^{(k)}\right) + P_s n_s + P_b n_b^{(k)} \right\}$$
4:     Identify the value of $n_b^{(k+1)}$: using binary search to solve
           $$n_b^{(k+1)} = \arg\min_{n_b} \left\{ C^*_{\text{ED}}\left(n_s^{(k+1)}, n_b\right) + P_s n_s^{(k+1)} + P_b n_b \right\}$$
5:     $k \leftarrow k+1$.
6: until $|n_b^{(k+1)} - n_b^{(k)}| < 1$ and $|n_s^{(k+1)} - n_s^{(k)}| < 1$.
7: Rounding the solution $n_s^\dagger = n_s^{(k+1)}$ and $n_b^\dagger = n_b^{(k+1)}$ to their nearest integer values.
Ensure: The optimal size $n_b^\dagger$ and $n_s^\dagger$, the total cost $C^*_{\text{ED}}\left(n_s^\dagger, n_b^\dagger\right) + P_s n_s^\dagger + P_b n_b^\dagger$
```

Since the integer relaxed version of (P1) is convex, it is convex in each individual variable. Thus CD can be performed to identify the integer-relaxed optimum solution of n_s and n_b by successively minimizing along coordinate directions. In each iteration, we fix the value of one variable and identify the optimum value of the other. Given the fact that the objective function is convex in n_s and n_b, the optimization in each direction can be performed by using binary search. The implementation of binary search requires an upper bound on the values of n_b and n_s, and both can be upper bounded by the area of installation as in (3.13) and (3.14).

Due to the convexity of the integer-relaxed optimization problem, the CD algorithm will converge to the integer-relaxed optimum solutions upon convergence. Denote the optimum solution to the integer-relaxed optimization problem as $n_s^\dagger$ and $n_b^\dagger$. Then the integer solution can be obtained by rounding $n_s^\dagger$ and $n_b^\dagger$ to their nearest integer values. It should be noted that the final integer solution might be suboptimum due to rounding of the solution of RIP.

3.3 Case studies

Simulation results are presented in this section to demonstrate the performance of the proposed optimum designs of PV system with BESS. The MILP is solved by CVX, a package for specifying and solving convex programs [128]. Both MILP and DP are solved in MATLAB. All simulations are performed on a workstation with a 6-core Intel Core i7-5820K CPU operating at 3.3 GHz, NVIDIA GeForce GTX 950 GPU, and 32 GB of random access memory (RAM).

3.3.1 Simulation Environment

The designs are performed by using load data from a public database provided by the Office of Energy Efficiency and Renewable Energy (EERE) at the United States Department of Energy (DOE) [129]. The database provides hourly load data in one year from different types of buildings at various locations. The data from a large hotel building in San Francisco in 2004 are used in this paper. The energy provided by solar panels is estimated by using the PVWatts calculator [130] from National Renewable Energy Laboratory (NREL). The utility charges are calculated by using the TOU rate (p_t, D_0) of Pacific Gas and Electric Company (PG&E) in Table 3.1, along with the time division in Table 3.2 [131]. The size of the time window is set at $\Delta t = 1$ hour. The inflation rate of electricity cost is $r_{\text{infl}} = 2\%$.

The batteries are modeled by using Tesla Powerwall, which has a charging/discharging rate of $q_c^{\max} = q_d^{\max} = 5$ kW, and a capacity of $c^{\max} = 13.5$ kWh, at a price of $P_b = \$5,900$ each. Based on the datasheet [132] and calculation, the energy efficiency of the batteries is $\gamma_e = 94\%$. According to the warranty on the data sheet, the calendar aging coefficient is $\alpha = 0.0036$, and the cycling aging coefficient is $\beta = 0.0155$. The initial SOC of battery in

Table 3.1: TOU rate in San Francisco

Total Energy Rates ($ per kWh)	Peak	Part-Peak	Off-Peak
Summer	0.15384	0.11333	0.08651
Winter	-	0.10779	0.09317
Total Demand Rates ($ per kW)		16.08	

Table 3.2: Times of the year and times of the day

Service Time	Summer (5/1 to 10/31)	Winter (11/1 to 4/30)
Peak	13:00 to 19:00	-
Part-Peak	10:00 to 13:00 and 19:00 to 22:00	10:00 to 22:00
Off-Peak	22:00 to 10:00	

the system is 0.

The price of a 10kW solar panel is set at $P_s = \$6,400$, including the price of products and installation. A maximum of $N_s = 120$ panels can be installed on the hotel rooftop limited by the area. Similarly, it is assumed that a maximum of $N_b = 100$ batteries can be installed in the system. The storage efficiency for solar panels described in month is set at $\gamma_s = 99.96\%$ [133].

To evaluate the profitability, the savings in monthly bills are discounted with an annual interest rate (r_{intr}) of 4% to calculate the net present value (NPV) up to the M-th month:

$$\text{NPV} = \sum_{m=1}^{M} C_{\text{saving}}(m) \cdot (1 + r_{\text{intr}})^{\lfloor \frac{m-1}{12} \rfloor} \tag{3.35}$$

The break-even point is defined as the month up to which the NPV of savings covers the total system cost for the first time.

3.3.2 One-Year Results with MILP

We first study the optimum designs of the PV system with MILP. Due to the high complexity of MILP, the design is restricted to a time horizon of one year. Optimum designs

Table 3.3: Annual utility bill under different configurations

System	Battery-assisted PV	PV-only
Utility bill ($)	361,030	446,413
Savings in bill ($)	252,561	167,178
Reduction in bill	41.2%	27.2%
Electricity bill without PV ($)	613,591	

with longer time horizons will be performed by using DP and CD later this section. Due to the short time horizon, the unit prices for solar panels and batteries are prorated to one year in the cost function. That is, based on the assumption of a 10-year life cycle of the solar panels and batteries, the unit price used in the cost function in the one-year study is obtained by dividing their actual prices by a factor of 10. The optimization is performed by solving the MILP defined in (P2). The optimization results indicate that the optimum performance can be achieved by using $n_b^* = 90$ Tesla Powerwall batteries and $n_s^* = 120$ 10 kW solar panels in the PV system. The total saving in annual utility bills after optimization are shown in Table 3.3. For reference, we also compare the performance of a system with solar panels but without batteries. The proposed optimum design of the battery-assisted PV system can achieve a 41.2% reduction in utility bills. For the PV-only system, the saving in utility bill drops to 27.2%.

Fig. 3.2 shows a snapshot of energy usage on a single day, July 1st, with or without the proposed PV system. The top figure shows the net load $q^{\text{net}}(i)$, which is the actual amount of energy bought from the utility at the i-th hour; the bottom figure shows the battery SOC $s(i)$. Both are shown as functions of the hour of the day. Throughout the day, the proposed system with both PV and battery has the lowest net load, followed by the PV-only and conventional systems, respectively. The largest energy saving is achieved between 10:00 to 16:00, where the net load drops to zero because the generated solar energy exceeds the actual load. During this time period, the extra energy is charged to the battery. During the evening hours between 19:00 to 24:00, the battery is gradually discharged in the proposed system, which maintains a steady net load at 450 kWh. The battery is depleted at 24:00 because of the low energy usage after that. On the other hand, for the conventional or PV-only systems, the load is peaked at 780 kW at 21:00. Therefore, a considerable amount

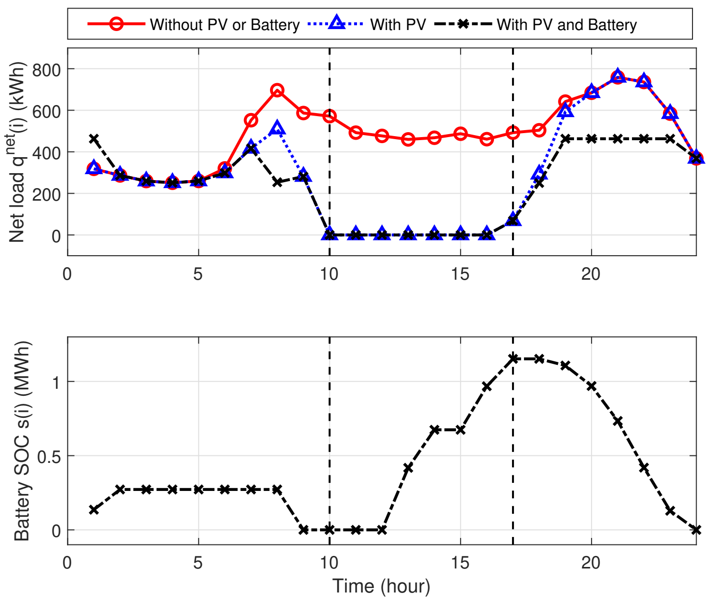
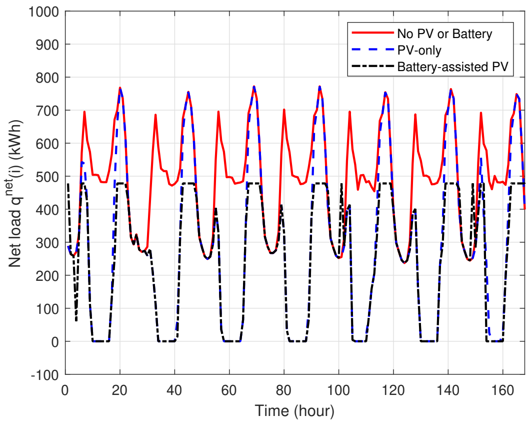
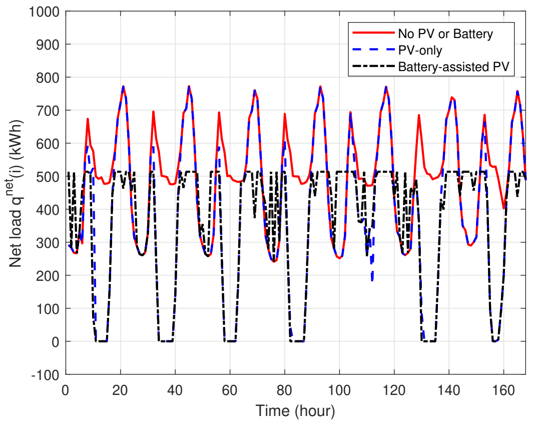


Figure 3.2: Snap shot of 1-day energy usage on July 1st.



(a) First week of June



(b) First week of December

Figure 3.3: Snap shots of 1-week energy usage in the first weeks of June and December, respectively.

of energy is saved with the proposed optimum design.

Figs. 3.3a and 3.3b show one-week snapshots of energy usage during the first week of June and December, respectively. Due to the relatively mild weather in San Francisco, the loads in June and December are similar. There are usually two peaks in one day: the early one is around 08:00 and the later one is around 20:00. The loads with PV in summer is much lower than it is in winter especially on the early peak. The integration of solar and

PV can achieve significant peak shaving. In both months, the peak of the original load is around 780 kW, and it is shaved to 480 kW and 510 kW in June and December, respectively. Even though employing PV without batteries can achieve similar performance as the battery integrated PV system during day time, the omission of batteries results in the same peak as the conventional system in the evening hours, when the energy demand is the highest.

Both figures have a significant peak shaving phenomenon after using batteries. The difference between the peaks of PV-only and battery-integrated PV systems corresponds to batteries' discharging rate. In these two figures, the load difference is about 300kW, which is smaller than the maximum discharging rate of 90 Tesla Powerwall batteries. The area between the curves of battery-integrated PV system and PV-only system is equal to the amount of energy discharged from the batteries.

3.3.3 Comparison between results from MILP and DP

Next we compare the results obtained from MILP and DP in terms of both accuracy and complexity. Due to the high complexity of MILP, the comparison is performed by optimizing the system over a time horizon of one year. Since DP relies on discretization of the battery capacity and RIP is employed in CD, the results in DP are suboptimum compared to their MILP counterparts.

Fig. 3.4 shows monthly bills in a year with optimizations performed by using MILP and DP, respectively. In DP, the battery capacity is discretized by using a step size of $q = 10$ kWh. Both MILP and DP obtain the same optimization results for the number of Tesla Powerwall batteries, $n_b^* = 90$, and the number of 10 kW solar panels, $n_s^* = 120$. The scheduling results of DP and MILP are different due to the discretization approximation employed by DP. The results obtained from DP and MILP are very close to each other. It should be noted that the bills obtained by MILP in some months are higher than those from DP, but the results from MILP always yield the lowest annual bill, which is the objective function of MILP.

We can increase the precision of DP by reducing the step size q, at the cost of a higher complexity. Thus the tradeoff between accuracy and complexity of DP can be adjusted by tuning the discretization step size q. Table 3.4 shows the percentage error of the DP results compared to the MILP results under different step size q. The percentage error increases in q as expected. Using a step size of $q = 10$ kWh or less can ensure that the approximation

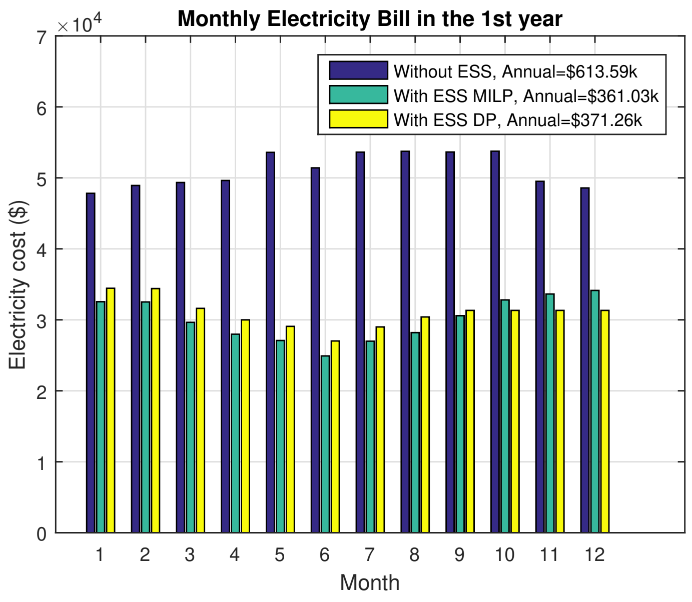


Figure 3.4: Comparison of monthly electricity bill with different scheduling approaches (q=10 kWh).

Table 3.4: Error of DP for different step size

Step size q (kWh)	5	10	20	25	50	100
Annual bill (k$)	365.45	371.56	372.61	373.49	375.59	395.83
Percent error (%)	1.22	2.92	3.21	3.45	4.03	9.64

error is below 3%. With 90 Tesla Powerwall batteries and $q = 10$ kWh, the battery capacity is discretized into $90 \times 13.5/10 = 121$ states.

Fig. 3.5 compares the complexity of MILP and DP algorithm as a function of the number of months in the optimization horizon. The complexity is measured as the amount of time (in seconds) required to solve the optimization problem. The complexity of MILP scales exponentially with m, the number of months in the time horizon. The complexity quickly becomes prohibitively high when the optimization horizon is long. On the other hand, the complexity of DP scales linearly with the number of months in the optimization horizon. The slope of the linearly scaled complexity increases as the step size q decreases. For a time horizon of 20 months, the optimization time required by MILP and DP with $q = 10$ kWh is 500.5 and 1833.0 seconds, respectively. The complexity difference will further

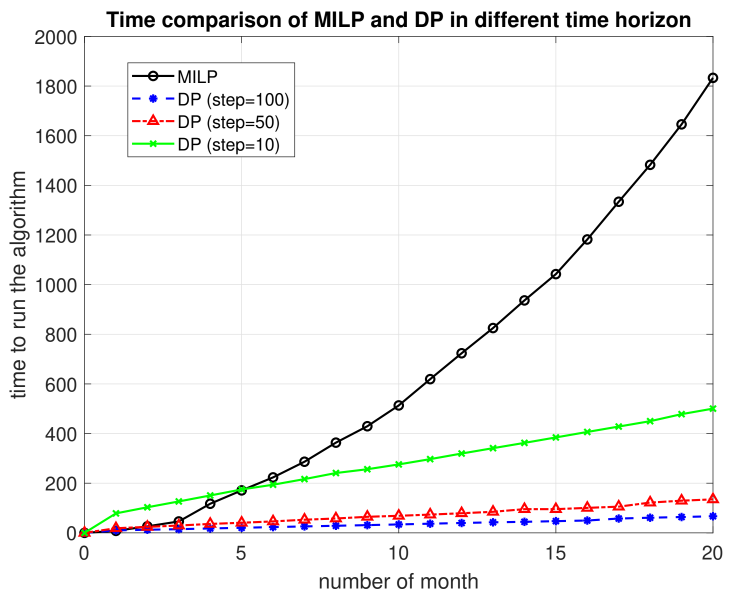


Figure 3.5: Time to run the MILP and DP ($q = 10, 50, 100$ kWh).

increase over longer time horizons.

3.3.4 Ten-Year results with DP

In this subsection, optimizations are performed over a time horizon of 10 years with the DP algorithm, and this corresponds to $T = 87,648$ hours. A period of 10 years is roughly on the same time scale as the life cycle of solar panels and batteries. Consequently, results obtained by optimizing over a 10-year time horizon can be better tuned based on the aging effects of the solar panels and batteries. However, the complexity of MILP is prohibitively high over a 10-year period, so only results from DP are shown. In DP, the step size is set at $q = 10$ kWh to achieve a balanced tradeoff between complexity and accuracy. Based on the optimization results, $n_b^* = 96$ Tesla Powerwall batteries and $n_s^* = 120$ 10 kW solar panels are required for the PV system. Compared to the one-year optimization results, 6 more Tesla Powerwall batteries are required to compensate for the decrease in both efficiency and storage capacity. The number of solar panels remains unchanged at the maximum number allowed by the area of the installation site.

The total utility bills under different system configurations over the 10-year period are shown in Table 3.5. The corresponding system costs are also shown in the table. If we do not consider the system cost, the battery-assisted PV system and PV-only system achieve a

Table 3.5: Utitlity and system cost under 10-year horizon

System	Battery-assisted PV	PV-only
Utility bill ($)	3,665,409	5,075,923
Savings in bill ($)	3,402,434 (48.1%)	1,991,920 (28.2%)
System cost ($)	1,334,400	768,000
Break-even point (month)	66	51
Total cost (utitlity + system) ($)	4,999,809	5,843,923
Total saving (utitlity + system) ($)	2,068,034 (29.3%)	1,223,920 (17.3%)
Electricity bill without PV ($)	7,067,843	

48.1% and 28.2% reduction in the total utility bills, respectively. The savings are substantial and much greater than the cost of solar panels and/or batteries. The battery-assisted PV system and the PV-only system achieve the break-even points at the 66th and 51st month, respectively.

In addition, the battery-assisted PV system significantly outperforms the PV-only system, and the extra savings in utility bill due to the addition of batteries far exceed the cost of batteries. For example, the battery-assisted PV system costs $566,400 more than the PV-only system, yet it can achieve an additional $1,410,514 saving in the 10-year utility bill. The total costs, which include both utility bill and system cost, of the battery-assisted PV and PV-only systems are $4,999,809 and $5,843,923, respectively, which are 29.3% and 17.3% lower than the total cost of the system without PV.

The impacts of the cost of batteries and solar panels are studied in Figs. 3.6 and 3.7. Fig. 3.6 shows the optimum number of solar panels as a function of the PV unit price, under various values of BESS unit price. When the PV unit price is low, e.g., under $4,000 per panel, then the system always selects the maximum number of solar panels allowed by the system to take advantage of low cost solar energy. The optimum number of solar panels decreases as the PV unit price increases. Under a given PV unit price, a higher battery unit price results in more solar panels, because the system needs more solar energy to compensate for the smaller battery capacity due to higher batter cost. Similar trends are also observed for the optimum number of batteries, which decreases in battery unit price but increases in

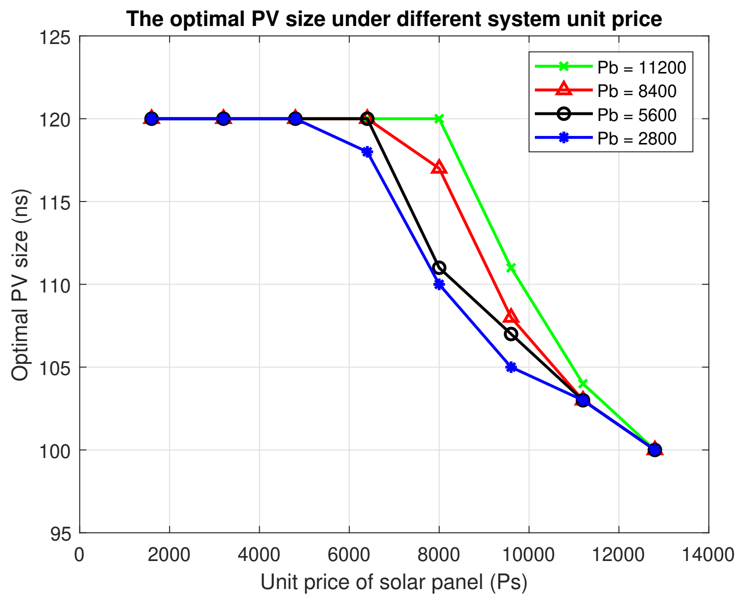


Figure 3.6: Change of solar panels under different unit price

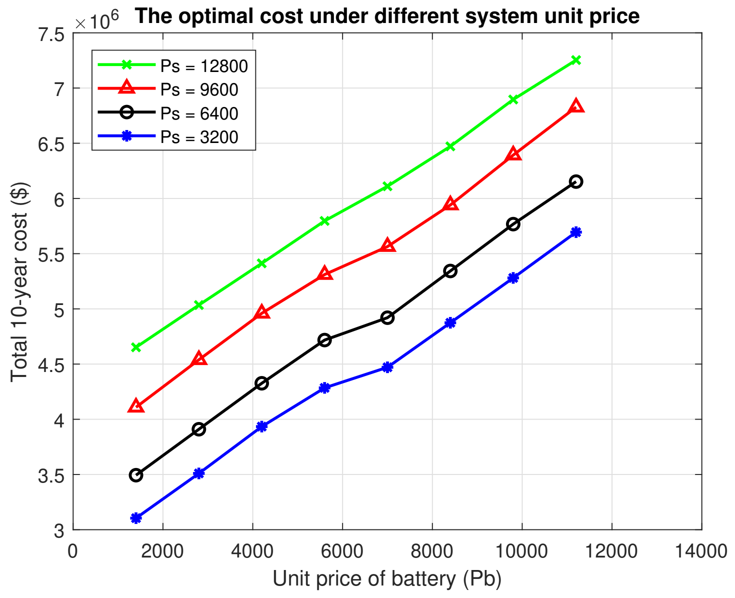


Figure 3.7: Change of total cost under different unit price of the battery and solar panel

PV unit price.

Fig. 3.7 demonstrates the impacts of PV and battery prices on the total 10-year cost of the system. Under a given battery unit price, the total 10-year cost increases almost linearly with the PV unit price. The rate of increasing is not affected by the battery unit price. Similar linear relationship is also observed between the total cost and battery unit price under a fixed PV unit price.

3.4 Conclusion

The optimum designs of photo-voltaic systems with battery energy storage systems have been studied in this paper. In order to accurately model and quantify the impacts of aging effects of solar panels and batteries, the optimum designs were performed over a long time horizon covering the entire life cycles of the battery-assisted PV systems. MILP- and DP-based methods have been proposed to solve the optimization problem. The MILP-based solution can obtain the globally optimum solution, but with prohibitively high complexity over long optimization time horizon. The DP-based solution provides a balanced tradeoff between accuracy and complexity. Case studies with real world data demonstrated that employing batteries in a PV system can achieve significant peak shaving and energy saving. Over a ten-year period, the total costs of a battery-assisted PV system and PV-only system are 29.3% and 17.3% lower than a conventional system without PV.

4 Reinforcement Learning Based On-Line Battery Energy Storage System Schedule Optimization

The last chapter has discussed the feasibility of saving electricity bill by storing the excessive solar energy during off-peak hours and discharging it during peak hours. It guides the consumers on investing a battery assisted PV system as an outlook of the whole off-line optimization problem, but does not provide an on-line ESS scheduling algorithm.

In this chapter, we propose a new DDPG-based online scheduling algorithm for a given PV systems with BESS. In specific, we propose a new DDPG reward function that can learn from the actions of an off-line non-causal optimum scheduling algorithm from last chapter. The new reward function can guide the learned strategy to achieve peak shaving and load shifting through a balanced process of exploration and exploitation. The newly proposed algorithm can minimize overall energy cost through online scheduling of BESS systems by learning the behaviors of renewable energy generation and grid operations.

4.1 Problem formulation

The same system in Fig. 3.1 is considered for this problem. However, the problem formulation needs to be slightly changed.

4.1.1 Battery Model

Based on the proof in the last chapter, the two action variables q_t^c and q_t^d are replaced by a single variable q_t represents the amount of energy change due to battery charging or discharging at time slot t, with $q_t > 0$ for charging and $q_t < 0$ for discharging. The battery dynamics can be described by the following difference equation,

$$c_{t+1} = \begin{cases} c_t + q_t\gamma_e, & q_t > 0, \\ c_t, & q_t = 0, \\ c_t + q_t/\gamma_e, & q_t < 0, \end{cases} \tag{4.1}$$

then the constraint on charging and discharging is changed correspondingly to:

$$-n_b q^{\min} \leq q_t \leq n_b q^{\max} \tag{4.2}$$

where $q^{\min}$ and $q^{\max}$ are the maximum discharging and charging rate. The SOC constraint remains the same:

$$0 \le c_t \le n_b c^{\max}[1 - \alpha \cdot (m-1)^{0.75} - \beta\sqrt{m-1}] \tag{4.3}$$

4.1.2 Power from the grid

Now we have the energy bought from the utility:

$$q_t^{\text{net}} = q_t^{\text{ld}} - q_t^{\text{sol}} + q_t, \tag{4.4}$$

4.1.3 Objective function

The objective of this problem is changed to minimize the expected monthly electricity bill by identifying the optimum charging and discharging schedule based on stochastic load and PV information in a given month.

With the same Time-Of-Use (TOU) rate plan and time division as the last chapter, the definitions of the energy charge and demand charge are restricted to the m-th month. The energy charge with the given TOU in the m-th month is:

$$C_{\text{E}} = \sum_{t \in \mathcal{H}_m} p_t q_t^{\text{net}} \tag{4.5}$$

where $\mathcal{H}_m$ is the time slots in the m-th month. The demand charge in the m-th month can be calculated as:

$$C_{\text{D}} = D \max_{t \in \mathcal{H}_m} q_t^{\text{net}} \tag{4.6}$$

where D is the unit demand charge in the unit of dollar per kWh.

The optimization problem can be formulated as:

$$\begin{aligned} \min_{\{q_t\}_{t \in \mathcal{H}_m}} \ & \mathbb{E}(C_{\text{E}} + C_{\text{D}}) = \mathbb{E}\left[\sum_{t \in \mathcal{H}_m} p_t q_t^{\text{net}} + D \max_{t \in \mathcal{H}_m} q_t^{\text{net}}\right], \\ \text{s.t.} \quad & (4.1) - (4.4), \end{aligned} \tag{P1}$$

where the expectation is performed with respect to the distributions of the load and PV information. Given the stochastic nature of the load and PV, the problem will be formulated under the framework of Markov decision process (MDP) in this section. The MDP formulation enables the RL-based solution to be presented in the next section.

4.2 Deep Deterministic Policy Gradient approach

4.2.1 Markov Decision Process

An MDP is a discrete time stochastic control process that consists of a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of possible actions, $\mathcal{P} = \{\mathbf{P}(s_{t+1}|s_t, a_t)\}$ is the set of state transition probabilities, with $\mathbf{P}(s_{t+1}|s_t, a_t)$ being the probability of transitioning from state $s_t \in \mathcal{S}$ to state $s_{t+1} \in \mathcal{S}$ with action $a_t \in \mathcal{A}$, $r_t = r(s_t, a_t) \in \mathcal{R}$ is the immediate scalar reward or cost by applying action $a_t \in \mathcal{A}$ to state $s_t \in \mathcal{S}$, and $0 < \gamma \leq 1$ is the discount factor. The learner or the decision maker is called the agent, and everything beside the agent in the system is called the environment E. Based on the system model of the PV-assisted energy storage system, we can formulate the problem under the MDP framework as follows.

State Space $\mathcal{S}$

The state at time window t includes the hourly index in a day $h_t \in \mathcal{H}$, the load q_t^{ld}, the solar energy q_t^{sol}, the state of charge c_t. Thus $s_t = \{h_t, q_t^{\mathrm{ld}}, q_t^{\mathrm{sol}}, c_t\}$. The TOU rate p_t is not included here because p_t is a deterministic function of the hour index t. The system has causal knowledge of q_t^{ld} and q_t^{sol}, but their future values are unknown.

Action Space $\mathcal{A}$

The actions or decisions that can be made at t is the battery charging/discharging amount q_t. The possible actions should always meet the constraint in (**??**). MDP with constraints can be modeled as a constrained MDP and solved by adding a safety layer as [134]. A similar method is adopted in the RL-based solution proposed in this paper and details are given in the next section.

Transition Probability $\mathcal{P}$

Given the current state $s_t = \{h_t, q_t^{\mathrm{ld}}, q_t^{\mathrm{sol}}, c_t\}$, the action $a_t = q_t$ only affects the transition of the state-of-charge c_t. Other transitions of load and solar energy are unknown and are affected by weather in the real world. Specifically, the transition of the hour index is deterministic as follows

$$h_{t+1} = \mod(h_t + 1, 24) \tag{4.7}$$

where mod stands for the modulo operator. The transition of state of charge is described in the battery dynamics in (4.1). To deal with the unknown transition probability of load and solar energy, one method is to discretize the possible states, and obtain the corresponding transition probabilities of discrete states by using historical data. However, the method in [135] cannot deal with continuous states such as the solar and load data used in this paper. Instead of explicitly identifying the transition probabilities, the problem can also be solved by using RL, where the agent can interact with the environment and implicitly learn the transition probability from past experiences. The MDP formulation presented in this section can also be applied to RL algorithms.

4.2.2 Reinforcement Learning

Policy and Reward

The agent chooses actions based on a policy. A stochastic policy, $\pi(s_t) = \{\mathbf{P}(a_t|s_t)|a_t \in \mathcal{A}\}$, defines the probability distribution of choosing different actions from the current state $s_t \in \mathcal{S}$. A deterministic policy function, denoted by $\mu(s_t)$, specifies the action to choose when in state s_t, that is, $a_t = \mu(s_t)$. The deterministic policy is adopted in this paper such that a certain action q_t can be taken given the current system state $s_t = \{h_t, q_t^{\text{ld}}, q_t^{\text{sol}}, c_t\}$.

Define the long-term discounted reward starting from time t as

$$R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i). \tag{4.8}$$

The discounted reward R_t is a random variable, the distribution of which depends on the stochastic nature of the environment and policy. The general objective of MDP is to find the optimal policy π^* or μ^* that will maximize the long-term expected discounted reward over an infinite time horizon:

$$\bar{R}_0 = \sum_{t=0}^{\infty} \mathbb{E}\left[\gamma^t r(s_t, a_t)\right], \tag{4.9}$$

where the expectation is performed with respect to both the environment and policy. For a system with a deterministic policy, the expectation in (4.9) is performed with respect to the environment only.

An episodic MDP is considered in this paper since the agent-environment interaction breaks naturally into months. Episodic MDP can be regarded as a special case of MDP, and it considers episode termination as an absorbing state that transits only to itself with zero rewards [136].

Next we will reformulate the objective function in (P1) in the form of an episodic MDP. Define $\phi_t = \max(q_t^{\text{net}}, \phi_{t-1})$ with $\phi_0 = 0$. The long-term expected reward in (4.9) with $\gamma = 1$ can be expressed as

$$\bar{R}_0 = \sum_{t \in \mathcal{H}_m} \mathbb{E}(r_t), \tag{4.10}$$

where

$$r_t = -\left[p_t q_t^{\text{net}} + D \cdot (\phi_t - \phi_{t-1})\right]. \tag{4.11}$$

In regular MDP, setting $\gamma < 1$ is used to ensure the convergence of the expected long term reward over an infinite horizon. In episodic MDP the reward is calculated over a finite horizon, thus we can set $\gamma = 1$. In this case, the expected reward is the same as the expected electricity bill. All the following discussions are still applicable to $\gamma < 1$.

Reward functions of RL systems are often carefully designed to achieve a balanced tradeoff between exploration and exploitation. The original reward function under the MDP framework might not be suitable for this role. Detailed reward function design for RL will be discussed in the next section.

Action-Value functions

The general action-value function used in MDP and RL is the expected discounted return after taking an action a_t from the state s_t by following a policy and reward. Both the policy and the reward could be stochastic.

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_{i \geq t}, s_{i>t} \sim E, a_{i>t} \sim \pi}\left[R_t | (s_t, a_t)\right] \tag{4.12}$$

where the expectations are performed with respect to $r_{i \geq t}, s_{i>t}, a_{i>t}$. The current and future reward and the future state depend on the environment, and the future action depends on the policy. R_t is the long-term discounted reward starting from t as defined in (5.36). Denoting expectations with respect to $r_t, s_t \sim E, a_t \sim \pi$ as $\mathbb{E}_{r_t, s_t, a_t}$, the action-value function in (4.12) can be expressed recursively as the Bellman equation,

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1}}\left[r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1}} Q^\pi(s_{t+1}, a_{t+1})\right] \tag{4.13}$$

The detailed derivation of (4.13) is given in the Appendix A.

For a system with a deterministic policy μ and a deterministic reward function $r(s_t, a_t)$, the action-value function, or Q function, is:

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{s_{i>t}}\left[R_t | s_t, \mu(s_t)\right], \tag{4.14}$$

where the expectations are performed with respect to the future states, $s_{i>t}$. The future actions, current and future rewards are all determined by the future states. The action-value function in (4.14) can be expressed in the recursive Bellman equation form as,

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1}))\right]. \tag{4.15}$$

If the optimal action-value function $Q^*(s, a)$ is known, then the optimal policy can be obtained as,

$$\mu^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a). \tag{4.16}$$

On the other hand, for a given optimal policy μ^*, the optimal action-value function $Q^*(s, a)$ can be obtained by applying the optimal policy to the action-value function. In practice, neither the optimal policy nor the optimal action-value function is known, thus they need to be estimated. Given the complexity of the problem formulated in this paper, it is difficult, if not impossible, to directly solve the Bellman equation either analytically or numerically.

To solve the Bellman equation, most MDP algorithms discretize the state spaces into finite states. For example, MDP with policy iteration updates to a better policy in each iteration by exhaustively searching and evaluating all possible policies (actions) along the state transition trellis [137]. MDP with value iteration is a special case of policy iteration where the policy is evaluated only once in one sweep [136]. Both algorithms suffer from the curse of dimensionality and incur prohibitive complexities when the size of the state space and/or action space is large. In addition, the discretization of the state or action space leads to loss in model accuracy.

4.2.3 Deep Q-learning Network

Q-learning is an RL algorithm that can learn the policy and update the Q function in an iterative manner. It does not need an explicit model of the environment, such as the transition probabilities in the MDP formulation. Since the expectation in the action-value function in (4.15) is not taken with respect to the policy, the optimal deterministic policy μ^* can be learned by transitions obtained by following other policies, i.e. Q-learning is an off-policy algorithm [136].

Q-learning uses an ε-greedy policy. At the beginning of the learning process, the action-value function $Q(s_0, a_0)$ can be initialized to an arbitrary value. Then the action-value function in the t-th iteration can be updated as:

$$Q(s_t, a_t) \leftarrow (1-\varepsilon)Q(s_t, a_t) + \varepsilon[r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})], \tag{4.17}$$

where $\varepsilon \in [0, 1]$ is the learning rate.

In Q-learning, the action is chosen to maximize the estimated expected discounted return for a given state s as

$$\lambda(s) = \arg\max_{a \in \mathcal{A}} Q(s, a). \tag{4.18}$$

Simple Q-learning approximates the action-value function by a lookup table (LUT). However, the LUT-based Q-learning method cannot be applied to complex environments because large non-linear approximators make the algorithm unstable. Deep Q-learning Network (DQN) uses the deep neural network (DNN) to estimate $Q(s, a)$, which can be parameterized as the DNN weight coefficients θ^Q. The parameterized Q-function is updated to the target value as

$$y_t = r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}|\theta^Q). \tag{4.19}$$

then the parameters θ^Q are learned by minimizing the following loss function:

$$L(\theta^Q) = \mathbb{E}_{s_{t+1}} \left[\left(Q(s_t, \lambda(s_t)|\theta^Q) - y_t \right)^2 \right]. \tag{4.20}$$

The expectation operation in the above equation can be approximated by using a size-$|\mathcal{R}|$ replay buffer $\mathcal{R}$, which stores the previous $|\mathcal{R}|$ transitions before the current time slot t, that is $(s_i, a_i, r_i, s_{i+1})_{i \in \{t-|\mathcal{R}|+1, \ldots, t-1, t\}}$. During the time slot t, a mini-batch $\mathcal{M} \subset \mathcal{R}$ with $|\mathcal{M}|$ transitions $(s_i, a_i, r_i, s_{i+1})_{i \in I(\mathcal{M})}$ are randomly sampled from $\mathcal{R}$ by following a uniform distribution without replacement, where $I(\mathcal{M})$ is the time index of transitions in the mini-batch $\mathcal{M}$. The expectation in (4.20) can then be approximated by averaging over the mini-batch $\mathcal{M}$. Then the weights are updated to minimize the following loss function,

$$\theta^Q \leftarrow \theta^Q + l \nabla_{\theta^Q} \frac{1}{|\mathcal{M}|} \sum_{i \in I(\mathcal{M})} \left(Q(s_i, a_i|\theta^Q) - y_i \right)^2, \tag{4.21}$$

where l is the learning rate of the DQN agent.

In addition to the DNN with parameter θ^Q, a target network parametrized by $\theta^{Q'}$ is updated every k iterations. This prevents the network from propagating too fast and reduces the risk of divergence since the target values are kept for k iterations.

Despite the success of DQN, it still suffers from the curse of dimensionality because of the discrete action space, where the action in each iteration is selected by exhaustively searching all possible actions. For systems with continuous actions, the action space has to be discretized in DQN to approximate the continuous actions, and this results in a loss of precision. We propose to solve this problem by adopting the DDPG approach, which can operate directly on a continuous action space.

4.2.4 Deep Deterministic Policy Gradient

We propose to use the DDPG learning method to solve the ESS scheduling problem, which has a continuous action space $\mathcal{A}$. Instead of using a greedy policy that requires a global maximization for continuous action space, the DDPG algorithm uses a policy gradient approach to search along the gradient of a policy-dependent value function with respect to the policy.

Two DNNs are employed in DDPG, and they are denoted as actor and critic networks, respectively. The actor network is a DNN used to model the deterministic policy μ for a given state s as $\mu(s|\theta^\mu)$, where the parameter θ^μ represents the weights of the actor network. The critic network is a DNN used to model the Q-function as $Q(s,a|\theta^Q)$ with parameter θ^Q. With the actor-critic pair, define the policy-dependent value function with respect to a policy μ as:

$$\begin{aligned} J(\mu) &= \int_{\mathcal{S}} \rho^\mu(s) Q(s, \mu(s|\theta^\mu)|\theta^Q) ds \\ &= \mathbb{E}_{s\sim\rho^\mu}\left[Q(s,\mu(s|\theta^\mu)|\theta^Q)\right], \end{aligned} \tag{4.22}$$

where $\mathcal{S}$ is the state space, ρ^μ is the state distribution under policy μ. The policy-dependent value function in (4.22) is parameterized by θ^μ instead of depending on the action. The parameters θ^μ can thus be updated by searching along the gradient of $J(\mu)$ with respect to θ^μ as [138],

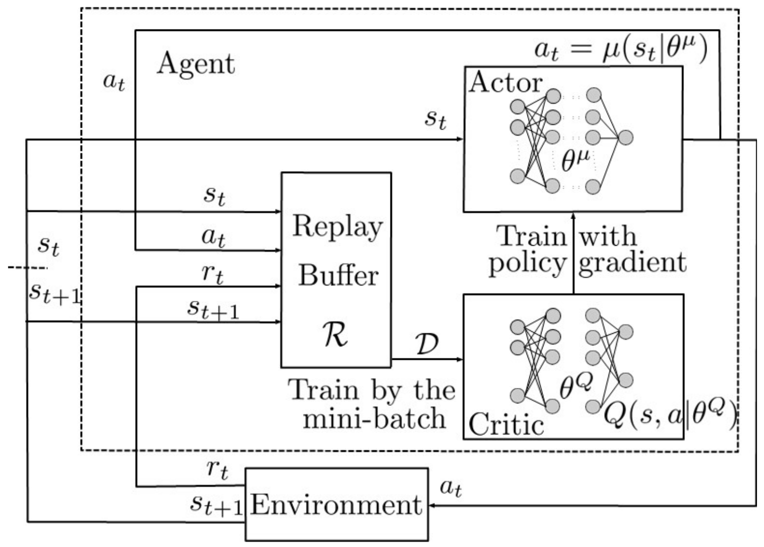


Figure 4.1: The training process of the actor and critic networks in DDPG.

$$\begin{aligned}\nabla_{\theta^\mu} J &\approx \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_{\theta^\mu} Q(s,a|\theta^Q)|_{a=\mu(s|\theta^\mu)}\right] \\ &= \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_a Q(s,a|\theta^Q)|_{a=\mu(s)}\nabla_{\theta^\mu}\mu(s|\theta^\mu)\right].\end{aligned} \tag{4.23}$$

In the equation above, we need to calculate the expected gradients of both the actor and critic networks. The expectation operation can be approximated by using the replay buffer $\mathcal{R}$ with mini-batch $\mathcal{M}$ as described in the previous subsection. The training process of the actor and critic networks in DDPG with replay buffer is shown in Fig. 5.3. In addition to the two networks in the figure, two target networks Q' and μ' with parameters $\theta^{Q'}$ and $\theta^{\mu'}$ are designed for the actor and critic networks, respectively. The target networks are used to prevent the networks from propagating too fast.

The critic network used to model $Q(s,a|\theta^Q)$ is similar to the DNN used in the DQN, and it evaluates how good an action is. Similar to the DQN target value in (4.19), the target critic value for the i-th transition in $\mathcal{M}$ under the current target actor is calculated by

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}\right),\ i \in I(\mathcal{M}). \tag{4.24}$$

Then the critic network can be updated by minimizing the mean squared error between the target critic value and the current critic value as

$$\theta^Q \leftarrow \theta^Q + l_c \nabla_{\theta^Q} \frac{1}{|\mathcal{M}|}\sum_{i\in I(\mathcal{M})}\left(Q(s_i,\mu(s_i|\theta^\mu)|\theta^Q) - y_i\right)^2, \tag{4.25}$$

where l_c is the learning rate of the critic network.

The actor network is used to model the action $\mu(s|\theta^\mu)$. The actor parameters θ^μ can be updated by following the policy gradient in (5.47). In order to reduce computation complexity, most practical DDPG implementations update the actor parameters by maximizing the action-value function averaged over the mini-batch $\mathcal{M}$ as

$$\theta^\mu \leftarrow \theta^\mu + l_a \nabla_{\theta^\mu} \frac{1}{|\mathcal{M}|} \sum_{i \in I(\mathcal{M})} \big(Q(s_i, \mu(s_i|\theta^\mu)|\theta^Q\big) \tag{4.26}$$

where l_a is the learning rate of the actor network. The two target networks are updated softly with parameter $\epsilon \ll 1$ by:

$$\begin{aligned} \theta^{Q'} &\leftarrow \epsilon\theta^Q + (1-\epsilon)\theta^{Q'}, \\ \theta^{\mu'} &\leftarrow \epsilon\theta^\mu + (1-\epsilon)\theta^{\mu'}. \end{aligned} \tag{4.27}$$

This update process slows down the update of the networks thus improves the stability of learning.

In addition, to ensure better exploration in a continuous action space, noise is added during action selection. In this paper, zero mean additive Gaussian noise is applied to the current policy parameter θ^μ, where the noise variance is adapted according to the distance measure between the original policy and noisy policy [139]. Details of the DDPG training process are given in Algorithm 6. The training process is divided into episodes, where each episode corresponds to one day with $T = 24$ time slots. The day in each episode is uniformly randomly chosen from the set of all summer or winter days. Summer and winter models are trained separately.

As discussed in Section 4.2.1, the actions performed in DQN or DDPG should meet the constraint in (4.2), so the actions are first bounded to safe values then used for reward calculation.

Since the output layer of DDPG is a tanh layer with output ranging from -1 to 1, the output $a_t(q_t)$ is first scaled up to $-n_b q^{\min}$ to $n_b q^{\max}$ to meet the constraint in (4.2). Then it is bounded to meet the constraint in (4.3) as:

$$\max\left\{[c_t - n_b c(m)]\gamma_e, -n_b q^{\min}\right\} \leq q_t \leq \min\left\{[n_b c(m) - c_t]/\gamma_e, n_b q^{\max}\right\}. \tag{4.28}$$

Algorithm 3 DDPG algorithm training

```
Require: initial actor parameters $\theta^\mu$, critic parameters $\theta^Q$, empty replay buffer $\mathcal{R}$, number of episodes $M$
 1: Initialization: Set target network parameters equal to main parameter: $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
 2: for episode = 1 to $M$ do
 3:     Initial $s_1$ for the first hour of the day.
 4:     for $t = 1$ to $T$ do
 5:         Output action from actor network $a_t = \mu(s_t|\theta^\mu)$
 6:         Execute action $a_t$ in the environment and observe reward $r_t$ and new state $s_{t+1}$
 7:         Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{R}$
 8:         Randomly sample a mini-batch $\mathcal{M}$ of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $\mathcal{R}$
 9:         Compute targets for all elements in the mini-batch $\mathcal{M}$:
                $$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}\right)$$
10:         Update the critic parameters $\theta^Q$ by one step gradient descent of the critic loss function in (6.29):
                $$\theta^Q \leftarrow \theta^Q + l_c \nabla_{\theta^Q} \frac{1}{|\mathcal{M}|} \sum_{i \in I(\mathcal{M})} \left(Q(s_i, \mu(s_i|\theta^\mu)|\theta^Q) - y_i\right)^2$$
11:         Update the actor parameters $\theta^\mu$ by one step gradient ascent of the actor value function in (6.27):
                $$\theta^\mu \leftarrow \theta^\mu + l_a \nabla_{\theta^\mu} \frac{1}{|\mathcal{M}|} \sum_{i \in I(\mathcal{M})} \left(Q(s_i, \mu(s_i|\theta^\mu)|\theta^Q)\right)$$
12:         Update the target networks with (6.30):
                $$\begin{aligned} \theta^{Q^{'}} &\leftarrow \epsilon\theta^Q + (1-\epsilon)\theta^{Q^{'}}, \\ \theta^{\mu^{'}} &\leftarrow \epsilon\theta^\mu + (1-\epsilon)\theta^{\mu^{'}}. \end{aligned} \tag{4.29}$$
13:     end for
14: end for
Ensure: Target actor parameters $\theta^{\mu'}$
```

Table 4.1: Desired actions for training

Scenarios	**Desired actions**
$t \in \mathcal{H}_{\text{pk}}$	Discharge to minimize $q_t^{\text{net}} \geq 0$.
$t \notin \mathcal{H}_{\text{pk}}$	Charge or discharge to minimize $\|q_t^{\text{net}} - q^{\max}\|$.

