ABSTRACT

FALLAH DIZCHE, AMIRHASSAN. Machine Learning Methods for Communication Systems and Control of Cyber-Physical Systems. (Under the direction of Dr. Alexandra Duel-Hallen and Dr. Aranya Chakrabortty).

This dissertation work focuses on data-driven and machine learning solutions for communication systems and optimal control design problems. First, we consider the problem of communication-efficient optimal control of uncertain cyber-physical systems and linear networked control systems (NCSs). In chapter 2, we present an online wide-area oscillation damping control (WAC) design for uncertain power systems models using reinforcement learning. We assume that the exact small-signal model of the power system is not known to the operator and use the nominal model and online measurements of the generator states and control inputs to find a state-feedback controller that minimizes a given quadratic energy cost. However, unlike conventional linear quadratic regulators (LQR), we intend our controller to be sparse, so its implementation reduces the communication costs. Therefore, we employ the gradient support pursuit (GraSP) optimization algorithm to impose sparsity constraints on the control gain matrix during learning. Using the IEEE 39-bus power system model, we show that the proposed learning method provides reliable LQR performance while significantly reducing the communication cost.

Large-scale control systems require a robust communication network to properly exchange sensor and actuator information. Millimeter wave (mmWave) employed in 5G cellular networks can provide connectivity in mobile NCSs, such as swarms of UAVs [72], connected autonomous vehicles [53], and electric vehicles integration into the smart grid [51], [69]. However, sensitivity to line-of-sight (LoS) blockage challenges the performance and robustness of mmWave communication systems. In chapter 4, we propose to use received signal strength (RSS) data in a data-driven method to predict LoS blockage far in

advance in mmWave cellular networks and improve reliability and robustness of the network. We apply the MiniRocket machine learning (ML) method to provide early warning of mobile mmWave signal blockage hundreds of milliseconds before it occurs, thus facilitates a proactive response. MmWave signal datasets for training and testing are created using our physics-based simulation tool that models blockage-induced diffraction accurately. Our insights and numerical results show that the proposed early warning capability relies on the diffraction-induced pre-blockage signal patterns and is independent of the topology and the speed of movement.

Machine Learning Methods for Communication Systems and Control of Cyber-Physical Systems

by
Amirhassan Fallah Dizche

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina
2022

APPROVED BY:

Dr. Hans Hallen	Dr. Huaiyu Dai
Dr. Alexandra Duel-Hallen Co-chair of Advisory Committee	Dr. Aranya Chakrabortty Co-chair of Advisory Committee

DEDICATION

To my parents, Alireza and Mehrasa. To my love, Yasaman.

BIOGRAPHY

Amirhassan Fallah Dizche was born in Tehran, Iran. He received his Bachelor of Science in Electrical and Computer Engineering from Shahid Beheshti University, Tehran, Iran, in 2011. He completed his Master of Science degree in Electrical Engineering from the North Carolina State University, Raleigh, NC, in 2012, focusing on communication and signal processing. In spring 2017, he joined the Ph.D. program of the North Carolina State University under the supervision of Dr. Alexandra Duel-Hallen and Dr. Aranya Chakrabortty. His current research interests include reinforcement learning, optimal control, machine learning, and communication systems.

ACKNOWLEDGEMENTS

First of all, I would like to extend my gratitude to my co-advisors, Dr. Alexandra Duel-Hallen and Dr. Aranya Chakrabortty, for their guidance, support, patience, and insights. I would like to thank you for sharing your knowledge and expertise in communications, optimization, control theory, and power systems with me to help me move forward and deeply understand my research problems. Thank you for your support, trust, and encouragement during difficult times and your patience in helping me with writing and presenting our work. Your insightful comments and questions during our numerous discussions helped me learn how to approach research problems thoroughly. Thanks for making this journey fulfilling for me. It was a pleasure exploring challenging topics with you.

Also, I would like to thank my committee members, Dr. Hans Hallen and Dr. Huaiyu Dai, for their encouragement, support, and insightful comments on my research work. Thanks for serving on my committee board. I extend thanks to all professors who taught me at NC State University. The excellent lectures have laid solid foundations for me to pursue challenging research topics.

I would like to thank my parents for their unlimited love and support during every step of my life. Without your continuous love and support, I could not accomplish this much. I would also like to show my appreciation to my wife for her company and support during my studies at NC State University. I feel so lucky to have met you and cherish every moment we spend together.

I would like to thank my fellow lab members, Feier Lian, Lu An, Prathistha Shukla, and Roghieh Mahdavi Haji, for your collaborative work and helpful discussions. I would also like to thank all other friends and fellow graduate students who study and work with me in the Electrical and Computer Engineering department of NC State University. I enjoyed the time we spent together attending lectures, working on coursework and projects, and sharing

ideas and thoughts. Special thanks to all my dear friends. Your company and support made this possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

Networked Control Systems (NCSs), such as power systems and swarms of unmanned aerial vehicles (UAVs), require exchanging an enormous amount of sensor and actuator data in real-time between various parts of their system. Having a highly robust communication network is vital for proper control of large-scale NCSs. Due to the high volume of data exchange, it is also economically desirable to optimize for sparse sensors or reduce the communication between sensors and controllers to reduce the costs of communication links or make the system more robust to communication impairments, e.g., delay and packet loss. Moreover, uncertainty in the system models of NCSs degrades the control performance and could potentially lead to instability of the NCS. Robust control methods can handle a limited amount of uncertainty in the NCS models. However, controlling com-

pletely unknown or highly uncertain large-scale NCSs require special methods to interact with the NCS and learn the uncertainties using learning methods and measurements from sensors and actuators within the system.

Sparse wide-area control (WAC) of power systems to save on communication costs is studied in [25], [47], [36]. A common limitation among these methods is that they assume having perfect knowledge of the NCS model and do not consider uncertainties. While in reality, the the power system model may change due to uncertainties, including renewable generation, power electronics, and active loads. Robust sparse control is proposed in [49, 6, 48] to consider a limited amount of uncertainty. However, it is generally challenging to find a reliable upper bound for uncertainty in large-scale NCSs. Hence, the control designers are interested in learning the NCS model online using measurements of the system.

Reinforcement learning (RL) [44, 40, 12, 45] is a suitable candidate for online optimal control of unknown and uncertain NCSs. Adaptive dynamic programming (ADP) [38, 76], Q-learning [43], and integral concurrent learning [60] are introduced in the literature for optimal control of uncertain dynamic systems. These methods have impractically long convergence times for large-scale dynamic systems, such as power systems. Motivated by the RL design proposed in [70], we propose a spares RL algorithm that speeds up convergence significantly by exploiting the structured uncertainty in linear power system models. In order to reduce the communication cost, we integrate the RL design with Gradient Support Pursuit (GraSP), which imposes sparsity constraints on the control gain matrix [8]. The proposed algorithm learns a sparse controller, thus simultaneously satisfying the communications cost constraint and overcoming the NCS model uncertainty.

An LQR controller designed with RL for continuous control of high-dimensional nonlinear dynamic systems is introduced in [55], where the complex uncertain dynamics are projected to a small dimension latent space using a neural network decoder. The RL model learns the control using the latent space and applies it to the nonlinear system via a decoder

network. This method is similar to designing control for reduced-order models and does not consider the communication cost of implementing the designed control policy in large-scale systems. A game-theoretic online solution for equilibrium seeking feedback control of uncertain nonlinear dynamic systems is studied in [11]. The offered solution is capable of following the desired trajectory with a bounded error for a time-varying generalized equation. An offline communication-efficient policy gradient RL method for NCSs with heterogeneous nodes is proposed in [18]. An off-policy RL optimal controller for Large-scale NCS based on state aggregation is introduced in [34], where sparsity in the feedback is promoted using ℓ_1 regularization. Continuous fitted value iteration RL for control of the small-scale unknown nonlinear dynamic system is introduced in [52]. A combination of Gaussian Process models to learn the nonlinear dynamics and Model Predictive Control (MPC) to control the learned nonlinear dynamics is proposed in [63] to control small-scale nonlinear dynamic systems. Deep Deterministic Policy Gradient (DDPG) deep RL method with neural networks as actor and critic function approximators is suggested in [32] to learn the uncertain stochastic distributions and an optimal control policy to damp low-frequency oscillations in power systems.

Approaches to improve RL controller design include utilization of temporal logic, which helps to mitigate unexpected changes in the dynamics [65, 46], and meta-learning, which facilitates sample-efficient learning of new tasks in dynamic systems [58, 27]. However, a common limitation of these methods is the extremely long convergence time in high-dimensional systems. Additionally, estimating the system model using conventional system ID methods [71, 67] is very time-consuming.

Implementing feedback control in large-scale NCSs require a robust and low-latency communication network to facilitate information exchange among sensors and actuators of the dynamic system. 5G and beyond cellular communication networks are a promising candidate to provide the network requirements of cyber-physical systems and NCSs. Millime-

terwave is the key enabling technology for 5G and beyond systems. However, mmWave communication systems rely heavily on line-of-sight (LoS) links to provide sufficient received signal strength. Due to weak diffraction properties and high penetration loss, mmWave signals are highly susceptible to blockage by physical objects [62]. Making the communication robust to this impairment is necessary to make the technology practical.

Multiple links with several Base Stations (BS) is proposed in [28, 61, 7] to react to LoS blockage in mmWave networks. Several mmWave systems assisted by sub-6 GHz links were proposed [7, 56] to enable fast and reliable link switching after LoS blockage. Frequency-dependent diffraction properties and our physics model-based insights were used to predict LoS blockage using sub-6 GHz observations without assuming a specific topology or environment [3]. A deep learning (DL) model was developed in [5] to predict link blockage in static environments using a sub-6 GHz signal. A method based on recurrent neural networks (RNN) used BS handoff data for a static environment, identical to that of the mobile user, to predict LoS blockage based on the handoff probability [4]. Blockage of neighboring paths in an environment with slowly moving receivers and obstacles was used to predict the mmWave LoS blockage [9]. Additional sensor information from cameras, along with a deep convolutional model, was proposed in [16, 59] for LoS blockage prediction. Simulated data and DL were used to predict mmWave blockage in indoor scenarios with limited speeds of obstacles [14]. A meta-learning framework and RNNs were proposed to predict mmWave link blockage using a Rician fading channel model for fixed BS and UE scenarios [39].

We propose a data-driven method to predict LoS blockage hundreds of milliseconds ahead using mmWave received signal strength (RSS) samples and time-series classification models. We generate a dataset of mobile mmWave scenarios, including LoS received signal as well as LoS-to-non-line-of-sight (NLoS) transitions, using our physical model [31, 54, 26, 3], which employs Fresnel diffraction and method of images to calculate the received signal. Our approach does not assume a specific environment (indoor, outdoor, static, fixed

topology), obstacle motion, direction, or speed and does not use simplistic simulation models. Moreover, it solely relies on in-band mmWave signal measurements and does not require corroborating information from sub-6 GHz signals or additional sensors such as cameras. Finally, our ML solution uses the extremely fast MiniRocket [22] time-series modeling technique with much lower computational complexity than DL and RNN models and does not incur high computation costs while finding the solution in real-time [22].

This thesis is organized as follows. In chapter 2, we present an online wide-area oscillation damping control (WAC) design for uncertain power systems models using reinforcement learning. We assume that the exact small-signal model of the power system at the onset of a contingency is not known to the operator and use the nominal model and online measurements of the generator states and control inputs to rapidly converge to a state-feedback controller that minimizes a given quadratic energy cost. However, unlike conventional linear quadratic regulators (LQR), we intend our controller to be sparse, so its implementation reduces the communication costs. We, therefore, employ the gradient support pursuit (GraSP) optimization algorithm to impose sparsity constraints on the control gain matrix during learning. Using the IEEE 39-bus power system model, it is demonstrated that the proposed learning method provides reliable LQR performance while the controller matched to the nominal model becomes unstable for severely uncertain systems.

In chapter 3, we use received signal strength (RSS) data to propose a data-driven method to predict line-of-sight (LoS) blockage in mmWave cellular networks. Sensitivity to blockage challenges performance of mmWave communication systems. We apply the MiniRocket machine learning (ML) method to provide early warning of mobile mmWave signal blockage hundreds of milliseconds, or tens of 5G frames, before it occurs. MmWave signal datasets for training and testing are created using our physics-based simulation tool that models diffraction accurately. Our insights and numerical results show that the proposed early warning capability relies on the diffraction-induced pre-blockage signal patterns and is

independent of the topology and speed. We demonstrate that the proposed ML method accurately predicts an upcoming blockage under practical impairments such as multipath fading and additive noise and thus facilitates a proactive response.

Chapter 4 includes conclusion of the thesis and discusses possible future research directions.

CHAPTER

2

SPARSE WIDE-AREA CONTROL OF POWER SYSTEMS USING DATA-DRIVEN REINFORCEMENT LEARNING

In this chapter we present an online wide-area oscillation damping control (WAC) design for uncertain models of power systems using ideas from reinforcement learning. We assume that the exact small-signal model of the power system at the onset of a contingency is not known to the operator and use the nominal model and online measurements of the generator states and control inputs to rapidly converge to a state-feedback controller that

minimizes a given quadratic energy cost. However, unlike conventional linear quadratic regulators (LQR), we intend our controller to be sparse, so its implementation reduces the communication costs. We, therefore, employ the gradient support pursuit (GraSP) optimization algorithm to impose sparsity constraints on the control gain matrix during learning. The sparse controller is thereafter implemented using distributed communication. Using the IEEE 39-bus power system model with 1149 unknown parameters, it is demonstrated that the proposed learning method provides reliable LQR performance while the controller matched to the nominal model becomes unstable for severely uncertain systems.

2.1 Introduction

Over the past few years, the occurrence of a series of blackouts in different parts of the world has led power system utility owners to look beyond the traditional approach of controlling the grid via local feedback and instead transition to system-wide control, often referred to as wide-area control (WAC). Several papers on WAC design for damping of electromechanical oscillations have been reported in the recent literature [17, 75, 35, 20].The basic approach is to linearize the system model around a given operating point, and design linear state-feedback or output-feedback LQR controllers for taking damping control action via the generator excitation control system. However, designing a traditional LQR controller is not suitable for WAC since it demands a dense all-to-all communication graph between every pair of generators. To save on communication costs, sparse control designs for WAC have been reported in several recent papers such as [25], [47], [36]. But a common limitation among all these designs is that they are based on perfect knowledge of the grid model.

In reality, however, the operating point of a grid may move over wide ranges, and therefore using just one fixed controller might not be optimal. The problem is becoming more notable with increasing penetration of renewables, power electronics, and active loads,

whose dynamic characteristics vary constantly over time. One way to counteract these uncertainties would be to design a robust WAC. The challenge, however, is that with millions of electric vehicles and inverter-based generation points being envisioned to be integrated to the US grid in very near future, primarily in a completely plug-and-play fashion, it is extremely difficult to quantify a reliable upper bound for these uncertainties that can be used for robust control designs. Recent papers such as [49, 6, 48] have proposed robust sparse control, but those designs usually work for fairly limited amount of uncertainty. Operators are, therefore, more interested in learning the power system model using online measurements available from sophisticated sensors such as Phasor Measurement Units (PMU) after contingencies, and in developing real-time control actions that result from learning.

Motivated by this problem, in this chapter we present a LQR-based WAC design using online reinforcement learning (RL). RL has been shown to be an excellent candidate for online optimal control design under model uncertainty in several recent papers such as [44, 40, 12, 45]. Other variants of online learning such as adaptive dynamic programming (ADP) [38, 76], Q-learning [43], and integral concurrent learning [60], for both continuous-time and discrete-time dynamic systems have also been proposed. In this chapter, we adopt the RL design proposed in [70], whereby online measurements of generator states and control inputs are used to learn an optimal LQR controller, given a choice of the objective function. However, the algorithm in [70] has very long convergence time due to the assumption of completely unknown system model. In this chapter, we exploit the knowledge of the approximate, or nominal, model to speed up convergence significantly. Using online measurements instead of labeled data-sets categorizes the proposed algorithm as unsupervised learning while using the objective function value as reinforcement signal will make it RL.

In order to reduce the communication cost, we integrate the RL design with Gradient

Support Pursuit (GraSP) that imposes sparsity constraints on the control gain matrix [8]. The proposed algorithm incorporates the advantages of RL control and offline sparse controllers. This algorithm learns a sparse controller, thus simultaneously satisfying the communications cost constraint and overcoming the model uncertainty. The proposed design is carried out in two sequential stages: (1) following a contingency, state estimates generated by decentralized Kalman filters at each generator, as well as the generator control inputs stream in to a central coordinator that serves as a 'critic' which simultaneously learns the sparse optimal controllers $\mathbf{K}_{SP}$ and applies the corresponding control input u; (2) Once the $\mathbf{K}_{SP}$-learning loop converges, the controller is implemented by a distributed sparse communication topology connecting the selected sets of generators. We validate these two stages using simulations of the IEEE 39-bus 10-generator power system model with 1149 unknown parameters. We highlight the numerical trade-offs of the two stages for learning versus implementation for different levels of uncertainty.

The *main contributions* of this chapter are:

- Reduce the convergence time of online RL control algorithm by exploiting the knowledge of the nominal model for WAC of power systems.
- Develop a a sparsity-constrained online learning control algorithm that reduces the communication cost.

The rest of the chapter is organized as follows. Section 2.2 formulates the proposed sparse WAC problem. Section 2.3 briefly reviews the use of RL for LQR designs and presents the main sparse learning algorithm by integrating RL with GraSP. Section 2.4 presents simulation results and numerical analysis. Section 2.5 concludes the chapter.

2.2 Problem Statement

2.2.1 Power System Model

Consider a power system network with n synchronous machines. Each machine is considered to be modeled by its *flux-decay* model [42], which is a common choice for designing wide-area damping controllers using excitation control. The model for the i^{th} generator can be written as

$$\dot{\delta}_i = \omega_i \tag{2.1}$$

$$M_i\dot{\omega}_i = P_{mi} - d_i\omega_i - \frac{|\mathbf{V}_i|E_i}{x'_{di}}\sin(\delta_i - \angle\mathbf{V}_i) + \frac{|\mathbf{V}_i|^2}{2}\Big(\frac{1}{x'_{di}} - \frac{1}{x_{qi}}\Big)\sin(2\delta_i - 2\angle\mathbf{V}_i) \tag{2.2}$$

$$\tau_{doi}\dot{E}_i = -\frac{x_{di}}{x'_{di}}E_i + \Big(\frac{x_{di}}{x'_{di}} - 1\Big)|\mathbf{V}_i|\cos(\delta_i - \angle\mathbf{V}_i) + V_{fdi}, \tag{2.3}$$

followed by active and reactive power balance equations

$$P_i = \frac{E_i|\mathbf{V}_i|}{x'_{di}}\sin(\delta_i - \angle\mathbf{V}_i) - \frac{|\mathbf{V}_i|^2}{2}\Big(\frac{1}{x'_{di}} - \frac{1}{x_{qi}}\Big)\sin(2\delta_i - 2\angle\mathbf{V}_i) \tag{2.4}$$

$$Q_i = \frac{E_i|\mathbf{V}_i|}{x'_{di}}\cos(\delta_i - \angle\mathbf{V}_i) - |\mathbf{V}_i|^2\left(\frac{\sin^2(\delta_i - \angle\mathbf{V}_i)}{x_{qi}} - \frac{\cos^2(\delta_i - \angle\mathbf{V}_i)}{x'_{di}}\right). \tag{2.5}$$

Equations (1)-(2) represent the swing dynamics, and (3) represents the electro-magnetic dynamics of the i^{th} generator. $\mathbf{V}_i$ is the voltage phasor at the generator bus, P_i and Q_i are the active and reactive power outputs of the generator, V_{fdi} is the exciter voltage, and the

remaining constants denote various model parameters whose definitions can be found in [42]. The generator model is coupled with the model of an exciter consisting of an automatic voltage regulator (AVR) and a power system stabilizer (PSS) whose combined dynamics can be written as

$$\tau_{e_i}\dot{V}_{fdi} = -V_{fdi} + V_{fdi}^{\bigstar} + K_{ai}(|\mathbf{V}_i| - |\mathbf{V}_i|^{\bigstar} - \nu_i + \gamma_i) \tag{2.6}$$

$$\dot{\zeta}_i = \mathbf{A}_{pss}\zeta_i + \mathbf{B}_{pss}\omega_i,\ \nu_i = C_{pss}\zeta_i + D_{pss}\omega_i \tag{2.7}$$

where superscript $^{\bigstar}$ means set-point. The signal γ_i serves as a control input representing an additional voltage reference signal to the AVR that can be designed to add damping to the slow or inter-area oscillation modes using state feedback from all generators spread across the grid. These controllers are referred to as wide-area controllers (WAC).