Reward Function Design

In order to solve (P1) using DQN or DDPG, we still need to make some modifications to the reward function. In the original reward function defined in (4.11), the reward is calculated based on the real month environment, that is, the demand charge in (P1) is calculated by using the peak load from all days within the same month. On the other hand, during the DDPG training process, the actor network is updated by using days randomly drawn from the replay buffer $\mathcal{R}$ as in (6.27). Such a random approach might not be able to learn key scheduling activities such as peak shaving during peak hours or load shifting during off-peak hours. For example, if too many off-peak hour samples are drawn to the mini-batch, then the actor network will not be able to fully explore the behavior of the environment during peak hours. As a result, the action learned during the training process might fail to recognize the importance of peak shaving needed during peak hours.

We propose to solve the above problem by designing a new reward function that can guide the learned strategy to balance the actions of peak shaving and load shifting. The new reward function is developed with the assistance of an off-line optimum nonlinear programming [33], which can obtain the optimum energy scheduling strategy in a month based on the training data. It should be noted that the off-line optimum algorithm is non-causal given that it requires the data from one entire month, thus it cannot be applied to practical systems. However, it provides the best achievable performance that can be used as a baseline. Denote the peak energy bought from the utility in a month under the optimal schedule as $q^{\max}$.

The reward function is designed by considering two different scenarios as shown in Table 4.1.

- Scenario 1: $t \in \mathcal{H}_{\text{pk}}$

During peak hour in summer, load shifting can be performed by discharging the battery to reduce $q_t^{\text{net}} \geq 0$ as much as possible to save on energy cost. The optimum energy scheduling and the corresponding step reward are

$$q_{t1}^{\text{opt}} = -\min\{q_t^{\text{ld}} - q_t^{\text{sol}}, n_b q^{\max}\} \tag{4.30}$$

In order to train the DQN and DDPG agents to learn this strategy, the step reward function for this scenario is defined as

$$r_t = 1 - \zeta_1 \frac{|q_t - q_{t1}^{\text{opt}}|}{n_b q^{\max}}, \tag{4.31}$$

where ζ_1 is a weight used to adjust the learning rate for scenario 1.

- Scenario 2: $t \notin \mathcal{H}_{\text{pk}}$

  During part-peak and off-peak hours in both summer and winter, the current energy demand should be close to the peak energy bought from the utility in the optimum off-line scheduling algorithm as much as possible. In this case, peak shaving and load shifting are performed by charging or discharging the battery. The optimum energy scheduling in this case is

$$q_{t2}^{\text{opt}} = \max\{\min\{q_{\max}^{\text{net}} - (q_t^{\text{ld}} - q_t^{\text{sol}}), n_b q^{\max}\}, -n_b q^{\max}\}, \tag{4.32}$$

$$r_t = 1 - \zeta_2 \frac{|q_t - q_{t2}^{\text{opt}}|}{n_b q^{\max}}, \tag{4.33}$$

  where ζ_2 is a weight used to adjust the learning rate for scenario 2.

The coefficients, ζ_1, ζ_2, can be adjusted to balance the actions of peak shaving and load shifting.

4.3 Case studies

Case studies with real world data are presented in this section to illustrate the performance of the proposed battery scheduling algorithm. All simulations are performed on a workstation with a 6-core Intel Core i7-5820K CPU operating at 3.3 GHz, NVIDIA GeForce GTX 950 GPU, and 32 GB of random access memory (RAM).

4.3.1 Data System Setup

The learning and testing of the scheduling algorithms are performed by using real world load data from four locations around the campus of the University of Arkansas

Table 4.2: Battery and solar parameters

Notation	Description	Value
$\{n_b, n_s\}$	Size of batteries and solar panels	5500,82
$c^{\max}$	Capacity of a single battery (kWh)	13.5
$q^{\max}, q^{\min}$	Maximum charging/discharging rate (kW)	5
γ_e	Energy efficiency of the battery	94%
γ_s	Efficiency of the solar panel	99.96%
α	Calendar aging coefficient	0.0036
β	Cycling aging coefficient	0.0155

(UARK), Fayetteville. The data include load information collected in 15-minute intervals over a period of 21 months, spanning from January, 2016 to September, 2017. In this paper, the 15-minute interval data are converted to hourly sampled data by adding the 4 samples within an hour. The data in 2016 are used as training set to learn the pattern and the policy, and the data in 2017 are used for testing. Specifically, two DDPG agents are trained for winter and summer months, respectively. Similarly, two DQN agents are also trained for winter and summer for comparison purposes.

The PV energy data are obtained from solar panels installed on the roof of Fayetteville Public Library [140], which provides hourly solar energy collected from the panels of a total capacity of 13.5 kW DC. Solar data from the same time period as the load data are used in the simulation. To match the surface of the installation site, the solar data are scaled to $N_s = 82$ solar modules with 10 kW DC capacity each.

The batteries are modeled by using Tesla Powerwall 2. The parameters of the batteries are set based on the battery datasheet [141]. The parameters for both batteries and solar planels are given in Table 4.2.

The utility charges are calculated by using the TOU rate (p_t, D) of Pacific Gas and Electric Company in Table 3.1, along with the time division in Table 3.2 [142].

The proposed scheduling algorithm with DDPG is compared to several baseline situations with details given as follows.

1. PV-only system with no batteries. This scenarios is used to benchmark the performance gain that can be achieved by employing BESS in the PV system.

Table 4.3: DQN and DDPG parameters

Notation	Description	Value
L	Number of DQN discretization level	128
	Number of neurons in DQN hidden layers	128,128
l	DQN learning rate	0.0005
k	Target network update frequency	500
	Number of neurons in DDPG hidden layers	128,128
l_a	DDPG actor learning rate	0.00001
l_c	DDPG critic learning rate	0.0001
ϵ	DDPG soft update coefficient	0.001
$\lvert\mathcal{R}\rvert$	Experience replay buffer size	50000
$\lvert\mathcal{M}\rvert$	Batch size	128
M	Number of episodes trained	1000
T	Episode length	24
ζ_1, ζ_2	Reward weight parameters	20,100

2. Off-line nonlinear programming (NLP). This is an optimum but non-causal algorithm that requires knowledge of the data for an entire month in order to achieve optimum scheduling for the same month. The nonlinear part comes from the maximum term in the demand charge in (4.6). The nonlinear programming problem is solved by using the algorithm presented in our previous work [143]. The results serve as the best achievable performance for a ESS-assisted PV system.

3. DQN based scheduling. The DQN algorithm is implemented for performance comparison under the framework of RL.The DQN has a discrete action space. If the lower and upper bound in (4.2) are denoted as $q_t^{\min}$ and $q_t^{\max}$, then the charge/discharge are discretized into $L = 128$ levels between $q_t^{\min}$ and $q_t^{\max}$. Other parameters are given in Table 4.3. The DQN algorithm and DDPG are both implemented by using Stable Baseline [144].

4.3.2 DDPG and DQN Training and Testing

The DDPG and DQN training parameters are given in Table 4.3. In DDPG, both the actor and critic networks are equipped with two hidden layers with 128 neurons each to match the DQN network. To ensure a fair comparison between DDPG and DQN, both networks share the same reward functions and system parameters, such as the sizes of replay buffers and batches.

The DDPG testing algorithm is given in Algorithm 7. The testing is performed by applying the target actor parameter $\theta^{\mu'}$ obtained from DDPG training. The reward function is calculated by using the step reward defined in (4.11). The testing of the DQN algorithm follows a similar procedure.

Algorithm 4 DDPG testing: Online ESS schedule

```
1: Initialization: $t \leftarrow 0$, obtain the current state $s_t = \{h_t, q_t^{\text{ld}}, q_t^{\text{sol}}, c_t\}$ as the beginning
   state.
2: while $t \in \mathcal{H}_m$ do
3:     $t \leftarrow t + 1$
4:     Obtain new observation and calculate the SOC to get $s(t+1)$.
5:     Output the schedule $q(t)$ based on target actor in Algorithm 1 by: $q(t) = \mu(s_t|\theta^{\mu'})$
6:     Calculate the reward by (4.11)
7: end while
```

4.3.3 Testing Results

Fig. 4.2 shows the amount of energy bought from the utility during the first week of January, which is the peak week in the month, under various scheduling scenarios. The corresponding battery SOC is shown in Fig. 4.3. Under the PV-only scenario, there are two peaks at the 136th and 154th hours. The NLP algorithm achieves the optimum off-line scheduling by using data from the entire month, and it successfully shaved those two peaks by charging the batteries to a higher capacity before the occurrences of the peaks. The DQN algorithm learns to shift the load by charging the batteries when the demand is low, but it fails to shave the two peaks. The DDPG algorithm succeeds in both load shifting and peak shaving. The peak of the DDPG energy curve is 7998.75 kWh at 119th hour, which is only 17.47% above the peak of the off-line NLP energy curve.

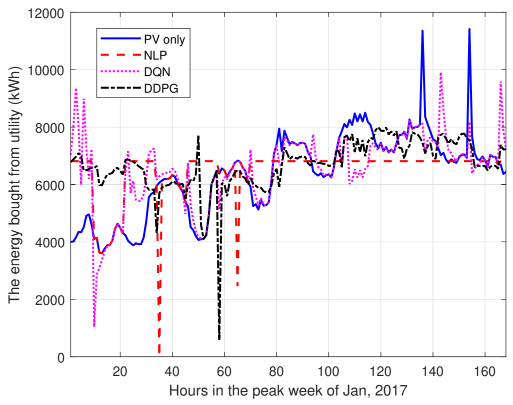


Figure 4.2: The energy bought from the utility on the first week of January, 2017.

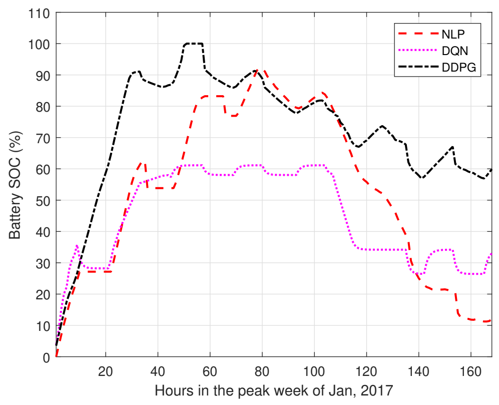


Figure 4.3: The SOC of the battery on the first week of January, 2017.

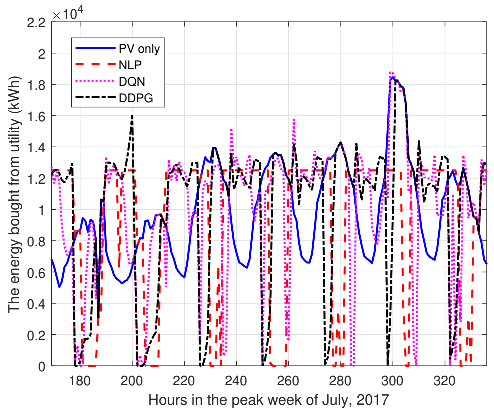


Figure 4.4: The energy bought from the utility on the second week of July, 2017.

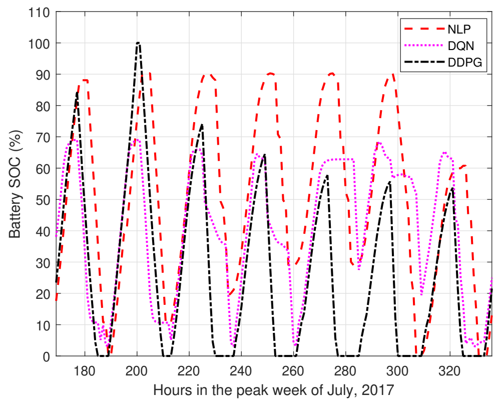


Figure 4.5: The SOC of the battery on the second week of July, 2017.

Table 4.4: Monthly Bill in 2017

Month	PV-only	NLP	DQN	DDPG
Jan	678,514	603,396	671,340	629,614
Feb	1,035,429	968,656	1,036,145	1,018,625
Mar	1,039,776	975,610	1,042,461	1,030,354
Apr	834,980	750,513	835,127	822,661
May	851,963	705,821	852,864	810,310
Jun	936,471	804,855	921,106	900,875
Jul	1,142,395	965,615	1,123,076	1,099,311
Aug	1,117,117	983,810	1,102,033	1,093,230
Sept	1,115,512	905,158	1,095,195	1,059,665
Total	8,752,158	7,663,434	8,679,347	8,464,645
Save	-	12.44%	0.83%	3.29%

As can be seen from the battery SOC shown in Fig. 4.3, both the DQN and DDPG agents learn to charge before the occurrences of peak days, but the DQN algorithm does not charge enough to shave the peaks. The behaviors of DQN are similar to that of NLP, but it discharges at a relatively slower rate compared to NLP.

Figs. 4.4 and 4.5 show, respectively, the total energy bought from the utility and the corresponding battery SOC during the second week of July 2017, which is the peak week in this summer month. The NLP algorithm shaves the peak around the 300th hour, and it shifts the load from peak hours to off-peak hours by charging and discharging the batteries on a daily basis. Both DDPG and DQN learn the strategy of daily load shifting. However, neither can shave the peak around the 300th hour. This is mainly due to the behavior of daily batter cycle in summer, which does not leave enough energy needed for potential peak shaving in the next day.

The monthly utility bill during the first nine months in 2017 under different scheduling scenarios are shown in Table 4.4. The non-causal NLP algorithm has the best performance as expected. Compared to the PV-only case, the DQN algorithm has lower bills in summer months, but its performance is worse in the winter months. The DDPG algorithm outperforms the DQN algorithm in every month.

4.4 Conclusion

This paper presents a new scheduling algorithm for BESS-assisted PV systems by using reinforcement learning with DDPG. Based on the unique behaviors of energy systems, a new reward function has been designed to guide the learning process of exploration and exploitation in DDPG. The new reward function can balance the actions of peak shaving and load shifting that are critical for energy system scheduling. Compared with other online scheduling methods, the proposed DDPG-based scheduling approach does not require an explicit model of the environment, and it can deal with continuous state and action spaces. Case studies with real world data show that the proposed scheduling algorithm outperforms exiting DQN-based algorithms, and it can learn the behaviors of the off-line NLP algorithm.

5 Intelligent Optimal Power Flow Control for Wind-Powered Microgrid with Deep Reinforcement Learning

Previous chapters discuss the investment plan and on-line schedule of the ESS-assisted PV system for consumers. Compared with solar panels that can be installed at the consumer's end, the wind turbines (WT) are integrated to the grid and controlled by the utilities. It demands a sophisticated control mechanism that can optimize the real-time operations of power generation and ESS charging/discharging schedules.

Traditional economic dispatch (ED) and optimal power flow (OPF) controllers are designed to minimize the grid operation cost by optimizing power generation schedules at each generator. Conventional OPF is usually performed over a single time period, e.g., an hour, without the need to consider the dynamics of the power grid over a longer time horizon. However, the addition of ESS such as battery, hydrogen fuel cells, or electrical vehicles, makes it necessary to consider the storage dynamics over a longer time period [9]. It is shown in [10] that for power systems with ESS, multi-period OPF is more economical than its single-period counterpart. The complexity of multi-period OPF increases exponentially with the time duration. Thus it might not be suitable for power systems with renewable energy sources (RES), which require real-time decision on the generation and storage schedule given the real-time uncertainty of RES.

In this chapter, we develop an intelligent RT-OPF control scheme for a microgrid equipped with wind power generators and ESS. The DDPG algorithm will be used to solve the ESS scheduling problem and interact with the environment where the OPF will be calculated. Both the critic and actor networks are trained through iterative interactions with historical data. The trained agent can then be applied for real time operation controls without the need of multi-period prediction or multi-period optimization as in conventional methods. The performance of the proposed DDPG solution is compared to those obtained from a classic Model Predictive Control (MPC)-based approach. Compared with MPC, the proposed DDGP approach can achieve a better performance in terms of both lower operation cost and lower complexity that can ensure its real-time operation.

5.1 Problem formulation

Power flow studies are employed by transmission and utility companies and are of extreme importance in transmission expansion planning (TEP), operation, and control.

5.1.1 Single-period Optimal Power Flow

Basics of AC circuit analysis

1. Single phase in vector form:

   (a) Voltage: $\boldsymbol{V} = |\boldsymbol{V}|\angle\delta_v$

   (b) Current: $\boldsymbol{I} = |\boldsymbol{I}|\angle\delta_i$

2. Volt-ampere characteristics:

   (a) Basics described in the table below, where ω is the angle frequency.

Type of elements	Time domain	Vector form
Resistor	$v(t) = Ri(t)$	$\boldsymbol{V} = R\boldsymbol{I}$
Inductor	$v(t) = L\frac{di(t)}{dt}$	$\boldsymbol{V} = j\omega L\boldsymbol{I}$ $(X_L = \omega L)$
Capacitor	$i(t) = C\frac{dv(t)}{dt}$	$\boldsymbol{V} = -j\frac{1}{\omega C}\boldsymbol{I}$ $(X_C = -\frac{1}{\omega C})$

   (b) Impedance (Ω): $\boldsymbol{Z} = R + jX$, $X = X_L + X_C = \omega L - \frac{1}{\omega C}$

   (c) Admittance (S): $\boldsymbol{Y} = \frac{1}{\boldsymbol{Z}} = \frac{1}{R+jX}$

   (d) Conductance: $G = Re\{\boldsymbol{Y}\} = \frac{R}{R^2+X^2}$

   (e) Susceptance: $B = Im\{\boldsymbol{Y}\} = -\frac{X}{R^2+X^2}$

3. Power analysis:

Type of power	1 Phase	3 Phase
Active power (W)	$P = \|\boldsymbol{V}\|\|\boldsymbol{I}\|\cos(\delta_v - \delta_i)$	$P = \sqrt{3}\|\boldsymbol{V}\|\|\boldsymbol{I}\|\cos(\delta_v - \delta_i)$
Reactive power (VAR)	$Q = \|\boldsymbol{V}\|\|\boldsymbol{I}\|\sin(\delta_v - \delta_i)$	$Q = \sqrt{3}\|\boldsymbol{V}\|\|\boldsymbol{I}\|\sin(\delta_v - \delta_i)$
Apparent power (VA)	$S = \|\boldsymbol{V}\|\|\boldsymbol{I}\| = \sqrt{P^2 + Q^2}$	$S = 3\|\boldsymbol{V}\|\|\boldsymbol{I}\| = \sqrt{P^2 + Q^2}$

Per-unit presentation

Assuming that the independent base values are apparent power and voltage magnitude, we have:

$$S_{\text{base}} = 1 \text{ pu}, V_{\text{base}} = 1 \text{ pu}$$

then from the equations in A 3), the rest of the units can be derived:

$$\begin{aligned} I_{\text{base}} &= \frac{S_{\text{base}}}{V_{\text{base}}} = 1 \text{ pu} \\ Z_{\text{base}} &= \frac{V_{\text{base}}^2}{S_{\text{base}}} = 1 \text{ pu} \\ Y_{\text{base}} &= \frac{1}{Z_{\text{base}}} = 1 \text{ pu} \end{aligned}$$

Bus Admittance Matrix

$$\boldsymbol{Y}_{\text{bus}} = \begin{bmatrix} Y_{11} & \cdots & Y_{1n} \\ \vdots & \ddots & \vdots \\ Y_{n1} & \cdots & Y_{nn} \end{bmatrix}$$

where each element $Y_{ij} = G_{ij} + jB_{ij}$, n is the total number of buses in the system. Denote the current and voltage of the buses as $\boldsymbol{I}_{\text{bus}}$ and $\boldsymbol{V}_{\text{bus}}$, respectively, we can describe a given network:

$$\boldsymbol{I}_{\text{bus}} = \boldsymbol{Y}_{\text{bus}} \boldsymbol{V}_{\text{bus}} \tag{5.1}$$

Using Kirchhoff's current law, the current entering the k-th bus can be expressed by:

$$\boldsymbol{I}_k = \sum_{i=1}^{n} \boldsymbol{Y}_{ki} \boldsymbol{V}_i \tag{5.2}$$

Then given the voltages on each bus $\boldsymbol{V}_{\text{bus}}$, the bus power injections can be calculated as:

$$\begin{aligned} P_k &= \sum_{i=1}^{n} |\boldsymbol{V}_i||\boldsymbol{V}_k|[G_{ki}\cos(\delta_k - \delta_i) + B_{ki}\sin(\delta_k - \delta_i)] \\ Q_k &= \sum_{i=1}^{n} |\boldsymbol{V}_i||\boldsymbol{V}_k|[G_{ki}\sin(\delta_k - \delta_i) - B_{ki}\cos(\delta_k - \delta_i)] \end{aligned} \tag{5.3}$$

Classical Formulation

Note that at each bus, there are four variables $(|\boldsymbol{V}|, \delta, P, Q)$ to be known to fully define the power flow. In classical power flow formulation, two of the four variables need to be specified (known) to calculate the other two. Based on the known variables, the buses in a system can be divided into three types:

Type of bus	$\|\boldsymbol{V}\|$	δ	P	Q	Represent
Constant power bus (PQ)	✗	✗	✓	✓	Loads
Voltage controlled bus (PV)	✓	✗	✓	✗	Generators
Swing (or slack) bus (SW)	✓	✓	✗	✗	Large generators control the frequency

The unknown variables that the system can control are bus voltages and angles, so the power flow calculation aims to solve $|\boldsymbol{V}|$ and δ in the PQ and PV buses. In specific, if bus 1 to m are PQ buses, $m+1$ to $n-1$ are PV buses, the n-th bus is the slack bus. Then there are $n-1$ unknown angels and m unknown magnitude. Our known vector $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\delta}_{n-1} \\ |\boldsymbol{V}|_m \end{bmatrix}$ has $n+m-1$ unknown variables and same number of known P and Q value based on the table above. Each of the known P and Q can be derived into a power equation in (5.3), so the problem is solvable.

Newton-Raphson Method

1. N-R method:

   Note that a set of differentiable nonlinear equations need to be solved to calculate the unknowns. A common way to solve nonlinear equation is to approximate it linearly at x_0 by the first to term of Taylor series:

   $$f(x) \approx f(x_0) + (x - x_0)\frac{\partial f(x_0)}{\partial x} \tag{5.4}$$

   then x can be approximated by:

   $$x = x_0 + [\,\frac{\partial f(x_0)}{\partial x}]^{-1}[f(x) - f(x_0)] \tag{5.5}$$

   This approximation can be repeated until the difference between current estimation and the previously estimated value is lower than a predefined tolerance. This iteration approach is called the Newton-Raphson Method:

$$x_{k+1} = x_k + [\frac{\partial f(x_k)}{\partial x}]^{-1}[f(x) - f(x_k)] \tag{5.6}$$

when the N-R method is used to find the root of the function, i.e. $f(x) = 0$, the iteration in (5.5) reduces to:

$$x_{k+1} = x_k - [\frac{\partial f(x_k)}{\partial x}]^{-1} f(x_k) \tag{5.7}$$

Similarly, the N-R update equation to find the root of a multi-dimensional nonlinear function $\boldsymbol{F}(\boldsymbol{x})$ is:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{J}^{-1}\boldsymbol{F}(\boldsymbol{x}_k) \tag{5.8}$$

where $\boldsymbol{x}$ is a n dimension vector of roots to $\boldsymbol{F}$, $\boldsymbol{J}$ is the Jacobian matrix:

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1} & \cdots & \frac{\partial F_n}{\partial x_n} \end{bmatrix}$$

2. Solve power flow by N-R method:

   Define the $n + m - 1$ dimension unknown and function:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\delta} \\ |\boldsymbol{V}| \end{bmatrix}, \boldsymbol{F}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{P_{spec}} \\ \boldsymbol{Q_{spec}} \end{bmatrix} - \begin{bmatrix} \boldsymbol{P}_{n-1} \\ \boldsymbol{Q}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta P}_{n-1} \\ \boldsymbol{\Delta Q}_m \end{bmatrix} \tag{5.9}$$

   where $\boldsymbol{P}_{n-1}$ and $\boldsymbol{Q}_m$ are calculated from (5.3). Then the solution can be obtained from the iteration formulated from (5.8):

$$\begin{bmatrix} \boldsymbol{\delta} \\ |\boldsymbol{V}| \end{bmatrix}_{k+1} = \begin{bmatrix} \boldsymbol{\delta} \\ |\boldsymbol{V}| \end{bmatrix}_k - \boldsymbol{J}^{-1} \begin{bmatrix} \Delta\boldsymbol{P}_{n-1} \\ \Delta\boldsymbol{Q}_m \end{bmatrix}_k \tag{5.10}$$

   The Jacobian matrix can be expressed as four sub-matrix:

$$\boldsymbol{J} = - \begin{bmatrix} \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{\delta}} & \frac{\partial \boldsymbol{P}}{\partial |\boldsymbol{V}|} \\ \frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{\delta}} & \frac{\partial \boldsymbol{Q}}{\partial |\boldsymbol{V}|} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H} & \boldsymbol{N} \\ \boldsymbol{K} & \boldsymbol{L} \end{bmatrix}$$

   where $\boldsymbol{H}$ is a $(n-1) \times (n-1)$ matrix with element $H_{ij} = -\frac{\partial P_i}{\partial \delta_j}$, plug (5.3) in, we can get:

$$H_{ij} = -|\boldsymbol{V}_i||\boldsymbol{V}_j|[G_{ij}\sin(\delta_i - \delta_j) - B_{ij}\cos(\delta_i - \delta_j)] \tag{5.11}$$

Similarly, we have:

$$N_{ij} = -|\boldsymbol{V}_i||\boldsymbol{V}_j|[G_{ij}\cos(\delta_i - \delta_j) + B_{ij}\sin(\delta_i - \delta_j)] \tag{5.12}$$

$$K_{ij} = |\boldsymbol{V}_i||\boldsymbol{V}_j|[G_{ij}\cos(\delta_i - \delta_j) + B_{ij}\sin(\delta_i - \delta_j)] \tag{5.13}$$

$$L_{ij} = -|\boldsymbol{V}_i||\boldsymbol{V}_j|[G_{ij}\sin(\delta_i - \delta_j) - B_{ij}\cos(\delta_i - \delta_j)] \tag{5.14}$$

when $i = j$:

$$H_{ii} = |\boldsymbol{V}_i|^2 B_{ii} + Q_i \tag{5.15}$$

$$N_{ii} = -|\boldsymbol{V}_i|^2 G_{ii} - P_i \tag{5.16}$$

$$K_{ii} = |\boldsymbol{V}_i|^2 G_{ii} - P_i \tag{5.17}$$

$$L_{ii} = |\boldsymbol{V}_i|^2 B_{ii} - Qi \tag{5.18}$$

5.1.2 Multi-period Optimal Power Flow

Now we extend the traditional OPF to a microgrid with WT and ESS as shown in Fig. 5.1.

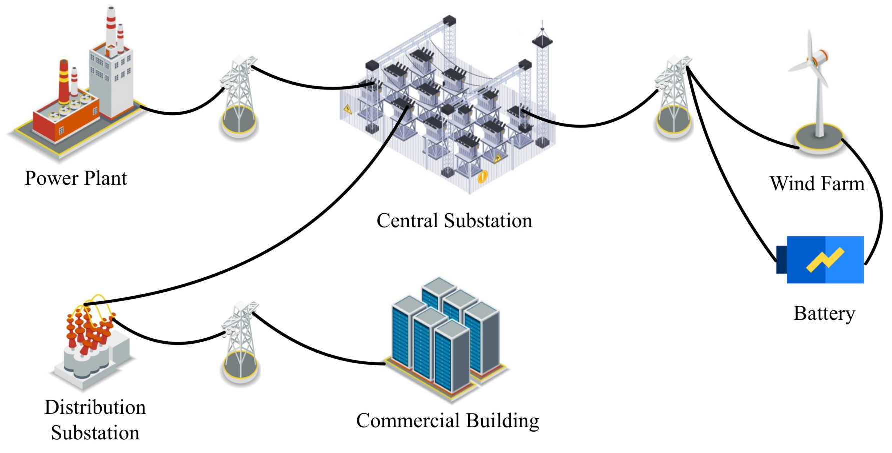


Figure 5.1: Distribution network with WT

Table 5.1 lists a summary of notations used in this section. We consider a power system with N buses, n traditional generators, one wind farm, l energy storage devices, and

m time-varying loads distributed at different buses. Define $\mathcal{N} = \{1, \ldots, N\}$ as the set of indices of all buses, and $\mathcal{G} \in \mathbb{N}^{1\times n}$, $\mathcal{B} \in \mathbb{N}^{1\times l}$, and $\mathcal{L} \in \mathbb{N}^{1\times m}$ as the set of indices of buses hosting generators, energy storage devices, and loads, respectively. The index of the bus with the wind farm is denoted as w.

The time is divided into non-overlapping time slots. The duration of each slot Δt is set to one hour in this paper. Denote the state of charge (SOC) of the storage device at the beginning of slot $t \in \mathcal{T} = \{t | t \geq t_0\}$ at bus $k \in \mathcal{B}$ as c_t^k. The initial SOC is set to be at half of its capacity in this paper. During slot $t \in \mathcal{T}$, each storage device can charge, discharge, or remain idle. Denote the charging/discharging rate of the storage device at bus k as q_t^k, and the charging/discharging efficiency as $\gamma_e \in (0, 1)$. The energy storage dynamics can be described by the following first-order difference equation:

$$c_{t+1}^k = \begin{cases} c_t^k + \Delta t q_t^k \gamma_e, & q_t^k > 0 \\ c_t^k, & q_t^k = 0 \\ c_t^k + \Delta t q_t^k / \gamma_e & q_t^k < 0. \end{cases} \tag{5.19}$$

Due to the physical limits of the battery, the SOC and charging/discharging rate must satisfy the following constraints,

$$0 \leq c_t^k \leq c_k^{\max} \tag{5.20}$$

$$-q_k^{\min} \leq q_t^k \leq q_k^{\max} \tag{5.21}$$

where $c_k^{\max}$ is the capacity of the energy storage device at bus k, $q_k^{\min}$ and $q_k^{\max}$ are the maximum discharging and charging rates at bus k, respectively.

Denote the active and reactive power generation at bus $k \in \mathcal{G}$ during slot $t \in \mathcal{T}$ as $P_{k,t}^g$ and $Q_{k,t}^g$, respectively. They are bounded by the power generation capacity at bus k as

$$\begin{aligned} P_k^{\min} \leq P_{k,t}^g \leq P_k^{\max}, \\ Q_k^{\min} \leq Q_{k,t}^g \leq Q_k^{\max}. \end{aligned} \tag{5.22}$$

The active power generation is also bounded by the ramp-rate limit at bus k as

$$-R_k^{\text{down}} \leq P_{k,t}^g - P_{k,t-1}^g \leq R_k^{\text{up}}. \tag{5.23}$$

The voltage magnitude $V_{k,t}$ at bus $k \in \mathcal{N}$ during slot $t \in \mathcal{T}$ is bounded by

$$V_k^{\min} \leq V_{k,t} \leq V_k^{\max}. \tag{5.24}$$

Table 5.1: Summary of notations

Notation	Description
B_{ij}	Line susceptance between buses i and j
$\mathcal{B}$	Set of buses hosting energy storage
c_{2k}, c_{1k}, c_{0k}	Coefficient of the generator cost function at bus k
c_t^k	State of charge of the energy storage at bus k at slot t
$c_k^{\max}$	Energy storage capacity at bus k
G_{ij}	Line conductance between buses i and j
$\mathcal{G}$	Set of buses hosting generators
h_t	Index of the hourly time in a day at slot t
l	Number of energy storage
$\mathcal{L}$	Set of buses with a changing demand
m	Number of buses with changing demand
n	Number of generators
N	Number of buses
$\mathcal{N}$	Set of buses
$P_{k,t}$,	Active power injected at bus k at slot t
$P_{k,t}^d, P_{k,t}^g$	Active power demand/generation at bus k at slot t
$P_k^{\min}, P_k^{\max}$	Minimum/maximum active power generation at bus k
$P_{k,t}^w$	Wind turbine power at bus k at slot t
$\mathbf{q}$	Schedule of all storage during OPF horizon
$\mathbf{q}_k$	Schedule of storage at bus k during OPF horizon
q_t^k	Charge/discharge power of the energy storage at bus k at slot t
$q_k^{\min}, q_k^{\max}$	Maximum discharge/charge rate of the energy storage at bus k
$Q_{k,t}$	Reactive power injected at bus k at slot t
$Q_{k,t}^d, Q_{k,t}^g$	Reactive power demand/generation at bus k at slot t
$Q_k^{\min}, Q_k^{\max}$	Minimum/maximum reactive power generation at bus k
$R_k^{\text{down}}, R_k^{\text{up}}$	Maximum down/up ramp-rate of the generator at bus k
$\mathcal{S}$	State space of a MDP
$S_{ij,t}$	Reactive power flow from bus i to bus j at slot t
$S_{ij}^{\max}$	Maximum reactive power flow from bus i to bus j
t	Index of the hourly time slots
t_0	RT-OPF starting time slot
T	Number of time slots in the multi-period OPF horizon
$\mathcal{T}$	Power flow time slots set
$V_{k,t}$	Voltage magnitude at bus k at slot t
$V_k^{\min}, V_k^{\max}$	Minimum/maximum voltage magnitude at bus k
w	Index of bus hosting wind turbines
γ_e	Charge/discharge efficiency of the energy storage
δ_{ij}	Difference between angles of buses i and j
Δt	Time slot solution

The active and reactive power flow balance equations at each bus $k \in \mathcal{N}$ are given by:

$$\begin{aligned} P_{k,t} &= \sum_{i \in \mathcal{N}} V_{k,t} V_{i,t} (G_{ki} \cos \delta_{ki} + B_{ki} \sin \delta_{ki}) \\ Q_{k,t} &= \sum_{i \in \mathcal{N}} V_{k,t} V_{i,t} (G_{ki} \sin \delta_{ki} - B_{ki} \cos \delta_{ki}) \end{aligned} \tag{5.25}$$

where G_{ki} and B_{ki} are the line conductance and susceptance between buses k and i, respectively, and δ_{ki} is the angle difference between buses k and i. $P_{k,t}$ and $Q_{k,t}$ are the active and reactive power injected to bus k at slot t given by

$$\begin{aligned} P_{k,t} &= P^g_{k,t} + P^w_{k,t} - q^k_t - P^d_{k,t} \\ Q_{k,t} &= Q^g_{k,t} - Q^d_{k,t} \end{aligned} \tag{5.26}$$

where $P^d_{k,t}$ and $Q^d_{k,t}$ are the active and reactive power demands at bus k during slot t, respectively. $P^w_{k,t}$ is the power from the WT at bus k.

The various power values need to satisfy the following constraints.

$$P^g_{k,t} = 0, \quad \forall k \notin \mathcal{G}, \tag{5.27}$$

$$Q^g_{k,t} = 0, \quad \forall k \notin \mathcal{G}, \tag{5.28}$$

$$P^w_{k,t} = 0, \quad \forall k \neq w, \tag{5.29}$$

$$q^k_t = 0, \quad \forall k \notin \mathcal{B}, \tag{5.30}$$

The reactive power flow on branch (i, j) at time slot $t \in \mathcal{T}$, $S_{ij,t}$ is bounded by:

$$0 \leq |S_{ij,t}| \leq S^{\max}_{ij} \tag{5.31}$$

The objective of the RT-OPF with battery storage and WT is to minimize the expected total future generation cost of the microgrid by optimizing the power generation and storage behavior for all future slots starting from the current slot $t = t_0$, without violating the storage dynamics and power flow constraints. The optimization problem can be formulated as

$$\begin{aligned} \min \quad & \mathbb{E}\left[\sum_{t=t_0}^{\infty} \sum_{k \in \mathcal{G}} c_{2k} {P^g_{k,t}}^2 + c_{1k} P^g_{k,t} + c_{0k} \right] \\ \text{s.t.} \quad & (5.19) - (5.31). \end{aligned} \tag{P1}$$

The optimization is performed over $P^g_{k,t}, Q^g_{k,t}$ for $k \in \mathcal{G}$ and q^k_t for $k \in \mathcal{B}$, where c_{2k}, c_{1k}, and c_{0k} are the coefficients of the generator cost function at bus $k \in \mathcal{G}$.

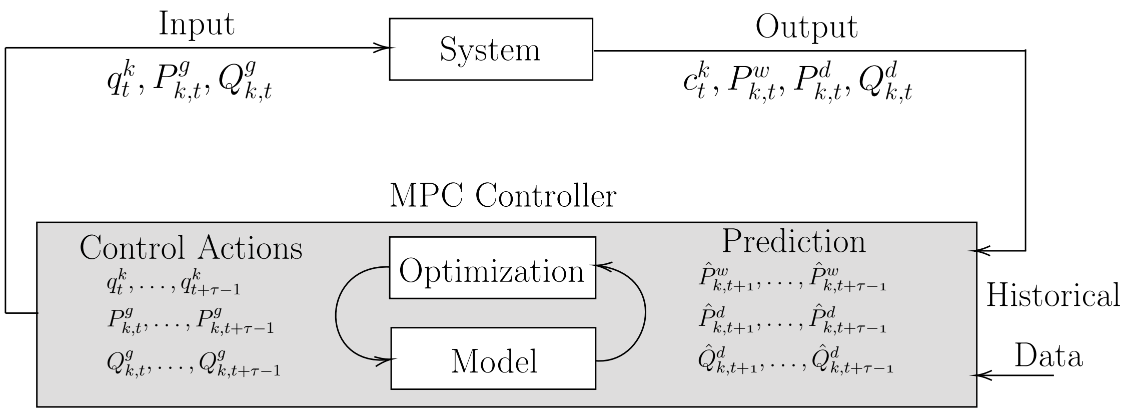

Input
$q_t^k, P_{k,t}^g, Q_{k,t}^g$
System
Output
$c_t^k, P_{k,t}^w, P_{k,t}^d, Q_{k,t}^d$
MPC Controller
Control Actions
$q_t^k, \dots, q_{t+\tau-1}^k$
$P_{k,t}^g, \dots, P_{k,t+\tau-1}^g$
$Q_{k,t}^g, \dots, Q_{k,t+\tau-1}^g$
Optimization
Model
Prediction
$\hat{P}_{k,t+1}^w, \dots, \hat{P}_{k,t+\tau-1}^w$
$\hat{P}_{k,t+1}^d, \dots, \hat{P}_{k,t+\tau-1}^d$
$\hat{Q}_{k,t+1}^d, \dots, \hat{Q}_{k,t+\tau-1}^d$
Historical
Data

Figure 5.2: Framework of MPC-based RT-OPF

5.2 Different approaches

5.2.1 Multi-period Optimal Power Flow with Model Predictive Control

The RT-OPF problem is formulated by using model predictive control (MPC) in this section. MPC is widely used and proven effective for industrial applications with slow dynamics such as chemical plants and supply chains. It is also popular in real-time energy management research such as real-time economic dispatch and RT-OPF. Figure 5.2 shows the framework of MPC-based RT-OPF. The MPC is used to form a prediction model to predict the wind power and load demands for the next $\tau - 1$ time slots. The RT-OPF is solved by using a rolling optimization over τ-period OPF with storage and predictions on load and wind.

ARMA Prediction model

The load and wind power can be modeled as stochastic time series and predicted by using statistical or machine learning methods. In this paper, the MPC controller employs the auto-regressive-moving-average (ARMA) model with the same orders for both auto-regression and moving-average. The prediction on wind power generation can be modeled as

$$P_{k,t}^w = c + \varepsilon_t + \sum_{i=1}^{p} \left[\varphi(i) P_{k,t-i}^w + \theta_i \varepsilon_{t-i}\right], \tag{5.32}$$

where c is a constant, ε_t is the model error at slot t, independent and identically distributed (i.i.d.) with a mean of zero, p is the order of the auto-regressive describing the pattern of the load, and it is also the order of the moving-average describing random effects, φ_i are regression parameters, and θ_i are the linear combination parameters for the error terms.

The prediction models for active and reactive load demands have the same order as the wind but use different model parameters learned from historical load data. Given the prediction model parameters, future wind and load demands from $t+1$ to $t+\tau-1$, $\{\hat{P}^w_{k,t+i}\}_{i=1}^{\tau-1}$, $\{\hat{P}^d_{k,t+i}\}_{i=1}^{\tau-1}$, and $\{\hat{Q}^d_{k,t+i}\}_{i=1}^{\tau-1}$, can be predicted and used for off-line τ-period OPF calculation combined with the current information on wind, demand, and storage SOC.

PSO-based τ-period OPF with storage and wind

The multi-period AC OPF with storage scheduling is a non-convex optimization problem. A commonly used approach to solve this problem in practice is to solve each slot as a single-period OPF with storage and wind, and then optimize the storage behavior over all periods.

Denote a possible storage scheduling vector during slots $\mathcal{T} = \{t_0, t_0+1, \ldots, t_0+\tau-1\}$ at bus $k \in \mathcal{B}$ as $\mathbf{q}^k = [q^k_{t_0}, q^k_{t_0+1}, \ldots, q^k_{t_0+\tau-1}]^T \in \mathcal{R}^{\tau\times 1}$, and define $\mathbf{q} = [\mathbf{q}^i | i \in \mathcal{B}] \in \mathcal{R}^{\tau\times l}$. Given the storage scheduling vector $\mathbf{q}$, the single-period OPF is converted to a normal single-period OPF problem with the following objective:

$$\begin{aligned} C(\mathbf{q}) = \min & \left[\sum_{t=t_0}^{t_0+\tau-1} \sum_{k\in\mathcal{G}} c_{2k} {P^g_{k,t}}^2 + c_{1k} P^g_{k,t} + c_{0k} \right] \\ \text{s.t.} \quad & (5.21) - (5.25), (5.26), (5.28) - (5.31) \end{aligned} \tag{P2}$$

over $P^g_{k,t}$ for $k \in \mathcal{G} \cup \mathcal{B} \cup \{w\}$ and $Q^g_{k,t}$ for $k \in \mathcal{G}$.

The optimal storage scheduling of the τ-period OPF with storage and wind can then be formulated as

$$\begin{aligned} & \min_{\mathbf{q}} C(\mathbf{q}) \\ & \text{s.t. } (5.19) - (5.20). \end{aligned} \tag{P3}$$

The storage optimization problem can be solved by using heuristic algorithms such as the particle swarm optimization (PSO). In particular, during solving (P3), a single-period OPF is calculated for each particle, which results in high computation complexity for large scale power systems. Combining (P2) and (P3) solves the τ-period OPF with storage and wind up to slot $t_0+\tau-1$.

Algorithm 5 MPC-based RT-OPF with storage and wind

```
Require: Historical wind and load data, and prediction models for wind and load.
1: Initialization: $t \leftarrow t_0$. Initial SOC $c_{t_0}^k = 0.5c_k^{\max}$ for bus $k \in \mathcal{B}$, wind power $P_k^w(t_0)$ for bus $k = w$, and load demand $P_{k,t_0}^d, Q_{k,t_0}^d$ for bus $k \in \mathcal{N}$.
2: while $t \geq t_0$ do
3:     Sample current system output $c_t^k, P_{k,t}^w, P_{k,t}^d$, and $Q_{k,t}^d$.
4:     Import wind and load profiles over the previous $p-1$ slots: $\{P_{k,t-p+1}^w, \dots, P_{k,t-1}^w\}$,$\{P_{k,t-p+1}^d, \dots, P_{k,t-1}^d\}$,$\{Q_{k,t-p+1}^d, \dots, Q_{k,t-1}^d\}$.
5:     Predict wind and load profiles over the next $\tau-1$ slots: $\{\hat{P}_{k,t+1}^w, \dots, \hat{P}_{k,t+\tau-1}^w\}$, $\{\hat{P}_{k,t+1}^d, \dots, \hat{P}_{k,t+\tau-1}^d\}$, and $\{\hat{Q}_{k,t+1}^d, \dots, \hat{Q}_{k,t+\tau-1}^d\}$.
6:     Calculate the $\tau$-period OPF with storage and wind from $t$ to $t+\tau-1$, get one step storage schedule $q_t^k$ for $k \in \mathcal{B}$ and power generation $P_{k,t}^g, Q_{k,t}^g$ for $k \in \mathcal{G}$.
7:     Operate the storage and generators correspondingly, update the SOC.
8:     $t \leftarrow t+1$
9: end while
```

MPC-based OPF with storage and wind

After exploring the state trajectories that emanate by different schedules from the current state and finding the optimal control actions, only the first step of the off-line strategy is applied to the real-time system. Then the system will output the updated wind, demand, and SOC. The MPC controller shifts forward by one slot and starts a new round of prediction and optimization. The detailed procedures are given in Algorithm 5.

5.2.2 Multi-period Optimal Power Flow with DDPG

As discussed in previous chapters, reinforcement learning (RL) trains an agent to make decisions by interacting with an environment $\mathcal{E}$ to maximize the expected cumulative future reward. The formulation is based on Markov Decision Process (MDP), which consists of a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} = \mathbf{P}(s_{t+1}|s_t, a_t)$ is the transition probability from states s_t to s_{t+1} by taking action a_t, $r_t = r(s_t, a_t) \in \mathcal{R}$ is the immediate scalar reward or cost by applying action a_t from state s_t, $0 < \gamma \leq 1$ is the discount factor and its value is usually chosen to be close to 1. A policy can be either stochastic or deterministic. A stochastic policy $\pi(s_t) = \mathbf{P}(a_t|s_t)$ gives the probability of taking different actions based on the state s_t. A deterministic policy gives a single action $a_t = \mu(s_t)$. The deterministic policy is adopted to solve the RT-OPF problem.

MDP Formulation *1) State Space $\mathcal{S}$.* The state at time slot t includes the hour index in a day $h(t) \in \{0, 1, \dots, 23\}$, the measured active and reactive demands, the wind power measured from the wind turbine, the power generation and the state of charge at slot $t-1$,

$$s_t = [\, h_t, P^d_{k\in\mathcal{L},t}, Q^d_{k\in\mathcal{L},t}, P^w_{k=w,t}, P^g_{k\in\mathcal{G},t-1}, Q^g_{k\in\mathcal{G},t-1}, c^{k\in\mathcal{B}}_{t-1}]. \tag{5.33}$$

2) Action Space $\mathcal{A}$. The actions or decisions that can be made at t is the energy charge or discharge at the storage: $a_t = q^{k\in\mathcal{B}}_t$. The action should meet constraints (6.14) and (6.13).

3) Transition Probability $\mathcal{P}$. Given the current state s_t, the action a_t affects power generation $P^g_{k\in\mathcal{G},t}, Q^g_{k\in\mathcal{G},t}$ under the constraints of power flow and ramp-rate limits, and the SOC charge $c_{k\in\mathcal{B}t}$ under the constraints of the battery dynamics. The transition of the hour index is deterministic as follows:

$$h_{t+1} = (h_t + 1) \mod 24 \tag{5.34}$$

where $\mod$ stands for the modulo operator. The transition probability of the demand and wind power is unknown.

4) Reward function $r(s_t, a_t)$. Once the transition is made by taking action a_t from state s_t, the power generation $P^g_{k\in\mathcal{G},t}$ and $Q^g_{k\in\mathcal{G},t}$ are known. The reward function of this transition is given by

$$r(s_t, a_t) = -\sum_{k\in\mathcal{G}} c_{2k} {P^g_{k,t}}^2 + c_{1k} P^g_{k,t} + c_{0k}. \tag{5.35}$$

Define the discounted cumulative future reward starting from time t_0 as

$$R_{t_0} = \sum_{t=t_0}^{\infty} \gamma^{t-t_0} r(s_t, a_t). \tag{5.36}$$

The discounted cumulative reward R_{t_0} is a random variable, the distribution of which depends on the environment $\mathcal{E}$ (or the transition probability $\mathcal{P}$). The general objective of MDP is to find the optimum policy μ^* that will maximize the expected cumulative discounted future reward from time t_0, that is, to maximize $\mathbb{E}[R_{t_0}]$,

$$\max\ \mathbb{E}\left[\sum_{t=t_0}^{\infty} -\gamma^{t-t_0} \sum_{k\in\mathcal{G}} c_{2k} {P^g_{k,t}}^2 + c_{1k} P^g_{k,t} + c_{0k}\right], \tag{5.37}$$

where the expectation is performed with respect to the environment $\mathcal{E}$ only. Compared to the objective of RT-OPF in (P1), the MDP objective in (5.37) is the same as (P1) if the discount factor $\gamma = 1$. The problem is formulated as

$$\begin{aligned} \max \ & \mathbb{E}_{r,s\sim\mathcal{E},\gamma=1}[R_{t_0}], \\ \text{s.t.} \quad & (5.19) - (5.31). \end{aligned} \tag{P4}$$

5) Action-value function $Q(s_t, a_t)$. The action-value function is defined as

$$Q(s_t, a_t) = \mathbb{E}_{r_{i\geq t}, s_{i>t}\sim\mathcal{E}}\left[R_t | s_t, a_t\right]. \tag{5.38}$$

It is used to estimate the expected discounted return after taking an action a_t from state s_t from any time $t \geq t_0$.

For simplicity, the action a_t under a deterministic policy μ is omitted, as well as the distribution under $\mathcal{E}$

$$Q^\mu(s_t) = \mathbb{E}_{r_{i\geq t}, s_{i>t}}\left[R_t | s_t, \mu(s_t)\right]. \tag{5.39}$$

The action-value function can be expressed recursively by following the *Bellman equation* as

$$Q^\mu(s_t) = \mathbb{E}_{r_t, s_{t+1}}[r(s_t, \mu(s_t)) + \gamma Q^\mu(s_{t+1})]. \tag{5.40}$$

Given the optimal action-value function denoted by Q^*, the optimal policy of the Bellman equation μ^* satisfies:

$$\mu^*(s_t) = \arg\max_{a_t\in\mathcal{A}} Q^*(s_t, a_t). \tag{5.41}$$

It is in general difficult, if not impossible, to obtain the analytical expression of the action-value function. In Q-learning, the Q function and the policy are learned in an iterative manner through the interaction with the environment.

Since the expectation in the Q function definition in (5.39) is not taken with respect to the policy, the Q function can be learned by transitions from a policy λ that is different from μ. A greedy policy that maximizes the current Q-function at time t is used in Q-Learning as

$$\lambda(s_t) = \arg\max_{a_t\in\mathcal{A}} Q^\lambda(s_t, a_t). \tag{5.42}$$

With the greedy policy, the Q function with an arbitrary initial value is updated in the $(t - t_0)$-th iteration as

$$Q^{\lambda}(s_t) \leftarrow (1-\alpha)Q^{\lambda}(s_t) + \alpha[r(s_t, \lambda(s_t)) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q^{\lambda}(s_{t+1})], \tag{5.43}$$

where $\alpha \in (0, 1)$ is the learning rate of the Q function. In Q-learning, the Q function values are stored in a look-up-table (LUT), the Q-table. Thus it requires a discretization of the state and action spaces, which will result in loss of precision. Deep Q-learning Network (DQN) uses a deep neural network (DNN) parameterized by θ^Q to estimate $Q(s, a)$, but it still can only deal with discrete action spaces. A replay buffer $\mathcal{R}$ and a target network $Q^{'}(s, a)$ with weight $\theta^{Q'}$ are introduced in DQN to stabilize the learning process. Details can be found in [145].

DDPG-based OPF with storage and wind

We propose to use the DDPG learning method to solve the BESS scheduling problem, which has a continuous action space $\mathcal{A}$. Instead of using a greedy policy that requires a global maximization in the continuous action space, the DDPG algorithm uses a policy gradient approach to search along the gradient of a policy-dependent value function with respect to the policy.

The DDPG agent employs two DNNs as shown in Fig. 5.3, and they are denoted as actor and critic networks, respectively. The actor network is used to model the deterministic policy μ for a given state s as $\mu(s|\theta^{\mu})$ with parameter θ^{μ}. The critic network is used to model the Q-function as $Q(s, a|\theta^Q)$ with parameter θ^Q. A replay buffer is still used along with the target critic network $Q'(s, a|\theta^{Q'})$ and target actor network $\mu'(s|\theta^{\mu'})$.

1) Critic Network. The parameters of the critic network are updated by using a target value calculated by using the target critic network and a subset of transitions from the replay buffer.

The replay buffer stores the previous $|\mathcal{R}|$ transitions before the current time slot t, that is $(s_i, a_i, r_i, s_{i+1})_{i\in\{t-|\mathcal{R}|+1,\ldots,t-1,t\}}$. During the time slot t, a mini-batch $\mathcal{M} \subset \mathcal{R}$ with $|\mathcal{M}|$ transitions $(s_i, a_i, r_i, s_{i+1})_{i\in I(\mathcal{M})}$ are randomly sampled from the replay buffer, where $I(\mathcal{M})$ is the time index of transitions in the mini-batch $\mathcal{M}$. Define the target critic value of transition $i \in I(\mathcal{M})$ under current target actor $\theta^{\mu^{'}}$ as y_i:

$$y_i = r_i + \gamma Q^{'}\left(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}.\right) \tag{5.44}$$

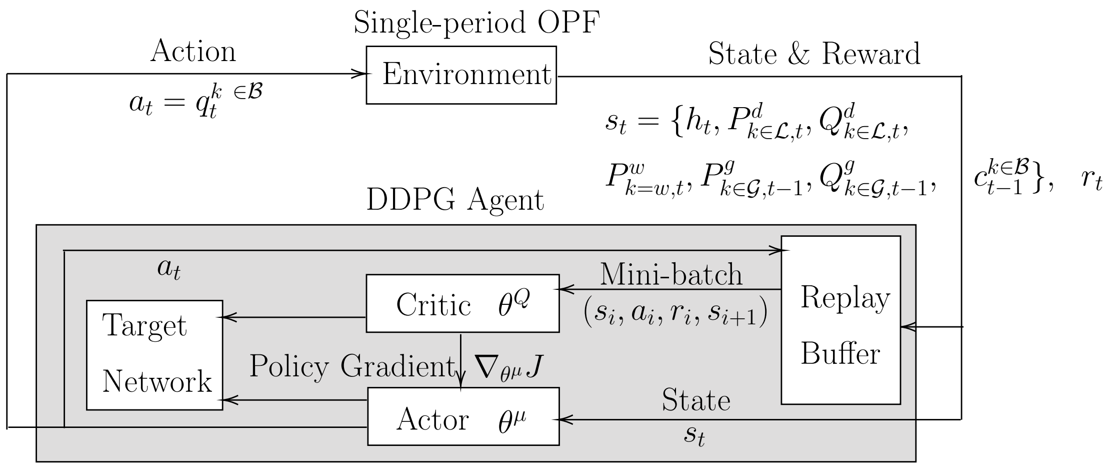

Single-period OPF
Environment
Action
$a_t = q_t^{k\in\mathcal{B}}$
State & Reward
$s_t = \{h_t, P^d_{k\in\mathcal{L},t}, Q^d_{k\in\mathcal{L},t}, P^w_{k=w,t}, P^g_{k\in\mathcal{G},t-1}, Q^g_{k\in\mathcal{G},t-1}, c^{k\in\mathcal{B}}_{t-1}\}, \quad r_t$
DDPG Agent
a_t
Critic θ^Q
Mini-batch
(s_i, a_i, r_i, s_{i+1})
Replay Buffer
Target Network
Policy Gradient $\nabla_{\theta^\mu} J$
Actor θ^μ
State
s_t

Figure 5.3: Framework of DDPG-based OPF

The parameters of the critic network θ^Q are updated by a one-step descent along the gradient of the mean squared error between the critic network output and the target value with learning rate l_c as

$$L(\theta^Q) = \frac{1}{|\mathcal{M}|}\sum_{i\in\mathcal{M}}\left(Q(s_i, a_i|\theta^Q) - y_i\right)^2, \tag{5.45}$$

$$\theta^Q \leftarrow \theta^Q + l_c\nabla_{\theta^Q}\frac{1}{|\mathcal{M}|}\sum_{i\in\mathcal{M}}\left(Q(s_i, a_i|\theta^Q) - y_i\right)^2. \tag{5.46}$$

2) Actor Network. The actor network is used to update the policy with the help of the critic network. The objective in (P3) can be written as a function of the policy θ^μ as $J(\theta^\mu)$. The *Policy Gradient Theorem* gives the policy gradient [138]

$$\nabla_{\theta^\mu}J(\theta^\mu) = \mathbb{E}_{s_t\sim\mu}\left[\nabla_a Q(s_t, \mu(s_t)|\theta^Q)\nabla_{\theta^\mu}\mu(s_t|\theta^\mu)\right]. \tag{5.47}$$

This provides a theoretical base of updating policy with its performance. A more efficient way of updating the actor network is to maximize the current Q-function value on $\mathcal{M}$ and with a learning rate l_a as

$$L(\theta^\mu) = \frac{1}{|\mathcal{M}|}\sum_{i\in\mathcal{M}}\left(Q(s, \mu(s_t|\theta^\mu))|\theta^Q\right) \tag{5.48}$$

$$\theta^\mu \leftarrow \theta^\mu + l_a\nabla_{\theta^\mu}\frac{1}{|\mathcal{M}|}\sum_{i\in\mathcal{M}}\left(Q(s, \mu(s_i|\theta^\mu)|\theta^Q\right) \tag{5.49}$$

The target critic and actor networks are updated in every iteration with a parameter ϵ as

$$\begin{aligned}\theta^{Q'} &\leftarrow \epsilon\theta^Q + (1-\epsilon)\theta^{Q'}\\ \theta^{\mu'} &\leftarrow \epsilon\theta^\mu + (1-\epsilon)\theta^{\mu'}\end{aligned} \tag{5.50}$$

Algorithm 6 DDPG-based OPF with storage and wind training

```
Require: initial actor parameters $\theta^{\mu}$, critic parameters $\theta^{Q}$, empty replay buffer $\mathcal{R}$
 1: Initialization: Set target network parameters equal to main parameter: $\theta^{Q'} \leftarrow \theta^{Q}$,
    $\theta^{\mu'} \leftarrow \theta^{\mu}$
 2: for episode $= 1$ to $M$ do
 3:     Initial $s_1$ for the first time slot
 4:     for $t = 1$ to $T$ do
 5:         Output action from actor network $a_t = \mu(s_t|\theta^{\mu})$
 6:         Execute action $a_t$ in the environment and observe reward $r_t$ and new state $s_{t+1}$
 7:         Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{R}$
 8:         Randomly sample a mini-batch $\mathcal{M}$ of $|\mathcal{M}|$ transitions $(s_i, a_i, r_i, s_{i+1})_{i\in I(\mathcal{M})}$ from $\mathcal{R}$
 9:         Compute target value $y_i$ with (6.28).
10:         Update the critic parameters $\theta^{Q}$:
                $$\theta^{Q} \leftarrow \theta^{Q} + l_c \nabla_{\theta^{Q}} \frac{1}{|\mathcal{M}|} \sum_{i\in\mathcal{M}} \big(Q(s_i, a_i|\theta^{Q}) - y_i\big)^2$$
11:         Update the actor parameters $\theta^{\mu}$:
                $$\theta^{\mu} \leftarrow \theta^{\mu} + l_a \nabla_{\theta^{\mu}} \frac{1}{|\mathcal{M}|} \sum_{i\in\mathcal{M}} \big(Q(s, \mu(s_i|\theta^{\mu})|\theta^{Q}\big)$$
12:         Update the target networks:
                $$\theta^{Q'} \leftarrow \epsilon\theta^{Q} + (1-\epsilon)\theta^{Q'}$$
                $$\theta^{\mu'} \leftarrow \epsilon\theta^{\mu} + (1-\epsilon)\theta^{\mu'}$$
13:     end for
14: end for
Ensure: Target actor parameters $\theta^{\mu'}$
```

To balance the tradeoff between exploration and exploitation, noise is added to the actions drawn from the deterministic policy. The noise model used in the actor network is similar to that in [146]. The DDPG training algorithm for the RT-OPF is shown in Algorithm 6.

5.3 Case study

The performance of the proposed DDPG-based RT-OPF algorithm are compared to that of the MPC-based RT-OPF algorithm. The MPC-based algorithm is implemented using Matpower [147] in Matlab. Note that no models are provided in Matpower for the wind turbine and energy storage, therefore they are all modeled as none-cost generators with a given maximum generation equal to the wind power and energy power in the single-period OPF during simulation. The DDPG-base algorithm is implemented using the Stable Baselines [144] and pandapower [148] in Python.

5.3.1 Data source and setup

The RT-OPF algorithms are tested on a modified IEEE 14-bus system as shown in Fig. 8.2, where energy storage units are installed at buses $\mathcal{B} = \{3, 6, 8, 11\}$ and the wind turbine is located at bus $w = 11$. Generators are located at bus $\mathcal{G} = \{1, 2, 3, 6, 8\}$. The load demand at buses $\mathcal{L} = \{10, 12, 13, 14\}$ are replaced by scaled hourly load data from four locations at the University of Arkansas, Fayetteville.

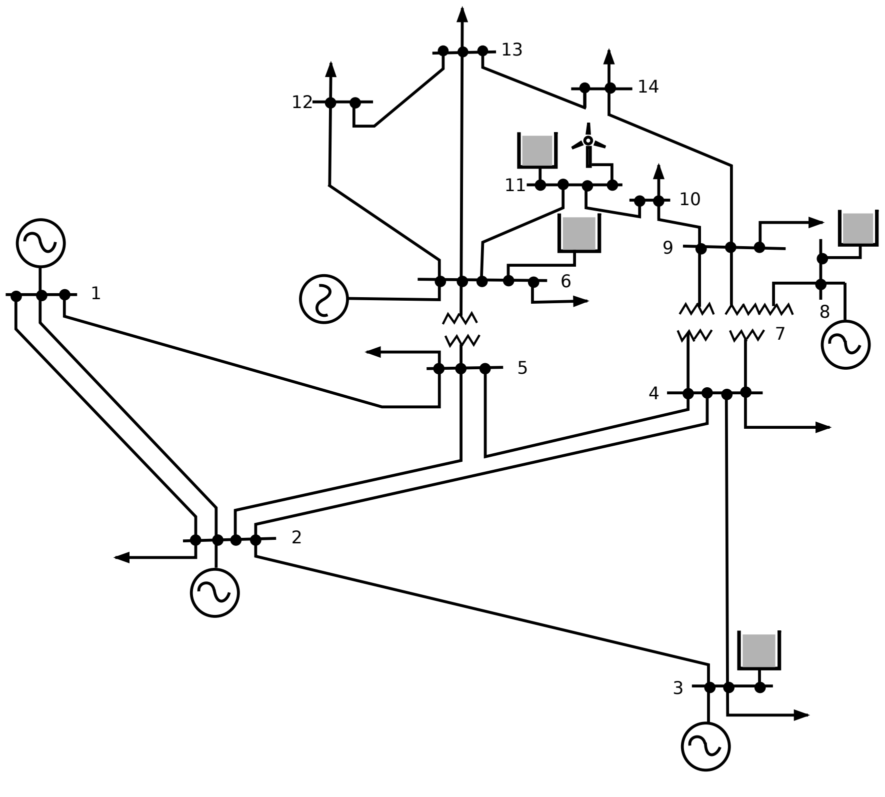


Figure 5.4: The modified IEEE 14-bus case with ESS and WT.

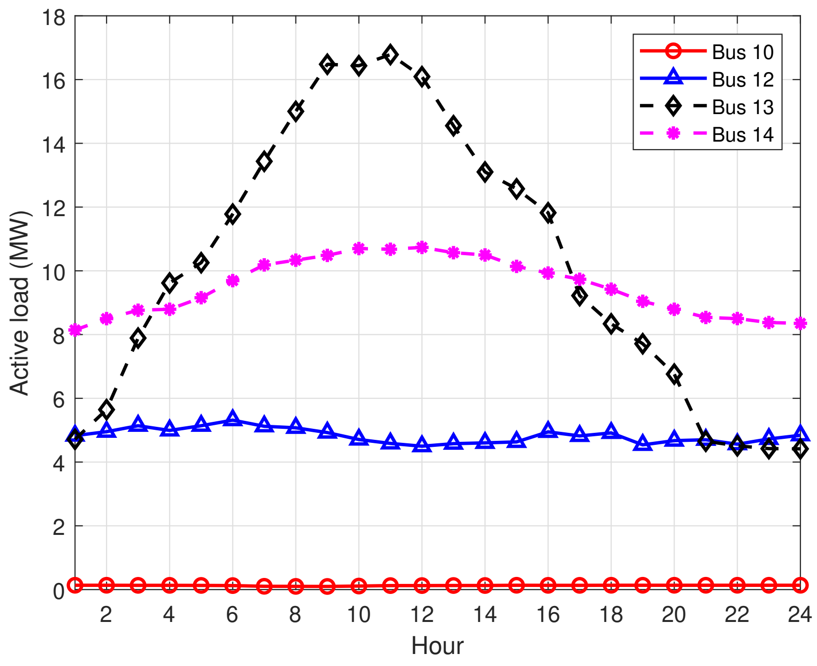


Figure 5.5: The active load profile at buses 10, 12, 13, 14.

The dataset provides active and reactive loads in 21 months from January, 2016 to September, 2017. The wind power is calculated from wind speed data at Drake Field weather station in Fayetteville in the Automated Surface Observing System (ASOS) dataset [149]. The wind power is scaled up to the same level of the base load of the system (275.5 MW), and the time period is chosen to be the same as the load dataset. The load and wind profiles in 2016 are used to train the ARMA model for MPC and the DDPG target actor network. The performance of the algorithms is tested by using the data from March, 2017. Fig. 6.8 shows the active load of the four buses on a testing day March 4th, 2017.

All energy storage units are identical, with a maximum charging/discharging rate of 4 MW ($q_{k\in\mathcal{B}}^{\min} = 4, q_{k\in\mathcal{B}}^{\max} = 4$), capacity 10 MWh ($c_{k\in\mathcal{B}}^{\max} = 10$), and storage efficiency $\gamma_e = 0.94$. Other system parameters and constraints remain the same as the original IEEE 14-bus system. Ramp limits and cost coefficients of the generators are listed in Table 5.2.

Two situations for comparison along with MPC and DDPG algorithms are introduced as follows:

Table 5.2: Generator parameters for OPF

Bus (k)	1	2	3	6	8
$R_k^{\text{down}}, R_k^{\text{up}}$ (MW)	40	70	50	60	6
c_{2k}	0.043	0.25	0.01	0.01	0.01
c_{1k}	20	20	40	40	40
c_{0k}	0	0	0	0	0

No storage used and no prediction (No storage)

Under this situation, a single-period OPF with wind is solved at each time slot. Solutions from the previous time slot only constraints the current OPF by the generator ramp-rate limit. This scenario serves as a baseline for comparison with more complicated systems.

OPF with known profiles (OPF)

The future load and wind profiles in the testing month are assumed to be known under this situation. An off-line 24-period OPF with storage and wind is solved by the PSO-based algorithm in 5.2.1 for each day instead of solving the OPF for the whole month. The initial SOC on the first day is at half capacity, and the initial SOC of the remaining days is the same as the final SOC of the previous day. This non-causal scheduling algorithm provides a benchmark with the best possible performance.

MPC-based RT-OPF (MPC)

ARMA models with order $p = 2$ are trained to predict the load and wind. Figs. 5.6 and 5.7 show the 1-hour ahead prediction results at bus 14 on March 4th, 2017. The mean absolute percentage error (MAPE) of the 1-hour ahead load prediction from January, 2017 to September, 2017 at four locations are given in Table 5.3. Predictions over longer period will accumulate the prediction error. A 5-hour prediction ($\tau = 6$) is employed to solve a 6-period OPF by the MPC controller.

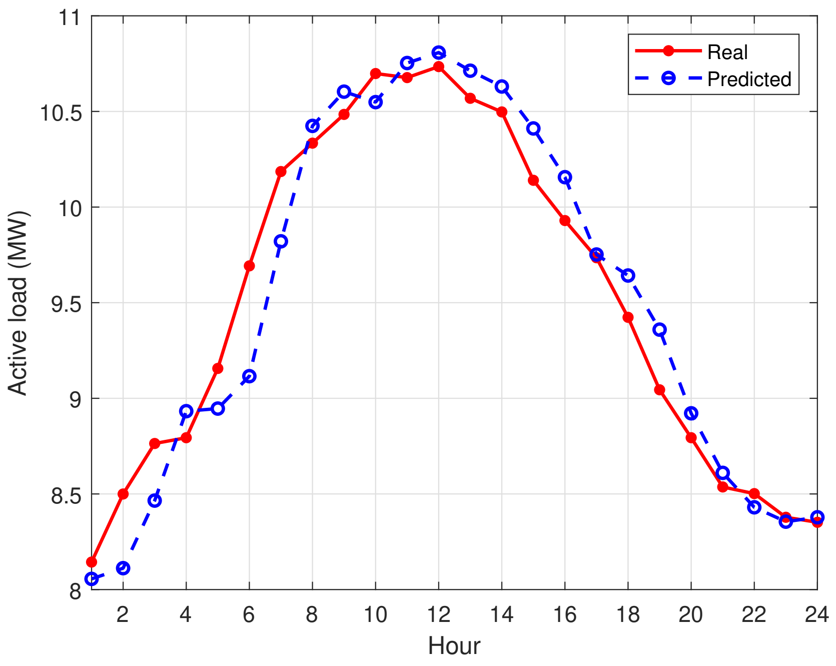


Figure 5.6: 1 hour ahead active load prediction at bus 14.

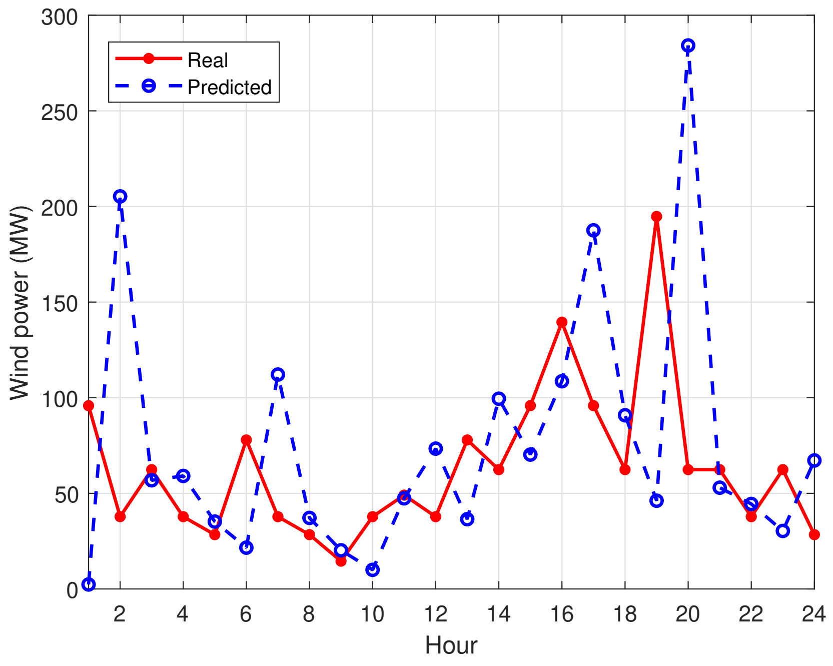


Figure 5.7: 1 hour ahead wind prediction at bus 14.

Table 5.3: 1 hour ahead load prediction performance

Bus	10	12	13	14
MAPE	0.074	0.044	0.061	0.057

DDPG-based RT-OPF (DDPG)

DDPG training parameters are given in Table 6.2. The target actor parameter $\theta^{\mu'}$ after DDPG training is applied to the testing month as shown in Algorithm 7. The framework of DDPG-based RT-OPF can be obtained by simply replacing the whole DDPG Agent block in Fig. 5.3 with the target actor network $\theta^{\mu'}$.

Table 5.4: DDPG parameters

Notation	**Description**	**Value**
Number of neurons in DDPG hidden layers		128,128
l_a	DDPG actor learning rate	0.00001
l_c	DDPG critic learning rate	0.0001
ϵ	DDPG soft update coefficient	0.001
$\lvert\mathcal{R}\rvert$	Experience replay buffer size	50000
$\lvert\mathcal{M}\rvert$	Batch size	240
M	Number of episodes trained	5000
T	Episode length	24

5.3.2 Simulation results

Fig. 5.8 shows the storage schedule results of OPF, MPC, and DDPG on each bus on March 4th, 2017. The MPC and DDPG results both have a similar storage behavior as the OPF case. The storage units are fully charged after the second hour and are discharged to zero from about 6 a.m. to 10 a.m., then are charged again from about 2 p.m. to 5 p.m., and are discharged after 9 p.m. The storage units are always fully discharged by the OPF at the end of the testing day because the daily operation cost will be minimized by using all the energy in the storage, while MPC and DDPG may save energy at the end of the day for tomorrow. The MPC result is closer to the OPF result for buses 3, 8, and 11 than the

Algorithm 7 DDPG-based RT-OPF with storage and wind

```
Require: Target network parameter $\theta^{\mu'}$.
1: Initialization: $t = t_0$, initialize $P^g_{k\in\mathcal{G},t_0}$ and $Q^g_{k\in\mathcal{G},t_0}$ by IEEE 14-bus case, $c^{k\in\mathcal{B}}_{t_0} = 0.5c^{\max}_{k\in\mathcal{B}}$.
2: while $t \geq t_0$ do
3:     Obtain new load and wind power, record the generation and the SOC to get $s_t$.
4:     Calculate the schedule $a_t = q^{k\in\mathcal{B}}_t$ based on the target actor in Algorithm 1 as $a_t = \mu'(s_t|\theta^{\mu'})$
5:     Calculate the power generation $P^g_{k\in\mathcal{G},t}$ and $Q^g_{k\in\mathcal{G},t}$ given the storage schedule by the single-period OPF.
6:     Operate the storage and generators correspondingly, and update the SOC.
7:     $t \leftarrow t + 1$
8: end while
```

DDPG. The DDPG learns a general policy based on the load and wind patterns on different days.

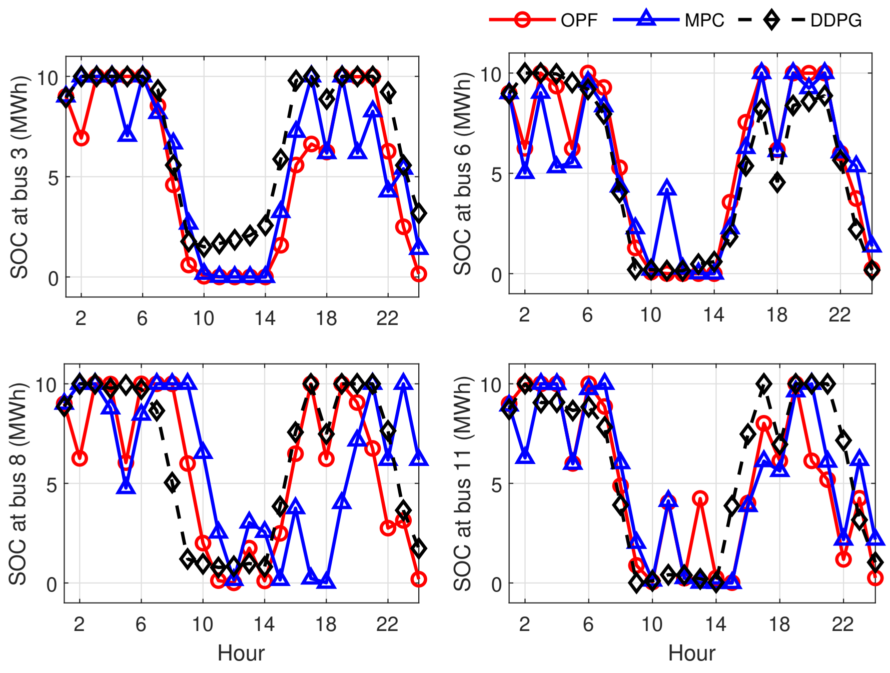


Figure 5.8: The storage schedule results at bus 3, 6, 8, 11.

Therefore, the DDPG schedule is smoother, and there is almost no alternative charging and discharging compared with the results from the other two algorithms.

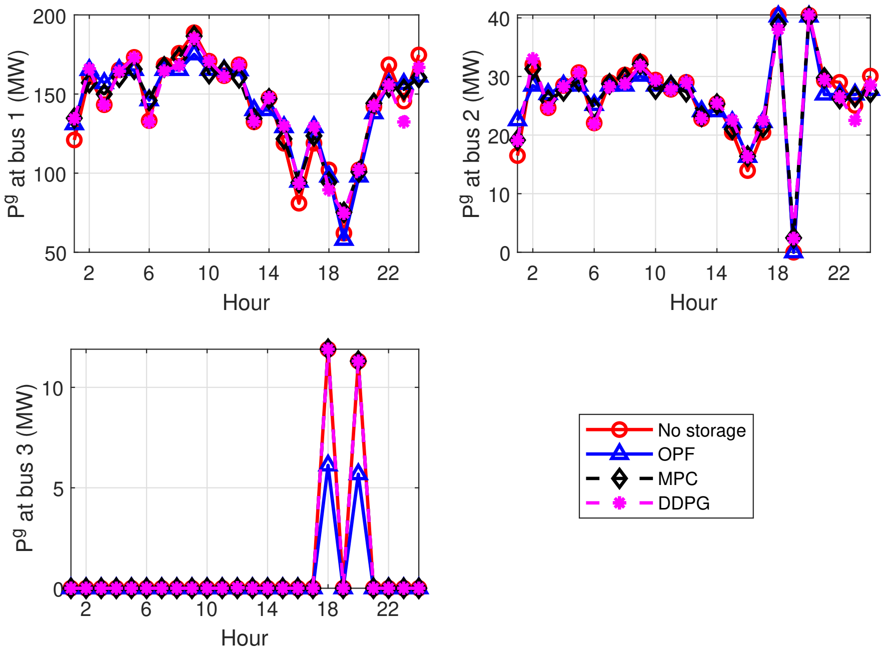


Figure 5.9: The active power generated at bus 1, 2, 3.

Figs. 5.9 and 5.10 show the active power generation results obtained by all algorithms at buses 1, 2, and 3. Generators at buses 6 and 8 are unused. Generator at bus 1 provides most of the power because of the low cost parameters c_{21} and c_{11}, generator at bus 2 with c_{12} also supplements the generation when generator 1 is bounded by the ramp-rate limits. Specifically, because of the wind power peak at 7 p.m. as shown in Fig. 5.7, P_1^g decreases with the maximum ramp down rate from 5 p.m. to 7 p.m. and increases after that. As a result, P_2^g and P_3^g increases at 6 p.m. and 8 p.m. to balance the demand when P_1^g decreases. The generation results by different algorithms are close, the average hourly operation cost and computation time of the testing month is given in Table 5.5.

Table 5.5: Average hourly operation result using different methods

Case	No storage	OPF	MPC	DDPG
Cost per hour ($)	6327.99	5781.18	6297.08	6272.31
Computation time (s)	1.03	13.58	64.77	1.28

The OPF algorithm operates with non-causal knowledge of the load and wind infor-

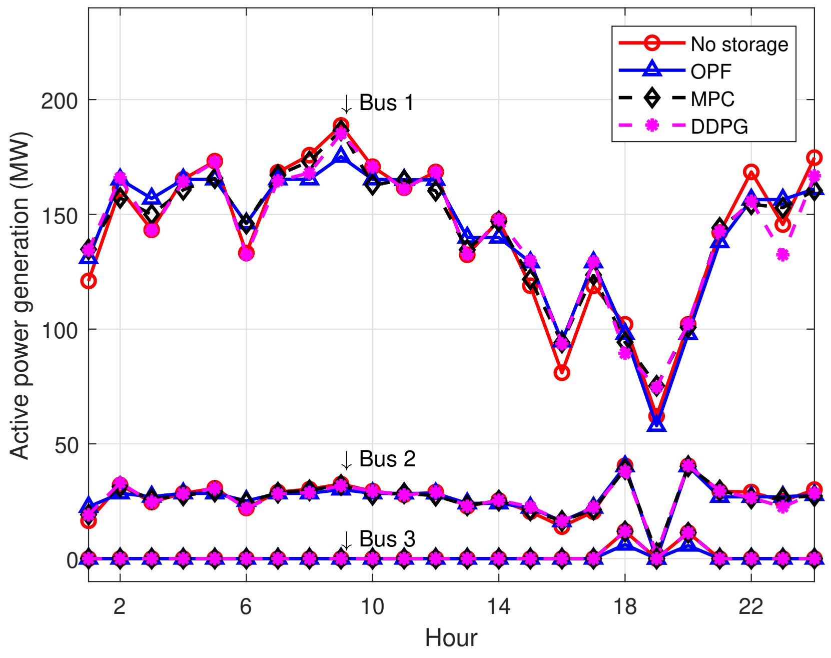


Figure 5.10: The active power generated at bus 1, 2, 3.

mation. Thus it cannot be implemented in practical systems but it provides the best possible performance that can be used as a benchmark. The DDPG algorithm obtains a lower average hourly cost than MPC, and the DDPG has a much lower computation complexity. The MPC performance is influenced by the prediction accuracy and the DDPG performance depends on parameters chosen in Table 6.2. Both require training before real-time operations. MPC needs to train the prediction model while DDPG needs to train the agent. However, no prediction is required during the real-time operation for DDPG. Therefore, given the same power system and storage setting, the computation time of DDPG is almost the same as a single-period OPF, while the MPC computation time depends on the chosen optimization horizon τ. When the complexity of the system and the number of storage units increase, the DDPG will still have the same order of complexity as the single-period OPF, but the complexity of MPC will increase dramatically.

5.4 Conclusion

This chapter has proposed a reinforcement learning based DDPG algorithm for real-time optimal power flow scheduling in a microgrid equipped with energy storage and power generations from both conventional and wind sources. Unlike conventional Q-learning algorithms that can work only with discrete action spaces, the proposed DDPG algorithm can deal with continuous actions by using the gradient of the policy. The agent of DDPG was trained by searching along the policy gradient of the cost function, and the trained agent was then applied to real world data for testing. The DDPG results were compared to those obtained from a classic MPC-based approach. Compared with MPC, DDPG has a lower average cost. In addition, DDPG does not need muti-period prediction and multi-period optimization as in MPC, thus the proposed DDPG algorithm has a much lower complexity than MPC and can operate efficiently in real time. Future works can be done to design a DDPG agent that can learn in real-time during power system operations. In this case, the parameters of the actor network can be updated in real time to adapt to the changing environment.

6 Optimum Scheduling of Truck-based Mobile Energy Couriers Using Deep Deterministic Policy Gradient

The intermittent nature of RES can be partly compensated through the employment of energy storage systems (ESS) such as batteries and hydrogen fuel cells. ESS can store excess energy during periods of high production and releasing it during periods of high demand. This helps to stabilize the grid and improve the reliability of renewable energy sources [7]. Recently there have been growing interests of adding mobility to ESS to achieve mobile energy storage. Mobile energy storage offers several advantages over SESS, including flexibility, rapid deployment, redundancy, reduced infrastructure costs, and environmental benefits. Mobile energy storage systems can be easily moved to different locations as needed, making it more flexible than its static counterpart [11]. It can also be rapidly deployed in response to emergencies or to support temporary events. Mobile energy storage provides backup power in situations where static energy storage systems have failed or are unavailable, ensuring a continuous supply of energy [12]. Additionally, mobile energy storage can reduce the need for expensive grid infrastructure upgrades and can be used to power electric vehicles, reducing greenhouse gas emissions and air pollution [13].

The control and scheduling of a distribution grid with mobile energy storage is complex and challenging due to the dynamic and uncertain nature of both the transportation networks and the RES, as well as the need to minimize operational costs while ensuring efficient and reliable energy distribution over the entire grid. This requires a comprehensive system model combining the scheduling of both transportation network and power flow. Consequently, various design and scheduling techniques have been proposed to optimize power grids with mobile energy storage systems.

In this chapter we propose a new platform of truck-based mobile energy couriers (MEC), which consists a fleet of trucks equipped with high-density inverters, converters, capacitor bands, and energy storage devices. The MEC design problem is formulated as a non-convex optimization problem that aims to minimize the grid operation cost under the constraints imposed by conventional and renewable energy sources, dynamic energy demands, power flows, energy storage, transportation networks, and their dynamic interactions.

nomenclature

Sets and Index

$\mathcal{A}$	Action space of DDPG
$\mathcal{B}$	Set of energy storage locations
$\mathcal{E}, e$	Set and index of links
$\mathcal{G}$	Set of generators
$\mathcal{K}, k$	Set and index of energy carriers
$\mathcal{L}$	Set of changeable loads
$\mathcal{N}, n, i, j$	Set and index of buses
$\mathcal{O}$	Set of origin-destination pairs
$\mathcal{P}_{uw}, p$	Set and index of paths between O-D pair (u, w)
$\mathcal{S}$	State space of DDPG
$\mathcal{T}, t$	Set and index of hours
$\mathcal{V}, u, w$	Set and index of nodes

Parameters for transportation and power network

$c^{\max}$	Energy storage and carrier capacity
δ_{ij}	Indicator of whether $i = j$
γ_e	Charge/discharge efficiency
λ_t^{ij}	Minimal travel time between bus i and bus j at hour t
$P_g^{\min}, P_g^{\max}$	Minimal/maximal active power generation at generator g
$q^{\min}, q^{\max}$	Minimal/maximal energy storage and carrier charge/discharge rate

$Q_g^{\min}, Q_g^{\max}$	Minimal/maximal reactive power generation at generator g
$S_{ij}^{\max}$	Maximal reactive power flow from bus i to bus j
τ_0^e	Free-flow travel time on link e
τ_t^{uw}	Minimal travel time between O-D pair (u, w) at hour t
θ_{ij}	Angle difference between bus i and bus j
$V_n^{\min}, V_n^{\max}$	Minimal/maximal voltage magnitude at bus n
ζ_{pe}	Binary indicator of whether path p contains link e
B_{ij}	Susceptance between bus i and bus j
c_t^k	State of charge of MEC k at hour t
c_{2i}, c_{1i}, c_{0i}	Cost coefficients of generator at bus i
d_t^{uw}	Traffic demand between O-D pair (u, w) at hour t
f^e	Capacity of link e
f_t^e	Traffic flow of link e at hour t
f_t^p	Path flow of path p at hour t
G_{ij}	Conductance between bus i and bus j
$P_{i,t}^b$	Renewable power generated at bus $i \in B$ at hour t
$P_{i,t}^g, Q_{i,t}^g$	Active/reactive power generated at bus $i \in \mathcal{G}$ at hour t
$P_{n,t}, Q_{n,t}$	Active/reactive power injected to bus n at hour t
q_t^k	Charging/discharging rate of MEC k at hour t
$S_{ij,t}$	Reactive power flow from bus i to bus j at hour t
$V_{n,t}$	Voltage magnitude at bus n at hour t

Parameters for DDPG

α	Parameter to adjust the weight of operation reward
ϵ	Soft update parameter
γ	Discount factor of DDPG
λ_t^k	Time slot left for k-th MEC to reach its destination at hour t
$\mathcal{M}$	Mini-batch of transitions from the replay buffer
$\mathcal{P}$	State transition probability of MDP
μ	Deterministic policy
$\mathcal{R}$	Replay buffer
θ^Q, θ^μ	Weights of critic network and actor network
$\theta^{Q^{'}}, \theta^{\mu^{'}}$	Weights of target networks
a_t	Action of the agent at time t
$C_{\text{tr}}, C_{\text{q}}, C_{\text{soc}}$	Penalty parameters for MEC to travel, exceed capacity, and not fully discharged
d_t^k	Destination of the k-th MEC at hour t
$I(\mathcal{M})$	Time index of transitions in B
l_c, l_a	Learning rate of critic network and actor network
M	Total training timesteps
o_t^k	Origin of the k-th MEC at hour t
$Q(s_t, a_t)$	Action-value function of taking a_t from s_t
r_t	Step reward from the environment at time t
r_t^{op}	Operation reward at time t
$r_t^{k,\text{q}}, r_t^{k,\text{soc}}$	Reward of capacity on k-th MEC at time t
$r_t^{k,\text{tr}}$	Reward of transition on k-th MEC at time t

s_t	State of the environment at time t
y_i	Target critic value of transition indexed by i

We propose to solve the problem by combining optimal power flow (OPF) with DDPG, where OPF is used to control the grid energy generation and power flow, and DDPG is used to control the mobility pattern and charging/discharging schedule of MEC. The proposed DDPG agent can learn the stochastic behaviors of various environment parameters, such as RES, different loads, and traffic conditions, by interacting with the training data through a balanced exploitation and exploration. The trained agent can optimize the MEC operations in real time by strategically dispatch them to different locations to agilely shape the peaks in generation and load while providing flexible controllability to the distribution grid. With such a concept, the excessive power from the RES can be directly routed to the point of local demand or stored in the MEC fleet. Simulation results demonstrated that the proposed MEC framework with DDPG can achieve significant cost reduction compared to conventional systems with static energy storage systems.

6.1 Problem formulation

The MEC scheduling problem divided into two sub-problems. The first one is a multi-period optimal power flow (MP-OPF) problem on a power distribution network equipped with RES, ESS, and MEC as shown in Fig. 6.1. The second one focuses on the optimum scheduling of the MECs, which can be formulated as a TA problem on a transportation network. In this chapter, we assume that the MECs are owned and controlled by the utility who also monitors and collects data from the distribution network. Each MEC will only travel in the transportation network according to its scheduled destination and established route. Compared to other vehicles in the network, the number of MECs is so small that they have negligible impact on the parameters of the transportation network such as the origin-destination (O-D) traffic demands, congestion time, and user equilibrium (UE). We will present the models of the distribution and transportation network in the following two subsections.

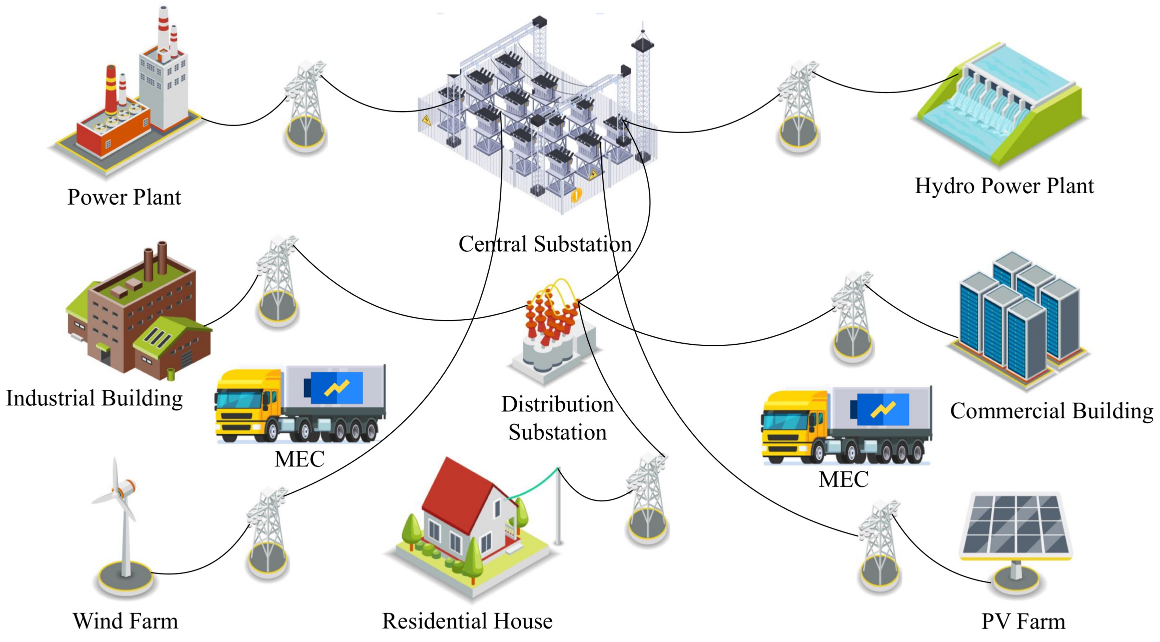


Figure 6.1: A distribution grid with both MEC and RES.

6.1.1 Transportation Network Model

The transportation network is usually defined by a connected graph $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denotes the set of vertices in the graph or nodes in the network, and $\mathcal{E}$ is the set of edges in the graph or links in the network.

The time is divided into non-overlapping time slots with the duration Δt as one hour in this paper. The vehicle entering rate on link $e \in \mathcal{E}$ at time t is called the traffic flow on e, denoted as f_t^e. The time for vehicles to travel through link e at time t due to the congestion of other vehicles is determined by the traffic flow f_t^e as the Bureau of Public Roads (BPR) function [150]

$$\tau_t^e(f_t^e) = \tau_0^e \left[1 + 0.15 \left(\frac{f_t^e}{f^e}\right)^4\right], \tag{6.1}$$

where τ_0^e is the free-flow travel time on link e without congestion and f^e is the capacity of link e.

Each vehicle traveling within the transportation network, including the MEC, leaves from its origin node $u \in \mathcal{V}$ and plans to arrives at its destination node $w \in \mathcal{V}$ through links. The origin-destination (O-D) pair is denoted by (u, w), and the set of all the O-D pairs is denoted by $\mathcal{O}$. The number of vehicles that all share the same O-D pair at time t is called the traffic demand between (u, w) at t, and it is denoted as d_t^{uw}. Since the graph is connected, there exits at least one path p connecting u and w for any chosen pair (u, w). Denote the set of paths between (u, w) as $\mathcal{P}_{uw}$.

The vehicle entering rate on path $p \in \mathcal{P}_{uw}$ at time t is called the path flow on p, denoted as f_t^p. The traffic flow f_t^e on link e can be calculated by adding the path flow of all the paths that contains link e as

$$f_t^e = \sum_{(u,w)\in\mathcal{O}} \sum_{p\in\mathcal{P}_{uw}} f_t^p \zeta_{pe}, \tag{6.2}$$

where $\zeta_{pe} = 1$ if link e is on path p and 0 otherwise.

User equilibrium assumes that each driver decides to use the fastest path, and the traffic flow will reach a stable state [151]. Under the UE assumption, the stable traffic flow is obtained by solving the following optimization problem [152]:

$$\min_{f_t^e} \quad \sum_{e\in\mathcal{E}} \int_0^{f_t^e} \tau_t^e(\omega) d\omega, \tag{6.3}$$

$$\text{s.t.} \quad (6.2),$$

$$d_t^{uw} = \sum_{p\in\mathcal{P}_{uw}} f_t^p, \tag{6.4}$$

$$f_t^p \geq 0, \tag{6.5}$$

where the objective function in (6.3) is the sum of the integration of the travel time between 0 and the traffic flow f_t^e over all links in the network. It has no physical meaning and is only used for the optimization. Besides the constraint of the relation between the traffic link and path link in (6.2), constraint (6.4) guarantees the satisfaction of the traffic demand, and constraint (6.5) ensures that path flow is non-negative.

Despite that (6.3) only solves the UE traffic flow at time t, if the O-D demands d_t^{uw} are given for different time slot t, the UE traffic flow for the whole time horizon $\mathcal{T}$ can be obtained by solving it for $|\mathcal{T}|$ times sequentially. Once the UE traffic flow f_t^e is obtained, the travel time $\tau_t^e(f_t^e)$ can be calculated. The Dijkstra's algorithm can be applied to find the fastest path for the MEC at time t between each O-D pairs, and the corresponding minimal travel time is denoted as τ_t^{uw}.