Our control design does not necessarily need the generators to follow this simple model. Detailed models of generators are allowed, provided all the generator states can be measured or estimated (a short description of decentralized state estimation will be given shortly). In general, we assume that the nonlinear model of the i^{th} generator has n_i states $\xi_i = [\delta_i, \omega_i, E_i, \dots] \in \mathbb{R}^{n_i}$, and one scalar control input γ_i as in (6), which is the field excitation voltage. Let the pre-disturbance equilibrium of the i^{th} generator be $\xi_i^* = [\delta_i{}^*, \omega_i{}^*, E_i{}^*, \dots]$. The differential-algebraic model of the generators and the power flow is converted to a state-space model using Kron reduction [42], and linearized about $\xi_i^*, i = 1, 2, \dots, n$. The small-signal model of the system with the i^{th} state defined as $\mathbf{x}_i = \xi_i - \xi_i^*$, is written as

$$\begin{bmatrix} \mathbf{x_1}(t) \\ \mathbf{x_2}(t) \\ \vdots \\ \mathbf{x_n}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A_{11}} & \mathbf{A_{12}} & \cdots & \mathbf{A_{1n}} \\ \mathbf{A_{21}} & \mathbf{A_{22}} & \cdots & \mathbf{A_{2n}} \\ \vdots & & & \\ \mathbf{A_{n1}} & \mathbf{A_{n2}} & \cdots & \mathbf{A_{nn}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1(t) \\ \mathbf{x}_2(t) \\ \vdots \\ \mathbf{x}_n(t) \end{bmatrix} + \begin{bmatrix} \mathbf{B}_1 & & & \\ & \mathbf{B}_2 & & \\ & & \ddots & \\ & & & \mathbf{B}_n \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_n(t) \end{bmatrix}. \quad (2.8)$$

Note that the state vector $\mathbf{x}_i$ includes the AVR and PSS states from (6)-(7) linearized around their respective equilibria. The small-signal control input is given by $u_i = \Delta\gamma_i$. The power system model (2.8) can be written in compact form by stacking state and input vectors as

$$\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{x}(0) = \mathbf{x}_0. \quad (2.9)$$

Model parameters and operating conditions in the grid change frequently between contingencies, and therefore the assumption of the exact knowledge of $\mathbf{A}$ and $\mathbf{B}$ is impractical. In the current state of art, utilities use offline models that may have been constructed years ago and simply depend on the inherent robustness of the grid to save the closed-loop response even if the control input $\mathbf{u}$ is not properly matched with the actual $(\mathbf{A}, \mathbf{B})$ matrices that apply to that situation. With rapid increase in renewable penetration and their power electronic interfaces as well as stochastic loads, such as electric vehicles, this robustness can no longer be counted on. Thus operators are more interested in designing the wide-area controller $\mathbf{u}$ by online learning.

One choice is to design a LQR controller for $\mathbf{u}$, as shown in [25]. For this, we will assume

the state $\mathbf{x}(t)$ to be available for feedback. This can be done by placing PMUs at geometrically observable set of buses, so that the voltage and current phasors at every generator bus are computable. A decentralized unscented Kalman filter is assumed to be installed at every generator. The computed (or measured if a PMU is already at the generator bus) values of the bus voltage and current phasors are used by the Kalman filter to estimate the generator state vector $\hat{\mathbf{x}}_i$. Assuming that the KF runs continuously and is sufficiently faster than the generator dynamics, for the rest of the chapter we will simply assume $\hat{\mathbf{x}}_i = \mathbf{x}_i$. For more details on this KF please see [66].

2.2.2 Optimal Wide-Area Control

The wide-area damping control problem for the power system model (2.9) is posed as a LQR problem, i.e. find matrix $\mathbf{K} \in \mathbb{R}^{m \times n}$ such that the state-feedback control $\mathbf{u}(t) = -\mathbf{K}\mathbf{x}$ minimizes the quadratic energy function:

$$J = \int_0^\infty (\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t))dt \tag{2.10}$$

where $\mathbf{Q} \geq 0$ and $\mathbf{R} > 0$ are design matrices with appropriate dimensions. In the vector form, the controller can be expressed as

$$\begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_n(t) \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} & \cdots & \mathbf{K}_{1n} \\ \mathbf{K}_{21} & \mathbf{K}_{22} & \cdots & \mathbf{K}_{2n} \\ \vdots & & & \\ \mathbf{K}_{n1} & \mathbf{K}_{n2} & \cdots & \mathbf{K}_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1(t) \\ \mathbf{x}_2(t) \\ \vdots \\ \mathbf{x}_n(t) \end{bmatrix} \tag{2.11}$$

where the sub-matrix $\mathbf{K}_{ij}$ indicates feedback gain from the states of generator j to the controller of generator i, while $\mathbf{K}_{ii}$ is the self-feedback gain for generator i. Following

Hamiltonian theory, the controller $\mathbf{K}$ can be designed by solving the algebraic Riccati equation [45]. When $\mathbf{K}$ is optimal, every $\mathbf{K}_{ij}$ and $\mathbf{K}_{ii}$, in general, are non-zero matrices. We define feedback links from states to control input within one generator as self-links and the feedback links between different generators as communication links. Solution of LQR will usually result in a dense $\mathbf{K}$ requiring a communication link from every state to every control input. Such dense communication graph requires large volume of data exchange among the generators. In order to reduce the communication cost [47], we therefore impose an extra constraint of reducing the cardinality of $\mathbf{K}$ to make it sparse and reduce the number of communication links

$$s = \mathrm{Card}_{\mathrm{off}}(\mathbf{K}) = \sum\nolimits_{i,j=1,i\neq j}^{n} \mathrm{nnz}(\mathbf{K}_{ij}) \tag{2.12}$$

where nnz(.) operator returns the number of nonzero elements of a matrix. The self-feedback gains $\mathbf{K}_{ii}$ are not counted in the definition (2.12) due to their negligible cost.

2.2.3 Model Uncertainties

We assume that the exact values of the matrices $\mathbf{A}$, $\mathbf{B}$ in (2.9) are not known to the power system operator. Only a nominal pair $(\mathbf{A_0}, \mathbf{B_0})$ is known. Typical uncertainties in $\mathbf{A}$, $\mathbf{B}$ in an actual power system may result from various unknown parameters such as inertias of the generators, especially when power electronic converters are added resulting in low-inertia equivalents [33], or exact values of the line reactances, especially when series compensation may be used in long transmission lines for certain unforeseen contingencies, or even the internal time constants of the generator circuits, uncertainties in load dynamics and associated load control mechanisms, unpredictable plug-and-play dynamics of converters, intermittent generation profiles of hundreds of wind, solar and storage devices, and so on.

We refer to the LQR controller based on the nominal model $\mathbf{A_0}, \mathbf{B_0}$ as the *mismatched*

LQR $\mathbf{K}_{mis}$. When the actual $\mathbf{A}, \mathbf{B}$ deviate significantly from the nominal model matrices $\mathbf{A_0}, \mathbf{B_0}$, the performance of the mismatched LQR controller suffers and can even become unstable as illustrated in section 4. Thus, we investigate RL under constrained communication cost given the knowledge of the nominal model $\mathbf{A_0}, \mathbf{B_0}$, but not of the actual model $\mathbf{A}$, $\mathbf{B}$.

2.3 RL control for WAC

Reinforcement learning has been proposed as a tool to implement optimal control for unknown or uncertain systems [40]. A combination of Q-learning and adaptive dynamic programming is proposed in [70], which provides an actor-critic structure capable of learning the optimal control policy for completely unknown, continuous-time dynamic systems using value iteration. This algorithm is implemented online using state and control input measurements of the system. Unlike a general RL problem, where $\mathbf{K}$ is learnt online starting from any arbitrary initial guess, uncertainties in power systems are typically not that unstructred and drastic. This means that if $(\mathbf{A_0}, \mathbf{B_0})$ is the model during one contingency, and $(\mathbf{A_1}, \mathbf{B_1})$ is the model during a contingency that occurs within a few hours, then most probably only a few entries of $(\mathbf{A_1}, \mathbf{B_1})$ will be different from those of $(\mathbf{A_0}, \mathbf{B_0})$ as only a few line and generator parameters may have changed between the two events. The initial guess for $\mathbf{K}$ can therefore be picked as the controller from the previous contingency. If the difference between the models is indeed not significant, then this choice would expedite the convergence of this loop considerably. As the discrepancy between the two models grows, choosing $\mathbf{K}$ corresponding to the nominal model $(\mathbf{A_0}, \mathbf{B_0})$ still increases the convergence speed of RL significantly relative to a random guess as will be illustrated in section 4. The notation used in this section is summarized in Table 2.1.

Table 2.1: Notation used in Algorithm 1.

Term	**Definition**
α_c	Critic convergence speed coefficient.
α_a	Actor convergence speed coefficient.
$\mathbf{U} = [\mathbf{x}^T \mathbf{u}^T]^T$	Concatenated vector of states and control inputs.
$\Phi(t) = \mathbf{U}(t) \otimes \mathbf{U}(t)$	Quadratic basis vector of states and control inputs.
$\sigma = \Phi(t) - \Phi(t-T)$	Change in the basis vector after time-step T.
$\mathbf{G} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{21} & \mathbf{G}_{22} \end{bmatrix}$	Kernel $\mathbf{G} \in \mathbb{R}^{(n+m)\times(n+m)}$
$\mathbf{W} = vech(Qf)$	Critic vector
$\mathbf{K}_{SP}$	Actor vector
$e_c = \mathbf{W}^T\sigma + \int_{t-T}^{t}(\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u})d\tau$	Critic error
$e_a = \hat{\mathbf{K}_{SP}}^T\mathbf{x} + \hat{\mathbf{G}}_{22}^{-1}\hat{\mathbf{G}}_{21}\mathbf{x}$	Actor error
$\mathbf{W} = -\alpha_c \frac{\sigma}{(1+\sigma^T\sigma)^2} {e_c}^T$	Critic update
$\mathbf{K}_{SP} = -\alpha_a \mathbf{x} {e_a}^T$	Actor update
$Qf(x,u) = \frac{1}{2}\mathbf{U}^T\mathbf{G}\mathbf{U}$	Q-function
$\mathbf{u}(t) = \arg\min_u Qf(\mathbf{x},\mathbf{u}) = -\mathbf{G}_{22}^{-1}\mathbf{G}_{21}x$	Optimized control input
vech(.)	Half vectorization operator, stacks elements of the upper triangular part of a matrix into a vector, multiplying diagonal elements by 2.
$\|\mathbf{K}\|_2$	Frobenius norm of the matrix $\mathbf{K}$, defined by trace($\mathbf{K}^T\mathbf{K}$).

Table 2.1 (continued).

$\text{supp}(\mathbf{K})$	The support set of the matrix $\mathbf{K}$, i.e., the set of indices of the nonzero entries of matrix $\mathbf{K}$.
$[\mathbf{K}]_s$	The matrix obtained by preserving only the s largest-magnitude entries of the matrix $\mathbf{K}$, and setting all other entries to zero.
$\nabla_{\mathbf{K}}(\|\|e_a\|\|^2)$	The gradient of $(\|\|e_a\|\|^2)$ w.r.t $\mathbf{K}$. Assuming $\mathbf{K} \in \mathbb{R}^{m\times n}$, $\nabla_{\mathbf{K}}(\|\|e_a\|\|^2)$ is given by $m \times n$ matrix with the elements $[\nabla_{\mathbf{K}}(\|\|e_a\|\|^2)]_{ij} = \partial \|\|e_a\|\|^2 / \partial \mathbf{K}_{ij}$.
$\Delta_{nwt}(\mathbf{K}, \tau)$	The restricted Newton step of an arbitrary function $f(\mathbf{K})$ at matrix $\mathbf{K} \in \mathbb{R}^{m\times n}$ under structural constraint $\text{supp}(\mathbf{K}) \subset \tau$. First, all elements of $\mathbf{K}$ is stacked in vector $\mathbf{x}$, and the function $g(\mathbf{x})$ is defined as $g(\mathbf{x}) \triangleq f(\mathbf{K})$. Then the $mn \times 1$ restricted Newton step vector $\Delta_{nwt}(\mathbf{x}, \tau)$ of $g(\mathbf{x})$ at $\mathbf{x}$ is computed using the conjugate gradient method [50]. The vector $\Delta_{nwt}(\mathbf{x}, \tau)$ is then converted back to $m \times n$ matrix by stacking consecutive $m \times 1$ segments of $\Delta_{nwt}(\mathbf{x}, \tau)$.

2.3.1 Sparsity-constrained WAC using RL

First, we briefly review the RL algorithm in [70]. The two-step learning iterative process starts from randomly generated actor and critic vectors and iterates until these vectors converge to their optimal values. The critic approximator is responsible for estimation of the Q-function while the actor approximator aims to find the optimal control policy. In

each step, the actor selects a control policy ($\mathbf{K}_{RL}$) and applies $\mathbf{u} = -\mathbf{K}_{RL}\mathbf{x}$ to the unknown system (2.9) in real-time. The critic evaluates the performance of the control policy applied by the actor using state and control input measurements. This evaluation is then used by the actor to update its control policy. This process continues until the actor update results in unchanged $\mathbf{K}_{RL}$ within a desired amount of error.

The critic and actor update rules in [70] are implemented by gradient descent and do not require the knowledge of the system model. Thus the algorithm in [70] converges to the optimal LQR controller for a completely unknown system. However, this method has long convergence time. Since in power systems partial system knowledge is available to the designer, we employ the known matrices $\mathbf{A_0}, \mathbf{B_0}$ of the nominal model to initialize the RL algorithm, which can reduce the convergence time. Note that using $(\mathbf{A_0}, \mathbf{B_0})$, however, does not classify Algorithm 1 as a model-dependent algorithm. The algorithm is still model-free; it is only *assisted* by the knowledge of the nominal model so that one can expedite the learning phase. In addition, to add sparsity in $\mathbf{K}$, we propose the control design problem as follows:

$$\begin{aligned} \min_{\mathbf{K}} \quad & J(\mathbf{K}) \\ \text{s.t.} \quad & \text{Card}_{\text{off}}(\mathbf{K}) \leq s \\ & \mathbf{x}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{x}(0) = \mathbf{x}_0 \\ & \mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t) \end{aligned} \tag{2.13}$$

where $\mathbf{A}$ and $\mathbf{B}$ are unknown matrices. The constrained optimization (2.13) results in a sparse $\mathbf{K}$ matrix denoted by $\mathbf{K}_{SP}$, which has at most s communication links. We employ GraSP algorithm [8] in the actor update step of [70] instead of gradient descent to impose this

constraint in RL. GraSP was shown to find sparse solutions for a wide class of optimization problems. The details of the RL controller for WAC given the sparsity constraint are provided in Alg. 1.

The disturbance happens in step 1, followed by all nodes sending their state information to the central controller in step 2. In step 3, the initial critic and actor vectors are generated. First, we solve the Riccati equation for the nominal model $\mathbf{A_0}, \mathbf{B_0}$ to find positive definite $\mathbf{P}_0$

$$\mathbf{A}_0^T\mathbf{P} + \mathbf{P}\mathbf{A}_0 - \mathbf{P}\mathbf{B}_0\mathbf{R}^{-1}\mathbf{B}_0^T\mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{2.14}$$

and then we form the kernel $\mathbf{G}$

$$\mathbf{G} = \begin{bmatrix} \mathbf{P}_0\mathbf{A}_0 + \mathbf{A}_0^T\mathbf{P}_0 + \mathbf{Q} + \mathbf{P}_0 & \mathbf{B}_0\mathbf{P}_0 \\ \mathbf{P}_0^T\mathbf{B}_0 & \mathbf{R} \end{bmatrix} \tag{2.15}$$

Next, in step 3 of Alg. 1, using $\mathbf{U}_0^T = [\mathbf{x}_0^T \ \mathbf{u}_0^T]$, we find the Q-function and $\mathbf{W}_0$

$$\mathbf{W}_0 = vech(Qf(\mathbf{x}_0, \mathbf{u}_0)) \tag{2.16}$$

The initial actor matrix $\mathbf{K}_{SP}^0$ is given by [70]

$$\mathbf{K}_{SP}^0 = \mathbf{R}^{-1}\mathbf{B}_0^T\mathbf{P}_0 \tag{2.17}$$

The actor update loop begins in step 4 and extends until end of the Algorithm 1, where ϵ_K is the desired actor error. The control input is calculated in step 5. Steps 6 and 7 implement addition of the exploration noise, where T_{PE} indicates the duration of the exploration noise $\mathbf{u}_{PE}$, chosen to provide sufficient persistence of excitation for the convergence of the critic approximator. The noise signal $\mathbf{u}_{PE}(t)$ is given by the sum of sinusoids with a sufficient

Algorithm 1 RL for WAC under sparsity constraint s

1: Time of disturbance: $t = 0$, iteration $i = 0$.
2: All nodes send states to central controller.
3: Central controller receives $\mathbf{x}_0$ and forms initial critic $\mathbf{W}^0$ (2.16) and actor $\mathbf{K}_{SP}^0$ (2.17).
4: **while** $\|\mathbf{K}_{SP}^{i+1} - \mathbf{K}_{SP}^{i}\|_2 \geq \epsilon_K$ **do**
5: Calculate control input $\mathbf{u}(t)$ (2.11)
6: **if** $t \leq T_{PE}$ **then** ▷ add exploration noise
7: $\mathbf{u}(t) \longleftarrow \mathbf{u}(t) + \mathbf{u}_{PE}(t)$
8: **end if**
9: Apply $\mathbf{u}(t)$ to the system $\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ for T sec.
10: All nodes send state and input measurements to central controller.
11: **if** $\|\mathbf{W}^{i+1} - \mathbf{W}^{i}\|_2 \geq \epsilon_W$ **then**
12: Update critic vector ($\mathbf{W}$) (Table 1)
13: **end if**
14: calculate the gradient of $e_a \rightarrow g = \nabla_{\mathbf{K}_{SP}^i}(\|e_a\|^2)$
15: Keep $2s$ largest entries of g $\rightarrow Z = \text{supp}([g]_{2s})$
16: Merge $\text{supp}(\mathbf{K}_{SP}^i)$ and Z $\rightarrow \tau = Z + \text{supp}(\mathbf{K}_{SP}^i)$
17: Descend based on restricted Newton step $\rightarrow \mathbf{K}_{SP}^{i+1} = \lambda \Delta_{nwt}(\mathbf{K}_{SP}^i, \tau)$
18: Prune extra entries: keep s largest elements of $\mathbf{K}_{SP}^{i+1}$ and set others equal to zero $\rightarrow \mathbf{K}_{SP}^{i+1} = [\mathbf{K}_{SP}^{i+1}]_s$
19: $i \longleftarrow i + 1$
20: **end while**
21: **return** $\mathbf{K}_{SP}^{i+1}$ ▷ $\mathbf{K}_{SP}^{i+1}$ is sparse with s non-zero elements

number of frequencies [70]. In step 8, the control input $\mathbf{u}(t)$ is applied to the system, and the resulting state and input measurements are sent to the central controller in step 9. Steps 11 and 12 update the critic as depicted in Table 1. Since the critic must converge faster than the actor, we choose the learning rate of the critic to be much greater than that of the actor ($\alpha_c \gg \alpha_a$). Moreover, ϵ_W is the desired error for the critic [70]. Steps 14-18 of Alg. 1 implement GraSP on the update of actor, which limits the directions in which this update is performed [8]. In each iteration, $\mathbf{K}_{SP}$ is extended along its $2s$ steepest gradient-descent directions (step 15). Then the set of descent directions is created by merging the indices of non-zero elements of $\mathbf{K}_{SP}$ found in the previous iteration and $2s$ largest elements of the gradient. As a result, the direction of descent will have at most $3s$ elements (16). After

descent based on these directions in step 17, an ℓ_0-norm is applied to the resulting $\mathbf{K}_{SP}$ to remove the extra entries and ensure $\text{Card}_{\text{off}}(\mathbf{K}_{SP}) = s$ (step 18).

The convergence time T_c of SRL in Alg. 1 is defined as

$$T_c = i\,T, \quad i = \text{ smallest integer s.t. } \|\mathbf{K}_{SP}^{i+1} - \mathbf{K}_{SP}^{i}\|_2 < \epsilon_K \tag{2.18}$$

2.3.2 Timeline and Cyber-Physical Implementation

Fig. 2.1 shows the Timeline, while Fig. 2.2 shows the closed-loop configuration as a cyber-physical system (CPS) and the stages of design and implementation of the proposed algorithm. As shown in Fig. 2.1, in stage 1, the learning Algorithm 1 finds a sparse controller ($\mathbf{K}_{SP}$) suitable for damping oscillations of the actual system. In stage 2, we apply $\mathbf{K}_{SP}$ right after learning it to damp oscillations. We also apply it at t_4, when new disturbances occur, assuming the system model remains unchanged. This assumption is based on typical power system conditions for disturbances within a short time interval of each other. In stage 1, centralized computation is carried out to learn the controller as shown in Fig. 2(a). Stage 1 consists of two phases. In *phase 1* (steps 4 to 18 of Alg. 1), both the critic and the actor estimators are updating. While the actor is sparse with the sparsity constraint s, its sparsity pattern can change, i.e. the indices of non-zero elements in $\mathbf{K}_{SP}$ might change. When the critic converges to its final value (sooner than the actor), the *phase 1* finishes, and the structure of the sparsity-constrained actor is fixed.