6.1.2 AC Multi-period Optimal Power Flow

The multi-period optimal power flow model for a distribution network with RES, ESS and MEC is formulated in this subsection.

Consider a power distribution network consists of $|\mathcal{N}|$ buses, $|\mathcal{G}|$ conventional generators, $|\mathcal{B}|$ buses hosting energy storage and renewable energy sources such as PV farm and wind farm, $|\mathcal{L}|$ changeable loads including residential, commercial, and industrial loads. There are $|\mathcal{K}|$ MECs in the distribution network, and they can be regarded as special cases of SESS.

Power generation is performed by both conventional power generators and renewable energy sources such as solar and wind. Denote the active and reactive power generated by the conventional generator on bus $i \in \mathcal{G}$ during time slot t as $P_{i,t}^g$ and $Q_{i,t}^g$, and they are bounded as follows

$$\begin{aligned} P_i^{\min} \leq P_{i,t}^g \leq P_i^{\max}, \\ Q_i^{\min} \leq Q_{i,t}^g \leq Q_i^{\max}. \end{aligned} \tag{6.6}$$

The active power generation is limited by the ramp rate as

$$-R_i^{\text{down}} \leq P_{i,t}^g - P_{i,t-1}^g \leq R_i^{\text{up}}. \tag{6.7}$$

The power generated by the PV panel and wind turbine on bus $i \in \mathcal{B}$ during time slot t is denoted as

$$P_{i,t}^b = P_{i,t}^{\text{pv}} + P_{i,t}^{\text{wind}}, \tag{6.8}$$

where $P_{i,t}^{\text{pv}}$ and $P_{i,t}^{\text{wind}}$ represent the active power generated by the PV panel and wind turbine, respectively. Both rely on the weather condition and changes dynamically with respect to time.

The network constraints contain constraints for bus voltages and branch flows. For any bus $n \in \mathcal{N}$ in the network, the voltage magnitude during time slot t should be bounded as

$$V_n^{\min} \leq V_{n,t} \leq V_n^{\max} \tag{6.9}$$

The active and reactive power injected to bus $n \in \mathcal{N}$ during time slot t are denoted by $P_{n,t}$ and $Q_{n,t}$, respectively. They need to satisfy the load flow equations

$$\begin{aligned} P_{n,t} &= \sum_{i \in N} V_{n,t} V_{i,t} (G_{ni} \cos \theta_{ni} + B_{ni} \sin \theta_{ni}), \\ Q_{n,t} &= \sum_{i \in N} V_{n,t} V_{i,t} (G_{ni} \sin \theta_{ni} - B_{ni} \cos \theta_{ni}), \end{aligned} \tag{6.10}$$

where G_{ni} and B_{ni} are the conductance and susceptance between buses n and i, respectively. $\theta_{ni} = \theta_n - \theta_i$ is the angle difference between buses n and i.

The reactive power flow on branch (i, j) at time slot t is denoted by $S_{ij,t}$, where $i, j \in \mathcal{N}$ and there is a branch between buses i and j. The reactive power should be bounded by

$$0 \leq |S_{ij,t}| \leq S_{ij}^{\max}. \tag{6.11}$$

The network delivers electricity from the transmission network to consumers, including residential, commercial, and industrial sectors. The active and reactive power demands at bus $i \in \mathcal{L}$ during time slot t are denoted as $P_{i,t}^l$ and $Q_{i,t}^l$, respectively.

MECs interact with the grid by traveling among the locations of the buses and charging or discharging their batteries, contributing to grid services or meeting their own energy needs. Denote the SOC of the k-th MEC at the beginning of time slot t as c_t^k. The initial SOC of all energy storage devices is set to be zero. In each time slot t, each energy storage device has the options to either charge, discharge, or stay idle. Denote the charging or discharging rate of the k-th MEC as q_t^k and the charging/discharging efficiency as γ_e. The dynamics of energy storage can be modeled by the following first-order difference equation as

$$c_{t+1}^k = \begin{cases} c_t^k + \Delta t q_t^k \gamma_e, & q_t^k > 0 \\ c_t^k, & q_t^k = 0 \\ c_t^k + \Delta t q_t^k / \gamma_e & q_t^k < 0. \end{cases} \tag{6.12}$$

The charging/discharging rates are bounded by the maximum rate and storage capacity as

$$-q^{\min} \leq q_t^k \leq q^{\max} \tag{6.13}$$

$$0 \leq c_t^k \leq c^{\max} \tag{6.14}$$

With all the generators, loads, and renewable energy sources, the active and reactive power at each bus $n \in \mathcal{N}$ should be balanced by

$$\begin{aligned} P_{n,t} &= \sum_{i \in \mathcal{G}} P^g_{i,t}\delta_{in} + \sum_{i \in \mathcal{B}} P^b_{i,t}\delta_{in} - \sum_{k \in \mathcal{K}} q^k_t \delta_{kn,t} - \sum_{i \in \mathcal{L}} P^l_{i,t}\delta_{in}, \\ Q_{n,t} &= \sum_{i \in \mathcal{G}} Q^g_{i,t}\delta_{in} - \sum_{i \in \mathcal{L}} Q^l_{i,t}\delta_{in}, \end{aligned} \tag{6.15}$$

where δ_{in} is the Kronecker delta with value 1 if $i = n$ and 0 otherwise. Similarly $\delta_{kn,t} = 1$ if the k-th MEC is at bus n during time slot t.

Define the invertible mapping from buses in the power system network to the nodes in the transportation network as $\phi : \mathcal{N} \to \mathcal{V}$. For example, buses $m, n \in \mathcal{N}$ are mapped to nodes $u, w \in \mathcal{V}$ by $\phi(m) = u, \phi(n) = w$, respectively. Then the minimal travel time between bus n and bus k at time t, in the unit of the number of slots with duration Δt, can be calculated as

$$\lambda^{mn}_t = \left\lceil \frac{\mathcal{T}^{\phi(m)\phi(n)}_t}{\Delta t} \right\rceil = \left\lceil \frac{\mathcal{T}^{uw}_t}{\Delta t} \right\rceil. \tag{6.16}$$

The MEC scheduling variable $\delta_{kn,t}$ is also constrained by

$$\sum_{n \in \mathcal{N}} \delta_{kn,t} \le 1, \qquad \forall t \in \mathcal{T}, \quad \forall k \in \mathcal{K}, \tag{6.17}$$

which indicates that the k-th MEC can only be at one bus or on a path at each time slot. The MEC scheduling variable is also constrained by the minimal travel time between buses. For example, if the k-th MEC arrives at bus m at time slot t_0, then the scheduling variable of the k-th MEC then needs to satisfy

$$\begin{aligned} \delta_{kn,t} = 0, \qquad & t \in \{t_0, t_0 + 1, \dots, t_0 + \lambda^{mn}_{t_0}\} \\ & \forall n \in \mathcal{N} \setminus \{m\}, \quad \forall k \in \mathcal{K}, \end{aligned} \tag{6.18}$$

which indicates that the k-th MEC at bus m will need at least $\lambda^{mn}_{t_0}$ time slots to arrive at bus n.

The objective of MP-OPF is to find the power flow solution that minimizes the cost function. In this paper, a polynomial cost function with c_{2i}, c_{1i}, and c_{0i} as the coefficients for the generator at bus $i \in \mathcal{G}$ is used. The MP-OPF can be formulated as

$$\begin{aligned} \text{min.} \quad & \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{G}} c_{2i} \left(P^g_{i,t}\right)^2 + c_{1i} P^g_{i,t} + c_{0i}, \\ \text{s.t.} \quad & (6.6) - (6.15), (6.17), (6.18). \end{aligned} \tag{P1}$$

where the decision variables are the power generation $P^g_{i,t}, Q^g_{i,t}$ for $i \in \mathcal{G}$, the MEC scheduling variable $\delta_{kn,t}$, and the MEC energy storage behavior q^k_t for $k \in \mathcal{K}, n \in \mathcal{N}$, during the whole time horizon $t \in \mathcal{T}$.

This is a non-convex optimization problem where no guarantees on the performance can be provided by theoretical calculation or solvers. It can be divided into two sub-problems. In the first sub-problem, we can fix the ESS scheduling variables q^k_t and the MEC scheduling variable $\delta_{kn,t}$ for $k \in \mathcal{K}$, $n \in \mathcal{N}$ and $t \in \mathcal{T}$, and solve a single-period OPF problem as

$$\begin{aligned} C(\delta_{kn,t}, q^k_t) = \min_{P^g_{i,t}, Q^g_{i,t}} & \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{G}} c_{2i} \left(P^g_{i,t}\right)^2 + c_{1i} P^g_{i,t} + c_{0i} \\ \text{s.t.} \quad & (6.6) - (6.11), (6.15). \end{aligned} \tag{P2}$$

This can be solved by classical OPF methods or solvers.

Once the OPF problem in (P2) is solved, the second sub-problem can solve the MEC and ESS scheduling problem as

$$\begin{aligned} \min_{\delta_{kn,t}, q^k_t} & \; C(\delta_{kn,t}, q^k_t) \\ \text{s.t.} \quad & (6.12) - (6.14), (6.17), (6.18). \end{aligned} \tag{P3}$$

6.2 DDPG approach

Recall that reinforcement learning (RL) is formulated based on Markov Decision Process (MDP), which consists of a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the state transition probability, r is the immediate scalar reward, and $\gamma \in (0, 1]$ is a discount factor. In this section, a brief DDPG framework will be introduced and a DDPG agent will be formulated to solve the MEC and ESS scheduling control in (P3).

6.2.1 DDPG formulation for MEC Control

An MEC control environment that fits the MDP framework is developed for the DDPG formulation. To simplify the environment, especially to reduce the dimension of the state space, the MEC scheduling problem is described by three pieces of information: o^k_t, d^k_t, and λ^k_t, which are, respectively, the origin, destination of the k-th MEC at slot t, and the number of remaining time slots for the MEC to reach its destination at slot t.

1. State space $\mathcal{S}$. The state at time slot t consists of the difference between active load demand and renewable power generation at the corresponding buses, the origin, desti-

nation, the number of time slots remaining to reach the destination, and SOC for each MEC. The state vector is defined as

$$s_t = \left[\sum_{i\in\mathcal{L}} P^l_{i,t}\delta_{in} - \sum_{i\in\mathcal{B}} P^b_{i,t}\delta_{in}, o^k_t, d^k_t, \lambda^k_t, c^k_t \right], \quad (n \in \mathcal{N}, k \in \mathcal{K}). \tag{6.19}$$

2. Action space $\mathcal{A}$. The action at time slot t includes the MEC destination and the battery charging/discharging schedule

$$a_t = \left[d^k_t, q^k_t\right], \qquad (k \in \mathcal{K}). \tag{6.20}$$

3. Transition Probability $\mathcal{P}$. The transition probability is used to describe the probability of state transitions for a given action. In the DDPG formulation, the DDPG agent will learn the transition probability through interactions with the environment, and there is no need to explicitly represent the transition probability.

4. Reward function r. The reward function is determined by the operation costs of generators and MEC. The objective of (P3) is to minimize the total operation cost of generators, so the corresponding reward can be calculated by using the negative generator operation cost as

$$r^{\text{op}}_t = -\left(\sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{G}} c_{2i} \left(P^g_{i,t}\right)^2 + c_{1i} P^g_{i,t} + c_{0i} \right). \tag{6.21}$$

In addition to the generator operation cost, we also need to consider several other costs and penalties that are related to MEC transition and constraints. If the k-th MEC is on the road, then the reward related to the transition cost of the k-th MEC at slot t is

$$r^{k,\text{tr}}_t = -C_{\text{tr}}, \;\; \text{if } \lambda^k_t > 0, \tag{6.22}$$

where C_{tr} is the cost associated with MEC transportation.

We define the following two rewards that are related to the negative of the penalties for violating of the energy storage constraints

$$r^{k,\text{q}}_t = -C_{\text{q}}, \;\; \text{if (6.14) is violated} \tag{6.23}$$

$$r^{k,\text{soc}}_t = -C_{\text{soc}} c^k_t, \;\; \text{if } t = 24 \tag{6.24}$$

where C_{q} is the penalty for violating the SOC constraints on k-th MEC at time t, and C_{soc} is the penalty if the final SOC of the k-th MEC is not zero.

The reward function is a linear combination of the rewards above as

$$r_t = \alpha r_t^{\text{op}} + \sum_{k \in \mathcal{K}} r_t^{k,\text{tr}} + r_t^{k,\text{q}} + r_t^{k,\text{soc}} \tag{6.25}$$

where α is a parameter to adjust the weight of operation reward compared with other rewards.

5. Discount factor γ: The discount factor is set to 0.99 in this paper.

6.2.2 DDPG framework

DDPG combines the policy gradient method and Deep Q-Network (DQN) by using an actor-critic framework. The DDPG agent contains two DNNs, which are denoted as the actor network and critic network, respectively.

The actor network is used to model the deterministic policy $a_t = \mu(s_t|\theta^\mu)$, that is, the action that the agent should take for a given state s_t, with θ^μ being the DNN weight coefficients for the actor network. The critic network is used to model the action-value function (also known as the Q-function), $Q(s_t, a_t)$, which is the expected discounted reward after taking an action a_t at the state s_t. The output of the critic network in DDPG can be represented as

$$Q(s_t, \mu(s_t|\theta^\mu)|\theta^Q), \tag{6.26}$$

where θ^Q represent the DNN weight coefficients for the critic network. With the critic network defined in (6.26), we can evaluate the gradient of the Q-function with respect to the policy parameters θ^μ, such that the DDPG agent can iteratively search along the policy gradient to optimize the policy (action) during the learning process.

DDPG training is performed by using mini-batches randomly sampled from a replay buffer $\mathcal{R}$, which stores the previous $|\mathcal{R}|$ transitions before the current time slot t, that is $(s_i, a_i, r_i, s_{i+1})_{i \in \{t-|\mathcal{R}|+1,\ldots,t-1,t\}}$. During the time slot t, a mini-batch $\mathcal{M} \subset \mathcal{R}$ with $|\mathcal{M}|$ transitions $(s_i, a_i, r_i, s_{i+1})_{i \in I(\mathcal{M})}$ are randomly sampled from the replay buffer, where $I(\mathcal{M})$ is the time index of transitions in the mini-batch $\mathcal{M}$.

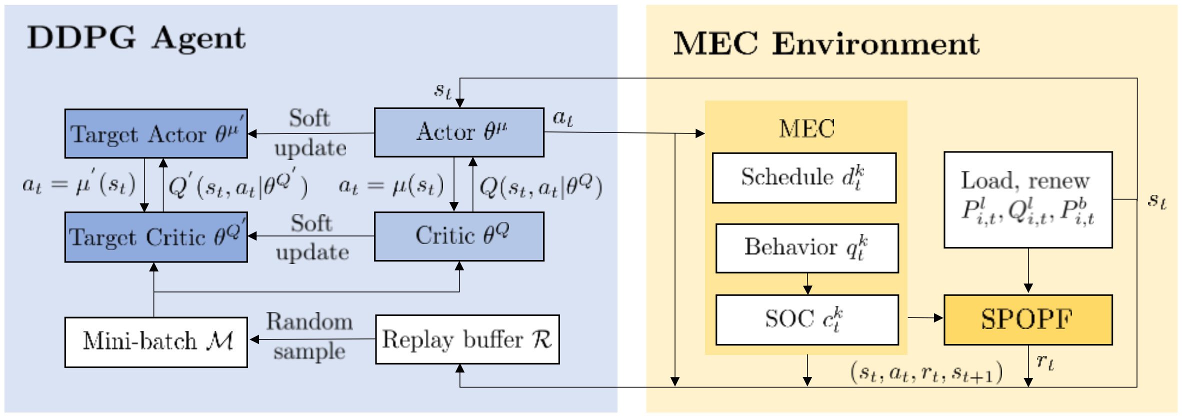


Figure 6.2: Diagram of DDPG training in MEC environment.

The actor network will be updated by maximizing the current Q-function value on $\mathcal{M}$ as

$$\theta^{\mu} \leftarrow \theta^{\mu} + l_a \nabla_{\theta^{\mu}} \frac{1}{|\mathcal{M}|} \sum_{i \in I(\mathcal{M})} \left(Q(s_i, \mu(s_i|\theta^{\mu})|\theta^{Q} \right) \tag{6.27}$$

where l_a is the learning rate of the actor network.

In order to improve the stability of the learning process, DDPG employs two target networks for the critic and actor networks with weights $\theta^{\mu'}$ and $\theta^{Q'}$, respectively. The target critic value for the i-th transition in $\mathcal{M}$ under the current target actor is calculated by

$$y_i = r_i + \gamma Q' \left(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'} \right), \; i \in I(\mathcal{M}). \tag{6.28}$$

Then the critic network can be updated by minimizing the mean squared error between the target critic value and the current critic value as

$$\theta^{Q} \leftarrow \theta^{Q} + l_c \nabla_{\theta^{Q}} \frac{1}{|\mathcal{M}|} \sum_{i \in I(\mathcal{M})} \left(Q(s_i, \mu(s_i|\theta^{\mu})|\theta^{Q}) - y_i \right)^2, \tag{6.29}$$

where l_c is the learning rate of the critic network. The two target networks are updated softly with parameter $\epsilon \ll 1$ by:

$$\begin{aligned} \theta^{Q'} &\leftarrow \epsilon \theta^{Q} + (1-\epsilon)\theta^{Q'}, \\ \theta^{\mu'} &\leftarrow \epsilon \theta^{\mu} + (1-\epsilon)\theta^{\mu'}. \end{aligned} \tag{6.30}$$

This update process slows down the update of the networks thus improves the stability of learning.

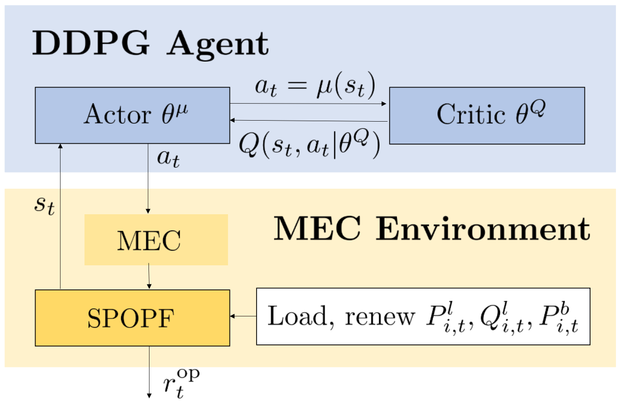


Figure 6.3: Diagram of DDPG testing in the MEC environment.

6.2.3 DDPG agent for MEC control

The DDPG agent will be trained for M episodes. During each training episode, one day is randomly sampled from the dataset. The MECs are initialized with zero SOC, and they are initially located at a bus with a renewable energy source $n \in B$. The initial destination for each MEC is set to be the same as its initial location. At each training time step, the agent interacts with the environment by executing actions from the actor network. MECs are scheduled to new destinations, and they might charge or discharge once they arrived at their destination. A single-period OPF with renewable energy source and MEC behavior is solved and the operation cost is recorded. Along with other rewards, the step reward is calculated. This single transition is then stored to the replay buffer.

At the same time, the actor and critic networks are updating their weights according to (6.27) and (6.29), respectively. The target networks are updated softly according to (6.30). The system diagram for DDPG training is shown in Fig. 6.2. The whole algorithm of DDPG training is given in Algorithm 8.

After the training is done, the trained agent can be used to interact with the MEC environment to solve the MEC control in real time. The system diagram for DDPG testing is shown in Fig. 6.3.

6.3 Case Studies

Simulation results are presented in this section to demonstrate the performance of the proposed MEC framework with DDPG. The DDPG algorithm is implemented using

Algorithm 8 DDPG training in MEC environment

```
Require: the transportation network $(\mathcal{V}, \mathcal{E})$, the power network with $\mathcal{B}, \mathcal{G}, \mathcal{L}, \mathcal{N}$, mapping
    function $\phi$, and $|\mathcal{K}|$ MECs. Structure and learning rate of actor and critic network,
    mini-batch $\mathcal{M}$.
 1: Initialization: Initialize MECs with zero SOC at bus $i \in \mathcal{B}$, initialize all network
    weights $\theta^{\mu}, \theta^{Q}, \theta^{\mu'}, \theta^{Q'}$, build replay buffer.
 2: for timestep = 1 to $M$ do
 3:     for $t \in \mathcal{T}$ do
 4:         Observe state $s_t$ from MECs and energy profiles, select action $a_t = \mu(s_t|\theta^{\mu})$
 5:         Execute $a_t$ in the MEC environment: control the MECs, run SPOPF, and calculate
            reward $r_t$
 6:         Store transition $(s_t, a_t, r_t, s_{t+1})$ in the replay buffer
 7:         Randomly sample a mini-batch $\mathcal{M}$ from the replay buffer
 8:         Compute targets for $i \in I(\mathcal{M})$ by (6.28)
 9:         Update the critic parameters $\theta^{Q}$ by (6.29)
10:         Update the actor parameters $\theta^{\mu}$ by (6.27)
11:         Update the target networks by (6.30)
12:     end for
13: end for
Ensure: Network parameters $\theta^{\mu}$, $\theta^{Q}$
```

Stable-Baselines3 [153] and pandapower [154] in Python. All simulations are performed on a workstation with a 6-core Intel Core i7-5820K CPU operating at 3.3 GHz, NVIDIA GeForce GTX 950 GPU, and 32 GB of random access memory (RAM).

6.3.1 Simulation Environment

The transportation network used for simulation is the Sioux Falls transportation network as shown in Fig. 6.4 where each marker represents a node labeled with its node ID. The data for all $|\mathcal{V}| = 24$ nodes, $|\mathcal{E}| = 74$ links, and all O-D traffic demand d^{uw} for $(u, w \in \mathcal{E})$ are downloaded from [155]. The original O-D traffic demands of Sioux Falls are shown in Fig. 6.5. In order to simulate the demand changes in the 24 hours in a day, the typical hourly traffic volume data on all links are downloaded at the City of Sioux Falls GIS website [156]. The traffic volume data are summed up to each hour and the proportion of

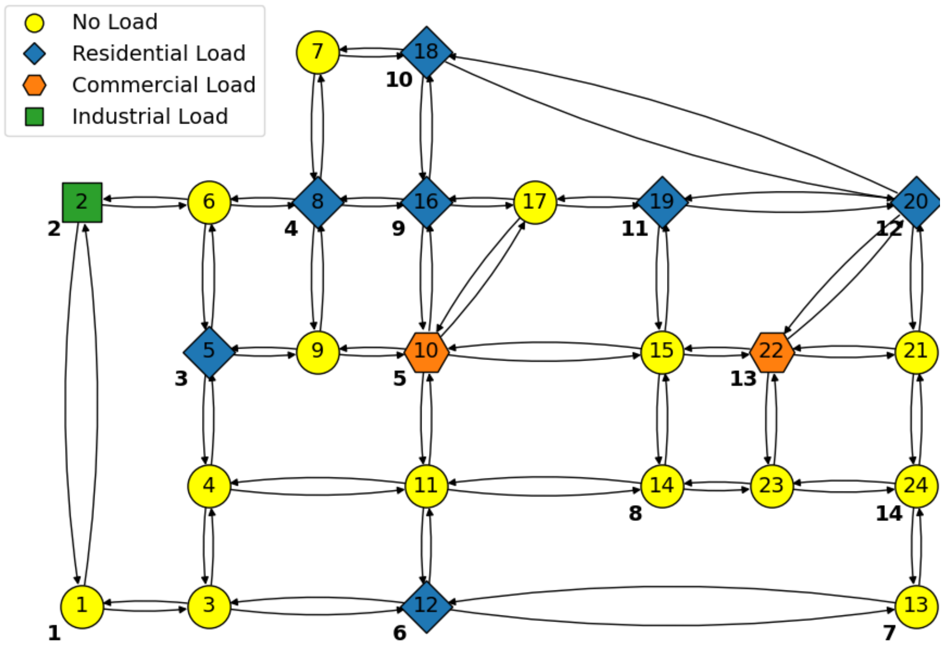


Figure 6.4: Sioux Falls Transportation Network.

the hourly traffic demand is calculated by dividing hourly demand by the total demand in the day. Then the original O-D demands are scaled by the this proportion as shown in Fig. 6.6 to hourly traffic demands d_t^{uw} for $(u, w \in E)$.

The distribution network is implemented on a modified IEEE 14-bus system as shown in Fig. 6.7, where generators are located at bus $\mathcal{G} = \{1, 2, 3, 6, 8\}$. The PVs and wind turbines are installed at buses $\mathcal{B} = \{3, 6, 8, 11\}$, with the profiles of year 2021 obtained from [157] and [158]. The load demands at bus $\mathcal{L} = \{2, 3, 4, 5, 6, 9, 10, 11, 12, 13\}$ change according to the load type in Table 6.1 and the corresponding load profile of the same year from [159–161]. The buses and loads in the distribution network can be found at Fig. 6.4, where the bold number at the left bottom corner of a node represents the bus ID. For illustration, the renewable energy profile and load profile on a bus on July 1st, 2021 are shown in Figs. 6.9 and Fig. 6.8, respectively.

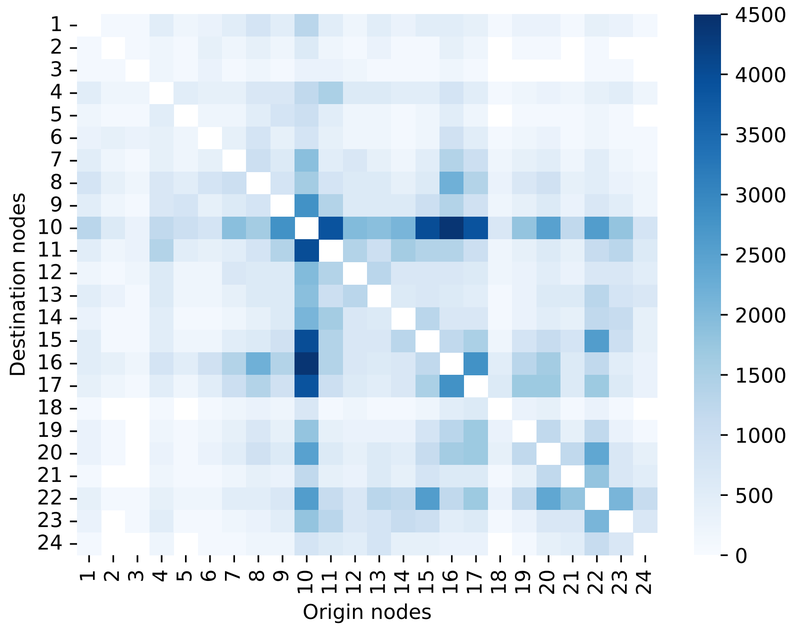


Figure 6.5: Sioux Falls O-D Demands.

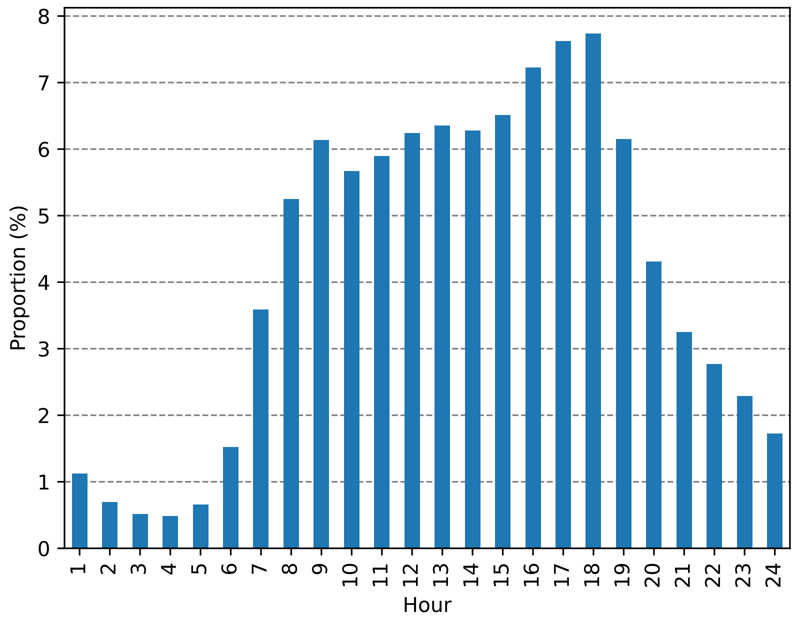


Figure 6.6: Typical hourly traffic demand proportion.

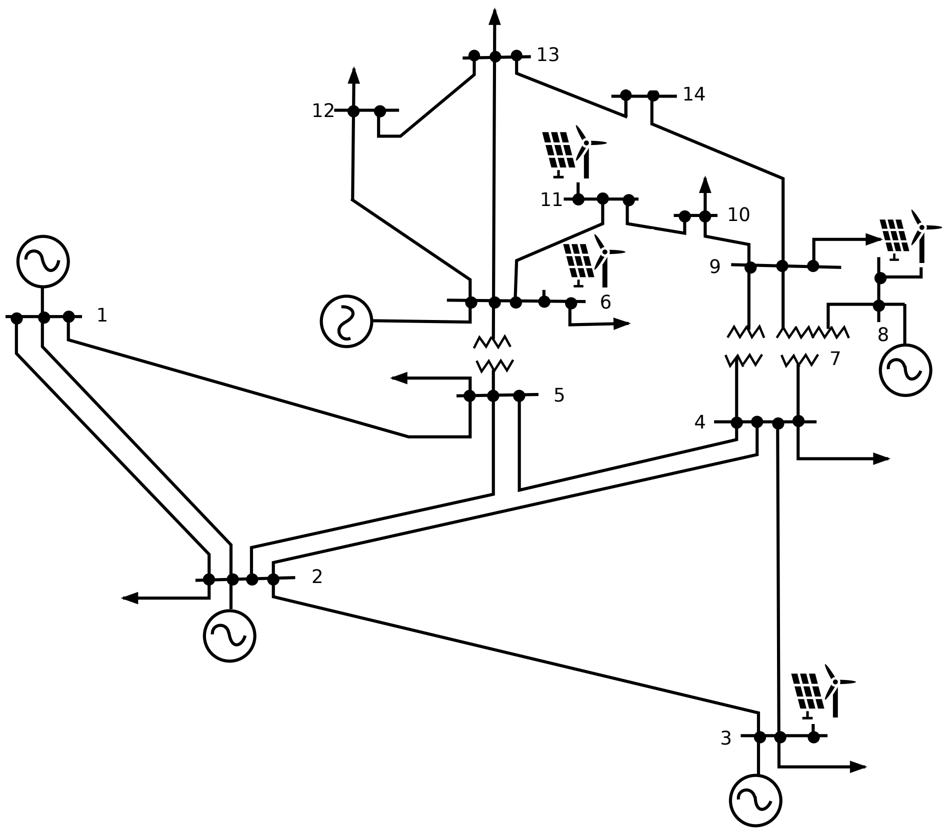


Figure 6.7: Modified IEEE 14-bus System.

Table 6.1: Bus to node mapping and load type

Bus ID	**Node ID**	**Load type**
1	1	-
2	2	I
3	5	R
4	8	R
5	10	C
6	12	R
7	13	-
8	14	-
9	16	R
10	18	R
11	19	R
12	20	R
13	22	C
14	24	-

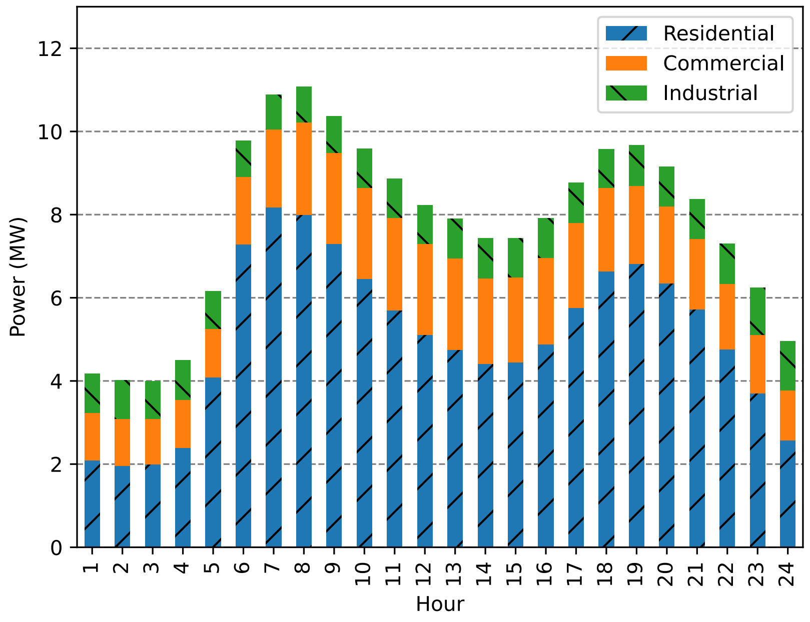


Figure 6.8: The single load profile on July 1st, 2021.

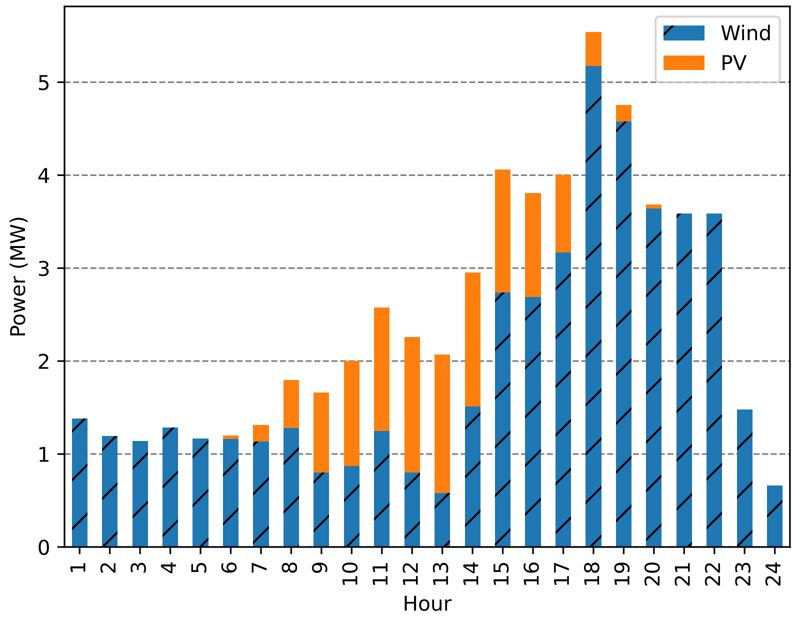


Figure 6.9: The single renewable energy profile on July 1st, 2021.

The cost coefficients and the other physical constraints remain the same as the original

Table 6.2: DDPG parameters

Notation	Description	Value
Number of neurons in DDPG hidden layers		128,128
l_a	DDPG actor learning rate	0.001
l_c	DDPG critic learning rate	0.001
ϵ	DDPG soft update coefficient	0.005
$\lvert\mathcal{R}\rvert$	Experience replay buffer size	1000k
$\lvert\mathcal{M}\rvert$	Batch size	100
M	Number of timesteps trained	100k
$\mathcal{T}$	Episode length	24
C_{tr}	Transition cost	0.5
C_{q}	SOC violation penalty	0.5
C_{soc}	Remain energy penalty	0.1
α	Weight of operation reward	0.01

IEEE 14-bus system. The energy storage units on each MEC are identical with a maximum charge/discharge rate $q^{\min} = q^{\max} = 3$ MW, capacity $c^{\max} = 10$ MWh, and storage efficiency $\gamma_e = 0.94$. A single-period OPF (SPOPF) is calculated on this modified system during each timestep of both training and testing using the pandapower package in Python.

The parameters for DDPG training are given in Table 6.2.

6.3.2 Simulation results

Fig. 6.10 shows the average episode reward during training with the shadow line as an exponential moving average of weight 0.6. The average episode reward converges to around -50 after being trained for 1×10^5 steps. This model obtained from training is stored and then used during the testing stage.

The agent is tested by using the data for 2021. The testing results are averaged into a daily cost, and the results for different scenarios are given in Table 6.3, where “No RES" represents the system without RES or ESS, “No ESS" represents the system with RES but no ESS, “SESS" represents the system with RES and static ESS co-located with the RES. Both SESS and MEC share the same storage parameters. Compared to the “No RES", “No ESS", and the SESS schemes, the proposed MEC scheme achieves a generation cost reduction of

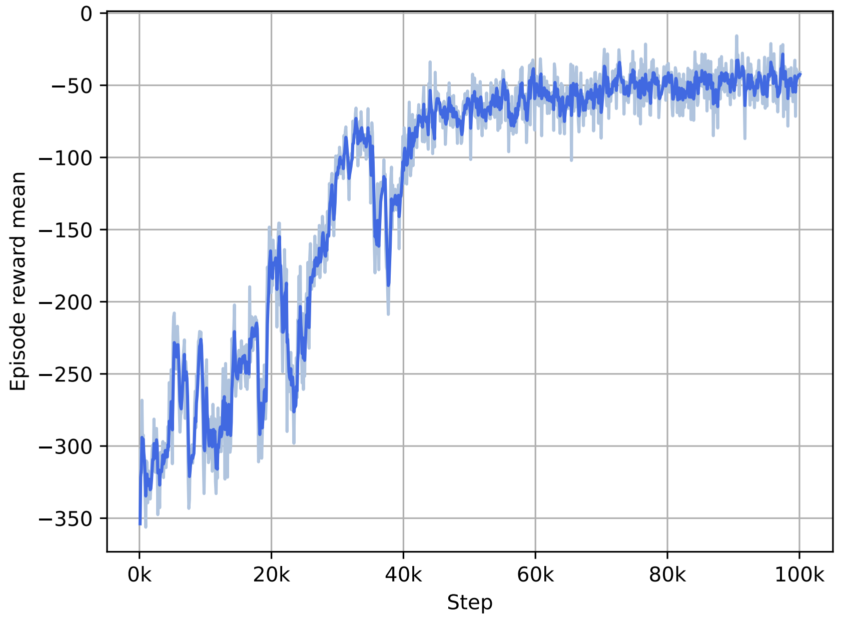


Figure 6.10: The training curve of DDPG agent.

Table 6.3: Simulation result

Case	No RES	No ESS	SESS	MEC
Generation cost ($)	6229.39	5928.28	5066.30	4391.48

29.5%, 25.9%, and 13.3% respectively. Thus the proposed MEC framework can significantly reduce the generation cost of a power system through the employment of the mobile ESS units.

The mobility pattern and ESS behaviors for MEC on July 1st, 2021 are shown in Fig. 6.11, where the lines are the MEC location (in bus ID) and the bars are the SOC of MEC. For example, MEC 2 is located at bus 6 at the beginning, and travels to bus 14 for the next 2 hours and stays there during 3 A.M. to 5 A.M. Then it travels to bus 5 to discharge during the early peak at 7 A.M. and goes back to bus 14 and charges the ESS at noon. Next it travels to bus 6 with RES, and it is fully charged in the afternoon. Later it discharges during the late peak at 7 P.M., travels to bus 14 to be fully discharged and travels back to the initial location. Similar behaviors are observed for MEC 4, which travels between buses to shave the two peak hours. The mobility of MEC 1 and 3 is not fully utilized, as they spend the majority of time at a single bus and act as an SESS.

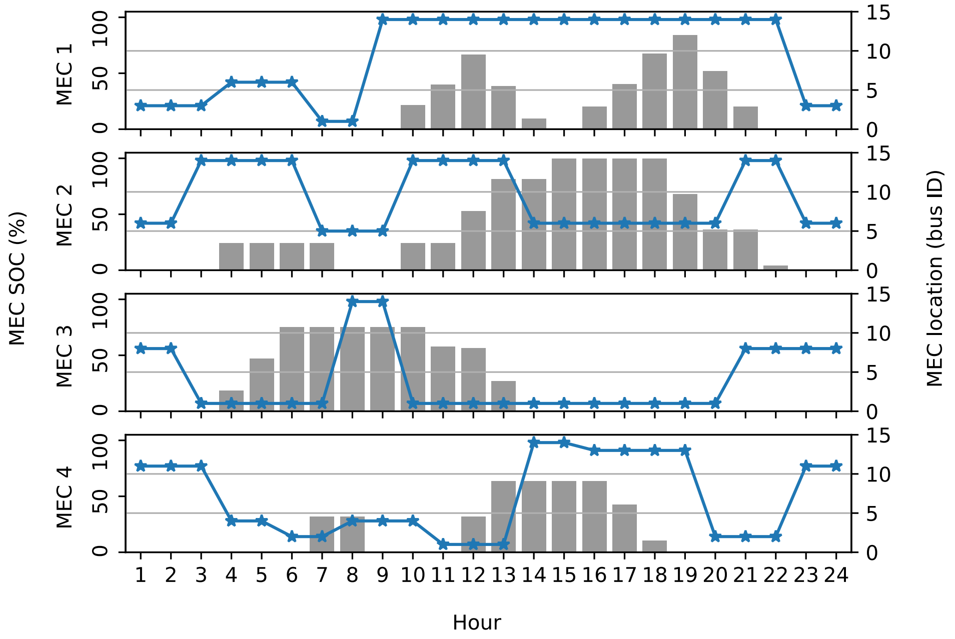


Figure 6.11: Simulation results of MECs on July 1st, 2021.

6.4 Conclusion

This chapter has proposed a new framework of truck-based MEC, which are used to strategically transport energy storage devices to different locations to facilitate the operations of a distribution network with renewable energy sources. The MEC scheduling involves both transportation scheduling and energy storage scheduling, and it has been solved by using deep deterministic policy gradient under the framework of deep reinforcement learning. Based on reinforcement learning of the dynamics of the loads, RES, and transportation networks, the MEC powered by DDPG can achieve dynamic spatial-temporal energy reallocation to optimize the grid operation. Simulation results have demonstrated that the employment of MEC can achieve better utilization of the dynamic renewable energy sources, thus achieve significant reduction in generation cost.

Several directions can be explored for future research. First, the problem formulation can be expanded by considering additional design objectives, such as the resilience and flexibility to enhance the overall performance of the power system. Second, the performance of the system can be further optimized by considering the number, capacity, and cost of MEC, and the results will be valuable for large power systems and transportation networks for practical implementations. Third, employing electrical vehicles as MEC can further improve the flexibility and efficiency of the overall system design.

7 Low Latency Attack Detection with Dynamic Watermarking for Grid-Connected Photovoltaic Systems

All previous chapters focus on the grid integration of RES and the control algorithms. The addition of these complex control and communication capabilities increases the vulnerability of the RES, and make them prone to cyberattacks [19]. Cyberattacks can disrupt normal grid operations by causing system instabilities such as line overloads, frequency and/or voltage violations, reverse power flow, and voltage collapse, especially during heavy load conditions [20, 21]. This necessitates the development of new cybersecurity technologies that can detect and/or mitigate the negative impacts of cyberattacks.

Many existing studies on the cybersecurity of energy systems focus on grid operations by using measurements from the supervisory control and data acquisition (SCADA) systems, the remote terminal units (RTUs), and/or the underlying communication network of the grid [2, 22]. These measurements are important indicators for grid operations, but they are insufficient given that attacks can be launched against local measurements from sensors and actuators of RES, or the local control policies for RES operations.

In this chapter we develop a low latency attack detection algorithm for a grid-connected PV systems with dynamic watermarking. The proposed algorithm has four main innovations. First, the algorithm is designed by using a hybrid model- and data-driven approach. We first construct a state-space model for a grid-connected PV farm, the knowledge of which is used to estimate and predict the state information, such as current and voltage, by using a Kalman filter. Key parameters of the filter are estimated and updated by using data collected from the system. Second, the algorithm performs active detection of cyberattacks by using a two-test dynamic watermarking scheme. The statistical tests of the dynamic watermarks are formulated by analyzing the statistical properties of the residuals from state estimation and measurements. Third, unlike existing methods that focus mainly on detection accuracy, the algorithm is developed to minimize the average detection delay (ADD), subject to an upper bound on the probability of false alarm (PFA). The low latency detection algorithm is designed by using a modified CUSUM algorithm that incorporates dynamic watermarks. Fourth, we propose to measure the stealthiness of various cyberattacks by using the Kullback-Leibler (KL) divergence between the pre- and post-attack distributions of the test statistics. The KL divergence provides a quantitative measure on the tradeoff between the stealthiness and the power of a given cyberattack. The KL divergences of several attacks, such as the false data injection (FDI) attack, replay attack, and destabilization attacks are analyzed and evaluated.

7.1 Problem formulation

7.1.1 PV model

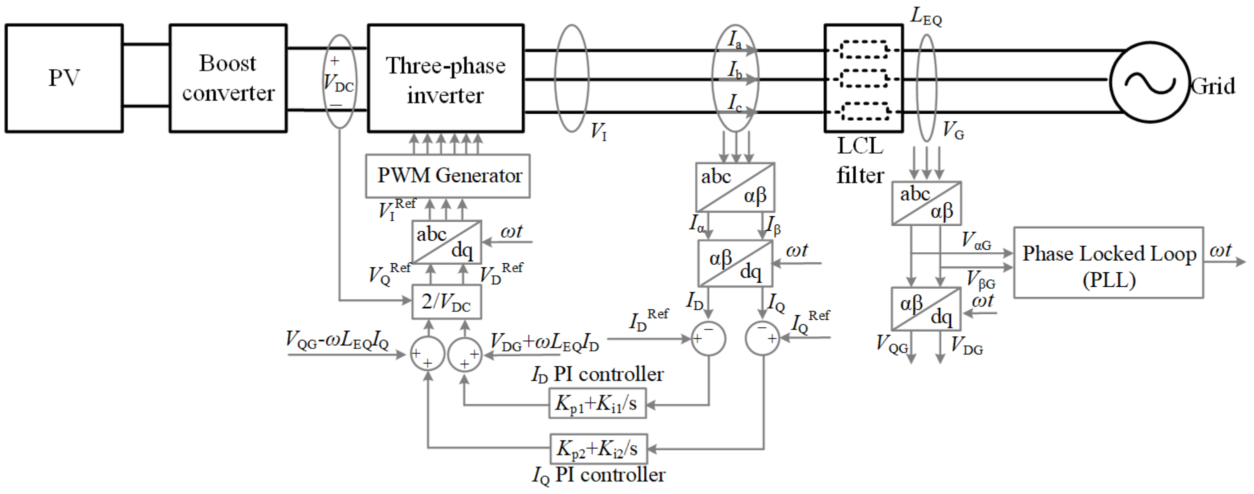


Figure 7.1: Diagram of PV inverter.

The diagram of a typical PV inverter is shown in Fig. 7.1. The magnitude and frequency of the PV inverter output voltage need to be controlled and regulated to ensure proper system operations. The DC voltage at the output of the boost converter, V_{DC}, is considered as an ideal DC voltage source. Denote the phase voltage magnitude connected to the grid as V_{G}, the DQ frame of which is represented as V_{DG} and V_{QG}, respectively. The three phases of the output current of the inverter are denoted as I_a, I_b, and I_c. The DQ frame of the three-phase current after the inverter is represented as I_{D} and I_{Q}, respectively. The PV inverter controller is controlled by using the signal V_{DG}, V_{QG}, I_{D}, I_{Q}, along with the DQ frame of the three-phase reference currents $I_{\mathrm{D}}^{\mathrm{Ref}}$ and $I_{\mathrm{Q}}^{\mathrm{Ref}}$, the values of which are set based on the output voltage and power to the grid.

Denote the phase voltage magnitude of the inverter as V_{I}, and the equivalent inductance between the inverter and the grid as L_{EQ}, which consists of the inductance of the LCL filter and the transmission line. Based on the input signals to the PV inverter controller, the reference DQ frames of V_{I} can be computed as

$$V_{\mathrm{DI}}^{\mathrm{Ref}} = \frac{2}{V_{\mathrm{DC}}}\left[K_{p1}\left(I_{\mathrm{D}}^{\mathrm{Ref}} - I_{\mathrm{D}}\right) + K_{i1}\int_0^t \left(I_{\mathrm{D}}^{\mathrm{Ref}} - I_{\mathrm{D}}\right) d\tau + V_{\mathrm{DG}} + \omega L_{\mathrm{EQ}} I_{\mathrm{Q}}\right], \tag{7.1a}$$

$$V_{\mathrm{QI}}^{\mathrm{Ref}} = \frac{2}{V_{\mathrm{DC}}}\left[K_{p2}\left(I_{\mathrm{Q}}^{\mathrm{Ref}} - I_{\mathrm{Q}}\right) + K_{i2}\int_0^t \left(I_{\mathrm{Q}}^{\mathrm{Ref}} - I_{\mathrm{Q}}\right) d\tau + V_{\mathrm{QG}} - \omega L_{\mathrm{EQ}} I_{\mathrm{D}}\right], \tag{7.1b}$$

where K_{p1}, K_{p2}, K_{i1} and K_{i2} are the corresponding proportion parameter and integration parameter tuned based on the desired static and dynamic performance of the output voltage. The reference voltage $V_{\mathrm{I}}^{\mathrm{Ref}}$ is calculated by inverse DQ transformation from its reference DQ frame, and then fed to the input of the PWM generator.

7.1.2 State-Space model

The PV farm model can be abstracted to a multiple-input multiple-output partial observed system model. Based on the grid connected PV farm model, define the system state vector, $\mathbf{x} \in \mathcal{R}^n$, the control system input vector, $\mathbf{u} \in \mathcal{R}^m$, and the output (or observation) vector, $\mathbf{y} \in \mathcal{R}^p$, as

$$\mathbf{x} = [I_{\mathrm{D}}, I_{\mathrm{Q}}, V_{\mathrm{DG}}, V_{\mathrm{QG}}]^T, n = 4, \tag{7.2a}$$

$$\mathbf{u} = [V_{\mathrm{DI}}, V_{\mathrm{QI}}]^T, \qquad m = 2, \tag{7.2b}$$

$$\mathbf{y} = [\omega, |V_{\mathrm{G}}|]^T, \qquad p = 2, \tag{7.2c}$$

where V_{DI} and V_{QI} are the DQ frames of the voltage after the inverter V_{I}, w and $|V_{\mathrm{G}}|$ are the frequency and magnitude of the output voltage connected to the grid, respectively.

The dynamics of the PV farm system can then be estimated by using the following linearized differential and algebraic equations (DAEs) as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{w}, \tag{7.3a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{n}, \tag{7.3b}$$

where $\mathbf{A} \in \mathcal{R}^{n\times n}$ is the state matrix, $\mathbf{B} \in \mathcal{R}^{n\times m}$ is the control matrix, $\mathbf{C} \in \mathcal{R}^{p\times n}$ is the output matrix, $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{w}})$ and $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{n}})$ are zero-mean Gaussian distributed process noise and the measurement noise, respectively, with their covariance matrices being $\boldsymbol{\Sigma}_{\mathbf{w}}$ and $\boldsymbol{\Sigma}_{\mathbf{n}}$, respectively.

The small-signal dynamics around a given operating condition can be estimated similarly by using the DEAs. Let $\Delta\mathbf{x}$ be the small deviation over the equilibrium state, and $\Delta\mathbf{u}$

and $\Delta\mathbf{y}$ are defined in a similar manner. Then the DAEs for the small-signal dynamics can be expressed as,

$$\Delta\dot{\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{u} + \mathbf{w}, \tag{7.4a}$$

$$\Delta\mathbf{y} = \mathbf{C}\Delta\mathbf{x} + \mathbf{n}. \tag{7.4b}$$

The control policy of the entire PV farm can be abstracted into a nonlinear vector function $\mathbf{h}(\cdot)$ as

$$\Delta\mathbf{u} = \mathbf{h}(\Delta\mathbf{y}). \tag{7.5}$$

The continuous-time state-space model in (7.4) can be discretized into discrete-time as

$$\Delta\mathbf{x}[t+1] - \Delta\mathbf{x}[t] = \mathbf{A}\Delta\mathbf{x}[t] + \mathbf{B}\Delta\mathbf{u}[t] + \mathbf{w}[t+1] \tag{7.6a}$$

$$\Delta\mathbf{y}[t+1] = \mathbf{C}\Delta\mathbf{x}[t+1] + \mathbf{n}[t+1] \tag{7.6b}$$

Similarly, the discrete form of (7.5) is:

$$\Delta\mathbf{u}[t] = \mathbf{h}(\mathbf{y}^t) \tag{7.7}$$

where $\mathbf{y}^t := \{\mathbf{y}[0], \mathbf{y}[1], \ldots, \mathbf{y}[t]\}$ are the collections of previous measurements reported by the sensors for the PI control.

It should be noted that the values of variables in the discrete-time difference equations generally differ from their continuous-time counterparts in the differential equations and their values are determined by the sampling rate. For convenience, the notations are kept the same and the small deviation notation Δ is omitted for the rest of the work. The DEAs in (7.6) can also be written as,

$$\mathbf{x}[t+1] = \mathbf{A_d}\mathbf{x}[t] + \mathbf{B}\mathbf{u}[t] + \mathbf{w}[t+1], \tag{7.8a}$$

$$\mathbf{y}[t+1] = \mathbf{C}\mathbf{x}[t+1] + \mathbf{n}[t+1], \tag{7.8b}$$

where $\mathbf{A_d} = \mathbf{A} + \mathbf{I}_n$. The state transition matrices for the discrete-time model can be obtained from the continuous-time model if the control model is known. They can also be estimated by measurements in practice without the knowledge of the control model.

7.1.3 Attack models

Suppose the system is attacked at the moment τ, and assume that the attacker has the knowledge of the control system, including the parameters $\mathbf{A_d}, \mathbf{B}, \mathbf{C}$, the control policy

$\mathbf{h}(\cdot)$ and all historical measurements $\mathbf{z}^t$. This is a very generous assumption to assume the worst possible attacks. In case the attacker has partial knowledge of the above parameters and/or control policy, the attack efficiency will be lower and it will be easier to detect.

The following types of cyberattacks are considered in this paper.

1. FDI attack. The measurement vector $\mathbf{y}[t]$ is injected with a deterministic attack vector $\mathbf{a}[t] \in \mathcal{R}^p$ or a noise vector $\mathbf{a}[t] \sim \mathcal{N}_p(\mathbf{0}, \mathbf{\Sigma_a})$ as

$$\mathbf{z}[t] = \begin{cases} \mathbf{y}[t], & t < \tau, \\ \mathbf{y}[t] + \mathbf{a}[t], & t \geq \tau. \end{cases} \tag{7.9}$$

2. Replay attack. The measurement vector $\mathbf{y}[t]$ is replaced by historical data from l moments ago with $l < \tau$ as,

$$\mathbf{z}[t] = \begin{cases} \mathbf{y}[t], & t < \tau, \\ \mathbf{y}[t-l], & t \geq \tau. \end{cases} \tag{7.10}$$

3. Destabilization attack. The control input $\mathbf{u}[t]$ is injected with a scaled controller input as

$$\mathbf{u}_a[t] = \mathbf{u}[t] + \mathbf{A_p}\mathbf{x}[t], \quad t \geq \tau \tag{7.11}$$

where $\mathbf{A_p} \in \mathcal{R}^{m \times n}$ is the scaling parameter for the attack. With the compromised control input $\mathbf{u}_a[t]$, the state transition in (7.8) becomes

$$\mathbf{x}[t+1] = (\mathbf{A_d} + \mathbf{B}\mathbf{A_p})\mathbf{x}[t] + \mathbf{B}\mathbf{u}[t] + \mathbf{w}[t+1]. \tag{7.12}$$

The system becomes unstable when the entries in $\mathbf{A_p}$ are chosen such that $||\mathbf{A_d} + \mathbf{B}\mathbf{A_p}|| \geq 1$ [162].

7.2 State Estimation

7.2.1 Kalman filter

Consider a multiple-input multiple-output partial observed system as shown in Fig. 7.2 with $\mathbf{x} \in \mathcal{R}^{n_x \times 1}$ as the **state vector**. A Kalman filter as shown in Fig. 7.3 is used to estimate the state. Denote $\hat{\mathbf{x}}_{a|b}$ as the estimation of $\mathbf{x}$ at the moment a given observations up

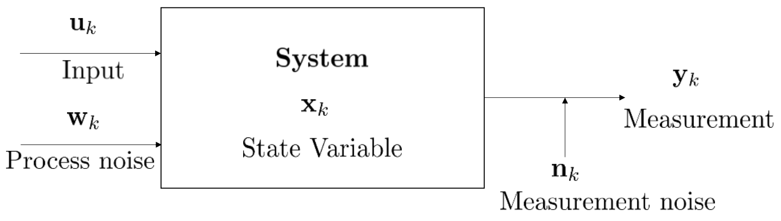


Figure 7.2: System model

to and including moment b. The **state extrapolation equation** predicts the next system state $\hat{\mathbf{x}}_{k+1|k}$ based on the knowledge of the current estimated state $\hat{\mathbf{x}}_{k|k}$.

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}\hat{\mathbf{x}}_{k|k} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k \tag{7.13}$$

where $\mathbf{u}_k \in \mathcal{R}^{n_u \times 1}$ is called the **control variable** or **input variable**, which is a measurable input to the system. $\mathbf{w}_k \in \mathcal{R}^{n_x \times 1}$ is called the **process noise** or **disturbance**, which is an unmeasurable input that affects the state. $\mathbf{A} \in \mathcal{R}^{n_x \times n_x}$ is the **state transition matrix**, $\mathbf{B} \in \mathcal{R}^{n_x \times n_u}$ is the **control matrix** or **input transition matrix**. The predictor covariance matrix is given by the **covariance extrapolation equation**:

$$\mathbf{P}_{k+1|k} = \mathbf{A}\mathbf{P}_{k|k}\mathbf{A}^T + \mathbf{\Sigma}_{\mathbf{w}} \tag{7.14}$$

where $\mathbf{P}_{k|k}$ is the estimation covariance matrix of the current state, $\mathbf{P}_{k+1|k}$ is the prediction covariance matrix of the next state. $\mathbf{\Sigma}_{\mathbf{w}}$ is the covariance matrix of the process noise. These two equations above are prediction equations.

Denote the **measurement vector** as $\mathbf{y}_k \in \mathcal{R}^{n_y \times 1}$, represents a linear transformation of the true state $\mathbf{x}_k \in \mathcal{R}^{n_x \times 1}$ with a random **measurement noise** $\mathbf{n}_k \in \mathcal{R}^{n_y \times 1}$, given by the **measurement equation**:

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{n}_k \tag{7.15}$$

where $\mathbf{C} \in \mathcal{R}^{n_y \times n_x}$ is the **observation matrix**. The randomness of the measurement noise is given by covariance matrix $\mathbf{\Sigma}_{\mathbf{n}}$.

There are also two equations for updating. Define $\boldsymbol{v}_{k+1} \in \mathcal{R}^{n_y \times 1}$ as the innovation vector at moment $k+1$ as:

$$\boldsymbol{v}_{k+1} = \mathbf{y}_{k+1} - \mathbf{C}\hat{\mathbf{x}}_{k+1|k}, \tag{7.16}$$

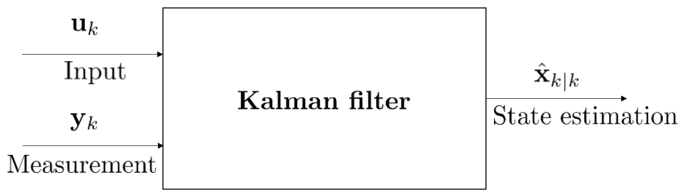


Figure 7.3: State estimation

The first is the **state update equation**:

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\boldsymbol{\upsilon}_{k+1}, \tag{7.17}$$

where $\mathbf{K}_{k+1} \in \mathcal{R}^{n_x \times n_y}$ is the Kalman Gain matrix. The **covariance update equation** is given by:

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})\mathbf{P}_{k+1|k}(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T + \mathbf{K}_{k+1}\boldsymbol{\Sigma}_{\mathbf{n}}\mathbf{K}_{k+1}^T \tag{7.18}$$

where $\mathbf{I} \in \mathcal{R}^{n_y \times n_y}$ is an identity matrix. The derivation can be found in Appendix B.1.

The Kalman Gain matrix is updated by:

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T + \boldsymbol{\Sigma}_{\mathbf{n}})^{-1} \tag{7.19}$$

the derivation is also given in Appendix B.2. Plugging equation (7.27) to equation (7.18) will give a simplified covariance update equation:

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})\mathbf{P}_{k+1|k} \tag{7.20}$$

This equation might look more elegant than (7.18), however it is numerically unstable and not recommended for computing the updated covariance. The operation of Kalman filter follows a “predict-estimate” loop, as shown in Fig.7.4

7.2.2 Dynamic watermarking

The dynamic watermarking is implemented in the form of a random signal $\mathbf{e}[t] \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{e}})$, and they are identically and independently distributed in time. The dynamic watermark signal is applied to the control input as,

$$\mathbf{u}[t] = \mathbf{h}(\mathbf{z}^t) + \mathbf{e}[t], \tag{7.21}$$

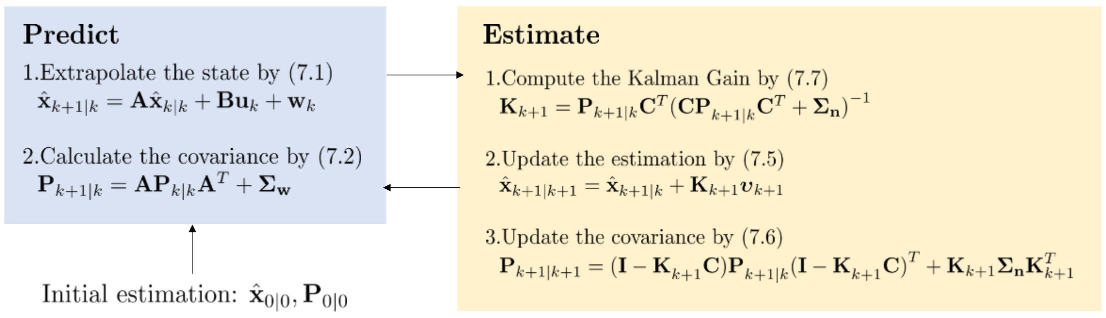

Predict
1.Extrapolate the state by (7.1)
$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}\hat{\mathbf{x}}_{k|k} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k$
2.Calculate the covariance by (7.2)
$\mathbf{P}_{k+1|k} = \mathbf{A}\mathbf{P}_{k|k}\mathbf{A}^T + \mathbf{\Sigma}_{\mathbf{w}}$
Initial estimation: $\hat{\mathbf{x}}_{0|0}, \mathbf{P}_{0|0}$
Estimate
1.Compute the Kalman Gain by (7.7)
$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T + \mathbf{\Sigma}_{\mathbf{n}})^{-1}$
2.Update the estimation by (7.5)
$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\boldsymbol{\upsilon}_{k+1}$
3.Update the covariance by (7.6)
$\mathbf{P}_{k+1|k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})\mathbf{P}_{k+1|k}(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T + \mathbf{K}_{k+1}\mathbf{\Sigma}_{\mathbf{n}}\mathbf{K}_{k+1}^T$

Figure 7.4: Kalman filter diagram

where $\mathbf{z}^t$ is the compromised observation vector after attack, and $\mathbf{z}^t = \mathbf{y}^t$ if there is no attack.

With the watermark signal, the system evolves as

$$\mathbf{x}[t+1] = \mathbf{A_d}\mathbf{x}[t] + \mathbf{B}\mathbf{h}(\mathbf{z}^t) + \mathbf{B}\mathbf{e}[t] + \mathbf{w}[t+1], \tag{7.22a}$$

$$\mathbf{y}[t+1] = \mathbf{C}\mathbf{x}[t+1] + \mathbf{n}[t+1]. \tag{7.22b}$$

It is shown in [86] that the addition of a dynamic watermark signal to the control input can facilitate the revelation of malicious tampering of the signals through two statistical tests.

The implementation of the low latency detection with dynamic watermarking requires state estimation of the solar farm. Using the Kalman filter equations we have:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A_d}\hat{\mathbf{x}}_{k|k} + \mathbf{B}\mathbf{h}(\mathbf{z}^k) + \mathbf{B}\mathbf{e}[k], \tag{7.23}$$

$$\mathbf{P}_{k+1|k} = \mathbf{A_d}\mathbf{P}_{k|k}\mathbf{A_d}^T + \mathbf{\Sigma}_{\mathbf{w}}. \tag{7.24}$$

Define $\boldsymbol{\upsilon}[k+1] \in \mathcal{R}^p$ as the innovation vector at moment $k+1$ as

$$\boldsymbol{\upsilon}[k+1] = \mathbf{z}[k+1] - \mathbf{C}\hat{\mathbf{x}}_{k+1|k}, \tag{7.25}$$

and the corresponding innovation covariance matrix is

$$\mathbf{R}_{k+1} = \mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T + \mathbf{\Sigma}_{\mathbf{n}}. \tag{7.26}$$

The optimal Kalman gain matrix at moment $k+1$ is

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k}\mathbf{C}^T\mathbf{R}_{k+1}^{-1}. \tag{7.27}$$

Then the posterior state estimation and the corresponding covariance matrix at the moment $k+1$ are updated by

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\boldsymbol{\upsilon}[k+1], \tag{7.28}$$

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I}_p - \mathbf{K}_{k+1}\mathbf{C})\mathbf{P}_{k+1|k}. \tag{7.29}$$

Substituting (7.23) and (7.25) into (7.28) yields

$$\hat{\mathbf{x}}_{k+1|k+1} = \mathbf{A_d}\hat{\mathbf{x}}_{k|k} + \mathbf{Bh}(\mathbf{z}^k) + \mathbf{Be}[k] + \mathbf{K}_{k+1}\boldsymbol{\upsilon}[k+1]. \tag{7.30}$$

Define a test statistic $\mathbf{g}[k+1]$ at the moment $k+1$ as:

$$\mathbf{g}[k+1] = \hat{\mathbf{x}}_{k+1|k+1} - \mathbf{A_d}\hat{\mathbf{x}}_{k|k} - \mathbf{Bh}(\mathbf{z}^k) - \mathbf{Be}[k] \tag{7.31}$$

If there is no attack, then $\mathbf{d}[k+1] = \mathbf{0}$, and we have the following distributions,

$$\hat{\mathbf{x}}_{k+1|k+1} - \mathbf{A_d}\hat{\mathbf{x}}_{k|k} - \mathbf{Bh}(\mathbf{z}^k) \sim \mathcal{N}_n(\mathbf{0}, \mathbf{B}\boldsymbol{\Sigma}_\mathbf{e}\mathbf{B}^T + \mathbf{K}_{k+1}\mathbf{R}_{k+1}\mathbf{K}_{k+1}^T), \tag{7.32}$$

$$\hat{\mathbf{x}}_{k+1|k+1} - \mathbf{A_d}\hat{\mathbf{x}}_{k|k} - \mathbf{Bh}(\mathbf{z}^k) - \mathbf{Be}[k] \sim \mathcal{N}_n(\mathbf{0}, \mathbf{K}_{k+1}\mathbf{R}_{k+1}\mathbf{K}_{k+1}^T). \tag{7.33}$$

The elements in $\mathbf{g}[k+1]$ might be mutually correlated because of the selected state of the system, which makes $\boldsymbol{\Phi}_{k+1} = \mathbf{K}_{k+1}\mathbf{R}_{k+1}\mathbf{K}_{k+1}^T$ singular.

To solve this problem, denote $\boldsymbol{\Phi} = \lim_{k\to\infty}\boldsymbol{\Phi}_{k+1}$ as the asymptotic estimate of $\boldsymbol{\Phi}_{k+1}$. Assume the rank of $\boldsymbol{\Phi}$ is $q \le p$ with nonzero eigenvalues $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_q]^T$, and the matrix $\bar{\mathbf{U}} \in \mathcal{C}^{p\times q}$ contains the corresponding eigenvectors on its column. We can perform dimension reduction on $\mathbf{g}[k+1]$ as

$$\bar{\mathbf{g}}[k+1] = \bar{\mathbf{U}}^H\mathbf{g}[k+1]. \tag{7.34}$$

Then we have

$$\lim_{k\to\infty}\mathbb{E}\left[\bar{\mathbf{g}}[k+1]\bar{\mathbf{g}}[k+1]^H\right] = \mathbf{D}, \tag{7.35}$$

where $\mathbf{D} = \mathrm{Diag}(\boldsymbol{\lambda}) \in \mathcal{C}^{q\times q}$ is a diagonal matrix with the q nonzero eigenvalues of $\boldsymbol{\Phi}$ on its main diagonal.

Based on the test statistic, the statistical tests that are used for dynamic watermarking are [86]

1. Test 1:
$$\lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{T-1} \mathbf{e}[k]\bar{\mathbf{g}}[k+1]^T = 0_{m\times q}. \tag{7.36}$$

2. Test 2:
$$\lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{T-1} \bar{\mathbf{g}}[k+1]\bar{\mathbf{g}}[k+1]^T = \mathbf{D}. \tag{7.37}$$

Following the similar procedure as in [86], it can be proved that passing tests (7.36) and (7.37) is sufficient to achieve an asymptotically zero attacking power defined as

$$\lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{T-1} ||\mathbf{d}[k+1]||^2 = 0, \tag{7.38}$$

which means there is no attack.

Define $\mathbf{r}[k+1] = [\bar{\mathbf{g}}[k+1]^T \ \mathbf{e}[k]^T]^T \in \mathcal{R}^{q+m}$, The the two tests described in (7.36) and (7.37) can be combined into one equivalent test as,

$$\begin{aligned}
&\lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{T-1} \mathbf{r}[k+1]\mathbf{r}[k+1]^T \\
&= \lim_{T\to\infty} \frac{1}{T} \sum_{k=0}^{T-1} \begin{bmatrix} \bar{\mathbf{g}}[k+1]\bar{\mathbf{g}}[k+1]^T & \bar{\mathbf{g}}[k+1]\mathbf{e}[k]^T \\ \mathbf{e}[k]\bar{\mathbf{g}}[k+1]^T & \mathbf{e}[k]\mathbf{e}[k]^T \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{D} & \mathbf{0}_{q\times m} \\ \mathbf{0}_{m\times q} & \mathbf{\Sigma_e} \end{bmatrix} := \mathbf{\Sigma_0} \in \mathcal{R}^{(q+m)\times(q+m)}.
\end{aligned} \tag{7.39}$$

7.3 Low Latency Detection with Dynamic Watermarking

Based on the test statistics designed for dynamic watermarking, we propose to develop low latency attack detection with dynamic watermarking.

7.3.1 Post-Attack Distributions and KL Divergence

We first study the pre- and post-attack distributions of the test statistics for dynamic watermarking with various attacks. In addition, the results are used to analytically identify the KL divergence between the pre- and post-attack distributions. The KL divergence can be used to measure the stealthiness of various cyberattacks. The analytical results will be used to facilitate the development of the low latency detection algorithm.

Under normal operation conditions without any attack, the limit distribution of $\mathbf{r}[k+1]$ is given based on the Law of large numbers (LLN) as

$$\lim_{k\to\infty} \mathbf{r}[k+1] \sim \mathcal{N}_{q+m}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{0}}). \tag{7.40}$$

Denote $\mathbf{K} = \lim_{k\to\infty} \mathbf{K}_{k+1}$ and $\mathbf{P} = \lim_{k\to\infty} \mathbf{P}_{k+1}$ as the asymptotic covariance matrix and Kalman gain matrix, respectively. The post-attack distribution of $\mathbf{r}[k+1]$ depends on the various attack models as analyzed in the following. The proof of the post-attack distribution of $\mathbf{r}[k+1]$ can be found in Appendix C.

1. FDI attack: Substituting (8.3) and (7.25) into (8.29) and (7.31), we have the post-attack distribution of $\mathbf{r}[k+1]$ under the FDI attack as

$$\lim_{k\to\infty} \mathbf{r}[k+1] \sim \mathcal{N}_{q+m}(\mu, \boldsymbol{\Sigma}) \tag{7.41}$$

with

$$\mu = \begin{bmatrix} \bar{\mathbf{U}}^H \mathbf{K}\mathbf{a}[k+1] \\ \mathbf{0}_m \end{bmatrix} \tag{7.42a}$$

$$\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{\mathbf{0}} \tag{7.42b}$$

under deterministic FDI. Under the noisy FDI attack, we have

$$\mu = \mathbf{0}_{q+m} \tag{7.43a}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathbf{D} + \bar{\mathbf{U}}^H \mathbf{K}\boldsymbol{\Sigma}_{\mathbf{a}}\mathbf{K}^T \bar{\mathbf{U}} & \mathbf{0}_{q\times m} \\ \mathbf{0}_{m\times q} & \boldsymbol{\Sigma}_{\mathbf{e}} \end{bmatrix} \tag{7.43b}$$

under noise FDI.

2. Replay attack: Substitute (7.10) to (8.29) and (7.31). Define the control matrix $\mathbf{L}$ to be the linear approximation of the control policy $\mathbf{h}(\cdot)$, such that

$$\mathbf{u}[k] = \mathbf{L}\hat{\mathbf{x}}_{k|k} + \mathbf{e}[k]. \tag{7.44}$$

Then the post-attack distribution of $\mathbf{r}[k+1]$ under the replay attack can be estimated as,

$$\lim_{k\to\infty} \mathbf{r}[k+1] \sim \mathcal{N}_{q+m}(\mathbf{0}_{q+m}, \boldsymbol{\Sigma}), \tag{7.45}$$

with

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{D} + 2\bar{\mathbf{U}}^H\mathbf{KCXC^TK}^T\bar{\mathbf{U}} & -\bar{\mathbf{U}}^{\mathbf{H}}\mathbf{KCB\Sigma_e} \\ -\mathbf{\Sigma_e B}^T\mathbf{C}^T\mathbf{K}^T\bar{\mathbf{U}} & \mathbf{\Sigma_e}, \end{bmatrix} \tag{7.46}$$

where $\mathbf{X}$ is the solution of the following Lyapunov equation

$$\mathbf{A_e X A_e^T} - \mathbf{X} + \mathbf{B\Sigma_e B^T} = \mathbf{0}, \tag{7.47}$$

and $\mathbf{A_e}$ is the estimated transition matrix:

$$\mathbf{A_e} = (\mathbf{A_d} + \mathbf{BL})(\mathbf{I}_p - \mathbf{KC}). \tag{7.48}$$

3. Destabilization attack: Substituting (7.8) into (7.30) leads to the post-attack distribution as

$$\mathbf{r}[k+1] \sim \mathcal{N}_{q+m}(\mu, \mathbf{\Sigma}), \tag{7.49}$$

with

$$\mu = \begin{bmatrix} \bar{\mathbf{U}}^H\mathbf{KCBA_p}\hat{\mathbf{x}}_{k|k} \\ \mathbf{0_m}, \end{bmatrix} \tag{7.50a}$$

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{D} + \bar{\mathbf{U}}^H\mathbf{KCP_a C}^T\mathbf{K}^T\bar{\mathbf{U}} & \mathbf{0}_{q\times m} \\ \mathbf{0}_{m\times q} & \mathbf{\Sigma_e}, \end{bmatrix} \tag{7.50b}$$

where

$$\mathbf{P_a} = \mathbf{BA_pPA_p}^T\mathbf{B}^T + \mathbf{A_dPA_p}^T\mathbf{B}^T + \mathbf{BA_pPA_d}^T. \tag{7.51}$$

Note that $\mathbf{r}[k+1]$ follows a normal distribution with $\mu \neq \mathbf{0}$ or $\mathbf{\Sigma} \neq \mathbf{\Sigma_0}$ under attacks in this paper, then the hypothesis test on $\mathbf{r}[k+1]$ is

$$\begin{aligned} \mathcal{H}_0 &: \mathbf{r}[k+1] \sim \mathcal{N}_{q+m}(\mathbf{0}, \mathbf{\Sigma_0}) \\ \mathcal{H}_1 &: \mathbf{r}[k+1] \sim \mathcal{N}_{q+m}(\mu, \mathbf{\Sigma}) \end{aligned} \tag{7.52}$$

We propose to measure the stealthiness of the various attacks by using the Kullback-Leibler (KL) divergence between the pre- and post-attack distributions of $\mathbf{r}[k+1]$. The KL divergence measures the difference between two statistical distributions. A larger KL divergence means a larger difference between the pre- and post-attack distributions, thus it is relatively easier to detect. A lower KL divergence means that the attack is more stealthy and harder to detect. The KL divergence of the pre- and post-attack distributions can be calculated as

$$D(\mathcal{H}_1||\mathcal{H}_0) = \frac{1}{2}[\mu^{\mathbf{T}}\mathbf{\Sigma_0^{-1}}\mu + \mathrm{Tr}(\mathbf{\Sigma_0^{-1}}\mathbf{\Sigma}) + \log\frac{|\mathbf{\Sigma_0}|}{|\mathbf{\Sigma}|} - m - q], \tag{7.53}$$

with μ and $\mathbf{\Sigma}$ being the post-attack mean and covariance matrices for the various attacks.

7.3.2 Low Latency Attack Detection

In the quickest attack detection, the objective is to minimize the average detection delay (ADD) subject to an upper bound of the probability of false alarm (PFA). Denote the attack time identified by the detector as $\hat{\tau}$. Then the detection problem can be formulated as:

$$\begin{aligned} \min \quad & \mathrm{ADD} = \mathbb{E}[\hat{\tau} - \tau | \hat{\tau} > \tau], \\ \text{s.t.} \quad & \mathrm{PFA} = \mathrm{P}(\hat{\tau} < \tau) \leq \alpha. \end{aligned} \tag{P1}$$

Based on the analysis in the previous subsections, define a new variable $\Gamma[k]$

$$\Gamma[k] = \mathbf{r}[k]^T\mathbf{\Sigma_0}^{-1}\mathbf{r}[k] \tag{7.54}$$

Under the null hypothesis, $\Gamma[k]$ follows a $\chi^2 - \textit{distribution}$ with $q+m$ degrees of freedom with mean and variance given as follows,

$$\begin{aligned} \mathbb{E}[\Gamma[k]] &= q+m \\ \mathrm{Var}[\Gamma[k]] &= 2(q+m) \end{aligned} \tag{7.55}$$

Based on the distribution of $\Gamma[k]$, we can define the test statistics used for CUSUM as [2]

$$U[k+1] = \max(0, U[k] + \frac{\Gamma[k+1] - (q+m)}{\sqrt{2(q+m)}}), \tag{7.56}$$

$$T[k] = \frac{U[k]}{k} \tag{7.57}$$

with $U[1] = 0$. The test sequence $T[k]$ accumulates the normalized variable, $\frac{\Gamma[k]-(q+m)}{\sqrt{2(q+m)}}$, over time. Under the null hypothesis, the test sequence $T[k]$ is always close to 0. Under the event of cyberattacks, the value of $T[k]$ will increase over time. Thus the CUSUM detector can be defined as a threshold test as

$$\hat{\tau} = \inf\{k \geq 1 | T[k] \geq \alpha\}, \tag{7.58}$$

where the threshold α is chosen to meet the PFA upper bound constraint. The Markov chain approach in [163] can be used for calculating the PFA and then selecting the threshold.

7.4 Simulation results

The PV farm is modeled using MATLAB Simulink, and all the attacks are simulated based on the Simulink model. The DC link voltage V_{DC} is set to 800 V, and the output AC phase voltage magnitude $|V_{\mathrm{G}}|$ is $400 \times \sqrt{\frac{2}{3}} = 326.60$ V with the frequency being 60 Hz. The reference DQ frame of the current are set to $I_{\mathrm{D}}^{\mathrm{Ref}} = -150$ A, $I_{\mathrm{Q}}^{\mathrm{Ref}} = 0$ A. The proportion parameters K_{p1}, K_{p2} are tuned to 10, and the integral parameters K_{i1}, K_{i2} are tuned to 20. The time interval of the simulation is set to $\Delta t = 10^{-6}$ s, which corresponds to a sampling rate of 1 MHz. This continuous state-space model is discretized with a sampling rate of 2 kHz, which is equivalent to a time interval of 5×10^{-4} s between each measurement. The covariance matrices for the process and measurement noises are set to $\mathbf{\Sigma_w} = 10^{-6}\mathbf{I}_4$ and $\mathbf{\Sigma_n} = 5 \times 10^{-7}\mathbf{I}_2$, respectively. The covariance matrix of the dynamic watermark is $\mathbf{\Sigma_e} = 10^{-6}\mathbf{I}_2$.

The system reaches the equilibrium after 2s. Once the system reaches the equilibrium, data are collected during the next one minute for parameter estimation. The state $\mathbf{x}$, the input $\mathbf{u}$, and the output $\mathbf{y}$ in the next one minute period are recorded, and are then used to estimate the corresponding matrices $\mathbf{A_d}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}, \mathbf{K}, \mathbf{P}$.

Cyberattacks and low latency attack detection are performed after parameter estimation. The attacks are launched at 4.5 s after the parameter estimation. State estimations are performed by using the control inputs and the measurements, and the results are then used to calculate the CUSUM test statistic. The ADD and PFA of the detector are calculated by using the results from 1,000 Monte-Carlo simulation trials.

7.4.1 Performance under different attacks

Deterministic FDI Attack

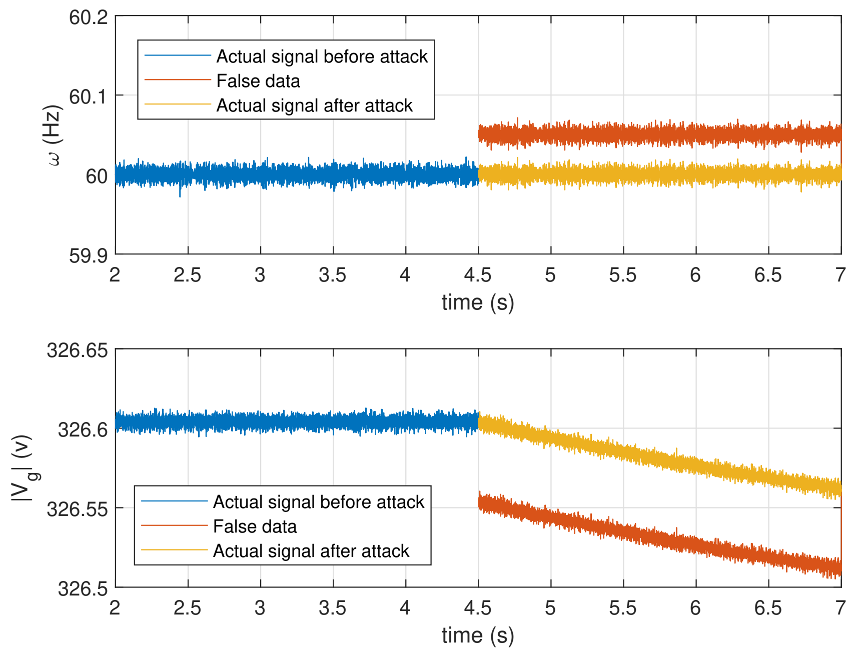


Figure 7.5: The voltage frequency (top) and magnitude (bottom) measurement under deterministic FDI attack on the PV system at 4.5s

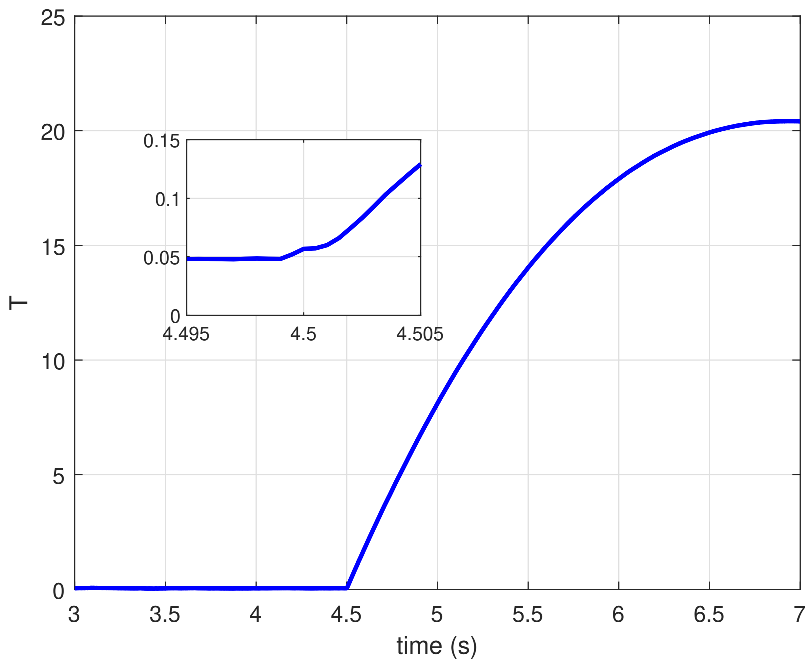


Figure 7.6: The detector statistic under deterministic FDI attack on the PV system at $4.5s$

The FDI attack with a deterministic attack vector is simulated by injecting the vector $\mathbf{a}[t] = [0.05, -0.05]^T$ to the measurement vector at 4.5 s. Fig. 7.5 shows the actual measurements and those with the attack. The attack does not cause apparent frequency deviation, but the voltage magnitude is decreased due to the attack despite the control system.

The CUSUM statistic under the deterministic FDI attack with a zoom-in around 4.5s is presented in Fig. 7.6. The CUSUM statistic remains around 0 before the attack, and its value increases dramatically after the attack. Thus the attack can be easily detected with minimum delay with the proposed quickest attack detection algorithm.

Noisy FDI Attack

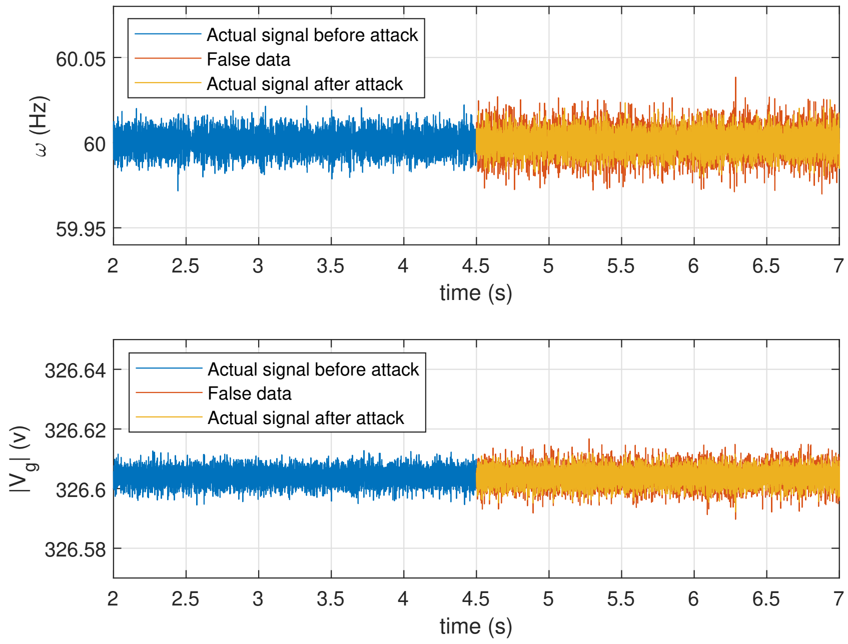


Figure 7.7: The voltage frequency (top) and magnitude (bottom) measurement under noise FDI attack on the PV system at 4.5s

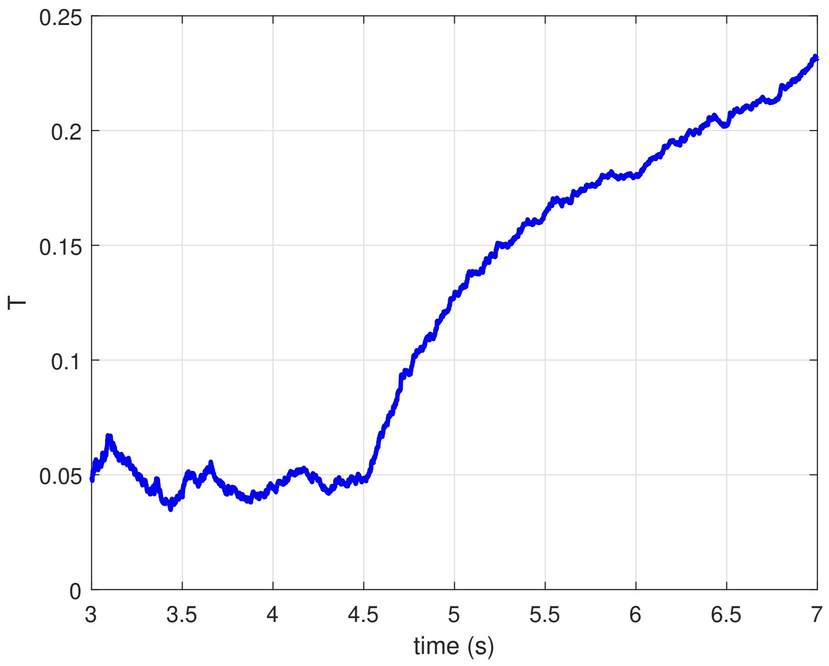


Figure 7.8: The detector statistic under noise FDI attack on the PV system at $4.5s$

The noisy FDI attack vector is generated from a random vector $\mathbf{a}[t] \sim \mathcal{N}_2(\mathbf{0}, \mathbf{\Sigma_a})$. We set the noise covariance to a level that is multiple times of the system and measurement noise, i.e., the vector on the main diagonal of $\mathbf{\Sigma_a}$ is set to $[3 \times 10^{-5}, 3 \times 10^{-6}]$. Fig.7.7 shows the actual measurements and those with noise injections. The injection only causes trivial fluctuation in both the frequency and voltage magnitude, and the actual measurements still fall in a normal range because of the control system.

The CUSUM statistic under the noisy FDI attack is presented in Fig. 7.8. Since the variance of the injected noise is very low, such an attack is hard to detect. However, it still causes a significant increase in the slope of the CUSUM statistic. Thus the noisy FDI attack can be easily detected with the proposed detection algorithm with low latency.

Replay attack

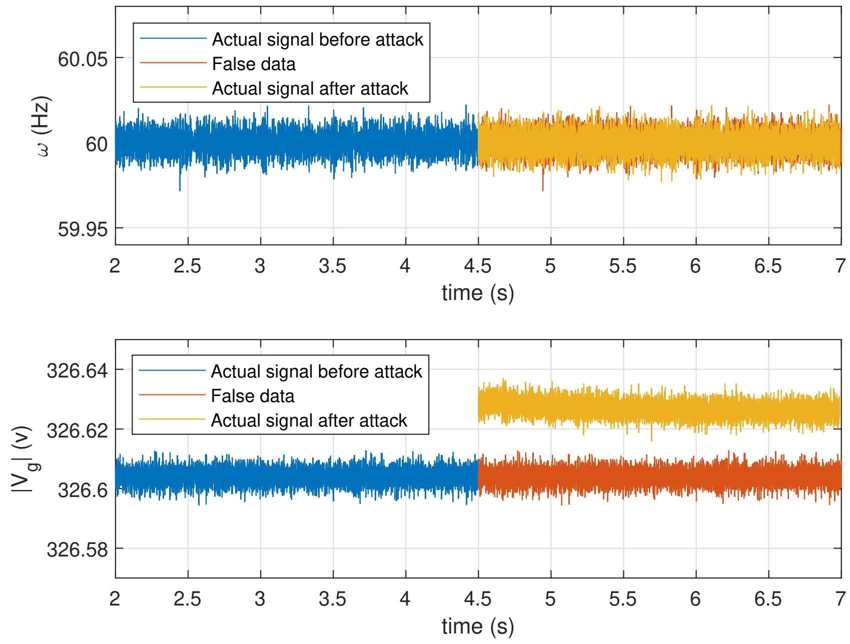


Figure 7.9: The voltage frequency (top) and magnitude (bottom) measurement under replay attack on the PV system at 4.5s

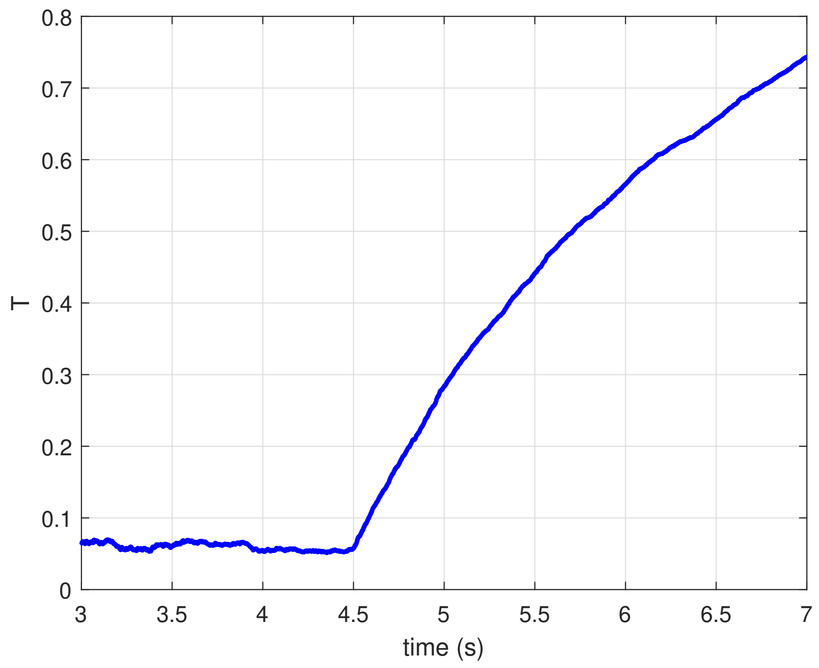


Figure 7.10: The detector statistic under replay attack on the PV system at $4.5s$

The replay attack is simulated by replacing the actual measurements from 4.5 s by historical measurements starting at 2.5 s (a delay of 2 seconds). Fig. 7.9 shows the measurements between 2s to 7s, where there is a 2 second delay between the attacked measurement and the actual measurement. The replay attack does not deviate the measurements from their normal range. However, the system will be out of normal control and cannot respond to load changes of the grid or faults in the PV farm, which can cause voltage fluctuations, reverse power flow, and real power curtailments.

The CUSUM statistic under the replay attack is presented in Fig. 7.10. The statistic $T[k]$ increases much slower than other attacks, i.e., the attack is much more stealthy compared to others. However, there is still an apparent increase in the slope of $T[k]$. Thus the replay attack can be easily detected with the proposed algorithm even if it does not cause significant deviation of the system states.