In *phase 2*, only the actor parameters are updated while the sparsity pattern, which determines the communication graph between the nodes of the plant, is fixed. Phase 2 extends over steps 4-18, but excludes steps 10-11, i.e. the critic vector update. Finally, Fig. 2.2(b) illustrates stage 2. It shows a sparse controller produced by the learning algorithm for a typical multi-machine grid. In this case, the sparse feedback gain matrix $\mathbf{K}_{SP}$ is fixed

and known to each generator. Thus there is no need for the central controller, and each generator can compute its own control input u_i using state measurements from other generators and $\mathbf{K}_{SP}$, thus implementing eq. (2.11) in a distributed fashion.

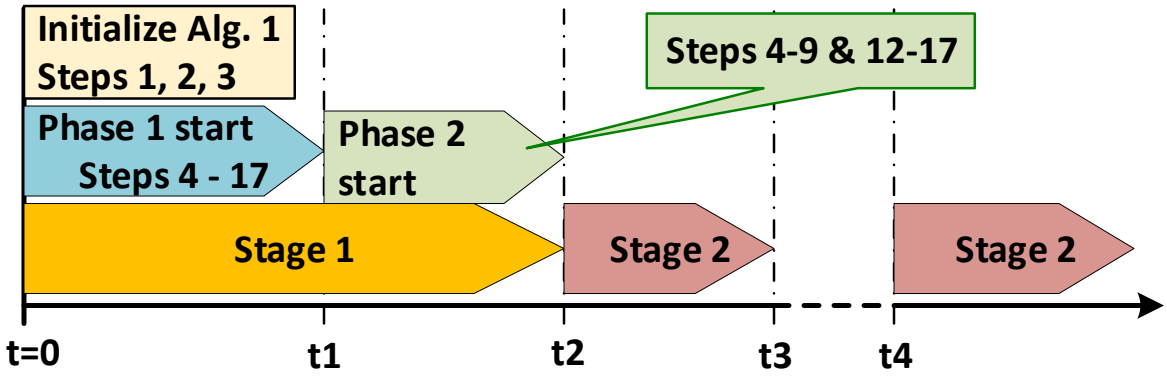


Figure 2.1: Timeline for Alg. 1.

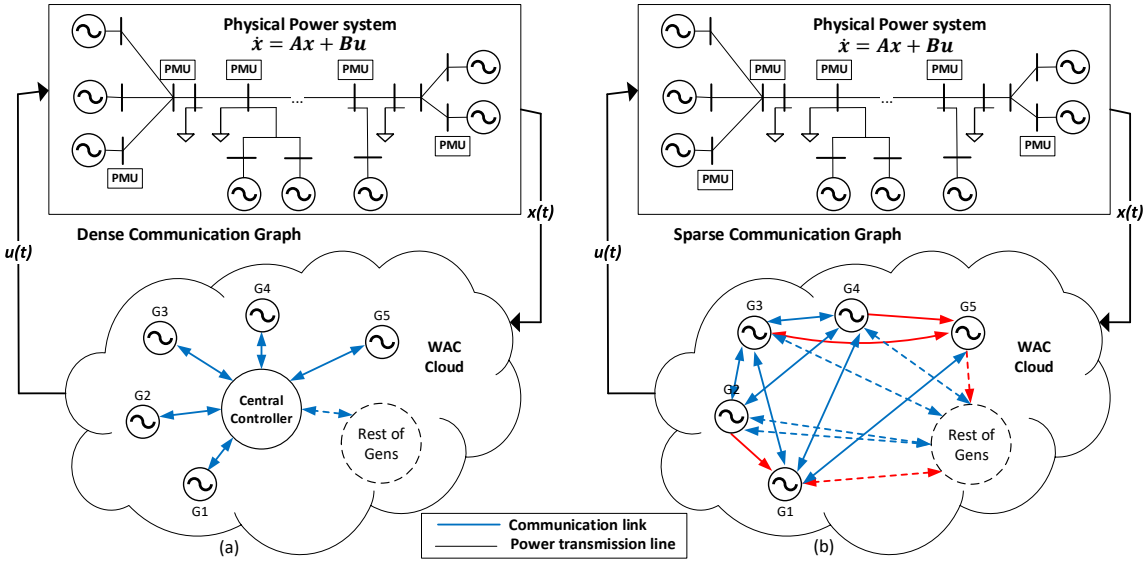


Figure 2.2: Stages of the learning algorithm and implementation of the sparse wide-area controller; t_2 is the convergence time of stage 1. (a) Learning of $\mathbf{K}_{SP}$, $t = 0$ to t_2 (Stage 1). (b) Implementation of the sparse wide-area controller $\mathbf{K}_{SP}$ (Stage 2). The blue and red lines indicate the required links for dense communication while only the red lined are required for sparse implementation of WAC. The dashed lines indicate communication links between indicated generators and remaining generators which are illustrated by a dashed circle. Hence, the dashed line may actually be more than one link.

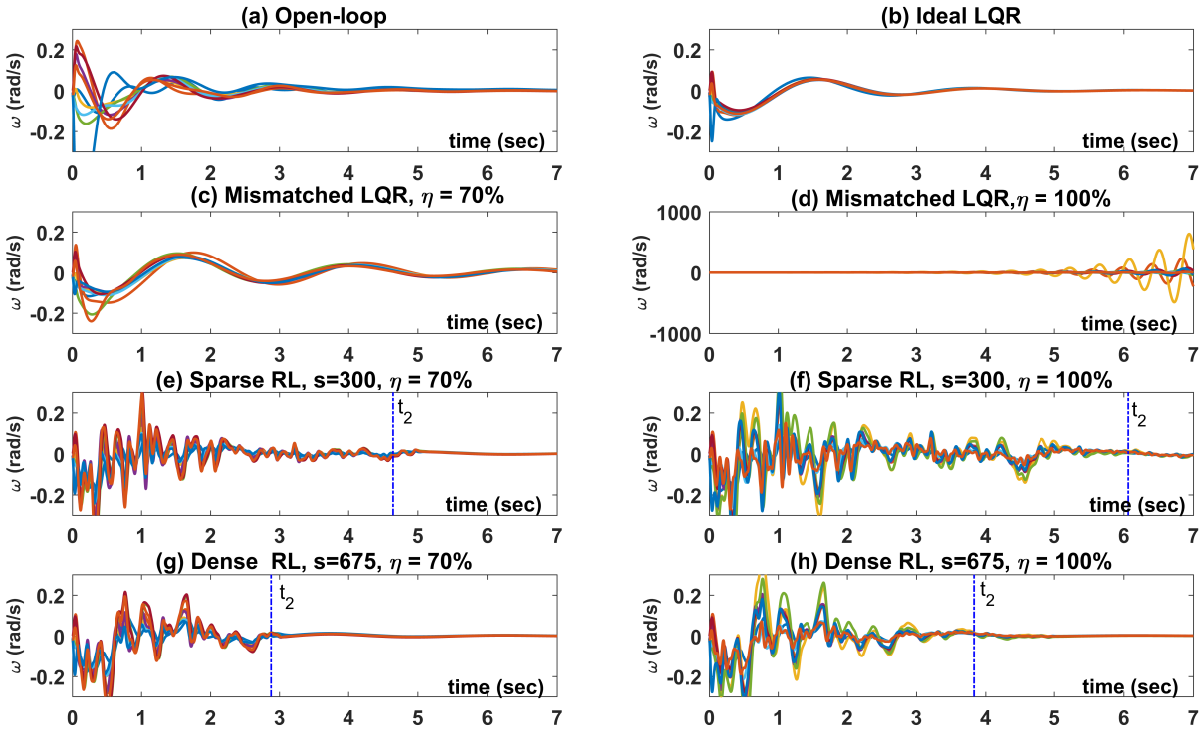


Figure 2.3: Performance of different controllers in damping oscillations; η indicates the uncertainty level, and t_2 is the end of stage 1 (learning).

2.4 Numerical Results

The IEEE 39-bus model of New England power system is used in this section to study the effectiveness and performance of the proposed algorithm. This model consists of 10 synchronous generators. The state-space model in the form of (2.11) has the state vector of size 75, i.e. $\mathbf{A} \in \mathbb{R}^{75\times 75}$ and 9 control inputs, resulting in $\mathbf{B} \in \mathbb{R}^{75\times 9}$ and $\mathbf{K} \in \mathbb{R}^{9\times 75}$. The details of the model can be found in [25]. The feedback gain matrix has 675 entries. Following the discussion in section 2.2.3, the uncertainty of the power system model is limited to the parameters corresponding to inertias of the machines and line reactances, which results in 1132 uncertain parameters in $\mathbf{A}$. Moreover, $\mathbf{B}$ matrix has 9 non-zero entries with known indices and uncertain values. The level of uncertainty in both $\mathbf{A}$ and $\mathbf{B}$ is measured by the parameter η. To generate uncertain parameters, we randomly vary the corresponding entries of $\mathbf{A}, \mathbf{B}$ using the uniform distribution within η% of their original values in $\mathbf{A}_0$ and $\mathbf{B}_0$ matrices, respectively. Finally, the duration of the exploration noise T_{PE} in step 6 of Algorithm 1 is 2 seconds, and the error terms ϵ_K and ϵ_W in steps 4 and 10 are equal to 10^{-5}.

Several WAC scenarios are simulated using two different η values of 70% and 100%. We compare four controllers: the *ideal* LQR controller ($\mathbf{K}_{lqr}$), designed assuming the knowledge of the actual system model $\mathbf{A}, \mathbf{B}$; the *mismatched* LQR controller ($\mathbf{K}_{mis}$), designed for the nominal system $\mathbf{A}_0$, $\mathbf{B}_0$ and applied to the actual system $\mathbf{A}, \mathbf{B}$; the *dense learning* controller ($\mathbf{K}_{dense}$) found by Algorithm 1 for the maximum value of $s = 675$; the *sparse learning* controller ($\mathbf{K}_{SP}$) designed using Algorithm 1 for a range of s values ($100 \leq s \leq 675$). Moreover, performance of the open-loop system controlled by local PSS (power system stabilizers), designed for $\mathbf{A}, \mathbf{B}$, is shown for comparison. Note that although by definition self-links does not incur any communication cost, the algorithm is capable of removing them if s is sufficiently small.