Destabilization attack

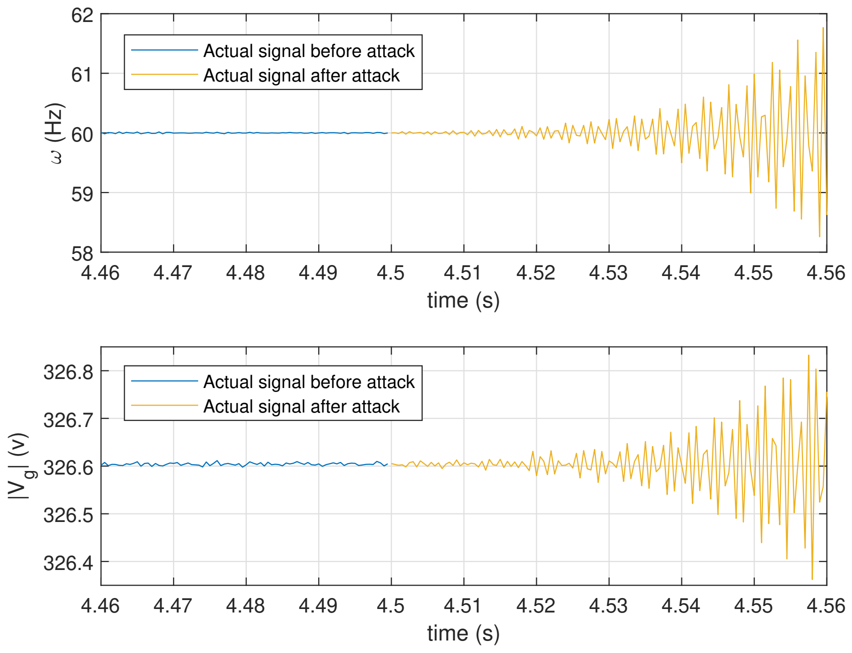


Figure 7.11: The voltage frequency (top) and magnitude (bottom) measurement under destabilization attack on the PV system at 4.5s

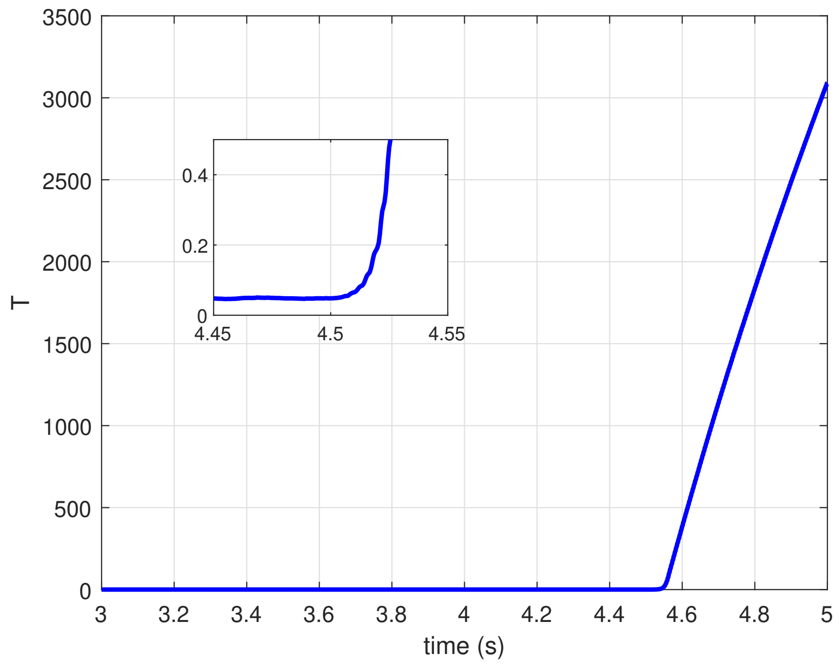


Figure 7.12: The detector statistic under destabilization attack on the PV system at $4.5s$

The destabilization attack is simulated by replacing the control inputs from 4.5 s by $\mathbf{v}(t)+\mathbf{A}_p(t)\mathbf{x}(t)$, where $\mathbf{A}_p(t)$ is a diagonal matrix with the main diagonal being $[1.5, 1.5, 0, 0]$. Fig. 7.11 shows the measurements from 4.46 s to 4.56 s. The attack rapidly causes instability in measurements which gradually exceeds its normal range.

The CUSUM statistic under destabilization attack is presented in Fig. 7.12. The statistic $T[k]$ increases much faster than other attacks, such that it is easier to detect such an attack.

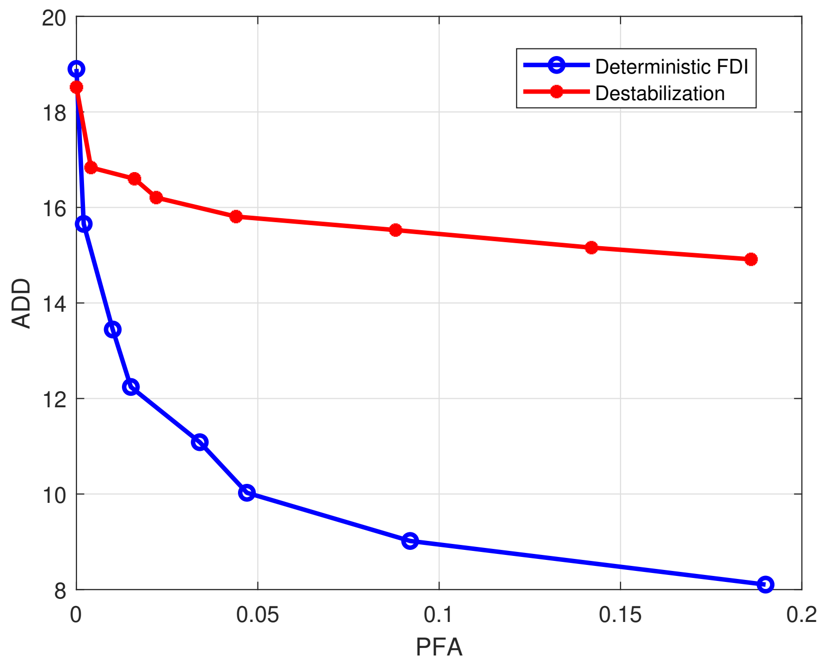


Figure 7.13: The PFA-ADD curve under the deterministic FDI attack and destabilization attack on the PV system

7.4.2 Detector performance

More powerful attacks can make the system rapidly drift away from its normal state and cause damage in a short time. However, they are usually easier to attack. The adversaries have more incentives to balance the stealthiness and power of the attack such that they can cause damage before being detected.

The stealthiness of the attacks can be measured by using the KL-divergence between the distributions of the CUSUM test vector r before and after the attack. The KL-divergence of various attacks at 4.5s is calculated by using the results in Section 7.3.1, and the results are shown in Table 7.1. The deterministic FDI and destabilization attacks have similar levels of KL divergence, and both are two or three orders of magnitude higher than that of the relay and noisy FDI attacks. Thus the deterministic FDI and destabilization attacks are relatively easier to detect. Among the 4 attacks, the noisy FDI attack has the best stealthiness with the lowest KL divergence.

The performance of the proposed low latency CUSUM detector is evaluated by using the ADD-PFA tradeoff curves shown in Fig.7.13 and Fig.7.14. Each point on the ADD-PFA tradeoff curve is obtained through 1,000 Monte Carlo trails for a given detection threshold. Due to the stealthiness of the noisy FDI and replay attack, under the same PFA, e.g. PFA = 0.02, the ADD of the deterministic FDI, destabilization, replay, and noisy attacks are 12.1, 16.5, 109, and 128 ms, respectively. This is consistent with the KL divergence analysis, that is, attacks with lower KL divergence are harder to detect, thus they have larger ADD under the same PFA.

Attack type	KL-divergence
FDI (deterministic)	0.2979
FDI (noise)	0.0053
Replay	0.0547
Destabilization	0.4567

Table 7.1: KL-divergence between distribution of $\mathbf{r}$ before and after attack.

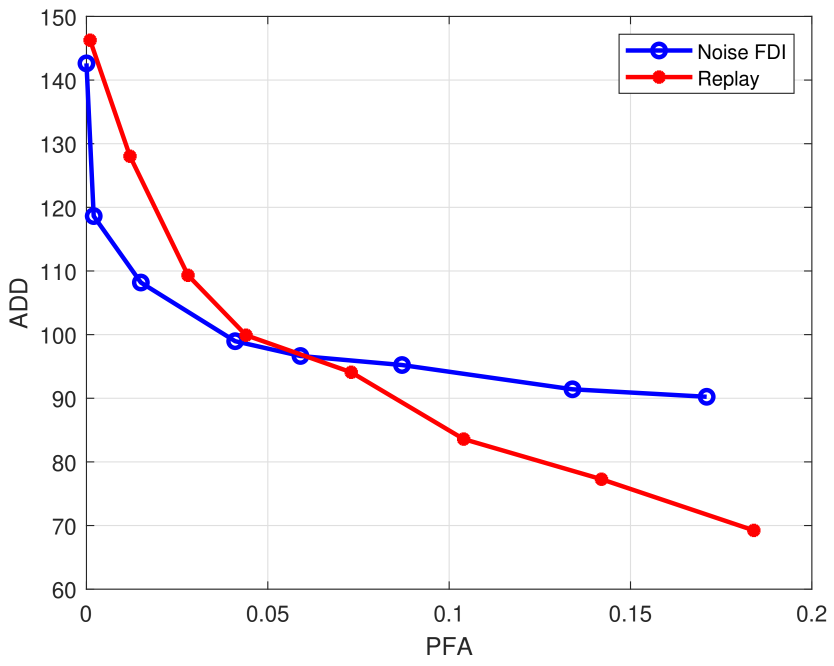


Figure 7.14: The PFA-ADD curve under noise FDI attack and replay attack on the PV system

7.5 Conclusion

This chapter has proposed an active low latency attack detection algorithm for grid-connected PV systems. We have developed a generalized CUSUM detector with dynamic watermarking by constructing and analyzing the physical model of a grid-connected PV system. The detection algorithm was developed to minimize detection delay while ensuring detection accuracy. In addition, we have proposed to use the KL divergence to measure the stealthiness of different cyberattacks. The algorithm was tested on a 400 V grid-connected PV system with various cyberattacks. Simulation results demonstrated that the proposed algorithm can achieve a detection delay of 50 ms with PFA below 5%.

8 Low Latency Cyberattack Detection in Smart Grids with Deep Reinforcement Learning

The ever-increasing power demands along with growing penetration rates of renewable energy necessitate designs of smart grids that are reliable, resilient, and secure [14]. The cyberattack on local RES such as PV farm can be quickly detected by the local measurement and the statistics used in last chapter. An important component of a smart grid is the supervisory control and data acquisition (SCADA) system, which monitors and controls power grid operations with the help of remote terminal units (RTUs). However, SCADA systems are prone to cyberattacks. For instance, cyberattacks on the SCADA system in the power grid of Kiev, Ukraine on December 23, 2015 led to a wide range blackout [15].

In this chapter, we propose to address this problem by developing a low latency detection algorithm that aims at minimizing the detection delay while maintaining good detection accuracy. The proposed algorithm adopts a hybrid model- and data-driven approach that relies on both the physical model of the power grid and the measurement data collected from the grid. In the model-based analysis, an AC model with an extended Kalman filter (EKF) is used to estimate and track the dynamic transitions of the power system. The data-driven analysis is performed by developing a deep reinforcement learning (DRL) based detection algorithm with a deep Q-network (DQN) on the framework of the Markov decision process (MDP). The new DQN design has two main innovations. First, the MDP state is designed as a sliding window of the Rao-statistics of the AC dynamic state estimation residuals. Such a state representation can accurately capture the dynamic state transitions in power systems over certain time periods, thus enabling real time detection. Second, a new reward function is proposed to enable flexible trade-offs between the detection delay and detection accuracy. The combination of the new MDP state and reward function allows us to achieve low latency attack detection in real time with high detection accuracy, and it can be used to detect both FDI and DoS attacks. In addition, the proposed DQN algorithm utilizes a continuous state space instead of the discrete state space used by most existing RL algorithms. The adoption of continuous state space can reduce detection complexity and improve detection accuracy, and it makes the algorithm less likely to suffer from the curse of dimensionality.

8.1 Problem formulation

8.1.1 System model

We consider a power system with N buses. Without loss of generality, the first bus is chosen to be the slack (reference) bus, which means the phase of the voltage at this bus is regarded as 0. The magnitudes and phases of voltages on the $N-1$ remaining buses are states of the system. Define the state vector of the system as $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in \mathcal{R}^{n\times 1}$, where $n = 2N-1$, and $\mathbf{A}^T$ is the matrix transpose operator. Denote the active and reactive power injected to bus i as P_i and Q_i, respectively, and the number of buses that are connected to bus i is c_i. Denote the active and reactive power flow from bus i to bus j as P_{ij} and Q_{ij}, respectively.

Each bus is equipped with a smart meter that collects the active and reactive power injections and power flows. The system provides $m = m_1 + m_2 + 1$ measurements in total, where $m_1 = 2N$ is the number of active and reactive power injections, and $m_2 = \sum_{i=1}^{N} c_i$ is the number of active and reactive power flows. To fully observe the power system, the voltage magnitude at the slack bus V_1 is also collected. Denote the measurement vector as $\mathbf{z} = [z_1, z_2, \ldots, z_m]^T \in \mathcal{R}^{m\times 1}$. The power and voltage results at time t can be modeled as nonlinear functions of the state vector $\mathbf{x}_t$ as $\mathbf{h}(\mathbf{x}_t) = [h_1(\mathbf{x}_t), h_2(\mathbf{x}_t), \ldots, h_m(\mathbf{x}_t)]^T$. The measurement vector can then be represented by

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{e}_t, \tag{8.1}$$

where $\mathbf{e}_t$ is the measurement error at time t, and it is modeled with a zero-mean Gaussian distributed random vector with length m and covariance matrix $\mathbf{R}$. Denote the estimated state vector with a dynamic state estimation at time t as $\hat{\mathbf{x}}_t$. Define a residual vector $\mathbf{v}_t$ as

$$\mathbf{v}_t = \mathbf{z}_t - \mathbf{h}(\hat{\mathbf{x}}_t). \tag{8.2}$$

8.1.2 Attack model

Assume that measurement elements in index $\mathcal{I} \subseteq \{1, 2, \cdots, m\}$ are attacked at time τ. The attack models of FDI and DoS are given as follows.

1. False Data Injection (FDI): The measurement vector is injected with a random attack

vector $\mathbf{A} = [a_1, a_2, \ldots, a_m]^T \in \mathcal{R}^{m\times 1}$, with $a_i = 0$ if $i \notin \mathcal{I}$:

$$\mathbf{z}_t = \begin{cases} \mathbf{h}(\mathbf{x}_t) + \mathbf{e}_t, & t < \tau \\ \mathbf{h}(\mathbf{x}_t) + \mathbf{e}_t + \mathbf{A}, & t \geq \tau \end{cases} \tag{8.3}$$

2. Denial of Service (DoS): In the DoS attack, a subset of the elements in the measurement vector are changed to zero. The DoS attack can be represented by using a diagonal matrix $\mathbf{A}$ with the main diagonal being a binary vector $d = [d_1, d_2, \cdots, d_m]^T \in \{0, 1\}^m$, that is $\mathbf{A} = \text{diag}(d)$, where

$$d_i = \begin{cases} 0, & i \in \mathcal{I} \\ 1, & i \notin \mathcal{I} \end{cases} \tag{8.4}$$

   The DoS attack is modeled as:

$$\mathbf{z}_t = \begin{cases} \mathbf{h}(\mathbf{x}_t) + \mathbf{e}_t, & t < \tau \\ \mathbf{A}\left[\mathbf{h}(\mathbf{x}_t) + \mathbf{e}_t\right], & t \geq \tau \end{cases} \tag{8.5}$$

   DoS attack can happen on different layers in the smart grid. Lack of measurements might cause the system to shut down in some cases.

In this paper the DoS attack is modeled by setting the unavailable measurements as zero. In this case, the DoS attack can be considered as a special case of FDI attack, because the DoS attack on $\mathcal{I}$ is equivalent to an FDI attack with an attack vector:

$$a_i = \begin{cases} -z_i, & i \in \mathcal{I} \\ 0, & i \notin \mathcal{I} \end{cases} \tag{8.6}$$

Thus model (8.3) will be used in this paper for both attacks.

In quickest attack detection, the objective is to minimize the average detection delay (ADD) subject to an upper bound of the probability of false alarm (PFA). Denote the attack time identified by the detector as $\hat{\tau}$. Then the ADD and PFA can be evaluated as

$$\text{ADD} = \mathbb{E}[\hat{\tau} - \tau | \hat{\tau} > \tau] \tag{8.7}$$

$$\text{PFA} = \text{P}(\hat{\tau} < \tau) \tag{8.8}$$

The design of optimum quickest attack detection algorithm usually requires the knowledge of the attack vector $\mathbf{a}$, which is not available in practical systems.

8.2 Dynamic state estimation

8.2.1 Extended Kalman filter

The idea of the EKF is a linearization of the dynamic model using the first-order Taylor expansion at the working point, which extend the linear Kalman filter in the previous section to complex systems with non-linear state transition and measurements. Modifications are made to the observation matrix $\mathbf{C}$ and the state transition matrix $\mathbf{A}$ in the original KF model, correspondingly.

The state transition matrix $\mathbf{A}$ is replaced by a non-linear transition function $\mathbf{f}(\cdot)$, so the state extrapolation equation looks like:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{f}(\hat{\mathbf{x}}_{k|k}) + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k \tag{8.9}$$

and the covariance extrapolation equation changes into:

$$\mathbf{P}_{k+1|k} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{P}_{k|k} \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^T + \mathbf{\Sigma}_{\mathbf{w}} \tag{8.10}$$

The observation matrix $\mathbf{C}$ is replaced by a non-linear measurement function $\mathbf{h}(\cdot)$, so the measurement equation looks like:

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{n}_k \tag{8.11}$$

so the innovation vector changes into:

$$\boldsymbol{\upsilon}_{k+1} = \mathbf{y}_{k+1} - \mathbf{h}(\hat{\mathbf{x}}_{k+1|k}) \tag{8.12}$$

the state update equation will use the innovation vector above.

The measurement function $\mathbf{h}(\cdot)$ is linearized to calculate the covariance matrix and Kalman gain. The covariance update equation changes into:

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I} - \mathbf{K}_{k+1} \frac{\partial \mathbf{h}}{\partial \mathbf{x}}) \mathbf{P}_{k+1|k} (\mathbf{I} - \mathbf{K}_{k+1} \frac{\partial \mathbf{h}}{\partial \mathbf{x}})^T + \mathbf{K}_{k+1} \mathbf{\Sigma}_{\mathbf{n}} \mathbf{K}_{k+1}^T \tag{8.13}$$

and the Kalman Gain is updated by:

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k} \frac{\partial \mathbf{h}}{\partial \mathbf{x}}^T (\frac{\partial \mathbf{h}}{\partial \mathbf{x}} \mathbf{P}_{k+1|k} \frac{\partial \mathbf{h}}{\partial \mathbf{x}}^T + \mathbf{\Sigma}_{\mathbf{n}})^{-1} \tag{8.14}$$

8.2.2 Forecasting-Aided State Estimation

The FASE is a particular case of applying DSE to a dynamic system with a quasi-steady state condition. The state-transition model in this case is almost linear, so the system behavior is modeled by:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}\mathbf{x}_k + \mathbf{g}_k + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k) + \mathbf{n}_k \end{aligned} \tag{8.15}$$

where $\mathbf{F} \in \mathcal{R}^{n_x \times n_x}$ is similar to the state transition matrix, $\mathbf{g}_k \in \mathcal{R}^{n_x \times 1}$ is the trend vector, which is a function of $\mathbf{u}_k$, identified from historical data. Compared with the traditional Kalman Filter where the prediction is calculated by state extrapolate, a variety of prediction models in time series analysis field can be applied for prediction in FASE.

Given the state vector sequence $\{\mathbf{x}_k\}$ is a non-linear time series. We propose to model the state transition by using Holt's linear trend method, which uses exponential smoothing to forecast a non-linear dynamic time series with a trend [164]. It involves a forecast equation and two smoothing constants, α and β.

Holt's Linear Trend Method

The Holt's (two parameter) linear exponential smoothing method can be convert into a predict model. A one-dimension example is introduced here. Consider a dynamic time series $\{y_t\}$. The h-step forecast equation for the time series is:

$$\tilde{y}_{t+h|t} = l_t + hb_t \tag{8.16}$$

where $\tilde{y}_{t+h|t}$ denotes the h-step forecast from y_t, and l_t and b_t are the estimations of the level and trend (slope) of the series at time t, respectively. The values of l_t and b_t are iteratively updated as

$$l_t = \alpha y_t + (1-\alpha)\tilde{y}_{t|t-1} = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{8.17}$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{8.18}$$

where both α and β are smoothing parameters between 0 and 1. The level equation (8.17) shows that l_t is a weighted average of observation y_t and one-step-ahead forecast $\tilde{y}_{t|t-1}$. The trend equation (8.18) shows that b_t is a weighted average of the estimated trend and the first order difference of the estimated level. The initial values l_0 and b_0 are estimated by minimizing the sum of squared errors for the one-step training errors. Since the level and trend are updated for each t, the forecasting method is dynamic.

Applying (8.16)-(8.18) with $h = 1$ to the system state vector $\mathbf{x}_k$, we can express the forecast state vector $\tilde{\mathbf{x}}_{k+1}$ from $\mathbf{x}_k$ as

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{F}_k\hat{\mathbf{x}}_k + \mathbf{g}_k \tag{8.19}$$

where

$$\mathbf{F}_k = \alpha(1+\beta)\mathbf{I}_n, \tag{8.20}$$

$$\mathbf{g}_k = (1+\beta)(1-\alpha)\hat{\mathbf{x}}_{k|k-1} - \beta\mathbf{l}_{k-1} + (1-\beta)\mathbf{b}_{k-1} \tag{8.21}$$

In the above equations, $\mathbf{I}_n$ is a size-n identity matrix, $\mathbf{l}_k$ and $\mathbf{b}_k$ are the estimates of the level and trend vectors, respectively. The corresponding error covariance matrix of state forecasting can then be calculated as

$$\mathbf{M}_{k+1} = \mathbf{F}_k\mathbf{\Sigma}_k\mathbf{F}_k + \mathbf{Q}_k. \tag{8.22}$$

State Filtering

The state estimation at time $k+1$, denoted as $\hat{\mathbf{x}}_{k+1}$, can be obtained by minimizing the following objective function,

$$J\left(\mathbf{x}_{k+1}\right) = \left[\mathbf{z}_{k+1} - \mathbf{h}(\mathbf{x}_{k+1})\right]^T \mathbf{R}_{k+1}^{-1}\left[\mathbf{z}_{k+1} - \mathbf{h}(\mathbf{x}_{k+1})\right] + \left[\left(\mathbf{x}_{k+1} - \tilde{\mathbf{x}}_{k+1}\right)^T \mathbf{M}_{k+1}^{-1}\left(\mathbf{x}_{k+1} - \tilde{\mathbf{x}}_{k+1}\right)\right], \tag{8.23}$$

where $\tilde{\mathbf{x}}_{k+1}$ is the forecast state vector in (8.19), and $\mathbf{z}_{k+1}$ is the newly received measurement at time $k+1$.

The optimum $\hat{\mathbf{x}}_{k+1}$ that minimizes $J\left(\mathbf{x}_{k+1}\right)$ can be obtained through an iterative EKF as [165]

$$\hat{\mathbf{x}}^{(i+1)} = \hat{\mathbf{x}}^{(i)} + \mathbf{\Sigma}^{(i)}\{\mathbf{H}^T(\hat{\mathbf{x}}^{(i)})\mathbf{R}^{-1}[\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}}^{(i)})] - \mathbf{M}^{-1}[\hat{\mathbf{x}}^{(i)} - \tilde{\mathbf{x}}]\}, \tag{8.24}$$

where i denotes the iteration counter, $\mathbf{H}(\mathbf{x}) = \frac{\partial\mathbf{h}(\mathbf{x})}{\partial\mathbf{x}}$ is the Jacobian matrix, and $\mathbf{\Sigma}^{(i)}$ is the error covariance matrix of the estimation $\hat{\mathbf{x}}^{(i)}$ as

$$\mathbf{\Sigma}^{(i)} = \left[\mathbf{H}^T(\hat{\mathbf{x}}^{(i)})\mathbf{R}^{-1}\mathbf{H}(\hat{\mathbf{x}}^{(i)}) + \mathbf{M}^{-1}\right]^{-1}. \tag{8.25}$$

It should be noted that the subscript $k+1$ was omitted in (8.24) and (8.25) for simplicity.

The EKF is initialized with $\hat{\mathbf{x}}_{k+1}^{(0)} = \tilde{\mathbf{x}}_{k+1}$. Under the assumption that state forecasting has a high accuracy, that is, $|\mathbf{x}_{k+1} - \tilde{\mathbf{x}}_{k+1}|$ is very small, the EKF initialized with $\tilde{\mathbf{x}}_{k+1}$ will converge very fast. Thus we only consider the estimation result after one EKF iteration. Performing the iteration in (8.24) and (8.25) once yields

$$\hat{\mathbf{x}}_{k+1} = \tilde{\mathbf{x}}_{k+1} + \mathbf{K}_{k+1}\mathbf{v}_{k+1}, \tag{8.26}$$

$$\mathbf{\Sigma}_{k+1} = \left[\mathbf{H}^T(\tilde{\mathbf{x}}_{k+1})\mathbf{R}_{k+1}^{-1}\mathbf{H}(\tilde{\mathbf{x}}_{k+1}) + \mathbf{M}_{k+1}^{-1}\right]^{-1}, \tag{8.27}$$

where $\mathbf{K}_{k+1}$ is the Kalman gain matrix defined as,

$$\mathbf{K}_{k+1} = \mathbf{\Sigma}_{k+1}\mathbf{H}^T(\tilde{\mathbf{x}}_{k+1})\mathbf{R}_{k+1}^{-1}, \tag{8.28}$$

and

$$\mathbf{v}_{k+1} = \mathbf{z}_{k+1} - \mathbf{h}(\tilde{\mathbf{x}}_{k+1}), \tag{8.29}$$

is the residual vector.

8.2.3 Hypothesis Test

The attack detection problem can be formulated in the form of a hypothesis test, where the null and alternate hypotheses correspond to the status of normal operation and attack, respectively.

Define the null hypothesis $\mathcal{H}_0$ and alternate hypothesis $\mathcal{H}_1$ at time $t+1$ as

$$\begin{aligned} \mathcal{H}_0 &: \mathbf{z}_{t+1} = \mathbf{h}(\mathbf{x}_{t+1}) + \mathbf{e}_{t+1}, \\ \mathcal{H}_1 &: \mathbf{z}_{t+1} = \mathbf{h}(\mathbf{x}_{t+1}) + \mathbf{e}_{t+1} + \mathbf{A}. \end{aligned} \tag{8.30}$$

Based on the assumption of high forecast accuracy, the nonlinear measurement function $\mathbf{h}(\mathbf{x}_{t+1})$ at time $t+1$ can be approximated by using its first order Taylor series expansion around point $\tilde{\mathbf{x}}_{t+1}$ as,

$$\mathbf{h}(\mathbf{x}_{t+1}) = \mathbf{h}(\tilde{\mathbf{x}}_{t+1}) + \mathbf{H}(\tilde{\mathbf{x}}_{t+1})(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}). \tag{8.31}$$

Combining (8.29), (8.3), and (8.31) yields

$$\mathbf{v}_{t+1} = \mathbf{H}(\tilde{\mathbf{x}}_{t+1})(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}) + \mathbf{e}_{t+1}. \tag{8.32}$$

The covariance matrix of $\mathbf{v}_{t+1}$ in the above equation is

$$\mathbf{S}_{t+1} = \mathbf{H}(\tilde{\mathbf{x}}_{t+1})\mathbf{M}_{t+1}\mathbf{H}^T(\tilde{\mathbf{x}}_{t+1}) + \mathbf{R}_{t+1}. \tag{8.33}$$

The residual vector after attack can be obtained in a similar manner. Then the hypothesis test given in (8.30) can be equivalently expressed in the form of the residual vector $\mathbf{v}_{t+1}$ as

$$\begin{aligned}\mathcal{H}_0 &: \mathbf{v}_{t+1} = \mathbf{H}(\tilde{\mathbf{x}}_{t+1})(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}) + \mathbf{e}_{t+1}, \\ \mathcal{H}_1 &: \mathbf{v}_{t+1} = \mathbf{H}(\tilde{\mathbf{x}}_{t+1})(\mathbf{x}_{t+1} - \tilde{\mathbf{x}}_{t+1}) + \mathbf{e}_{t+1} + \mathbf{A}.\end{aligned} \tag{8.34}$$

The residual $\mathbf{v}_{t+1}$ is assumed to be Gaussian distributed, with its mean vector being zero and $\mathbf{A}$ under the null and alternate hypothesis, respectively [166]. The covariance matrix remains $\mathbf{S}_{t+1}$ with or without attacks. Thus the hypothesis test can be equivalently written as

$$\begin{aligned}\mathcal{H}_0 &: \mathbf{v}_{t+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{S}_{t+1}), \\ \mathcal{H}_1 &: \mathbf{v}_{t+1} \sim \mathcal{N}(\mathbf{A}, \mathbf{S}_{t+1}).\end{aligned} \tag{8.35}$$

To further simplify the detection process, we perform the eigen-decomposition of $\mathbf{S}_{t+1}$ as

$$\mathbf{S}_{t+1} = \mathbf{U}_{t+1}^T \mathbf{D}_{t+1} \mathbf{U}_{t+1}. \tag{8.36}$$

Define a whitened residual vector

$$\bar{\mathbf{v}}_{t+1} = \mathbf{W}_{t+1} \mathbf{v}_{t+1} \tag{8.37}$$

where $\mathbf{W}_{t+1} = \mathbf{D}_{t+1}^{-\frac{1}{2}} \mathbf{U}_{t+1}$ is the whitening matrix. Then the hypothesis test on $\bar{\mathbf{v}}_{t+1}$ can be alternatively expressed as

$$\begin{aligned}\mathcal{H}_0 &: \bar{\mathbf{v}}_{t+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_m), \\ \mathcal{H}_1 &: \bar{\mathbf{v}}_{t+1} \sim \mathcal{N}(\mathbf{W}_{t+1}\mathbf{A}, \mathbf{I}_m).\end{aligned} \tag{8.38}$$

8.3 DQN approach

8.3.1 DQN formulation for Quickest Change Detection

The detail of DQN has been introduced in previous sections. DQN is used for solving this problem instead of DDPG is because that DQN has discrete action space, which fits the behavior of the detector: decide on two actions, whether the system is under attack or not.

The attack detection problem is formulated into the MDP framework for DQN training and testing as following:

- State

  The state of the system should be able to reveal the status change of the power grid before and after an attack. The system measurement vector $\mathbf{z}_t$ is in general not a good candidate for state, because it is possible to obtain the same measurement before and after a carefully designed attack. Various system operation and measurement statistics have been used as state in the literature, such as the state estimation residual [111], the negative log-likelihood function of the DC dynamic state estimation [109], etc. Motivated by the Rao-CUSUM method presented in [2], we propose to solve this problem by using the Rao-test statistic of the whitened residual as [2, 167]

$$Y_t = \bar{\mathbf{v}}_t^T \bar{\mathbf{v}}_t \tag{8.39}$$

  With the this Rao-test statistic, we represent the state at time t by using a size-w sliding window of Rao-test statistics as

$$o(t) = [Y_{t-w+1}, Y_{t-w+2}, \ldots, Y_t]. \tag{8.40}$$

  The state at time t contains the Rao-test statistics calculated from the current and the past $w-1$ measurements. The time evolution of the state can then be used to detect the presence of attacks.

- Action

  Since the objective of the system is attack detection, the action at time t can be simply defined as:

$$a(t) = \begin{cases} 1, & \text{attack detected} \\ 0, & \text{no attack detected} \end{cases} \tag{8.41}$$

- Reward function

  Denote the reward function of taking action $a(t)$ from state $o(t)$ as $r(o(t), a(t))$. The design of the reward function plays an important role in the accuracy, efficiency, and convergence of the learning algorithm. The reward function should take into considerations of both detection accuracy and detection delay. The function will give rewards to correct detection and low detection delays, while false alarms and long detection delays should be penalized. Considering both detection accuracy and detection delay, we propose a new reward function as follows

$$r(o(t), a(t)) = \begin{cases} \frac{1}{\tau}[1 - a(t)], & t < \tau, \\ \frac{1}{\tau}a(t), & t = \tau, \\ -\phi[1 - a(t)], & t > \tau, \end{cases} \tag{8.42}$$

where $\phi \in [0, 1]$ is a parameter adjusting the trade-off between ADD and PFA. If ϕ is close to 0, then there is a very small penalty to delayed detection, which leads to a long ADD but a low PFA. On the other hand, a larger ϕ will lead to a severe penalty to long detection delays, which results in low ADD at the cost of potentially higher PFA. The stage transitions related to various actions and the corresponding reward functions are illustrated in Fig. 8.1.

Assume the attack is detected at time $\hat{\tau}$, that is, $a(t) = 0$ for $t < \hat{\tau}$ and $a(\hat{\tau}) = 1$. Define an episode $\mathcal{E}$ as a sequence of state and action pairs between $t \in \{1, 2, \ldots, \hat{\tau}\}$, that is, $\mathcal{E} = \{(o(1), a(1)), (o(2), a(2)), \cdots, (o(\hat{\tau}), a(\hat{\tau}))\}$. Then the accumulated episode reward function is

$$r(\mathcal{E}) = \sum_{t=1}^{\hat{\tau}} r(o(t), a(t)). \tag{8.43}$$

An action of $a = 1$ will always lead to the terminal stage, that is, the end of an episode. The reward function in (8.42) is designed to have an accumulated episode reward of 1 for perfect detection, that is, $r(\mathcal{E}) = 1$ if $a(\tau) = 1$. The reward function will be strictly less than 1 for false alarm or delayed detection. For excessive long delays, the reward function will be negative. In (8.43), the accumulated episode reward will be negative if the detection delay is larger than $\frac{\tau-1}{\phi\tau}$.

Assume the entire DQN training time horizon is divided into multiple sequential training episodes. Each episode has at most T time steps. A training episode ends if an attack is detected before T time steps, or no attack is detected in T time steps. For each new episode, the power grid is initialized to the same normal operating condition, but with different random attack vectors applied at different time steps. The entire training process consists E training episodes.

In order to stabilize the training process and avoid big swings from step to step, we adopt a target network $Q^{'}(o, a)$, which is built along the main network $Q(o, a)$. The weights of the main network, θ, are updated every step, yet the weights of the target network, $\theta^{'}$, are

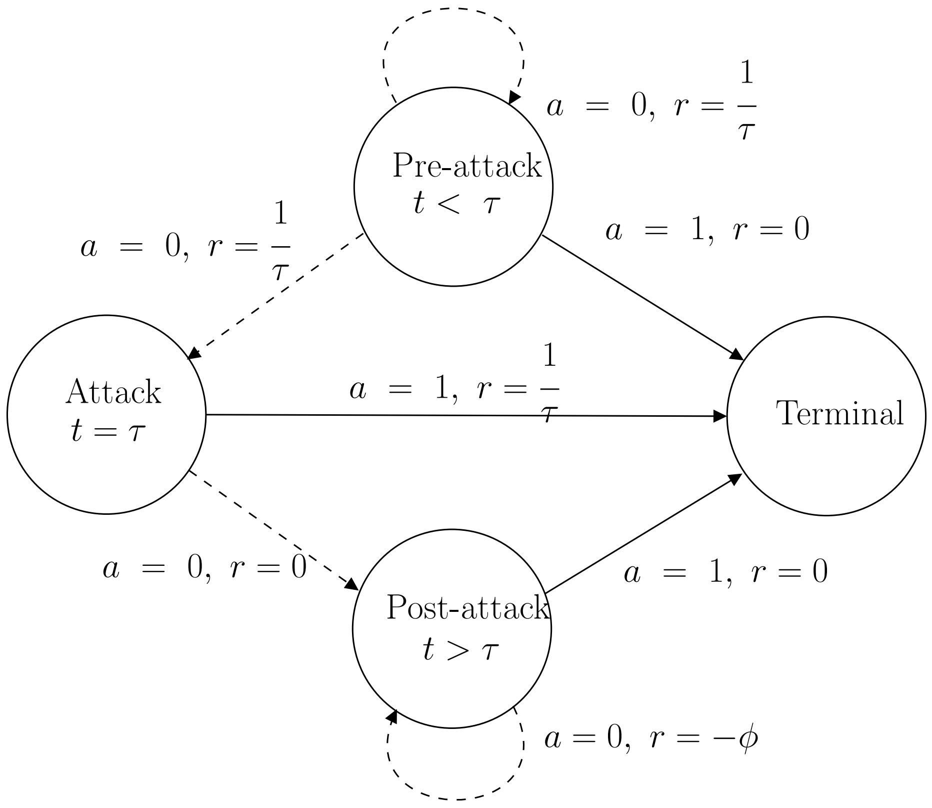


Figure 8.1: Stage transitions

copied from the main network every C steps, with C being an integer. The action selection is performed by using the target network $Q^{'}(o, a)$.

In order to broaden the exploration of the action space, we adopt an ϵ-greedy policy in action selection. In the ϵ-greedy policy, a given probability parameter $\epsilon \in [0, 1]$ is chosen. During the action selection process, we either select a random action with probability ϵ, or greedily select an action by using greedy policy with probability $1 - \epsilon$. Such a randomized action selection approach can broader the search space and avoid being trapped in a local optimum early during the training process.

Experience replay is used to improve the sample efficiency and stability of the learning process. In experience replay, we can obtain in each time step an experience tuple, $e(t) = \{o(t), a(t), r(t), o(t + 1)\}$, which is stored in a replay buffer $\mathcal{R}$. The weights of the main network are updated by randomly sample a mini-batch $\mathcal{M} \subseteq \mathcal{R}$ experience tuples from the replay buffer. For each experience tuple in the mini-batch, we first calculate the corresponding target value as

$$y(i) = \begin{cases} r(i), & \text{if } i = T \\ r(i) + \gamma \max_{a \in \mathcal{A}} Q^{'}(o(i+1), a; \theta^{'}), & \text{otherwise} \end{cases} \tag{8.44}$$

Then the weights of the main network can be updated by minimizing the mean squared error between the main network and the target values of the mini-batch as

$$\min_{\theta} \frac{1}{|\mathcal{M}|} \sum_{e(i) \in \mathcal{M}} [y(i) - Q(o(i), a(i); \theta)]^2 \tag{8.45}$$

The minimization can be performed by using gradient descent. With experience replay, the main network weights are updated by using a random subset of experience tuples. Such an approach allows the algorithm to learn from uncorrelated experiences from the past, recall rare occurrences, and learn from individual experiences multiple times. As a result, it can learn more efficiently from the past experience with better stability.

Details of the DQN training procedure are shown in Algorithm 9.

Algorithm 9 DQN Learning

```
Require: Bus number $N$, measurement size $m$, window size $w$, attack time $\tau$, initial action-value network parameter $\theta$, empty replay buffer $\mathcal{R}$
 1: Initialization: Set target network weights: $\theta^{'} \leftarrow \theta$.
 2: for $e = 1$ to $E$ do
 3:     Set $t \leftarrow 1$; calculate $o(1)$ by dynamic state estimation.
 4:     while $t \leq T$ do
 5:         Select action $a(t)$ by following the $\epsilon$-greedy policy.
 6:         if $a(t) = 1$ then
 7:             $t \leftarrow T + 1$
 8:         else
 9:             Calculate reward $r(t)$ by using (8.42).
10:             Calculate $o(t + 1)$ by dynamic state estimation.
11:             Store experience tuple $(o(t), a(t), r(t), o(t + 1))$ in the replay buffer $\mathcal{R}$.
12:             Randomly sample a mini-batch of $|\mathcal{M}|$ experience tuples $\mathcal{M} = \{(o(i), a(i), r(i), o(i + 1))\}$ from $\mathcal{R}$.
13:             Calculate the target value $y(i)$ for all experience tuples in $\mathcal{M}$ by using (8.44).
14:             Update the main network weights $\theta$ by minimizing the cost function in (8.45).
15:             if mod $(t, C) = 0$ then
16:                 Update the target network weights $\theta' \leftarrow \theta$.
17:             end if
18:             $t \leftarrow t + 1$
19:         end if
20:     end while
21:     $e \leftarrow e + 1$
22: end for
Ensure: Target network parameters $\theta^{'}$
```

8.3.2 Complexity Analysis

The computation complexity of the proposed algorithm comes from two sources: the dynamic state estimation (DSE) with extended Kalman filter (EKF), and the DQN algorithm.

During DSE with an AC system model, the EKF is used to estimate and track the dynamic state transition of the power grid. The estimation results are then used to calculate the Rao-statistics to form the state vector for the DQN. This procedure is performed at each time step during the training and testing process.

Consider a power system with N buses and M lines. The size of the state vector $\mathbf{x}$ is $n \times 1$, where $n = 2N - 1$. The dimension of the measurement vector $\mathbf{z}$ is $m \times 1$, where $m = 2(M + N) + 1$. The Holt's linear trend method in section III-A requires 2 vector add (VA) in (8.21), and multiple scalar multiplications in (8.20) and (8.21). The computational cost of scalar multiplication is much lower than matrix operations such as VA, matrix-vector product (MVP), matrix-matrix product (MMP), and matrix inversion (MI). Therefore, only VA, MVP, MMP, MI are counted during the complexity analysis. The state forecasting in (8.19) has 1 VA and 1 MVP, the calculation of error covariance matrix in (8.22) has 2 MMPs and 1 VA.

The state estimation is obtained from one EKF iteration in (8.26), which has 1 MVP and 1 VA. The calculation of the residual vector requires 1 VA in (8.29). The Kalman gain matrix requires 2 MMPs in (8.28) and 2 MMPs, 1 VA, and 2 MI in (8.27). A summary of vector and matrix operations of the DSE-EKF is given in Table 8.1. Given $M > N$ in most cases, the DSE has a computation complexity of $\mathcal{O}(N(M + N)^2)$.

Regarding the DQN algorithm, we only need to consider the complexity of the online detection process, because the training is performed offline. The computational complexity of the online DQN algorithm comes from the computation cost in the target network. During the online DQN detection process, the target network takes an input $o(t)$ of dimension w. It has two hidden layers with dimension k each, and generates an output of dimension 2, $Q(o(t), 0; \theta')$ and $Q(o(t), 1; \theta')$. The target network has 4 layers with a total of $w + 2k + 2$ $(w, k, k, 2)$ neurons. For each layer, an MVP and an activation function are computed. Thus the online DQN detection requires 3 MVPs with complexity $\mathcal{O}(wk), \mathcal{O}(k^2)$, and $\mathcal{O}(k)$, respectively, and activation computation with complexity $\mathcal{O}(k)$. Given $k > w$ in the DQN, the complexity of the online DQN algorithm is $\mathcal{O}(k^2)$ and it is independent of the size of the power grid.

Operation	Number	Complexity
VA	7	$\mathcal{O}(M+N)$
MVP	2	$\mathcal{O}(N(M+N))$
MMP	6	$\mathcal{O}(N(M+N)^2)$
MI	2	$\mathcal{O}(N^3)$

Table 8.1: Number and complexity of vector and matrix operations

8.4 Simulation results

8.4.1 System Setup

The simulations are performed on a 13-bus system with two areas as shown in Fig. 8.2 using MATLAB Power System Toolbox (PST V3.0) [168]. Bus 1 is used as the reference bus. The measurement vector consists of $m = 55$ components, including the voltage magnitude of bus 1, the active and reactive power injections at all 13 buses, and the active and reactive power flows at all 14 lines. The state vector consists of $n = 25$ components, which are the voltage magnitudes at all 13 buses and the phase angles at the 12 non-reference buses. The time interval of the simulation is $\Delta t = 0.01s$, which corresponds to a sampling rate of 100 Hz. The maximum length of each episode is $T = 200$ samples, which corresponds to a time duration of 2 seconds. The measurement and state vectors are considered as the true values of $\mathbf{z}$ and $\mathbf{x}$, respectively. For the AC system DSE parameters, the covariance matrix of measurement error is set as $\mathbf{R} = \text{diag}(10^{-5}, 10^{-6}, \ldots, 10^{-6})$. The parameters for Holt's linear trend method are $\alpha = 0.95$ and $\beta = 0.001$. The covariance matrix of state transition

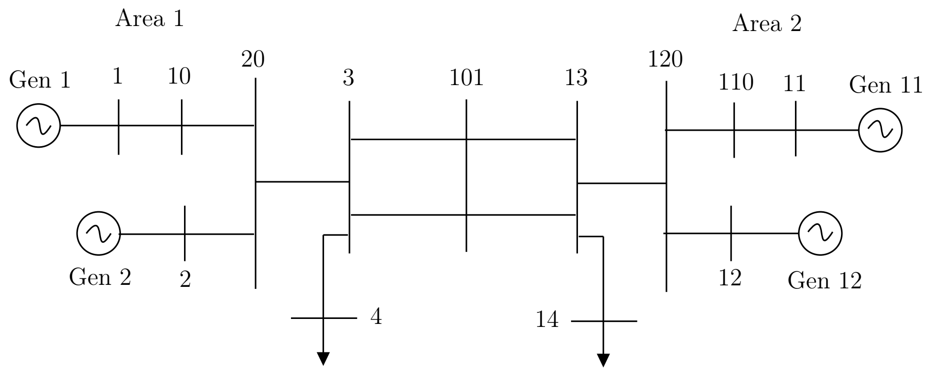


Figure 8.2: 13-bus Two Area System [1]

error is $\mathbf{Q}_t = 10^{-6}\mathbf{I}_n$.

The DQN algorithm is implemented in Python using Stable Baselines [144], with which we built a customized power system environment for our simulations. Hyperparameters are tuned manually based on the episode reward curve and state distributions in Tensorboard. The Q-network and the target network each have two hidden layers with 32 nodes per layer. The discount factor is $\gamma = 1$, and the learning rate for updating θ in (8.45) is 0.0005. The size of the buffer $\mathcal{R}$ is $50,000$. The exploration probability of the ϵ-greedy policy is $\epsilon = 0.1$, and the number of transitions in a mini-batch is $|\mathcal{M}| = 32$. The frequency of the target network update is $C = 500$.

8.4.2 Training Results

During the training stage, the number of buses under attack is generated by uniformly sampling from $\{1, 2, \ldots, m\}$. Once the number of buses under attack is determined, the set of indices of buses under attack, $\mathcal{I}$, are sampled uniformly without replacement from the index set of all buses. The attack time τ is uniformly sampled from $\{1, 2, \ldots, T = 200\}$. The elements of the attack vector $\mathbf{a}$ are uniformly sampled from $[-2, 2]$ p.u. for FDI attacks. The system is only trained for FDI attacks. The model trained with FDI attacks will be tested against both FDI and DoS attacks during the testing stage. In addition to cyberattacks, the normal system dynamic and state transitions of the power grid are simulated by increasing the active load at bus 4 by 0.5 per unit (p.u.) at $t = 0$. Model trained under such a deterministic load change will be tested against systems with random load changes during the testing stage. The length of the sliding window used in the MDP state $o(t)$ in (8.40) is $w \in \{1, 2, 4\}$. The trade-off parameter ϕ used in the MDP reward function in (8.42) is chosen from $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$. Each agent with a given set of parameters is trained for $E = 10^5$ episodes, which takes about 4 hours on a workstation with a 6-core Intel Core i7-5820 K CPU operating at 3.3 GHz and 32 GB of random access memory (RAM).

Fig. 8.3 shows the episode reward curves for $w \in \{1, 2, 4\}$ and $\phi = 0.1$. The shadow lines are the real episode rewards and the solid lines are the episode rewards after a Gaussian-weighted moving average over 20 consecutive samples. Only the positive episode reward is shown in the figure. The agent with $w = 1$ failed to reach a high episode reward after being trained for $E = 10^5$ episodes, thus it failed to learn a useful detection strategy. This is due to the fact that it makes decisions based on the Rao-test statistic from only the current

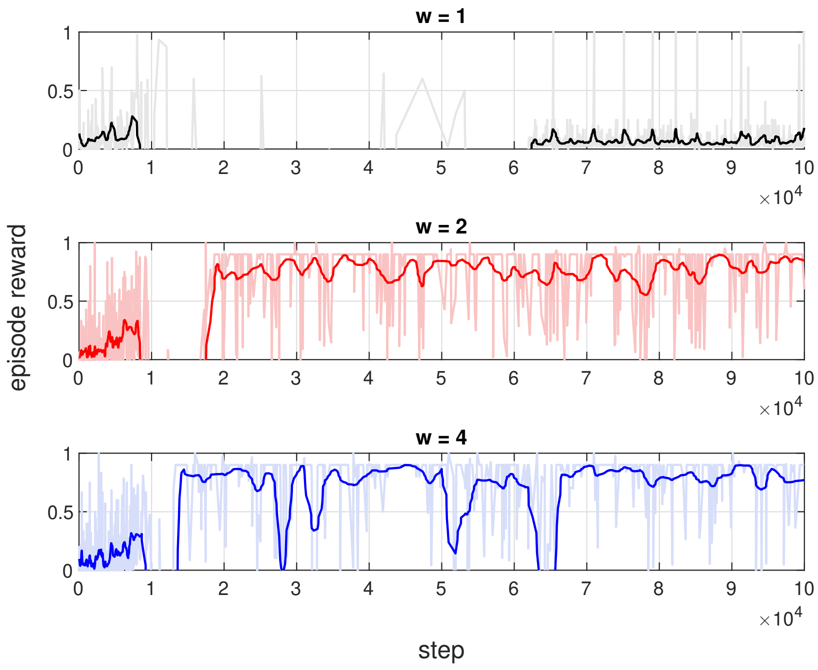


Figure 8.3: Learning curve for $\phi = 0.1$ and different w

measurement while ignoring all previous measurements. The training results for agents with $w = 2$ and $w = 4$ successfully reached an episode reward that is close to 1 after being trained for 2×10^4 episodes. The models obtained from the training stage are then used during the testing stage.

8.4.3 Testing Results

The DQN models are tested on the same power system but with different system dynamics and random cyberattacks. For FDI attacks, the attack indices, attack time, and attack vector values are all randomly generated by following the same distributions as described in the training stage. For DoS attacks, the attack indices are generated as the same way as FDI attacks, and the attack matrix $\mathbf{A}$ is generated according to (8.4). Each agent is tested for 1,000 Monte Carlo simulations, the PFA and ADD are calculated from the simulated detection results according to (8.7) and (8.8).

Algorithm 10 DQN Testing: Online Detection

```
Require: Target Q-network parameters $\theta^{'}$ obtained from Algorithm 1, network measurements, episode length $T$.
 1: Initialization: $t \leftarrow 1$; $\hat{\tau} = \infty$.
 2: while $t \leq T$ do
 3:     Calculate $o(t)$.
 4:     Update $a(t)$ as
            $$a(t) \leftarrow \arg\max_{a} Q(o(t), a; \theta^{'})$$
 5:     if $a(t) = 1$ then
 6:         $\hat{\tau} \leftarrow t$
 7:         Break
 8:     end if
 9:     $t \leftarrow t + 1$
10: end while
Ensure: $\hat{\tau}$
```

During one testing episode (Monte Carlo trial), the agent works as an online detector. At the t-th time-step, it obtains the real measurement $\mathbf{z}_t$ and then calculates the MDP state $o(t)$ from $\mathbf{z}_t$ and historical data. The optimal action is made according to $a(t) = \arg\max_a Q(o, a; \theta^{'})$, where $\theta^{'}$ are the target network parameters obtained through the training stage. The testing episode ends if an attack is detected or the end of the episode is reached. Detailed testing procedures in each testing episode are presented in Algorithm 10.

The testing results of FDI attacks for agents with $w \in \{1, 2, 4\}$ and $\phi \in \{0.5, 1\}$ are given in Table 8.2, where each entry represents the (PFA, ADD) pair obtained for a given configuration. As discussed in section 8.3, the parameter ϕ can be used to tune the trade-off between ADD and PFA, with a larger ϕ leading to a bigger penalty for detection delay. Such a trade-off relationship can be observed in Table 8.2, where increasing ϕ from 0.5 to 1 leads to a shorter ADD but a slightly larger PFA. As shown in Fig. 8.3, the agent with $w = 1$ fails to learn during the training phase, thus the testing PFA is close to 1. The agent with $w = 1$ is just a one-shot soft threshold detector without utilizing historical data. Increasing w from 1 to 2 or 4 leads to systems with considerably better performance. The agents with $w = 2$ slightly outperform their $w = 4$ counterparts in terms of both PFA and ADD. In addition,

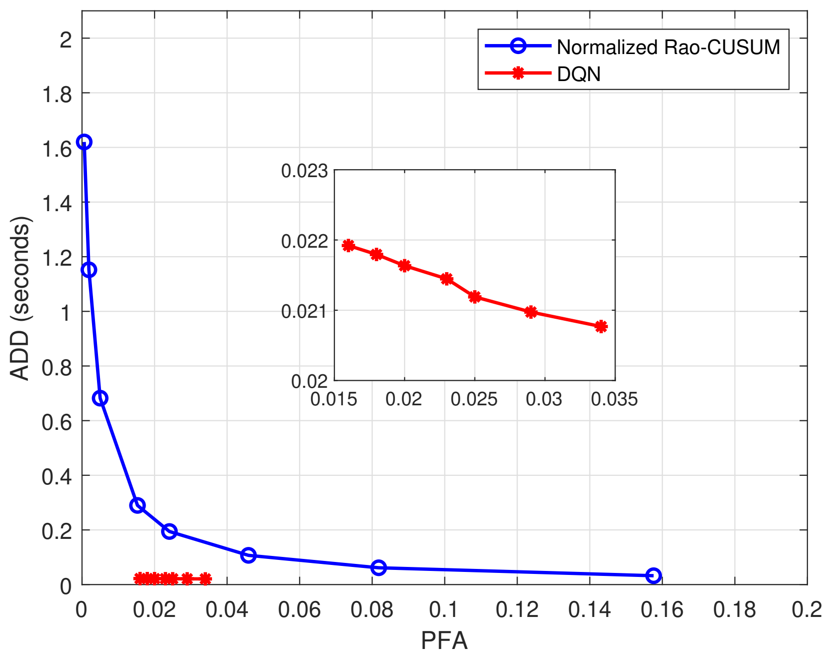


Figure 8.4: Performance of DQN detector ($w = 2$) and Normalized Rao-CUSUM detector [2] under FDI attack

the model complexity of $w = 2$ is lower than that of $w = 4$ due to the lower dimension of $o(t)$.

Table 8.2: FDI Testing results (PFA, ADD)

Parameters	$\phi = 0.5$	$\phi = 1$
$w = 1$	0.993, 0	0.999, 0
$w = 2$	0.029, 2.0974e-2	0.034, 2.0768e-2
$w = 4$	0.033, 4.0513e-2	0.036, 4.0156e-2

The performances of the proposed DQN-based detector under FDI and DoS attacks are shown in Figs. 8.4 and 8.5, respectively, where the ADD is plotted as a function of the PFA. In the simulations we have $w = 2$, and the results are compared to that from the Rao-CUSUM detector [2]. The multiple points on the ADD-PFA trade-off curve of the DQN detectors are obtained by setting $\phi \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$. The trade-off curve of the Normalized Rao-CUSUM detector is obtained by choosing different values for detection threshold as described in [2]. Every point on the curves is obtained by 1,000 Monte Carlo

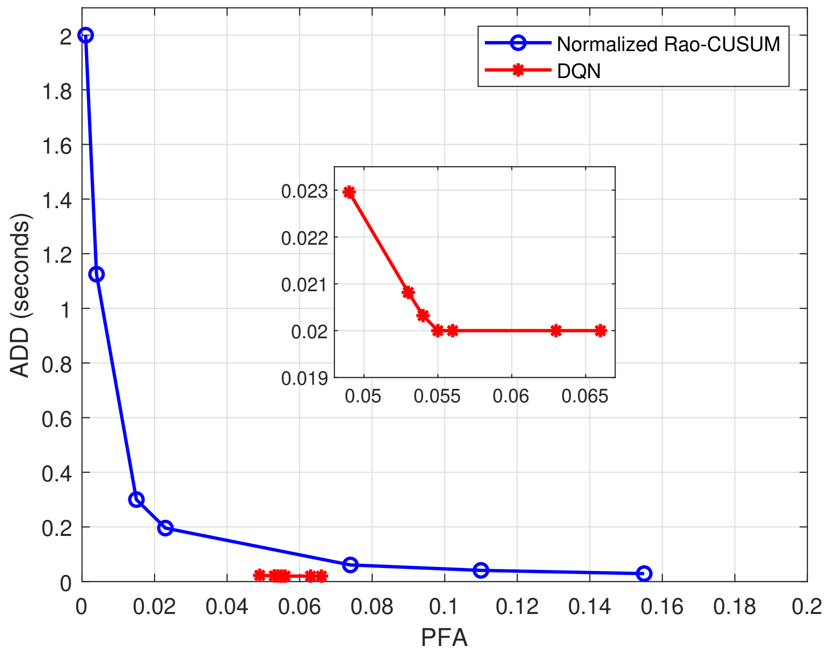


Figure 8.5: Performance of DQN detector ($w = 2$) and Normalized Rao-CUSUM detector [2] under DoS attack

trials. The proposed DQN-based detector outperforms the Rao-CUSUM detector in terms of both PFA and ADD under both FDI and DoS attacks. Note that the ADD for both detectors are based on the time interval of the simulation. In practice, the SCADA updates every 2-5 s, the computation time of DSE in our detector is within 5s for a system with 200 buses or less [169], and the complexity of online the online DQN detection process is much smaller compared to that of DSE. Since the DQN model is trained under FDI attacks, systems with FDI attacks slightly outperform those with DoS attacks.

Fig. 8.6 shows the real power at bus 14 under FDI attacks. In case of FDI attacks, the real power measurement at bus 13 is falsely decreased by 1.5 p.u. and that at bus 14 is falsely increased by 1 p.u. between 0.25 and 0.6 seconds. The proposed DQN-based detector can correctly detect the presence of FDI. Upon detection of FDI, we can remove the false data and replace them with estimated and predicted power values, which are very close to their true values.

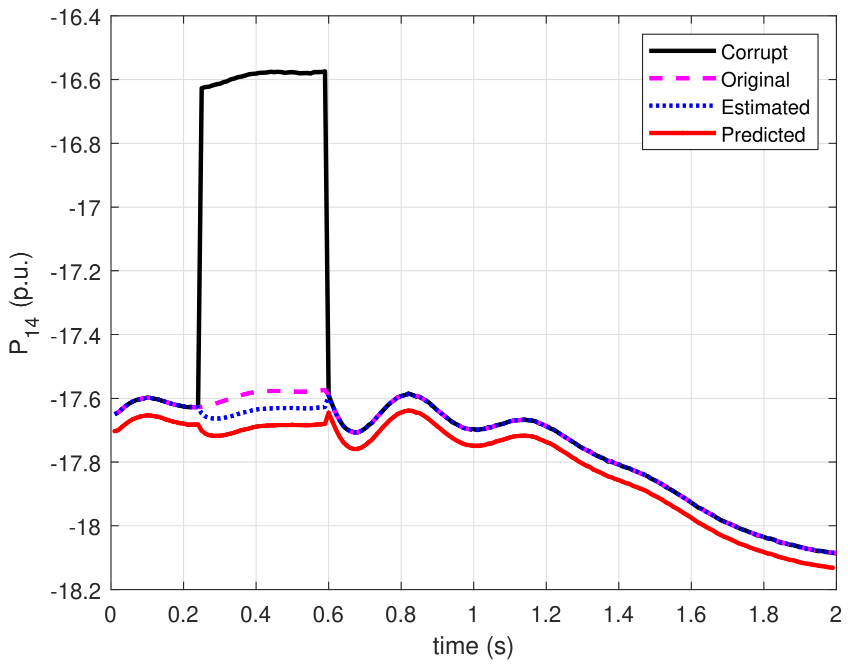


Figure 8.6: The real power at bus 14 with FDI at $0.25 < t < 0.6$

8.5 Conclusion

A DQN-based deep reinforcement learning algorithm has been proposed for the low latency detection of cyberattacks, such as FDI and DoS attacks, in smart grids. Unlike conventional detection methods that focus solely on detection accuracy, the proposed algorithm aims at minimizing the average detection delay while maintaining a low probability of false alarm. The design objective was achieved by developing a DQN-based reinforcement learning algorithm with dynamic AC power system models, which can accurately model dynamic state transients in power systems and identify cyberattacks in real time. The DQN-based reinforcement learning algorithm was developed by following an MDP framework. The MDP state was formulated by using a sliding window of Rao-statistics that can accurately capture the dynamic state evolution of the power grid in real time. A new reward function was designed to allow a flexible trade-off between ADD and PFA. Simulation results demonstrated that the DQN-based RL detection algorithm can achieve very low detection delays while maintaining good PFA performance, and it can achieve considerable performance gains over the existing Rao-CUSUM algorithm. For future works, we plan to apply and improve the proposed algorithm to more sophisticated cyberattacks, e.g., cyberattacks generated by using machine learning algorithms such as generative adversary network (GAN).

9 Conclusion and Future Work

This dissertation demonstrates an overlook research procedures to achieve high renewable energy integration in smart grid with machine learning, which includes the integration of solar energy to a consumer's house or building with ESS and the integration of solar and wind energy to the grid with ESS or MEC. The overall contributions of this dissertation can be summarized as follows:

This dissertation demonstrates several machine learning based algorithms for achieving high renewable energy integration in smart grids which includes the individual consumer's perspective, the grid control, and the cybersecurity concerns. The overall contributions of this dissertation can be summarized as follows: First, this dissertation creates different models for renewable energy integration in different scenarios, and proposes low complexity and real-time algorithms to solve the schedule problem. Next, this dissertation presents a variety of optimization techniques which includes traditional LP, MILP, DP, and MDP, heuristic algorithms such as PSO, and RL-based sequential decision making algorithms such as DQN, DDPG. Some of them are combined with QCD framework for cybersecurity research.

Several training and testing environments for simulating the power system with ESS in Python are developed in this dissertation. Some environments integrates the matlab model and utilizing reinforcement learning techniques, those models can be further used for researchers who are not familiar with building RL environments and data process in Python. Additionally, some environments are build solidly on Python following the Gynmasium environment rules, which can be used for training and testing by researchers who are not familiar with power system simulations.

If we review the research objectives at beginning of the dissertation. The optimum design of ESS-assisted PV system considering parameters such as system cost, battery and solar panel aging, is modeled and solved for a long-horizon of 10 years in Chapter 3. Not only a MINLP model is proposed for optimum design, but also a low-complexity DP based suboptimal solution is proposed. The total savings and break-even points are given as a reference. The DDPG based on-line scheduling algorithm is proposed in Chapter 4 for a given ESS-assisted PV system. The DDPG agent learns from the off-line algorithm and requires no environment models and provides a real-time continuous control.

A DDPG-based OPF for microgrid with renewable integration is proposed in Chapter 5, which requires no prediction model on future renewable energy generation and user load. The research gap of low computational complexity and prediction model is solved. Such algorithm is extended to MEC control in Chapter 6 where the transportation model of the MECS are integrated into the state of the DDPG agent, where no complex transition models and constraints are needed.

The cybersecurity of the smart grid integrated with high renewable energy is partially solved. The cyberattacks on the PV farm can be quickly detected by dynamic watermarking-based algorithm with a minimal delay in Chapter 7, while the attacks on the grid can be quickly detected by a DQN-based algorithm in Chapter 8. Thus, all of the objectives listed at the beginning of the dissertation are successfully achieved.

There are several future directions that can be moved from this dissertation. They are outlined below:

- **Train and test the RL agent on different networks:**

  In the OPF control simulation, we used IEEE-14 bus system and make modifications to the original system. IEEE-14 bus system is a small distribution network for simulation, therefore the agent trained for this system need to be further trained on larger networks or even a hardware implementation. Similarly, in the cybersecurity simulation, the two area 13-bus system is used. The proposed detection agent should be tested on other networks.

- **Use the state-of-art RL agent:**

  The development of reinforcement learning algorithms is changing rapidly. Many actor-critic structured algorithms such as Asynchronous Advantage Actor Critic (A3C) [170], Proximal Policy Optimization (PPO) [171], and Twin Delayed DDPG (TD3) [172] have been applied for control problems and out-performance DDPG.

- **Use machine learning-based attack:**

  In our study on cybersecurity, we design and test attacks using traditional methods such as FDI, DoS, while many attack models are generated using ML algorithms. New attack model needs to be studied and the detection algorithms proposed in this paper need to be improved for these attacks.

Bibliography

[1] G. Rogers, *Power system oscillations.* Springer Science & Business Media, 2012.

[2] S. Nath, I. Akingeneye, J. Wu, and Z. Han, "Quickest detection of false data injection attacks in smart grid with dynamic models," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 10, no. 1, pp. 1292–1302, 2019.

[3] I. O'MALLEY, "U.S. renewable electricity surpassed coal in 2022," *Associated Press*, 2023.

[4] J. Collins. (2018) 2020 solar power: California officially codifies mandate for homes. [Online]. Available: https://www.governing.com/topics/transportation-infrastructure/tns-california-solar-power-homes.html

[5] B. Palmintier, R. Broderick, B. Mather, M. Coddington, K. Baker, F. Ding, M. Reno, M. Lave, and A. Bharatkumar, "On the path to sunshot. emerging issues and challenges in integrating solar with the distribution system," National Renewable Energy Lab.(NREL), Golden, CO (United States), Tech. Rep., 2016.

[6] A. Qazi, F. Hussain, N. A. Rahim, G. Hardaker, D. Alghazzawi, K. Shaban, and K. Haruna, "Towards sustainable energy: a systematic review of renewable energy sources, technologies, and public opinions," *IEEE access*, vol. 7, pp. 63 837–63 851, 2019.

[7] K. M. Tan, T. S. Babu, V. K. Ramachandaramurthy, P. Kasinathan, S. G. Solanki, and S. K. Raveendran, "Empowering smart grid: A comprehensive review of energy storage technology and application with renewable energy integration," *Journal of Energy Storage*, vol. 39, p. 102591, 2021.

[8] X. Xia and A. Elaiw, "Optimal dynamic economic dispatch of generation: A review," *Electric power systems research*, vol. 80, no. 8, pp. 975–986, 2010.

[9] D. Gayme and U. Topcu, "Optimal power flow with large-scale storage integration," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 709–717, 2012.

[10] N. T. Nguyen, D. D. Le, C. Bovo, and A. Berizzi, "Optimal power flow with energy storage systems: Single-period model vs. multi-period model," in *2015 IEEE Eindhoven PowerTech.* IEEE, 2015, pp. 1–6.

[11] J. Kim and Y. Dvorkin, "Enhancing distribution system resilience with mobile energy storage and microgrids," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 4996–5006, 2018.

[12] S. Yao, P. Wang, X. Liu, H. Zhang, and T. Zhao, "Rolling optimization of mobile energy storage fleets for resilient service restoration," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1030–1043, 2019.

[13] M. C. Bozchalui and R. Sharma, "Analysis of electric vehicles as mobile energy storage in commercial buildings: Economic and environmental impacts," in *2012 IEEE Power and Energy Society General Meeting*. IEEE, 2012, pp. 1–8.

[14] C. Greer, D. A. Wollman, D. E. Prochaska, P. A. Boynton, J. A. Mazer, C. T. Nguyen, G. J. FitzPatrick, T. L. Nelson, G. H. Koepke, A. R. Hefner Jr *et al.*, "Nist framework and roadmap for smart grid interoperability standards, release 3.0," Tech. Rep., 2014.

[15] D. U. Case, "Analysis of the cyber attack on the ukrainian power grid," *Electricity Information Sharing and Analysis Center (E-ISAC)*, vol. 388, 2016.

[16] A. S. Musleh, G. Chen, and Z. Y. Dong, "A survey on the detection algorithms for false data injection attacks in smart grids," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2218–2234, 2019.

[17] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, pp. 1–33, 2011.

[18] A. Huseinovic, S. Mrdovic, K. Bicakci, and S. Uludag, "A survey of denial-of-service attacks and solutions in the smart grid," *IEEE Access*, 2020.

[19] M. Z. Gunduz and R. Das, "Cyber-security on smart grid: Threats and potential solutions," *Computer networks*, vol. 169, p. 107094, 2020.

[20] X. Liu, M. Shahidehpour, Y. Cao, L. Wu, W. Wei, and X. Liu, "Microgrid risk analysis considering the impact of cyber attacks on solar pv and ess control systems," *IEEE transactions on smart grid*, vol. 8, no. 3, pp. 1330–1339, 2016.

[21] A. Teymouri, A. Mehrizi-Sani, and C.-C. Liu, "Cyber security risk assessment of solar pv units with reactive power capability," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 2872–2877.

[22] I. Akingeneye and J. Wu, "Pmu-assisted bad data detection in power systems," in *2018 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)*. IEEE, 2018, pp. 1–5.

[23] A. Borghetti, C. D'Ambrosio, A. Lodi, and S. Martello, "An MILP Approach for Short-Term Hydro Scheduling and Unit Commitment With Head-Dependent Reservoir," *IEEE Transactions on Power Systems*, vol. 23, no. 3, pp. 1115–1124, Aug. 2008.

[24] S. Chouhan, D. Tiwari, H. Inan, S. Khushalani-Solanki, and A. Feliachi, "DER optimization to determine optimum BESS charge/discharge schedule using Linear Programming." IEEE, Jul. 2016, pp. 1–5.

[25] T. A. Nguyen and R. H. Byrne, "Maximizing the cost-savings for time-of-use and net-metering customers using behind-the-meter energy storage systems." IEEE, Sep. 2017, pp. 1–6.

[26] K. Kwan and D. Maly, "Optimal battery energy storage system (BESS) charge scheduling with dynamic programming," *IEE Proceedings - Science, Measurement and Technology*, vol. 142, no. 6, pp. 453–458, Nov. 1995.

[27] Y. Riffonneau, S. Bacha, F. Barruel, and S. Ploix, "Optimal Power Flow Management for Grid Connected PV Systems With Batteries," *IEEE Transactions on Sustainable Energy*, vol. 2, no. 3, pp. 309–320, Jul. 2011.

[28] V. Marano, G. Rizzo, and F. A. Tiano, "Application of dynamic programming to the optimal management of a hybrid power plant with wind turbines, photovoltaic panels and compressed air energy storage," *Applied Energy*, vol. 97, pp. 849–859, Sep. 2012.

[29] R. Kamyar and M. M. Peet, "Multi-objective dynamic programming for constrained optimization of non-separable objective functions with application in energy storage," in *Decision and Control (CDC), 2016 IEEE 55th Conference on.* IEEE, 2016, pp. 5348–5353.

[30] Y. Choi and H. Kim, "Optimal Scheduling of Energy Storage System for Self-Sustainable Base Station Operation Considering Battery Wear-Out Cost," *Energies*, vol. 9, no. 6, p. 462, Jun. 2016.

[31] M. Jones and M. M. Peet, "Solving dynamic programming with supremum terms in the objective and application to optimal battery scheduling for electricity consumers subject to demand charges." IEEE, Dec. 2017, pp. 1323–1329.

[32] H. Dagdougui, R. Minciardi, A. Ouammi, M. Robba, and R. Sacile, "A dynamic decision model for the real-time control of hybrid renewable energy production systems," *IEEE Systems Journal*, vol. 4, no. 3, pp. 323–333, Sep. 2010.

[33] Y. Li and J. Wu, "Optimum design of battery-assisted photo-voltaic energy system for a commercial application," in *2019 IEEE Power & Energy Society General Meeting (PESGM).* IEEE, 2019.

[34] A. T. Nguyen and S. Chaitusaney, "Optimum schedule and size of BESS in the low voltage network with high penetration of solar rooftops to maintain voltages within acceptable limit." IEEE, Jun. 2017, pp. 194–197.

[35] S.-W. Hwangbo, B.-J. Kim, and J.-H. Kim, "Application of economic operation strategy on battery energy storage system at Jeju." IEEE, Apr. 2013, pp. 1–8.

[36] R. T. de Salis, A. Clarke, Z. Wang, J. Moyne, and D. M. Tilbury, "Energy storage control for peak shaving in a single building," in *PES General Meeting| Conference & Exposition, 2014 IEEE.* IEEE, 2014, pp. 1–5.

[37] J. von Appen and M. Braun, "Sizing and Improved Grid Integration of Residential PV Systems With Heat Pumps and Battery Storage Systems," *IEEE Transactions on Energy Conversion*, vol. 34, no. 1, pp. 562–571, Mar. 2019.

[38] P. Harsha and M. Dahleh, "Optimal Management and Sizing of Energy Storage Under Dynamic Pricing for the Efficient Integration of Renewable Energy," *IEEE Transactions on Power Systems*, vol. 30, no. 3, pp. 1164–1181, May 2015.

[39] L. Hernandez, C. Baladron, J. M. Aguiar, B. Carro, A. J. Sanchez-Esguevillas, J. Lloret, and J. Massana, "A survey on electric power demand forecasting: future trends in smart grids, microgrids and smart buildings," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1460–1495, 2014.

[40] F. Hafiz, M. Awal, A. R. de Queiroz, and I. Husain, "Real-time stochastic optimization of energy storage management using deep learning-based forecasts for residential pv applications," *IEEE Transactions on Industry Applications*, vol. 56, no. 3, pp. 2216–2226, 2020.

[41] L. Tang, Y. Yi, and Y. Peng, "An ensemble deep learning model for short-term load forecasting based on arima and lstm," in *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2019, pp. 1–6.

[42] C. Feng and J. Zhang, "Reinforcement learning based dynamic model selection for short-term load forecasting," in *2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE, 2019, pp. 1–5.

[43] L. Yu, S. Qin, M. Zhang, C. Shen, T. Jiang, and X. Guan, "A review of deep reinforcement learning for smart building energy management," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 046–12 063, 2021.

[44] B. V. Mbuwir, M. Kaffash, and G. Deconinck, "Battery scheduling in a residential multi-carrier energy system using reinforcement learning," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2018, pp. 1–6.

[45] Y. Wei, F. R. Yu, M. Song, and Z. Han, "User scheduling and resource allocation in hetnets with hybrid energy supply: An actor-critic reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 680–692, 2017.

[46] V. François-Lavet, D. Taralla, D. Ernst, and R. Fonteneau, "Deep reinforcement learning solutions for energy microgrids management," in *European Workshop on Reinforcement Learning (EWRL 2016)*, 2016.

[47] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, and J. G. Slootweg, "On-line building energy optimization using deep reinforcement learning," *IEEE Transactions on Smart Grid*, 2018.

[48] N. Tsang, C. Cao, S. Wu, Z. Yan, A. Yousefi, A. Fred-Ojala, and I. Sidhu, "Autonomous household energy management using deep reinforcement learning," in *2019 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. IEEE, 2019, pp. 1–7.

[49] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[50] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," *arXiv preprint arXiv:1812.07394*, 2018.

[51] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Deep deterministic policy gradient (ddpg) based energy harvesting wireless communications," *IEEE Internet of Things Journal*, 2019.

[52] R. Liessner, C. Schroer, A. M. Dietermann, and B. Bäker, "Deep reinforcement learning for advanced energy management of hybrid electric vehicles." in *ICAART (2)*, 2018, pp. 61–72.

[53] L. Yu, W. Xie, D. Xie, Y. Zou, D. Zhang, Z. Sun, L. Zhang, Y. Zhang, and T. Jiang, "Deep reinforcement learning for smart home energy management," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2751–2762, 2019.

[54] Y. Gao, J. Yang, M. Yang, and Z. Li, "Deep reinforcement learning based optimal schedule for a battery swapping station considering uncertainties," *IEEE Transactions on Industry Applications*, vol. 56, no. 5, pp. 5775–5784, 2020.

[55] E. Mohagheghi, A. Gabash, and P. Li, "Real-time optimal power flow under wind energy penetration-part i: Approach," in *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*. IEEE, 2016, pp. 1–6.

[56] S. S. Reddy and P. Bijwe, "Day-ahead and real time optimal power flow considering renewable energy resources," *International Journal of Electrical Power & Energy Systems*, vol. 82, pp. 400–408, 2016.

[57] L. Gan and S. H. Low, "An online gradient algorithm for optimal power flow on radial networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 625–638, 2016.

[58] Y. Tang, K. Dvijotham, and S. Low, "Real-time optimal power flow," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2963–2973, 2017.

[59] Y. Zhang, E. Dall'Anese, and M. Hong, "Dynamic admm for real-time optimal power flow," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2017, pp. 1085–1089.

[60] A. Papavasiliou, Y. Mou, L. Cambier, and D. Scieur, "Application of stochastic dual dynamic programming to the real-time dispatch of storage under renewable supply uncertainty," *IEEE Transactions on Sustainable Energy*, vol. 9, no. 2, pp. 547–558, 2017.

[61] A. Di Giorgio, F. Liberati, and A. Lanna, "Real time optimal power flow integrating large scale storage devices and wind generation," in *2015 23rd Mediterranean Conference on Control and Automation (MED)*. IEEE, 2015, pp. 480–486.

[62] R. A. Jabr, S. Karaki, and J. A. Korbane, "Robust multi-period opf with storage and renewables," *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2790–2799, 2014.

[63] K. Utkarsh, D. Srinivasan, A. Trivedi, W. Zhang, and T. Reindl, "Distributed model-predictive real-time optimal operation of a network of smart microgrids," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 2833–2845, 2018.

[64] T. Lu, Z. Wang, Q. Ai, and W.-J. Lee, "Interactive model for energy management of clustered microgrids," *IEEE Transactions on Industry Applications*, vol. 53, no. 3, pp. 1739–1750, 2017.

[65] C. Zhao and X. Li, "A novel real-time energy management strategy for gird-friendly microgrid: Harnessing internal fluctuation internally," *arXiv preprint arXiv:2006.11521*, 2020.

[66] P. Siano, C. Cecati, H. Yu, and J. Kolbusz, "Real time operation of smart grids via fcn networks and optimal power flow," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 4, pp. 944–952, 2012.

[67] Q. Zhang, K. Dehghanpour, Z. Wang, and Q. Huang, "A learning-based power management method for networked microgrids under incomplete information," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1193–1204, 2019.

[68] V.-H. Bui, A. Hussain, and H.-M. Kim, "Double deep q-learning-based distributed operation of battery energy storage system considering uncertainties," *IEEE Transactions on Smart Grid*, vol. 11, no. 1, pp. 457–469, 2019.

[69] Z. Yan and Y. Xu, "Real-time optimal power flow: A lagrangian based deep reinforcement learning approach," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 3270–3273, 2020.

[70] Y. Sun, Z. Li, M. Shahidehpour, and B. Ai, "Battery-based energy storage transportation for enhancing power system economics and security," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2395–2402, 2015.

[71] Y. Sun, J. Zhong, Z. Li, W. Tian, and M. Shahidehpour, "Stochastic scheduling of battery-based energy storage transportation system with the penetration of wind power," *IEEE Transactions on Sustainable Energy*, vol. 8, no. 1, pp. 135–144, 2016.

[72] S.-Y. Kwon, J.-Y. Park, and Y.-J. Kim, "Optimal operation of mobile energy storage devices to minimize energy loss in a distribution system," in *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*. IEEE, 2018, pp. 1–6.

[73] H. H. Abdeltawab and Y. A.-R. I. Mohamed, "Mobile energy storage scheduling and operation in active distribution systems," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 9, pp. 6828–6840, 2017.

[74] S. Lei, C. Chen, H. Zhou, and Y. Hou, "Routing and scheduling of mobile power sources for distribution system resilience enhancement," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5650–5662, 2018.

[75] Z. Yang, P. Dehghanian, and M. Nazemi, "Enhancing seismic resilience of electric power distribution systems with mobile power sources," in *2019 IEEE Industry Applications Society Annual Meeting*. IEEE, 2019, pp. 1–7.

[76] Z. Pan, Z. Qu, Y. Chen, H. Li, and X. Wang, "A distributed assignment method for dynamic traffic assignment using heterogeneous-adviser based multi-agent reinforcement learning," *IEEE Access*, vol. 8, pp. 154 237–154 255, 2020.

[77] T. Qian, C. Shao, X. Li, X. Wang, and M. Shahidehpour, "Enhanced coordinated operations of electric power and transportation networks via ev charging services," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3019–3030, 2020.

[78] D. M. Shilay, K. G. Lorey, T. Weiz, T. Lovetty, and Y. Cheng, "Catching anomalous distributed photovoltaics: An edge-based multi-modal anomaly detection," *arXiv preprint arXiv:1709.08830*, 2017.

[79] K. G. Lore, D. M. Shila, and L. Ren, "Detecting data integrity attacks on correlated solar farms using multi-layer data driven algorithm," in *2018 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2018, pp. 1–9.

[80] Q. Li, F. Li, J. Zhang, J. Ye, W. Song, and A. Mantooth, "Data-driven cyberattack detection for photovoltaic (pv) systems through analyzing micro-pmu data," in *2020 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE, 2020, pp. 431–436.

[81] S. Nath and J. Wu, "Quickest change point detection with multiple postchange models," *Sequential Analysis*, vol. 39, no. 4, pp. 543–562, 2020.

[82] Y. Mo, R. Chabukswar, and B. Sinopoli, "Detecting integrity attacks on scada systems," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1396–1407, 2013.

[83] R. Tunga, C. Murguia, and J. Ruths, "Tuning windowed chi-squared detectors for sensor attacks," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 1752–1757.

[84] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *2009 47th annual Allerton conference on communication, control, and computing (Allerton)*. IEEE, 2009, pp. 911–918.

[85] K. Manandhar, X. Cao, F. Hu, and Y. Liu, "Detection of faults and attacks including false data injection attack in smart grid using kalman filter," *IEEE transactions on control of network systems*, vol. 1, no. 4, pp. 370–379, 2014.

[86] B. Satchidanandan and P. R. Kumar, "Dynamic watermarking: Active defense of networked cyber–physical systems," *Proceedings of the IEEE*, vol. 105, no. 2, pp. 219–240, 2016.

[87] J. Ramos-Ruiz, J. Kim, W.-H. Ko, T. Huang, P. Enjeti, P. Kumar, and L. Xie, "An active detection scheme for cyber attacks on grid-tied pv systems," in *2020 IEEE CyberPELS (CyberPELS)*. IEEE, 2020, pp. 1–6.

[88] P. Hespanhol, M. Porter, R. Vasudevan, and A. Aswani, "Dynamic watermarking for general lti systems," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1834–1839.

[89] M. Porter, P. Hespanhol, A. Aswani, M. Johnson-Roberson, and R. Vasudevan, "Detecting generalized replay attacks via time-varying dynamic watermarking," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3502–3517, 2020.

[90] A. Monticelli, "Electric power system state estimation," *Proceedings of the IEEE*, vol. 88, no. 2, pp. 262–282, 2000.

[91] R. Moslemi, A. Mesbahi, and J. M. Velni, "A fast, decentralized covariance selection-based approach to detect cyber attacks in smart grids," *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 4930–4941, 2017.

[92] I. Akingeneye and J. Wu, "Low latency detection of sparse false data injections in smart grids," *IEEE Access*, vol. 6, pp. 58 564–58 573, 2018.

[93] J. Zhao, A. Gómez-Expósito, M. Netto, L. Mili, A. Abur, V. Terzija, I. Kamwa, B. Pal, A. K. Singh, J. Qi *et al.*, "Power system dynamic state estimation: Motivations, definitions, methodologies, and future work," *IEEE Transactions on Power Systems*, vol. 34, no. 4, pp. 3188–3198, 2019.

[94] M. Khalaf, A. Youssef, and E. El-Saadany, "Detection of false data injection in automatic generation control systems using kalman filter," in *2017 IEEE Electrical Power and Energy Conference (EPEC)*. IEEE, 2017, pp. 1–6.

[95] M. N. Kurt, Y. Yılmaz, and X. Wang, "Distributed quickest detection of cyber-attacks in smart grid," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2015–2030, 2018.

[96] H. Karimipour and V. Dinavahi, "On false data injection attack against dynamic state estimation on smart power grids," in *2017 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*. IEEE, 2017, pp. 388–393.

[97] ——, "Robust massively parallel dynamic state estimation of power systems against cyber-attack," *IEEE Access*, vol. 6, pp. 2984–2995, 2017.

[98] Y. Li, Z. Li, and L. Chen, "Dynamic state estimation of generators under cyber attacks," *IEEE Access*, vol. 7, pp. 125 253–125 267, 2019.

[99] M. A. Hasnat and M. Rahnamay-Naeini, "A data-driven dynamic state estimation for smart grids under dos attack using state correlations," in *2019 North American Power Symposium (NAPS)*. IEEE, 2019, pp. 1–6.

[100] Y. Ding and J. Liu, "Real-time false data injection attack detection in energy internet using online robust principal component analysis," in *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*. IEEE, 2017, pp. 1–6.

[101] Y. He, G. J. Mendis, and J. Wei, "Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2505–2516, 2017.

[102] L. Wei, D. Gao, and C. Luo, "False data injection attacks detection with deep belief networks in smart grid," in *2018 Chinese Automation Congress (CAC)*. IEEE, 2018, pp. 2621–2625.

[103] G. Fenza, M. Gallo, and V. Loia, "Drift-aware methodology for anomaly detection in smart grid," *IEEE Access*, vol. 7, pp. 9645–9657, 2019.

[104] M. Ashrafuzzaman, Y. Chakhchoukh, A. A. Jillepalli, P. T. Tosic, D. C. de Leon, F. T. Sheldon, and B. K. Johnson, "Detecting stealthy false data injection attacks in power grids using deep learning," in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2018, pp. 219–225.

[105] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[106] K. Hamedani, L. Liu, R. Atat, J. Wu, and Y. Yi, "Reservoir computing meets smart grids: Attack detection using delayed feedback networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 734–743, 2017.

[107] K. Hamedani, L. Liu, S. Hu, J. Ashdown, J. Wu, and Y. Yi, "Detecting dynamic attacks in smart grids using reservoir computing: A spiking delayed feedback reservoir based approach," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 3, pp. 253–264, 2019.

[108] J. James, Y. Hou, and V. O. Li, “Online false data injection attack detection with wavelet transform and deep neural networks,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3271–3280, 2018.

[109] M. N. Kurt, O. Ogundijo, C. Li, and X. Wang, “Online cyber-attack detection in smart grid: A reinforcement learning approach,” *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5174–5185, 2018.

[110] Y. Chen, S. Huang, F. Liu, Z. Wang, and X. Sun, “Evaluation of reinforcement learning-based false data injection attack to automatic voltage control,” *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 2158–2169, 2018.

[111] D. An, Q. Yang, W. Liu, and Y. Zhang, “Defending against data integrity attacks in smart grid: A deep reinforcement learning-based approach,” *IEEE Access*, vol. 7, pp. 110 835–110 845, 2019.

[112] A. S. Willsky, “A survey of design methods for failure detection in dynamic systems,” *Automatica*, vol. 12, no. 6, pp. 601–611, 1976.

[113] F. Gustafsson and F. Gustafsson, *Adaptive filtering and change detection.* Citeseer, 2000, vol. 1.

[114] M. Basseville, “Detecting changes in signals and systems—a survey,” *Automatica*, vol. 24, no. 3, pp. 309–326, 1988.

[115] S. Nath and J. Wu, “Bayesian quickest change point detection with multiple candidates of post-change models,” in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP).* IEEE, 2018, pp. 51–55.

[116] C. Murguia and J. Ruths, “Cusum and chi-squared attack detection of compromised sensors,” in *2016 IEEE Conference on Control Applications (CCA).* IEEE, 2016, pp. 474–480.

[117] C. Truong, M. Naumann, R. Karl, M. Müller, A. Jossen, and H. Hesse, “Economics of Residential Photovoltaic Battery Systems in Germany: The Case of Tesla’s Powerwall,” *Batteries*, vol. 2, no. 2, p. 14, May 2016.

[118] A. Berrueta, J. Pascual, I. S. Martın, P. Sanchis, and A. Ursua, “Influence of the Aging Model of Lithium-Ion Batteries on the Management of PV Self-Consumption Systems,” p. 5.

[119] I. Quesada and I. Grossmann, “An LP/NLP based branch and bound algorithm for convex MINLP optimization problems,” *Computers & Chemical Engineering*, vol. 16, no. 10-11, pp. 937–947, Oct. 1992.

[120] R. Ahuja, “Minimax linear programming problem,” *Operations Research Letters*, vol. 4, no. 3, pp. 131–134, Oct. 1985.

[121] A. Gupta, R. Jain, and P. W. Glynn, "An empirical algorithm for relative value iteration for average-cost mdps," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 5079–5084.

[122] V. P. Pribylov and A. I. Plyasunov, "A convolutional code decoder design using viterbi algorithm with register exchange history unit," in *2005 Siberian Conference on Control and Communications*. IEEE, 2005, pp. 13–18.

[123] R. Bellman, "On a routing problem," *Quarterly of applied mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[124] P. K. Kitanidis and E. Foufoula-Georgiou, "Error analysis of conventional discrete and gradient dynamic programming," *Water Resources Research*, vol. 23, no. 5, pp. 845–858, May 1987.

[125] K. Ponnambalam and B. J. Adams, "Comment on "Error analysis of conventional discrete and gradient dynamic programming" by P. K. Kitanidis and Efi Foufoula-Georgiou," *Water Resources Research*, vol. 24, no. 6, pp. 888–889, Jun. 1988.

[126] J. Stachurski, "Continuous State Dynamic Programming via Nonexpansive Approximation," *Computational Economics*, vol. 31, no. 2, pp. 141–160, Mar. 2008.

[127] A. Heydari, "Theoretical and Numerical Analysis of Approximate Dynamic Programming with Approximation Errors," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 2, pp. 301–311, Feb. 2016.

[128] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[129] "Commercial Reference Buildings | Department of Energy." [Online]. Available: https://www.energy.gov/eere/buildings/commercial-reference-buildings

[130] "PVWatts Calculator." [Online]. Available: https://pvwatts.nrel.gov/pvwatts.php

[131] "PG&E - Electric Schedule E-19: Medium general demand-metered tou service." [Online]. Available: https://www.scribd.com/document/253380587/Pg-e-Electric-Schedule-E-19

[132] "Tesla Powerwall." [Online]. Available: https://www.tesla.com/powerwall

[133] C. A. F. Fernandes, J. P. N. Torres, M. Morgado, and J. A. Morgado, "Aging of solar PV plants and mitigation of their consequences," in *2016 IEEE International Power Electronics and Motion Control Conference (PEMC)*. Varna, Bulgaria: IEEE, Sep. 2016, pp. 1240–1247.

[134] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *arXiv preprint arXiv:1801.08757*, 2018.

[135] S. Nath and J. Wu, "Online battery scheduling for grid-connected photo-voltaic systems," *Journal of Energy Storage*, vol. 31, p. 101713, 2020.

[136] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[137] D. K. Maly and K.-S. Kwan, "Optimal battery energy storage system (bess) charge scheduling with dynamic programming," *IEE Proceedings-Science, Measurement and Technology*, vol. 142, no. 6, pp. 453–458, 1995.

[138] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.

[139] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," *arXiv preprint arXiv:1706.01905*, 2017.

[140] "Solrenview fayetteville public library," https://www.solrenview.com/cgi-bin/cgihandler.cgi?view=0,2,0,0&cond=site_ID=316.

[141] "Tesla powerwall 2 datasheet - north america," https://www.tesla.com/sites/default/files/pdfs/powerwall/Powerwall%202_AC_Datasheet_en_northamerica.pdf.

[142] "Pge - electric schedule e-19: Medium general demand-metered tou service," https://www.pge.com/tariffs/assets/pdf/tariffbook/ELEC_SCHEDS_E-19.pdf.

[143] Y. Li and J. Wu, "Optimum integration of solar energy with battery energy storage systems," *IEEE Transactions on Engineering Management*, 2020.

[144] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[145] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[146] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[147] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2010.

[148] L. Thurner, A. Scheidler, F. Schäfer, J.-H. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, "pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510–6521, 2018.

[149] V. Nadolski, "Automated surface observing system (asos) user's guide," *National Oceanic and Atmospheric Administration, Department of Defense, Federal Aviation Administration, United States Navy*, vol. 20, 1998.

[150] U. B. of Public Roads. Office of Planning. Urban Planning Division, *Traffic Assignment Manual for Application with a Large, High Speed Computer.* US Department of Commerce, 1964.

[151] J. G. Wardrop, "Road paper. some theoretical aspects of road traffic research." *Proceedings of the institution of civil engineers*, vol. 1, no. 3, pp. 325–362, 1952.

[152] M. S. Daskin, "Urban transportation networks: Equilibrium analysis with mathematical programming methods," 1985.

[153] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[154] L. Thurner, A. Scheidler, F. Schäfer, J. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, "pandapower — an open-source python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 6510–6521, Nov 2018.

[155] T. Cabannes, "Transportationnetworks," https://github.com/bstabler/TransportationNetworks/tree/master/SiouxFalls, 2019.

[156] "Dataworks: City of sioux falls gis." [Online]. Available: https://dataworks.siouxfalls.org/datasets/b169d6ec2d8947f9a18f8a7977cf8d28/explore?location=43.551843%2C-96.699982%2C11.93

[157] "National renewable energy laboratory: Pvwatts calculator." [Online]. Available: https://pvwatts.nrel.gov/

[158] "Us department of commerce, noaa: Automated surface observing system (asos) dataset." [Online]. Available: https://www.weather.gov/asos/

[159] "Sioux falls residential load." [Online]. Available: https://openei.org/datasets/files/961/pub/EPLUS_TMY2_RESIDENTIAL_BASE/USA_SD_Sioux.Falls.726510_TMY2.csv

[160] "Sioux falls foss field commercial load." [Online]. Available: https://openei.org/datasets/files/961/pub/COMMERCIAL_LOAD_DATA_E_PLUS_OUTPUT/USA_SD_Sioux.Falls-Foss.Field.726510_TMY3/

[161] F. Braeuer, "Load profile data of 50 industrial plants in Germany for one year," Jun. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3899018

[162] M. Izbicki, S. Amini, C. R. Shelton, and H. Mohsenian-Rad, "Identification of destabilizing attacks in power systems," in *2017 American Control Conference (ACC).* IEEE, 2017, pp. 3424–3429.

[163] J. Tang, J. Song, and A. Gupta, “A dynamic watermarking algorithm for finite markov decision problems,” *arXiv preprint arXiv:2111.04952*, 2021.

[164] E. S. Gardner Jr and E. McKenzie, “Forecasting trends in time series,” *Management Science*, vol. 31, no. 10, pp. 1237–1246, 1985.

[165] A. L. Da Silva, M. Do Coutto Filho, and J. De Queiroz, “State forecasting in electric power systems,” in *IEE Proceedings C (Generation, Transmission and Distribution)*, vol. 130, no. 5. IET, 1983, pp. 237–244.