As described earlier, we assume that only the nominal model $\mathbf{A}_0$, $\mathbf{B}_0$ is known to the

designer. Modeling the fault as an impulse input that is cleared at $t = 0$, we start the simulation from the initial state $\mathbf{x}(0)$ at $t = 0$. Figure 2.3 shows the time response of the generator speeds for the different controllers and uncertainty levels. The convergence time of the learning algorithm (end of stage 1) is denoted as t_2 in Fig. 2.3 (e-h). Note that when $\eta = 100\%$, the *mismatched* LQR designed for the nominal model becomes unstable while the RL controller suppresses oscillations in both sparse and dense cases. The convergence time of Algorithm 1 increases with the level of uncertainty for both sparse and dense controllers.

Next, we apply $\mathbf{K}_{SP}$ learned in stage 1 (fig 2.2(a)) to damp the oscillations caused by new disturbances (see fig 2.1) for both uncertainty levels. Figure 2.4 indicates the rotor speed of the generators in this scenario. Note that the mismatched LQR is still unstable when $\eta = 100\%$ while $\mathbf{K}_{SP}$ successfully damps the oscillations for both s-values. We found that $\mathbf{K}_{SP}$ does not depend on the initial condition $\mathbf{x}_0$ at $t = 0$. Therefore, $\mathbf{K}_{SP}$ designed by Alg. 1 performs well for any incoming disturbance.

The performance of the controllers in terms of the increase in the closed-loop energy J in (2.10) compared to the *ideal* LQR is reported in Table 2. The J-value is calculated from $t = 0$ (when the initial disturbance happens) till $t = 10$ seconds when the oscillations are practically damped. Note that in this case the overall energy is not significantly affected by learning due to relatively short duration of Stage 1 when the nominal knowledge of the system is used. It can be seen that the values of J in figures 2.3 and 2.4 are comparable for fixed values of η.

When Algorithm 1 does not employ the sparsity constraint and the nominal model, it is identical to the RL method in [70], which was proven to converge to the optimal stable solution of LQR (Theorem 2, [70]). Addition of the nominal model does not change the convergence result since the latter applies to any initial control matrix including (2.17). However, sparse control problems elude theoretical convergence guarantees [25, 47]. If an ideal closed-form solution to (2.13) was available for known system matrices (similar to the

Riccati equations (2.14) for the LQR problem), the arguments of [70] would apply to prove convergence of the sparsity-constrained RL design to this ideal solution.

In practice, even when the system matrices $\mathbf{A}, \mathbf{B}$ are known, convergence of GraSP is not assured for a given value of s [47], implying that Algorithm 1 of this chapter does not necessarily converge. However, if it converges, step (15) guarantees that $\mathbf{K}_{SP}$ satisfies the following property

$$\nabla_{\mathbf{K}}(\|e_a\|^2)|_{\text{supp}(\hat{\mathbf{K}}_{SP})}(\mathbf{K}_{SP} = \hat{\mathbf{K}}_{SP}) = 0, \tag{2.19}$$

which is the weak necessary condition for the optimality of (2.13) [10]. Thus, the feedback matrix $\mathbf{K}_{SP}$ at convergence of Algorithm 1 corresponds to the global or a local minimum of the optimization problem (2.13). Extensive numerical studies show that the proposed algorithm converges in many practical scenarios for all uncertainly values and sparsity ranges $s = 100-675$, and the closed-loop system is observed to be stable.

As expected, the value of the objective function (2.10) increases as the sparsity level grows (decreasing s). This is due not only to sub-optimality of the sparse controller, but also to increased duration of learning, and thus more persistent oscillations for lower s values.

Table 2.2: Increase in J-values compared to ideal LQR.

Figure	2.3	2.3	2.4	2.4
η	70%	100%	70%	100%
Open-loop J	1470%	1470%	221.49%	221.49%
Mismatched LQR J	13.68%	∞	16.55%	∞
Sparse RL J $s = 300$	6.84%	11.43%	5.14%	6.87%
Dense RL J $s = 675$	1.8%	3.27%	0.23%	0.23%

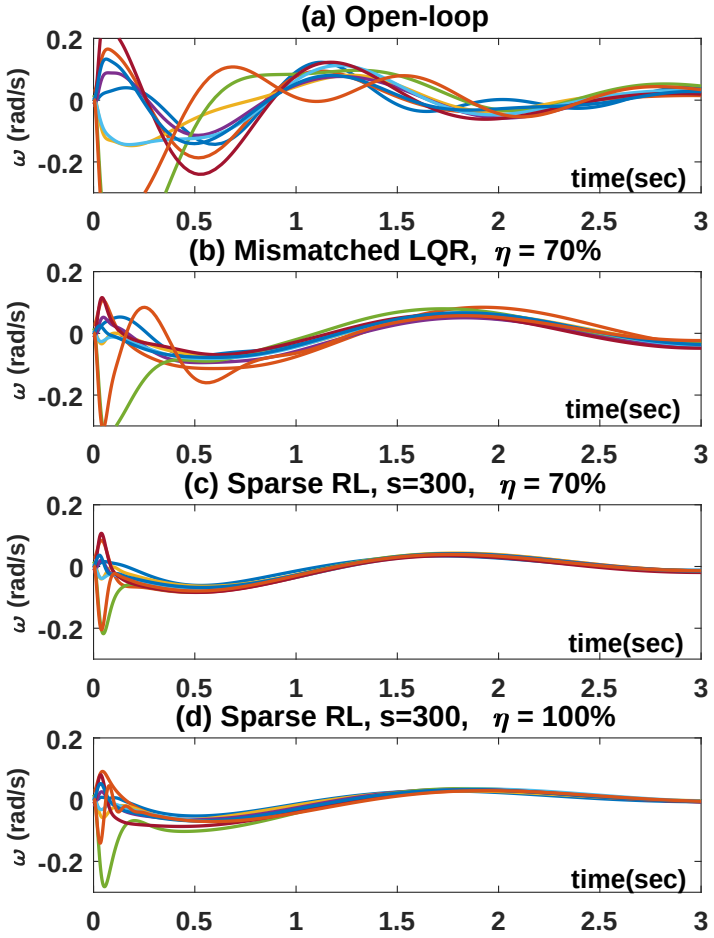


Figure 2.4: Performance for future disturbances using previously learnt controller; η indicates the uncertainty level .Note that the mismatched LQR becomes unstable for $\eta = 100\%$ (not shown).

Finally, the convergence time of Alg. 1 depends on two factors. First, it increases with the uncertainty level η. As the deviation between the nominal model and the actual model increases, the convergence time also increases sharply since the initial critic and actor do not approximate their optimal values closely for large η-values. Moreover, decreasing the sparsity constraint s results in slower convergence. For $s < 100$, the convergence time increases dramatically, thus precluding utilization of extremely sparse controllers in WAC when employing RL. If Alg. 1 was initialized with randomly generated actor and critic as in

[70], the convergence time would be on the order of hours and thus exceed significantly the acceptable range for WAC applications. Moreover, the exploration noise duration would increase greatly as well, adding extra oscillations to the system and degrading the quality of service during that period. Hence, using the knowledge of the nominal model to initialize the RL algorithm enables its application in WAC.

2.5 Conclusions

We proposed a sparse wide-area control design for power systems using online, data-driven reinforcement learning. First, the convergence time was reduced significantly by using the knowledge of the nominal model. Second, the communication cost of WAC was reduced by sparsifying the controller using the GraSP method. The effectiveness of the proposed controller was illustrated on the IEEE New England power system model with uncertain parameters. It was demonstrated that the proposed sparse controller successfully damps the wide-area oscillations even for highly uncertain models while the LQR controller matched to the nominal model destabilizes the system as the uncertainty level grows.

CHAPTER

3

EARLY WARNING OF MMWAVE SIGNAL BLOCKAGE USING DIFFRACTION PROPERTIES AND MACHINE LEARNING

Sensitivity to blockage challenges performance of millimeter-wave (mmWave) communication systems. We apply the MiniRocket machine learning (ML) method to provide reliable early warning of mobile mmWave signal blockage hundreds of milliseconds ahead, thus facilitates a proactive response. MmWave signal datasets for training and testing the ML method are created using our low-complexity physics-based simulation tool, which models

diffraction accurately. Our insights and numerical results illustrate that the proposed early warning method is facilitated by the diffraction-induced pre-blockage signal patterns and is robust to diverse environmental and mobility conditions.

3.1 Introduction

MillimiterWave (mmWave) is an integral part of the 5G and beyond cellular communications technology. However, there are many challenges to successful deployment of mmWave communications. First, these systems rely heavily on line-of-sight (LoS) links for sufficient received signal power. Second, due to weaker diffraction and higher penetration loss, mmWave signals are more susceptible to blockage by physical objects than sub-6 GHz signals [62]. Such blockages can degrade the performance severely in environments with moving obstacles [68, 4, 9].

In this paper, we apply the MiniRocket Machine Learning (ML) time-series classifier method [22] to provide early warning (EW) of LoS blockage. The proposed method employs only in-band mmWave received signal strength (RSS) samples to forecast blockages hundreds of milliseconds, or tens of 5G frames [74], ahead in scenarios with mobile blockers and user equipment (UE), thus enabling proactive response to an upcoming blockage, e.g., base station (BS) or beam switching.

A dataset of mobile mmWave scenarios, including LoS received signal as well as LoS-to-non-line-of-sight (NLoS) transitions, is generated using our physical model [31, 54, 26, 3], which employs Fresnel diffraction and method of images to calculate the received signal. These scenarios and physical insights demonstrate unique diffracted signal patterns associated with approaching blockages. Our extensive simulation results demonstrate that the ML method is able to identify features of the approaching blockages and thus provides accurate early warning of blockage in various scenarios. An important observation

is that this capability is due to physical properties of diffraction and does not rely on specific environmental assumptions. Thus, a single training and testing dataset works for many scenarios. We show that distances, speeds, directions, and other environmental factors alter the diffracted signal pattern in well-defined ways, but both the large training sets permitted by model generation and choice of ML technique make the early warning generally applicable. Moreover, we show that the proposed method is robust to additive noise and multipath fading and is feasible for various channel sampling rates.

Related Work Blockage prediction assisted by sub-6 GHz links was investigated in [3, 56, 5]. A method based on recurrent neural networks (RNN) used BS handoff data for a static environment to predict LoS blockage [4]. Blockage of neighboring paths in an environment with slowly moving receivers and obstacles was used to predict the mmWave blockage [9]. A deep convolutional model that used additional sensor information from cameras was employed in [16, 59] for LoS blockage prediction. Simulated data and DL were used to predict mmWave blockage in indoor scenarios with limited speeds of obstacles [14]. A meta-learning framework and RNNs were proposed to predict mmWave link blockage using Rician fading channel model for fixed BS and UE scenarios [39].

In contrast to the above methods, our approach does not assume a specific environment (indoor, outdoor, static, fixed topology), obstacle motion, direction, or speed and does not use simplistic simulation models. Moreover, it solely relies on in-band mmWave signal measurements and does not require corroborating information from sub-6 GHz signals or additional sensors, such as cameras. Finally, our ML solution uses the extremely fast MiniRocket [22] time-series modeling technique with much lower computational complexity than DL and RNN models and does not incur high computation costs while finding the solution in real-time [22].

The physical model employed in this paper [31, 54, 26, 3] accurately captures frequency-dependent diffraction, uses full Fresnel formalism to avoid quantization onto rays, includes

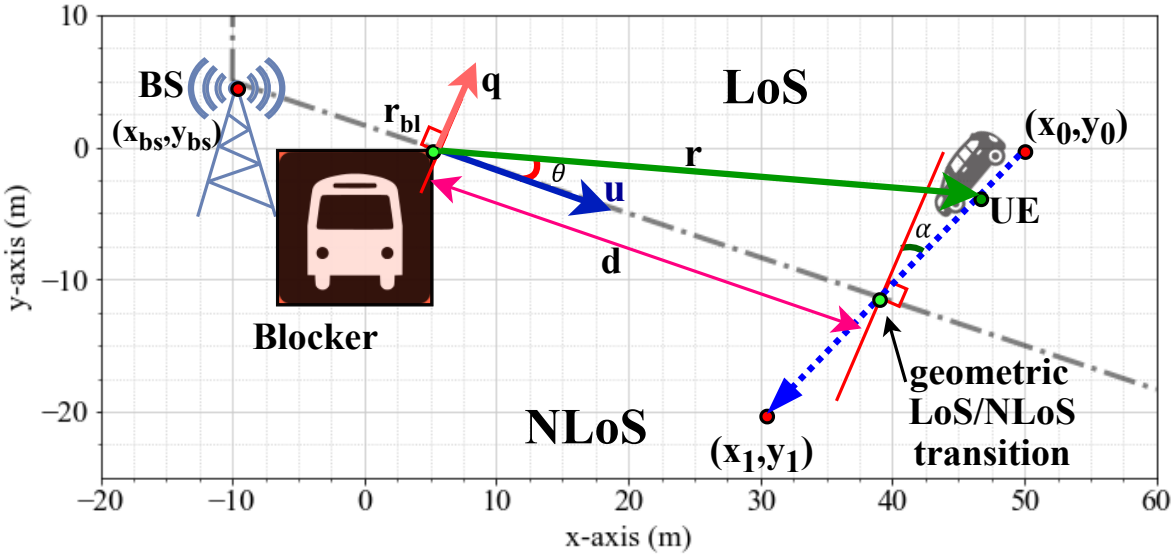


Figure 3.1: Example simulation scenario for the physical model. Stationary BS = (-10,5) *m* and blocker = (5,0) *m*; UE moving from (50,0) to (30,-20) *m*.

diffraction modification to the non-reflected component, and models small reflectors. These features are not addressed in ray tracing [41] or enhanced ray tracing [73].

The physical model enables us to generate a large and diverse dataset of realistic mobile scenarios to train the proposed ML method inexpensively rather than relying solely on costly and time-consuming measurements. The key features (oscillations) present in the diffraction pattern prior to blockage are not predicted by Fraunhoffer diffraction, but instead require Fresnel diffraction, which is valid when the LoS/NLoS transition approaches [29]. Our model is used to show how these oscillations vary in a variety of scenarios to obtain insights into the training sets needed for accurate and robust ML.

The main contributions of the paper are: 1) We propose and validate a novel, fast, robust, and accurate ML approach for early warning of upcoming LoS blockage for mobile mmWave systems using in-band observations. 2) We employ an accurate, low-complexity physical model of mmWave propagation to train and test the proposed ML early warning method

and provide insights into the RSS diffraction-induced oscillation patterns, which warn about approaching blockages.

The rest of the paper is organized as follows. Section 3.2 summarizes the physics-based model and the diffraction-induced pre-blockage patterns. The ML method for early warning of mmWave blockage is presented in Section 3.3. Simulation results are discussed in Section 3.4, and Section 3.5 concludes the paper.

3.2 Physics-Based Channel Model

We use our physical model [31, 54, 3, 30], based on the Fresnel diffraction and method of images to model and simulate transitions from LoS to NLoS in the received down-link signal. The model calculates the corresponding equivalent lowpass spatial coefficient $h(x, y)$ point-by-point along a given mobile trajectory in the defined calculation plane. For a mobile receiver moving on a straight line starting from (x_0, y_0) at $t = 0$ with constant velocity of $\mathbf{v} = (v_x, v_y)$, the spatial coefficient can be converted into time to produce an equivalent lowpass received signal

$$h(t) = h(x(t), y(t)) = h\left(x_0 + v_x t, y_0 + v_y t\right). \tag{3.1}$$

In this paper, $h(t)$ represents the direct (non-reflected, but possibly diffracted) received signal component.

An example scenario is depicted in Fig. 3.1. The BS is at (x_{bs}, y_{bs}), and the bus acting as the blocker at (x_{bl}, y_{bl}) is implemented as an edge for diffraction in the physical model [3]. The UE is a car moving from (x_0, y_0) in the LoS region to (x_1, y_1) in the NLoS region along the straight dotted line. Equivalently, the blocker could move and block the LoS path from BS to UE. The RSS $|h(t)|$ of this scenario is plotted in Fig. 3.2. More generally, both the UE

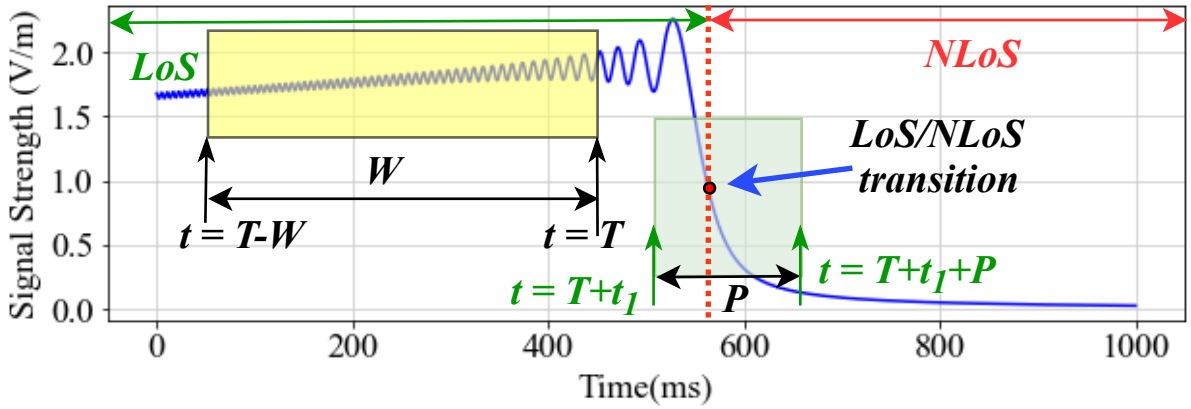


Figure 3.2: MmWave RSS $|h(t)|$ along the UE path from (x_0, y_0) to (x_1, y_1) in Fig. 3.1. Examples of the observation window W, prediction window P, and the LoS/NLoS transition are shown in ms with UE speed $v = 28\ m/s$, $\alpha \approx 26.6°$, $d \approx 35.6\ m$.

and blocker can move in arbitrary speeds and directions, and all distances and angles vary to produce diverse model scenarios of LoS as well as LoS/NLoS transition as described in section 3.3.2.

Details on the model parameters and the code are available in [3, 30]. To create a dataset, we specify the locations of the BS, the blocker edge at point $\mathbf{r}_{bl}$ and along a line perpendicular to the page, and the UE. Each of these can move along their own specified line segment during the simulation. The blocker edge at $\mathbf{r}_{bl}$ defines a Fresnel diffraction [29] for this blocker, and the dash-dotted line containing $\mathbf{r}_{bl}$ and BS specifies the geometric LoS/NLoS transition along which the RSS is reduced by a factor of two from its value in the absence of the blocker [3, 29].

Prior to the LoS/NLoS transition, the received signal exhibits the diffraction-induced pattern evident in Fig. 3.2. Oscillations grow in amplitude and decrease in frequency, then stop prior to the rapid signal decrease near the geometric LoS/NLoS transition. Insights into these patterns can be obtained from the physical model equation for the signal (3.1) at $\mathbf{r}_{\mathrm{UE}} = (x_{\mathrm{UE}}, y_{\mathrm{UE}})$:

$$h(\mathbf{r}_{\mathrm{UE}}) = 0.5(j-1)e^{j(2\pi r f_c/c)} F r(w)/|\mathbf{r}|. \tag{3.2}$$

The Fresnel Integral Fr is given by

$$\begin{gathered} Fr(w) = 0.5 - C(w) - 0.5j + jS(w), \text{ where} \\ C(w) = \int_0^w \cos(\frac{\pi\xi^2}{2})d\xi \text{ and } S(w) = \int_0^w \sin(\frac{\pi\xi^2}{2})d\xi. \end{gathered} \tag{3.3}$$

The argument of the Fresnel integral is $w = \sqrt{2f_c/c\rho}(q_1 - q_0)$, where c is the speed of light, ρ an effective distance given by (5) in [3], and q_0, q_1 are coordinates along the direction q in Fig. 3.1. The point q_0 specifies the intersection of the line from BS to UE with the q axis, and q_1 refers to the blocker edge at $\mathbf{r}_{bl}$. The oscillation pattern originates from the sinusoids in (3.3). We found that the specific features of this pattern depend strongly on the carrier frequency f_c, distance of the UE path from the blocker edge d, and angle α from normal at which the UE path crosses the geometric transition line, shown in Fig. 3.1. In this paper, we assume the mmWave frequency is $f_c = 30$GHz, but our results are applicable to other frequencies. Besides a phase factor from the overall propagation distance, frequency only appears in w in (3.2, 3.3). To reach the same point (e.g., first maximum) in the diffraction pattern measured in time from the geometrical transition point, we need to reach the same value of w. Since ρ is approximately constant, when the time t is approximated using $vt = \zeta(q_1 - q_0)$, ζ a constant, in the above formula for w, the timing t of the pattern of oscillations measured from the geometric transition scales as the inverse square root of the frequency, $t = \zeta w/v\sqrt{c\rho/2f_c}$.