[166] K. Nishiya, J. Hasegawa, and T. Koike, “Dynamic state estimation including anomaly detection and identification for power systems,” in *IEE Proceedings C (Generation, Transmission and Distribution)*, vol. 129, no. 5. IET, 1982, pp. 192–198.

[167] A. De Maio, “Rao test for adaptive detection in gaussian interference with unknown covariance matrix,” *IEEE transactions on signal processing*, vol. 55, no. 7, pp. 3577–3584, 2007.

[168] J. H. Chow and K. W. Cheung, “A toolbox for power system dynamics and control engineering education and research,” *IEEE transactions on Power Systems*, vol. 7, no. 4, pp. 1559–1564, 1992.

[169] H. Karimipour and V. Dinavahi, “Extended kalman filter-based parallel dynamic state estimation,” *IEEE transactions on smart grid*, vol. 6, no. 3, pp. 1539–1549, 2015.

[170] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning.* PMLR, 2016, pp. 1928–1937.

[171] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[172] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *International conference on machine learning.* PMLR, 2018, pp. 1587–1596.

A Bellman equation (4.13) derivation

Based on R_t in (5.36), the discounted reward function R_t can be written in a recursive manner as $R_t = r(s_t, a_t) + \gamma R_{t+1}$. Substitute this result into (4.12) and denoting expectations with respect to $r_t, s_t \sim E, a_t \sim \pi$ as $\mathbb{E}_{r_t,s_t,a_t}$ yields

$$\begin{aligned} Q^\pi(s_t, a_t) = \mathbb{E}_{r_{i\geq t}, s_{i>t}, a_{i>t}} \left[r(s_t, a_t)|s_t, a_t\right] \\ + \mathbb{E}_{r_{i\geq t}, s_{i>t}, a_{i>t}} \left[\gamma R_{t+1}|s_t, a_t\right] \end{aligned} \tag{A.1}$$

The first term is independent of $a_{i>t}$, so:

$$\begin{aligned} & \mathbb{E}_{r_{i\geq t}, s_{i>t}, a_{i>t}} \left[r(s_t, a_t)|s_t, a_t\right] \\ = \; & \mathbb{E}_{r_{i\geq t}, s_{i>t}} \left[r(s_t, a_t)|s_t, a_t\right] \\ = \; & \mathbb{E}_{r_t, s_{t+1}} \left[\mathbb{E}_{r_{i\geq t+1}, s_{i>t+1}} \left[r(s_t, a_t)|s_t, a_t, r_t, s_{t+1}\right]\right] \\ = \; & \mathbb{E}_{r_t, s_{t+1}} \left[r(s_t, a_t)\right] \end{aligned} \tag{A.2}$$

Similarly, the second term is independent of s_t and a_t:

$$\begin{aligned} & \mathbb{E}_{r_{i\geq t}, s_{i>t}, a_{i>t}} \left[\gamma R_{t+1}|s_t, a_t\right] \\ = \; & \mathbb{E}_{r_{i\geq t}, s_{i>t}, a_{i>t}} \left[\gamma R_{t+1}\right] \\ = \; & \mathbb{E}_{r_t, s_{t+1}, a_{t+1}} \left[\mathbb{E}_{r_{i>t}, s_{i>t+1}, a_{i>t+1}} \left[\gamma R_{t+1}|r_t, s_{t+1}, a_{t+1}\right]\right] \\ = \; & \mathbb{E}_{r_t, s_{t+1}, a_{t+1}} \left[\mathbb{E}_{r_{i\geq t+1}, s_{i>t+1}, a_{i>t+1}} \left[\gamma R_{t+1}|s_{t+1}, a_{t+1}\right]\right] \\ = \; & \mathbb{E}_{r_t, s_{t+1}, a_{t+1}} \left[\gamma Q^\pi(s_{t+1}, a_{t+1})\right] \\ = \; & \mathbb{E}_{r_t, s_{t+1}} \left[\gamma \mathbb{E}_{a_{t+1}} \left[Q^\pi(s_{t+1}, a_{t+1})\right]\right] \end{aligned} \tag{A.3}$$

Combine (A.2) and (A.3) results in the recursive equation of the action-value function in (4.13).

B Kalman filter equations derivation

B.1 Covariance Update Equation Derivation

Plug the measurement equation (7.15) into the state update equation (7.17):

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}(\mathbf{C}\mathbf{x}_{k+1} + \mathbf{n}_{k+1} - \mathbf{C}\hat{\mathbf{x}}_{k+1|k}) \tag{B.1}$$

so the estimation error:

$$\begin{aligned} &\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k+1} \\ &= \mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k} - \mathbf{K}_{k+1}(\mathbf{C}\mathbf{x}_{k+1} + \mathbf{n}_{k+1} - \mathbf{C}\hat{\mathbf{x}}_{k+1|k}) \\ &= \mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k} - \mathbf{K}_{k+1}\mathbf{C}\mathbf{x}_{k+1} - \mathbf{K}_{k+1}\mathbf{n}_{k+1} + \mathbf{K}_{k+1}\mathbf{C}\hat{\mathbf{x}}_{k+1|k} \end{aligned} \tag{B.2}$$

the estimation covariance:

$$\begin{aligned} \mathbf{P}_{k+1|k+1} &= \mathbb{E}\left[(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k+1})(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k+1})^T\right] \\ &= \mathbb{E}\Big[\big((\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k}) - \mathbf{K}_{k+1}\mathbf{n}_{k+1}\big)\times \\ &\qquad \big((\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k}) - \mathbf{K}_{k+1}\mathbf{n}_{k+1}\big)^T\Big] \\ &= \mathbb{E}\Big[\big((\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k}) - \mathbf{K}_{k+1}\mathbf{n}_{k+1}\big)\times \\ &\qquad \big((\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})^T(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T - (\mathbf{K}_{k+1}\mathbf{n}_{k+1})^T\big)\Big] \\ &= \mathbb{E}\Big[(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k}) \times (\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})^T(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T \\ &\qquad - (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})(\mathbf{K}_{k+1}\mathbf{n}_{k+1})^T \\ &\qquad - \mathbf{K}_{k+1}\mathbf{n}_{k+1}(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})^T(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T \\ &\qquad + \mathbf{K}_{k+1}\mathbf{n}_{k+1}(\mathbf{K}_{k+1}\mathbf{n}_{k+1})^T\Big] \\ &= \mathbb{E}\left((\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k}) \times (\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})^T(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T\right) \\ &\qquad - \mathbb{E}\left((\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})(\mathbf{K}_{k+1}\mathbf{n}_{k+1})^T\right) \\ &\qquad - \mathbb{E}\left(\mathbf{K}_{k+1}\mathbf{n}_{k+1}(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})^T(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T\right) \\ &\qquad + \mathbb{E}\left(\mathbf{K}_{k+1}\mathbf{n}_{k+1}(\mathbf{K}_{k+1}\mathbf{n}_{k+1})^T\right) \end{aligned} \tag{B.3}$$

where

$$\mathbb{E}\left((\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})(\mathbf{K}_{k+1}\mathbf{n}_{k+1})^T\right) = 0$$
$$\mathbb{E}\left(\mathbf{K}_{k+1}\mathbf{n}_{k+1}(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})^T(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T\right) = 0$$

because the error of the prior estimation $(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})$ is uncorrelated with the post measurement noise $\mathbf{n}_{k+1}$. Then the covariance is:

$$\begin{aligned}\mathbf{P}_{k+1|k+1} &= \mathbb{E}\left((\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k}) \times (\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})^T(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T\right)\\ &\qquad + \mathbb{E}(\mathbf{K}_{k+1}\mathbf{n}_{k+1}\mathbf{n}_{k+1}^T\mathbf{K}_{k+1}^T)\\ &= (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})\mathbb{E}\left((\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k}) \times (\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1|k})^T\right)(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T\\ &\qquad + \mathbf{K}_{k+1}\mathbb{E}(\mathbf{n}_{k+1}\mathbf{n}_{k+1}^T)\mathbf{K}_{k+1}^T\\ &= (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})\mathbf{P}_{k+1|k}(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T + \mathbf{K}_{k+1}\boldsymbol{\Sigma}_{\mathbf{n}}\mathbf{K}_{k+1}^T\end{aligned} \tag{B.4}$$

B.2 Kalman Gain Equation Derivation

First, the covariance update equation should be rearranged:

$$\begin{aligned}\mathbf{P}_{k+1|k+1} &= (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})\mathbf{P}_{k+1|k}(\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})^T + \mathbf{K}_{k+1}\boldsymbol{\Sigma}_{\mathbf{n}}\mathbf{K}_{k+1}^T\\ &= (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C})\mathbf{P}_{k+1|k}(\mathbf{I} - \mathbf{C}^T\mathbf{K}_{k+1}^T) + \mathbf{K}_{k+1}\boldsymbol{\Sigma}_{\mathbf{n}}\mathbf{K}_{k+1}^T\\ &= (\mathbf{P}_{k+1|k} - \mathbf{K}_{k+1}\mathbf{C}\mathbf{P}_{k+1|k})(\mathbf{I} - \mathbf{C}^T\mathbf{K}_{k+1}^T) + \mathbf{K}_{k+1}\boldsymbol{\Sigma}_{\mathbf{n}}\mathbf{K}_{k+1}^T\\ &= \mathbf{P}_{k+1|k} - \mathbf{P}_{k+1|k}\mathbf{C}^T\mathbf{K}_{k+1}^T - \mathbf{K}_{k+1}\mathbf{C}\mathbf{P}_{k+1|k} + \mathbf{K}_{k+1}\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T\mathbf{K}_{k+1}^T\\ &\qquad + \mathbf{K}_{k+1}\boldsymbol{\Sigma}_{\mathbf{n}}\mathbf{K}_{k+1}^T\\ &= \mathbf{P}_{k+1|k} - \mathbf{P}_{k+1|k}\mathbf{C}^T\mathbf{K}_{k+1}^T - \mathbf{K}_{k+1}\mathbf{C}\mathbf{P}_{k+1|k} + \mathbf{K}_{k+1}(\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T + \boldsymbol{\Sigma}_{\mathbf{n}})\mathbf{K}_{k+1}^T\end{aligned} \tag{B.5}$$

The Kalman Filter is an optimal filter, which means the Kalman Gain matrix needs to minimize the estimation variance. This is equivalent to minimize the race of the covariance matrix $\mathrm{Tr}(\mathbf{P}_{k+1|k+1})$:

$$\begin{aligned}\mathrm{Tr}(\mathbf{P}_{k+1|k+1}) &= \mathrm{Tr}(\mathbf{P}_{k+1|k}) - \mathrm{Tr}(\mathbf{P}_{k+1|k}\mathbf{C}^T\mathbf{K}_{k+1}^T) - \mathrm{Tr}(\mathbf{K}_{k+1}\mathbf{C}\mathbf{P}_{k+1|k})\\ &\qquad + \mathrm{Tr}(\mathbf{K}_{k+1}(\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T + \boldsymbol{\Sigma}_{\mathbf{n}})\mathbf{K}_{k+1}^T)\\ &= \mathrm{Tr}(\mathbf{P}_{k+1|k}) - 2\,\mathrm{Tr}(\mathbf{K}_{k+1}\mathbf{C}\mathbf{P}_{k+1|k})\\ &\qquad + \mathrm{Tr}(\mathbf{K}_{k+1}(\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T + \boldsymbol{\Sigma}_{\mathbf{n}})\mathbf{K}_{k+1}^T)\end{aligned} \tag{B.6}$$

To minimize the trace, we differentiate it with respect to $\mathbf{K}_{k+1}$ and set the result to zero:

$$\begin{aligned}\frac{d\left(\mathrm{Tr}(\mathbf{P}_{k+1|k+1})\right)}{d\mathbf{K}_{k+1}} &= \frac{d\left(\mathrm{Tr}(\mathbf{P}_{k+1|k})\right)}{d\mathbf{K}_{k+1}} - \frac{d\left(2\,\mathrm{Tr}(\mathbf{K}_{k+1}\mathbf{C}\mathbf{P}_{k+1|k})\right)}{d\mathbf{K}_{k+1}}\\ &\qquad + \frac{d\left(\mathrm{Tr}(\mathbf{K}_{k+1}(\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T + \boldsymbol{\Sigma}_{\mathbf{n}})\mathbf{K}_{k+1}^T)\right)}{d\mathbf{K}_{k+1}}\\ &= 0 - 2(\mathbf{C}\mathbf{P}_{k+1|k})^T + 2\mathbf{K}_{k+1}(\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T + \boldsymbol{\Sigma}_{\mathbf{n}}) = 0\end{aligned} \tag{B.7}$$

where

$$\frac{d\left(\mathrm{Tr}(\mathbf{AB})\right)}{d\mathbf{A}} = \mathbf{B}^T$$
$$\frac{d\left(\mathrm{Tr}(\mathbf{ABA}^T)\right)}{d\mathbf{A}} = 2\mathbf{AB}$$

are used in the final step. So we get:

$$\begin{aligned}(\mathbf{CP}_{k+1|k})^T &= \mathbf{K}_{k+1}(\mathbf{CP}_{k+1|k}\mathbf{C}^T + \mathbf{\Sigma_n}) \\ \mathbf{K}_{k+1} &= (\mathbf{CP}_{k+1|k})^T(\mathbf{CP}_{k+1|k}\mathbf{C}^T + \mathbf{\Sigma_n})^{-1} \\ &= \mathbf{P}_{k+1|k}\mathbf{C}^T(\mathbf{CP}_{k+1|k}\mathbf{C}^T + \mathbf{\Sigma_n})^{-1}\end{aligned} \tag{B.8}$$

C Post-attack distribution derivation

C.1 Proof of equation (7.42) and (7.43)

Under the FDI attack, the measurements are replaced by:

$$\mathbf{z}[k+1] = \mathbf{y}[k+1] + \mathbf{a}[k+1] \tag{C.1}$$

The posterior state estimation is:

$$\begin{aligned} &\hat{\mathbf{x}}_{k+1|k+1} \\ &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}(\mathbf{z}[k+1] - \mathbf{C}\hat{\mathbf{x}}_{k+1|k}) \\ &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}(\mathbf{y}[k+1] - C\hat{\mathbf{x}}_{k+1|k} + \mathbf{a}[k+1]) \end{aligned} \tag{C.2}$$

The test statistic $\mathbf{g}[k+1]$ is:

$$\begin{aligned} \mathbf{g}[k+1] &= \hat{\mathbf{x}}_{k+1|k+1} - \hat{\mathbf{x}}_{k+1|k} \\ &= \mathbf{K}_{k+1}(\boldsymbol{\upsilon}[k+1] + \mathbf{a}[k+1]) \end{aligned} \tag{C.3}$$

Then the whitened statistic $\bar{\mathbf{g}}[k+1]$ is

$$\bar{\mathbf{g}}[k+1] = \bar{\mathbf{U}}^H\mathbf{g}[k+1] = \bar{\mathbf{U}}^H\mathbf{K}_{k+1}(\boldsymbol{\upsilon}[k+1] + \mathbf{a}[k+1]) \tag{C.4}$$

For deterministic $\mathbf{a}[k+1]$, the mean of $\bar{\mathbf{g}}[k+1]$:

$$\mathbb{E}[\bar{\mathbf{g}}[k+1]] = \bar{\mathbf{U}}^H\mathbf{K}\mathbb{E}[\boldsymbol{\upsilon}[k+1] + \mathbf{a}[k+1]] = \bar{\mathbf{U}}^H\mathbf{K}\mathbf{a}[k+1] \tag{C.5}$$

The covariance of $\bar{\mathbf{g}}[k+1]$ is $\mathbf{D}$ since the mean is deterministic:

$$\mathrm{Cov}[\bar{\mathbf{g}}[k+1]] = \mathbf{D} \tag{C.6}$$

The covariance of $\bar{\mathbf{g}}[k+1]$ and $\mathbf{e}[k]$ is:

$$\begin{aligned} \mathbb{E}[\bar{\mathbf{g}}[k+1]\mathbf{e}[k]^T] &= \mathbb{E}[\bar{\mathbf{U}}^H\mathbf{K}_{k+1}(\boldsymbol{\upsilon}[k+1] + \mathbf{a}[k+1])\mathbf{e}[k]^T] \\ &= \bar{\mathbf{U}}^H\mathbf{K}\mathbb{E}[(\boldsymbol{\upsilon}[k+1] + \mathbf{a}[k+1])\mathbf{e}[k]^T] \\ &= \bar{\mathbf{U}}^H\mathbf{K}\mathbb{E}[(\boldsymbol{\upsilon}[k+1] + \mathbf{a}[k+1])]\mathbb{E}[\mathbf{e}[k]^T] \\ &= \mathbf{0}_{q\times m} \end{aligned} \tag{C.7}$$

Then the mean of $\mathbf{r}[k+1]$ is:

$$\mu = \begin{bmatrix} \mathbb{E}[\bar{\mathbf{g}}[k+1]] \\ \mathbb{E}[\mathbf{e}[k]] \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{U}}^H \mathbf{K}\mathbf{a}[k+1] \\ \mathbf{0}_m \end{bmatrix} \tag{C.8}$$

and the covariance of $\mathbf{r}[k+1]$ is:

$$\Sigma = \begin{bmatrix} \mathbf{D} & \mathbf{0}_{q\times m} \\ \mathbf{0}_{m\times q} & \boldsymbol{\Sigma}_{\mathbf{e}} \end{bmatrix} = \boldsymbol{\Sigma}_{\mathbf{0}} \tag{C.9}$$

which completes the proof of (7.42).

For $\mathbf{a}[k+1] \sim \mathcal{N}_p(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{a}})$, the mean of $\bar{\mathbf{g}}[k+1]$ is:

$$\begin{aligned} \mathbb{E}[\bar{\mathbf{g}}[k+1]] &= \bar{\mathbf{U}}^H \mathbf{K}\mathbb{E}[\boldsymbol{v}[k+1] + \mathbf{a}[k+1]] \\ &= \bar{\mathbf{U}}^H \mathbf{K}(\mathbb{E}[\boldsymbol{v}[k+1] + \mathbb{E}[\mathbf{a}[k+1]]) = \mathbf{0}_q \end{aligned} \tag{C.10}$$

The covariance of $\bar{\mathbf{g}}[k+1]$ is:

$$\begin{aligned} &\mathrm{Cov}[\bar{\mathbf{g}}[k+1]] = \mathbb{E}[\bar{\mathbf{g}}[k+1]\bar{\mathbf{g}}[k+1]^T] \\ &= \bar{\mathbf{U}}^H \mathbf{K}\mathbb{E}[(\boldsymbol{v}[k+1] + \mathbf{a}[k+1])(\boldsymbol{v}[k+1] + \mathbf{a}[k+1])^T]\mathbf{K}^T\bar{\mathbf{U}} \\ &= \bar{\mathbf{U}}^H \mathbf{K}\mathbb{E}[\boldsymbol{v}[k+1]\boldsymbol{v}[k+1]^T + \mathbf{a}[k+1]\mathbf{a}[k+1]^T]\mathbf{K}^T\bar{\mathbf{U}} \\ &= \mathbf{D} + \bar{\mathbf{U}}^H \mathbf{K}\boldsymbol{\Sigma}_{\mathbf{a}}\mathbf{K}^T\bar{\mathbf{U}} \end{aligned} \tag{C.11}$$

using the fact that innovation $\boldsymbol{v}[k+1]$ and $\mathbf{a}[k+1]$ are independent. The covariance of $\bar{\mathbf{g}}[k+1]$ and $\mathbf{e}[k]$ has the same form as deterministic case, then the mean and covariance of $\mathbf{r}[k+1]$ is:

$$\mu = \mathbf{0}_{q+m} \tag{C.12a}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathbf{D} + \bar{\mathbf{U}}^H \mathbf{K}\boldsymbol{\Sigma}_{\mathbf{a}}\mathbf{K}^T\bar{\mathbf{U}} & \mathbf{0}_{q\times m} \\ \mathbf{0}_{m\times q} & \boldsymbol{\Sigma}_{\mathbf{e}} \end{bmatrix} \tag{C.12b}$$

which completes the proof of (7.43).

C.2 Proof of equation (7.45) and (7.46)

Under the replay attack, the measurements are replaced by:

$$\mathbf{z}[k+1] = \mathbf{y}[k+1-l] \tag{C.13}$$

The innovation after attack is:

$$\begin{aligned}\boldsymbol{\upsilon}[k+1] &= \mathbf{z}[k+1] - \mathbf{C}\hat{\mathbf{x}}_{k+1|k} \\ &= \mathbf{y}[k+1-l] - \mathbf{C}\hat{\mathbf{x}}_{k+1|k} \\ &= \mathbf{C}(\mathbf{x}[k+1-l] - \hat{\mathbf{x}}_{k+1|k}) + \mathbf{n}[k+1-l] \\ &= \mathbf{C}(\mathbf{x}[k+1-l] - \mathbf{A_d}\hat{\mathbf{x}}_{k|k} - \mathbf{B}\mathbf{h}(\mathbf{z}^k)) \\ &\qquad + \mathbf{n}[k+1-l] - \mathbf{C}\mathbf{B}\mathbf{e}[k]\end{aligned} \tag{C.14}$$

Suppose we have a virtual system that satisfied the following system equations:

$$\mathbf{x}^{'}[t+1] = \mathbf{A_d}\mathbf{x}^{'}[t] + \mathbf{B}\mathbf{h}(\mathbf{z}^{'t}) + \mathbf{B}\mathbf{e}^{'}[t] + \mathbf{w}^{'}[t+1], \tag{C.15a}$$

$$\mathbf{y}^{'}[t+1] = \mathbf{C}\mathbf{x}^{'}[t+1] + \mathbf{n}^{'}[t+1]. \tag{C.15b}$$

and the Kalman filter update at $k+1$:

$$\hat{\mathbf{x}}^{'}_{k+1|k} = \mathbf{A_d}\hat{\mathbf{x}}^{'}_{k|k} + \mathbf{B}\mathbf{h}(\mathbf{z}^{'k}) + \mathbf{B}\mathbf{e}^{'}[k] \tag{C.16}$$

$$\hat{\mathbf{x}}^{'}_{k+1|k+1} = \hat{\mathbf{x}}^{'}_{k+1|k} + \mathbf{K}_{k+1}\boldsymbol{\upsilon}^{'}[k+1] \tag{C.17}$$

and also satisfies the linear approximation of control policy:

$$\mathbf{h}(\mathbf{z}^{'k}) = \mathbf{L}\hat{\mathbf{x}}^{'}_{k|k} \tag{C.18}$$

with the initial state $\mathbf{x}^{'}[0]$ and initial prior state estimation $\hat{\mathbf{x}}^{'}_{1|0}$. In addition, the virtual system is a delayed version of the real system without any attack, which satisfies:

$$\mathbf{x}^{'}[k+1] = \mathbf{x}[k+1-l] \tag{C.19}$$

$$\hat{\mathbf{x}}^{'}_{k|k} = \hat{\mathbf{x}}_{k-l|k-l} \tag{C.20}$$

when $k \geq l$. Then the replay attack can be regarded as replacing $\mathbf{y}[k]$ with $\mathbf{y}^{'}[k]$ starting from τ.

Define the estimated control and transition matrix:

$$\mathbf{u}[k] := \mathbf{L}\hat{\mathbf{x}}_{k|k} + \mathbf{e}[k] \tag{C.21}$$

$$\mathbf{A_e} := (\mathbf{A_d} + \mathbf{B}\mathbf{L})(\mathbf{I}_p - \mathbf{K}\mathbf{C}) \tag{C.22}$$

Assume $\mathbf{A_e}$ is stable, otherwise the measurements will soon be unbounded and the attack can be detected as a destabilization attack.

The Kalman filter estimation after the system is attacked and becomes stable can be rewritten as:

$$\begin{aligned} & \hat{\mathbf{x}}_{k+1|k} = \mathbf{A_d}\hat{\mathbf{x}}_{k|k} + \mathbf{Bu}[k] \\ &= (\mathbf{A_d} + \mathbf{BL})\hat{\mathbf{x}}_{k|k} + \mathbf{Be}[k] \\ &= (\mathbf{A_d} + \mathbf{BL})(\hat{\mathbf{x}}_{k|k-1} + \mathbf{K}(\mathbf{y}^{'}[k] - \mathbf{C}\hat{\mathbf{x}}_{k|k-1}) + \mathbf{Be}[k] \\ &= \mathbf{A_e}\hat{\mathbf{x}}_{k|k-1} + (\mathbf{A_d} + \mathbf{BL})\mathbf{K}\mathbf{y}^{'}[k] + \mathbf{Be}[k] \end{aligned} \tag{C.23}$$

This update also holds true for the virtual system that:

$$\hat{\mathbf{x}}^{'}_{k+1|k} = \mathbf{A_e}\hat{\mathbf{x}}^{'}_{k|k-1} + (\mathbf{A_d} + \mathbf{BL})\mathbf{K}\mathbf{y}^{'}[k] + \mathbf{Be}^{'}[k] \tag{C.24}$$

Therefore, we consider the difference between the prior estimation of the two systems at $k+1$:

$$\begin{aligned} & \hat{\mathbf{x}}^{'}_{k+1|k} - \hat{\mathbf{x}}_{k+1|k} \\ &= \mathbf{A_e}(\hat{\mathbf{x}}^{'}_{k|k-1} - \hat{\mathbf{x}}_{k|k-1}) + \mathbf{B}(\mathbf{e}^{'}[k] - \mathbf{e}[k]) \\ &= \mathbf{A_e}^2(\hat{\mathbf{x}}^{'}_{k-1|k-2} - \hat{\mathbf{x}}_{k-1|k-2}) \\ &\quad + \mathbf{A_e}\mathbf{B}(\mathbf{e}^{'}[k-1] - \mathbf{e}[k-1]) + \mathbf{B}(\mathbf{e}^{'}[k] - \mathbf{e}[k])) \\ &= \cdots \\ &= \mathbf{A_e}^k(\hat{\mathbf{x}}^{'}_{1|0} - \hat{\mathbf{x}}_{1|0}) + \sum_{i=1}^{k} \mathbf{A_e}^{k-i}\mathbf{B}(\mathbf{e}^{'}[i] - \mathbf{e}[i]) \end{aligned} \tag{C.25}$$

The limit mean of $\bar{\mathbf{g}}[k+1]$ is:

$$\begin{aligned} & \lim_{k\to\infty} \mathbb{E}[\bar{\mathbf{g}}[k+1]] = \bar{\mathbf{U}}^H\mathbf{K}\mathbb{E}[\boldsymbol{\upsilon}[k+1]] \\ &= \lim_{k\to\infty} \bar{\mathbf{U}}^H\mathbf{K}\mathbb{E}[\mathbf{y}^{'}[k+1] - \mathbf{C}\hat{\mathbf{x}}^{'}_{k+1|k} + \mathbf{C}(\hat{\mathbf{x}}^{'}_{k+1|k} \\ &\quad - \hat{\mathbf{x}}_{k+1|k})] \\ &= \lim_{k\to\infty} \bar{\mathbf{U}}^H\mathbf{K}\mathbb{E}[\boldsymbol{\upsilon}^{'}[k+1]] + \bar{\mathbf{U}}^H\mathbf{KC}\mathbb{E}[\mathbf{A_e}^k(\hat{\mathbf{x}}^{'}_{1|0} - \hat{\mathbf{x}}_{1|0}) \\ &\quad + \sum_{i=1}^{k} \mathbf{A_e}^{k-i}\mathbf{B}(\mathbf{e}^{'}[i] - \mathbf{e}[i])] \\ &= \lim_{k\to\infty} \bar{\mathbf{U}}^H\mathbf{K}\mathbb{E}[\boldsymbol{\upsilon}^{'}[k+1]] + \bar{\mathbf{U}}^H\mathbf{KC}\mathbb{E}[\mathbf{A_e}^k(\hat{\mathbf{x}}^{'}_{1|0} - \hat{\mathbf{x}}_{1|0})] \\ &\quad + \sum_{i=1}^{k} \bar{\mathbf{U}}^H\mathbf{KC}\mathbb{E}[\mathbf{A_e}^{k-i}\mathbf{B}(\mathbf{e}^{'}[i] - \mathbf{e}[i])] \\ &= \mathbf{0}_q \end{aligned} \tag{C.26}$$

where the first term is the innovation of the virtual system, which has zero mean. The second term will converge to zero because $\mathbf{A_e}$ is stable. The third term is zero because the watermark has zero mean.

The limit covariance of $\bar{\mathbf{g}}[k+1]$ is:

$$\begin{aligned}
&\lim_{k\to\infty} \mathrm{Cov}[\bar{\mathbf{g}}[k+1]] \\
&= \lim_{k\to\infty} \bar{\mathbf{U}}^H \mathbf{K} \mathrm{Cov}[\mathbf{y}'[k+1] - \mathbf{C}\hat{\mathbf{x}}'_{k+1|k} + \mathbf{C}(\hat{\mathbf{x}}'_{k+1|k} \\
&\quad - \hat{\mathbf{x}}_{k+1|k})]\mathbf{K}^T\bar{\mathbf{U}} \\
&= \lim_{k\to\infty} \bar{\mathbf{U}}^H \mathbf{K}(\mathrm{Cov}[\boldsymbol{v}'[k+1]] + \sum_{i=0}^{k} \mathrm{Cov}[\mathbf{C}\mathbf{A_e}^i\mathbf{B}\mathbf{e}'[k-i]] \\
&\quad + \sum_{i=0}^{k} \mathrm{Cov}[\mathbf{C}\mathbf{A_e}^i\mathbf{B}\mathbf{e}[k-i]])\mathbf{K}^T\bar{\mathbf{U}} \\
&= \mathbf{D} + 2\sum_{i=0}^{\infty} \bar{\mathbf{U}}^H\mathbf{K}\mathbf{C}\mathbf{A_e}^i\mathbf{B}\boldsymbol{\Sigma}_\mathbf{e}\mathbf{B}^T(\mathbf{A_e}^T)^i\mathbf{C}^T\mathbf{K}^T\bar{\mathbf{U}}
\end{aligned} \tag{C.27}$$

using the fact that the innovation is independent of the dynamic watermark. Define $\mathbf{X}$ as the solution of the following Lyapunov equation:

$$\mathbf{A_e}\mathbf{X}\mathbf{A_e^T} - \mathbf{X} + \mathbf{B}\boldsymbol{\Sigma}_\mathbf{e}\mathbf{B^T} = \mathbf{0} \tag{C.28}$$

since $\mathbf{A_e}$ is stable,

$$\mathbf{X} = \sum_{i=0}^{\infty} \mathbf{A_e}^i\mathbf{B}\boldsymbol{\Sigma}_\mathbf{e}\mathbf{B}^T(\mathbf{A_e}^T)^i \tag{C.29}$$

thus, the covariance of $\bar{\mathbf{g}}[k+1]$ is:

$$\mathrm{Cov}[\bar{\mathbf{g}}[k+1]] = \mathbf{D} + 2\bar{\mathbf{U}}^H\mathbf{K}\mathbf{C}\mathbf{X}\mathbf{C}^T\mathbf{K}^T\bar{\mathbf{U}} \tag{C.30}$$

The covariance of $\bar{\mathbf{g}}[k+1]$ and $\mathbf{e}[k]$ is:

$$\begin{aligned}
&\mathbb{E}[\bar{\mathbf{g}}[k+1]\mathbf{e}[k]^T] \\
&= \mathbb{E}[\bar{\mathbf{U}}^H\mathbf{K}_{k+1}(\mathbf{C}(\mathbf{x}[k+1-l] - \mathbf{A_d}\hat{\mathbf{x}}_{k|k} - \mathbf{B}\mathbf{h}(\mathbf{z}^k)) \\
&\qquad + \mathbf{n}[k+1-l] - \mathbf{C}\mathbf{B}\mathbf{e}[k])\mathbf{e}[k]^T] \\
&= \bar{\mathbf{U}}^H\mathbf{K}\mathbb{E}[-\mathbf{C}\mathbf{B}\mathbf{e}[k]\mathbf{e}[k]^T] \\
&= -\bar{\mathbf{U}}^H\mathbf{K}\mathbf{C}\mathbf{B}\boldsymbol{\Sigma}_\mathbf{e}
\end{aligned} \tag{C.31}$$

since other terms are independent of the current dynamic watermark $\mathbf{e}[k]$. Combined with the proofs of mean and covariance completes the proof of equation (7.45) and (7.46).

C.3 Proof of equation (7.50)

Under the destabilization attack, the attacked control input:

$$\mathbf{u}_a[k] = \mathbf{u}[k] + \mathbf{A_p}\mathbf{x}[k] \tag{C.32}$$

the state at $k+1$ if there is no attack is:

$$\mathbf{x}[k+1] = \mathbf{A_d}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] + \mathbf{w}[k+1] \tag{C.33}$$

and denote the state at $k+1$ after attack as:

$$\begin{aligned} \mathbf{x}_a[k+1] &= \mathbf{A_d}\mathbf{x}[k] + \mathbf{B}\mathbf{u}_a[k] + \mathbf{w}[k+1] \\ &= \mathbf{x}[k+1] + \mathbf{B}\mathbf{A_p}\mathbf{x}[k] \end{aligned} \tag{C.34}$$

the measurement at $k+1$ is:

$$\begin{aligned} \mathbf{z}[k+1] &= \mathbf{C}\mathbf{x}_a[k+1] + \mathbf{n}[k+1] \\ &= \mathbf{y}[k+1] + \mathbf{C}\mathbf{B}\mathbf{A_p}\mathbf{x}[k] \end{aligned} \tag{C.35}$$

The innovation after attack is:

$$\begin{aligned} \boldsymbol{\upsilon}_a[k+1] &= \mathbf{z}[k+1] - \mathbf{C}\hat{\mathbf{x}}_{k+1|k} \\ &= \mathbf{y}[k+1] - \mathbf{C}\hat{\mathbf{x}}_{k+1|k} + \mathbf{C}\mathbf{B}\mathbf{A_p}\mathbf{x}[k] \end{aligned} \tag{C.36}$$

so the innovation mean is:

$$\begin{aligned} \mathbb{E}[\boldsymbol{\upsilon}_a[k+1]] &= \mathbb{E}[\mathbf{y}[k+1] - \mathbf{C}\hat{\mathbf{x}}_{k+1|k} + \mathbf{C}\mathbf{B}\mathbf{A_p}\mathbf{x}[k]] \\ &= \mathbf{C}\mathbf{B}\mathbf{A_p}\mathbb{E}[\mathbf{x}[k]] \\ &= \mathbf{C}\mathbf{B}\mathbf{A_p}\hat{\mathbf{x}}_{k|k} \end{aligned} \tag{C.37}$$

the first two terms are the innovation without attack which has zeros mean, and by the definition the posterior estiation should be unbiased. The mean of whitened statistic $\bar{\mathbf{g}}[k+1]$ is:

$$\mathbb{E}[\bar{\mathbf{g}}[k+1]] = \bar{\mathbf{U}}^H\mathbf{K}\mathbf{C}\mathbf{B}\mathbf{A_p}\hat{\mathbf{x}}_{k|k} \tag{C.38}$$

The covariance of the innovation is:

$$\begin{aligned} &\mathrm{Cov}[\boldsymbol{\upsilon}_a[k+1]] \\ &= \mathrm{Cov}[\boldsymbol{\upsilon}[k+1] + \mathbf{C}\mathbf{B}\mathbf{A_p}(\mathbf{x}[k] - \hat{\mathbf{x}}_{k|k})] \\ &= \mathbf{R} + \mathbf{C}\mathbf{B}\mathbf{A_p}\mathbf{P}\mathbf{A_p}^T\mathbf{B}^T\mathbf{C}^T \\ &\qquad + \mathbb{E}[\boldsymbol{\upsilon}[k+1](\mathbf{x}[k] - \hat{\mathbf{x}}_{k|k})^T\mathbf{A_p}^T\mathbf{B}^T\mathbf{C}^T] \\ &\qquad + \mathbb{E}[\mathbf{C}\mathbf{B}\mathbf{A_p}(\mathbf{x}[k] - \hat{\mathbf{x}}_{k|k})\boldsymbol{\upsilon}[k+1]^T] \end{aligned} \tag{C.39}$$

where

$$\begin{aligned}
&\mathbb{E}[\boldsymbol{\upsilon}[k+1](\mathbf{x}[k]-\hat{\mathbf{x}}_{k|k})^T\mathbf{A_p}^T\mathbf{B}^T\mathbf{C}^T] \\
&=\mathbb{E}[(\mathbf{C}(\mathbf{x}[k+1]-\hat{\mathbf{x}}_{k+1|k})+\mathbf{n}[k+1])(\mathbf{x}[k]-\hat{\mathbf{x}}_{k|k})^T \\
&\qquad \mathbf{A_p}^T\mathbf{B}^T\mathbf{C}^T] \\
&=\mathbf{C}\mathbb{E}[(\mathbf{x}[k+1]-\hat{\mathbf{x}}_{k+1|k})(\mathbf{x}[k]-\hat{\mathbf{x}}_{k|k})^T]\mathbf{A_p}^T\mathbf{B}^T\mathbf{C}^T \\
&=\mathbf{C}\mathbb{E}[(\mathbf{A_d}\mathbf{x}[k]+\mathbf{B}\mathbf{u}[k]+\mathbf{w}[k+1]-\mathbf{A_d}\hat{\mathbf{x}}_{k|k}-\mathbf{B}\mathbf{u}[k]) \\
&\qquad (\mathbf{x}[k]-\hat{\mathbf{x}}_{k|k})^T]\mathbf{A_p}^T\mathbf{B}^T\mathbf{C}^T \\
&=\mathbf{C}\mathbf{A_d}\mathbb{E}[(\mathbf{x}[k]-\hat{\mathbf{x}}_{k|k})(\mathbf{x}[k]-\hat{\mathbf{x}}_{k|k})^T]\mathbf{A_p}^T\mathbf{B}^T\mathbf{C}^T \\
&=\mathbf{C}\mathbf{A_d}\mathbf{P}\mathbf{A_p}^T\mathbf{B}^T\mathbf{C}^T
\end{aligned} \tag{C.40}$$

Plug it to the covariance equation:

$$\begin{aligned}
&\mathrm{Cov}[\boldsymbol{\upsilon}_a[k+1]] \\
&=\mathbf{R}+\mathbf{C}\mathbf{B}\mathbf{A_p}\mathbf{P}\mathbf{A_p}^T\mathbf{B}^T\mathbf{C}^T+\mathbf{C}\mathbf{A_d}\mathbf{P}\mathbf{A_p}^T\mathbf{B}^T\mathbf{C}^T \\
&\qquad +\mathbf{C}\mathbf{B}\mathbf{A_p}\mathbf{P}\mathbf{A_p}^T\mathbf{B}^T\mathbf{C}^T
\end{aligned} \tag{C.41}$$

Define

$$\mathbf{P_a}=\mathbf{B}\mathbf{A_p}\mathbf{P}\mathbf{A_p}^T\mathbf{B}^T+\mathbf{A_d}\mathbf{P}\mathbf{A_p}^T\mathbf{B}^T+\mathbf{B}\mathbf{A_p}\mathbf{P}\mathbf{A_d}^T \tag{C.42}$$

then the covariance of $\bar{\mathbf{g}}[k+1]$ is:

$$\mathrm{Cov}[\bar{\mathbf{g}}[k+1]]=\mathbf{D}+\bar{\mathbf{U}}^H\mathbf{K}\mathbf{C}\mathbf{P_a}\mathbf{C}^T\mathbf{K}^T\bar{\mathbf{U}} \tag{C.43}$$

The covariance of $\bar{\mathbf{g}}[k+1]$ and $\mathbf{e}[k]$ is zero since all terms in $\boldsymbol{\upsilon}[k+1]$ are independent of $\mathbf{e}[k]$. Combined with the proofs of mean and covariance completes the proof of (7.50).

D Description of Softwares

D.1 Optimization and simulation of power system in Matlab

D.1.1 CVX Toolbox

CVX is a Matlab-based modeling system for convex optimization such as least squares (LS), linear programming (LP), and quadratic programming (QP). Academic users may utilize the CVX Professional capability at no charge. The CVX Professional license can be required by an educational email address. Solvers such as Mosek and Gurobi are highly recommanded for speeding up the optimization. To do so, it is necessary to obtain licenses for Mosek and/or Gurobi directly from the vendors. The solver accessibility needs to be check before optimization.

Version 2.0 of CVX also supports mixed integer disciplined convex programming (MIDCP) such as mixed integer linear programming (MILP).

D.1.2 MATPOWER

MATPOWER is a package of free, open-source Matlab-language M-files for solving steady-state power system simulation and optimization problems, such as optimal power flow (OPF), unit commitment (UC). Many IEEE standard systems are included as cases and can be imported to simulation easily for OPF calculation. Modifications can be made to the standard cases by changing the data while still keeping the data format, adding or deleting elements like generators, loads, or even branches. The OPF can also be extended by adding user-defined variables, constraints and costs. The details of extending the OPF can be found in Chapter 7 in the MATPOWER user manual.

MATPOWER uses an interior solver called MIPS, while there are multiple solvers can be choosen to solve OPF in MATPOWER, such as CPLEX, Gurobi, and MOSEK. When these solvers are installed, they can be used by setting the solver option in the OPF code.

Note that MATPOWER can only solve OPF for single period. The MATPOWER Optimal Scheduling Tool (MOST) is an extension of MATPOWER for solving generalized steady-state electric power scheduling problems. MOST can be used to solve problems as complex as a stochastic, security-constrained, combined unit-commitment and **multiperiod** optimal power flow problem with locational contingency and load-following reserves, ramping costs and constraints, deferrable demands, lossy storage resources and uncertain renewable generation.

MOST is designed to solve multiperiod OPF. It models the renewable resources as extra generators and it adds new elements such as energy storage and other time-varying parameters such as price. Many additional functions are provided to add these new elements, load the date for extra generators and storages, and load the profiles. Tutorials of determinstic multiperiod OPF with wind profile can be found in the MOST user manual as a very good starting point of simulations.

D.1.3 Power System Toolbox

Power System Toolbox (PST) is a package of Matlab M-files for performing stability simulations of power system, such as transient stability and small signal analysis and damping controller design. While very few standard models are included in PST, users can customize the power system model following the set of rules according to the Single Line Diagram Two Area System tutorial given in the manual. Descriptions of functions for dynamic simulations such as generating disturbances to the control input can also be found in the manual.

However, this toolbox is pretty old and the user manual is not well maintained. Matlab Simulink is used for model the dynamic of PV farm in this dissertation. New simulation tools are highly recommended for future students.

D.2 Optimization and simulation of power system in Python

To simulate the power system in Python is unnecessary unless machine learning algorithms are used for optimization.

D.2.1 PYPOWER

PYPOWER is a power flow and OPF solver that ported from MATPOWER to Python language. All the cases included in MATPOWER can be imported to PYPOWER directly. However, a solver is all it can be, modifications on these cases are really hard to make directly in PYPOWER unless the user is familar with the data format used in MATPOWER and no other functions are provided.

D.2.2 Pandapower

Pandapower combines the popular data analysis library "Pandas" and the power flow solver PYPOWER in Python to create an easy to use network calculation program for network analysis and optimization in power systems. Many bench mark grids are included in pandapower besides the IEEE cases, and the data structures for different elements including load, static generator, generator, and storage are all given in its document. Modifications on the cases are also easy to make for users who are familiar with data manipulations using Pandas.

The OPF in pandapower is solved by PYPOWER using the "runopp" function. However, the convergence properties are not guaranteed. Any modified cases must be checked to have a convergent OPF solution before further simulations. Sometimes the OPF is not convergent because the modified load is too small or more constraints are needed for the modified system to get a solution.

D.3 Reinforcement learning in Python

D.3.1 OpenAI Baselines

OpenAI Baselines is a package of implementations of reinforcement learning algorithms using Tensorflow. Although the codes are all of high quality and open-sourced by the OpenAI company, they have not been maintained for years, and there is no document for researchers who are not from a solid computer science background and just want to use the package for optimization.

D.3.2 Stable Baselines

Stable Baselines is a package that improved the implementations of RL algorithms in OpenAI Baselines. The documents are well organized and many state-of-art RL algorithms are included. It also provides examples on training, saving, and loading agents to learn in a Gym environment. The RL algorithms in this dissertation are all implemented by Stable Baselines except for chapter 6. However, it only supports Tensorflow versions before 2.0.0. With the popularity of machine learning frameworks switching from Tensorflow to Pytorch, this package is migrated to Stable-Baselines3.

D.3.3 Stable Baselines3

Stable Baselines3 (SB3) is based on the Pytorch framework and has all feasures in Stable Baselines. It is highly recommended for future RL researches. Some experience for installation and environment configuration will be given in this section.

Anaconda or VScode are recommended for Windows users to install Python and Python packages. Most of the time, a stable version of software is better than the latest version. For a large training set, GPU is recommended for training. Users need to visit the Cuda website where the supporting version of Pytorch will be listed, then install the Pytorch package. GPU can be used for training by setting the "device" attribute in the agent functions.

Most RL algorithms are trained and tested on environments belong to Gymnasium, a standard API for RL algorithms and a diverse collections of environments such as classic control and Atari games. To train and test an agent in the power system, users have to customize an environment that follows the Gymnasium interface. In specific, a new class has to be defined inherited from the gymnasium environment class, which has functions such as "step", "reset" and so on. A Gym Environment Checker function is provided in SB3, checking the custom environment before training an agent on this environment can save a lot of debugging time. Examples can be found on the documentations of both the SB3 and Gymnasium. The environment need to be imported and specified for training and testing an agent.

To monitor the training process of a RL agent needs the tensorboard, where training parameters such as the mean of episode reward and the mean of the step reward can be plotted and updated during training. The usage of tensorboard is also provided in the SB3 documentation.

E All Publication Published, Submitted, and Planned

E.1 Journals

1. Y. Li and J. Wu, “Optimum integration of solar energy with battery energy storage systems,” IEEE Transactions on Engineering Management, 2020.

2. Y. Li and J. Wu, “Low latency cyberattack detection in smart grids with deep reinforcement learning,” International Journal of Electrical Power & Energy Systems, vol. 142, p. 108265, 2022.

3. Y. Li, J. Wu, and Y. Pan, “Deep reinforcement learning for online scheduling of photovoltaic systems with battery energy storage systems,” Intelligent and Converged Networks. Tsinghua University Press (Under review).

4. Y. Li, J. Wu, and Y. Pan, “Intelligent Optimal Power Flow Control for Wind-Powered Microgrid with Deep Reinforcement Learning,” Intelligent and Converged Networks. Tsinghua University Press (Under review).

5. Y. Li, N. Lin, J. Wu, Y. Pan, and Y. Zhao, “Low Latency Attack Detection with Dynamic Watermarking for Grid-Connected Photovoltaic Systems,” Journal of Emerging and Selected Topics in Industrial Electronics. IEEE (Under review).

E.2 Conference

1. Y. Li and J. Wu, “Optimum design of battery-assisted photo-voltaic energy system for a commercial application,” in 2019 IEEE Power & Energy Society General Meeting (PESGM). IEEE, 2019.

2. S. Nath, Y. Li, J. Wu, and P. Fan, “Multi-user multi-channel computation offloading and resource allocation for mobile edge computing,” in ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE, 2020, pp. 1–6.

E.3 Posters presentation

1. Y. Li and J. Wu, “Optimum design of battery-assisted photo-voltaic energy system for a commercial application,” in 2019 IEEE Power & Energy Society General Meeting (PESGM). IEEE, 2019.

Chapter 3 is reproduced from journal paper 1. Chapter 4 is reproduced from journal paper 3. Chapter 5 is reproduced from journal paper 4. Chapter 7 is reproduced from journal paper 5. Chapter 8 is reproduced from journal paper 2.