The spatial periods of the oscillations prior to the signal decrease are found to be generally tens of wavelengths or longer. Thus, these oscillations can be easily distinguished from multipath fading since the latter is composed of sinusoids with much higher frequencies [26, 1] as illustrated in Fig. 3.3. Therefore, diffraction-induced patterns are reliable EW indicators of an approaching blockage, and presence of reflected multipath components does not present significant obstacles to proposed early warning capability as demonstrated in

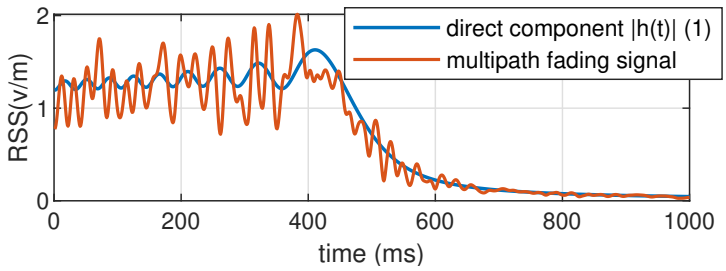


Figure 3.3: RSS of the multipath fading UE path with LoS/NLoS transition; Rician fading with K=11dB [1], UE speed = 24 (m/s).

Section 3.4.

Finally, realistic physical model-based examples of oscillation patterns prior to the Los/NLoS transition that illustrate the impacts of the distance d, angle α, and speeds are shown in Fig. 3.4(a-c). From Fig. 3.4(a), we note that the overall oscillation frequency reduces if the blocker is located further from the UE. Fig. 3.4(b) shows that the overall frequency increases as α decreases while Fig. 3.4(c) demonstrates that increasing UE and blocker speeds result in higher oscillation frequency and sharper magnitude drop prior to the blockage. Similar trends have been observed for other LoS/NLoS transition scenarios. Also, the characteristic frequency decrease and envelope changes were observed only prior to the blockage, not in other model scenarios as illustrated in Fig. 3.4(d).

3.3 ML-based Early Warning of Blockage

3.3.1 Problem Formulation and Overview of MiniRocket Method

Consider the example in Fig. 3.2. At time $t = T$, the proposed method predicts whether a blockage occurs within the upcoming time interval $P = [T + t_1, T + t_1 + P]$ by observing the RSS $|h(t)|$ in the observation window $W = [T - W, T]$. Given t_1, the observation and prediction sliding windows in Fig. 3.2 retain their separation, but may be found at any point

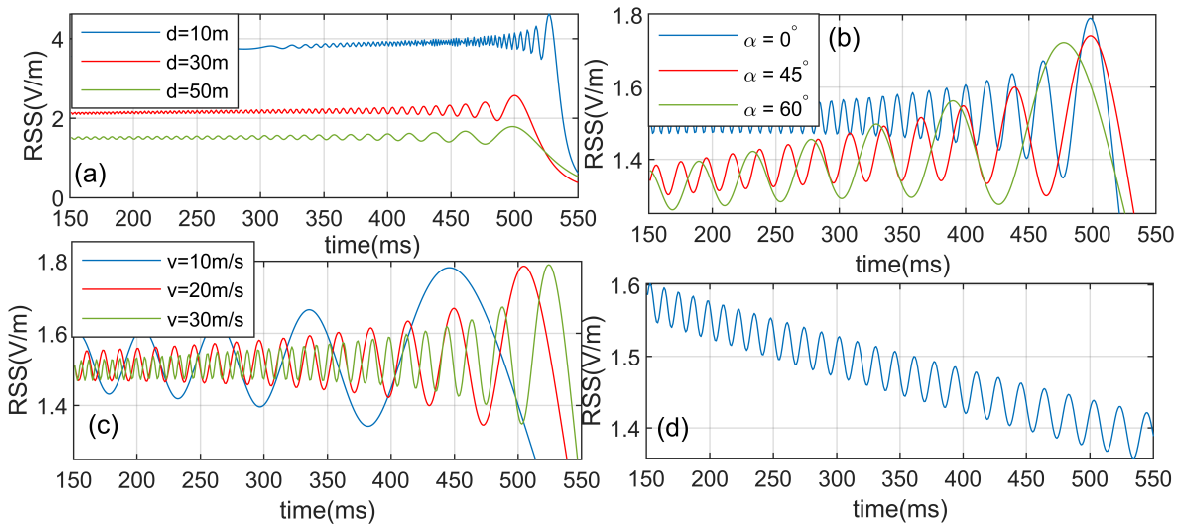


Figure 3.4: RSS $|h(t)|$; BS and blocker as in Fig. 3.1, (a-c) pre-blockage pattern; transition time $t_t = 563\ ms$. (a) $\alpha = 0°$, $d = \{10, 30, 50\}\ m$, $v \approx 28\ m/s$, (b) $\alpha = \{0°,45°,60°\}$, $d = 50\ m$, $v \approx 28\ m/s$, (c) $\alpha = 0°$, $d = 50\ m$, $v = \{10,20,30\}\ m/s$, (d) No LoS/NLoS transition, UE starts from (50,0) m at $t = 0$, and moves to (20,0) m, $v = 30\ m/s$.

before, during, or after the transition. The positions shown are the most relevant for early warning.

The input to the ML model is $|h(t)|$ (possibly modified to capture multipath fading and additive noise as described in Section 3.4) sampled at the rate f_s(Hz) within the observation window W. The output is given by the label $\mathscr{L}$ that predicts absence ($\mathscr{L}$=0) or presence ($\mathscr{L}$=1) of blockage within the prediction window P. The samples of $|h(t)|$ are time-correlated and exhibit specific diffraction-induced features discussed in Section 3.2 and illustrated in Fig. 3.2, 3.4, thus justifying utilization of the time-series classification ML methods.

We employ the MiniRocket classifier [22], which improves on the Rocket method [21]. MiniRocket extracts features from time-series data by convolving it with a large number (10,000) of convolutional kernels of length 12 with random weights and dilation. A unique feature of the Rocket-type methods is utilization of diverse dilation and padding, which

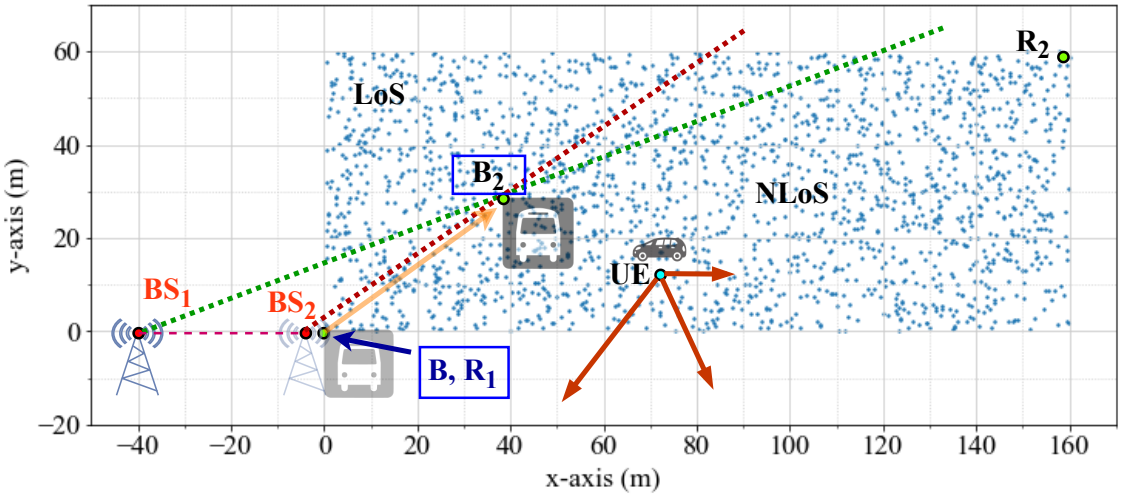


Figure 3.5: Physical model configuration for dataset $\mathscr{D}$ Generation.

produces a huge variety of kernels and allows to capture the diffraction-induced patterns at different frequencies and scales. The extracted features are used to train a Ridge regression model, which is a linear least-square regression with ℓ_2 regularization penalty imposed on the size of the coefficients [13]. Finally, a comprehensive review of the time-series classification methods in [64] shows that Rocket outperforms other ML methods, including the state-of-the-art Long Short-Term Memory (LSTM) classification model TapNet, in terms of the training and prediction time, making MiniRocket a suitable choice for the EW task at UE.

3.3.2 ML Dataset Generation

We use the physical model described in section 3.2 to generate the dataset for training and testing the proposed ML model for early warning of mmWave LoS blockage.

The training and testing dataset $\mathscr{D} = \{\mathscr{X}_i, \mathscr{L}_i\},\ i = 1, \ldots, N$, contains randomly generated scenarios where the UE trajectory originates in LoS and terminates in either LoS or NLoS. It is formed by collecting N simulations of randomized scenario examples generated using

the physical model where $\mathscr{X}_i$ is the mmWave signal RSS within the sliding observation window W sampled with frequency f_s (Fig. 3.2) and $\mathscr{L}_i \in \{0,1\}$, provided by the physics-based model, indicates the actual presence or absence of blockage in the prediction window P. We found that $N = 10{,}000$ is a suitable size for $\mathscr{D}$ for all the results in the paper by varying N from 1000 to 100,000 scenarios. Smaller N values degrade the EW performance while higher N do not enhance it. We vary the relative locations of the BS, the UE, and the blocker as well as the UE and blocker velocities and directions of movement, thus creating a rich mapping of the features of the signals $\mathscr{X}_i$ to their labels $\mathscr{L}_i$. Half of the scenarios in $\mathscr{D}$ have label $L_i = 1$, and the rest have label $L_i = 0$, resulting in a balanced dataset $\mathscr{D}$.

Figure 3.5 illustrates the physical model configuration for the dataset $\mathscr{D}$ within a 5G/mmWave cell with radius of 200 m. All the random variations in the dataset are modeled using independent, uniform distributions. We make a realistic assumption that the BS is stationary and located at a point selected randomly along the dashed line between $\mathrm{BS}_1 = (-40,0)\,m$ and $\mathrm{BS}_2 = (-0.5,0)\,m$. The blocker starts its motion from point $\mathrm{B} = (0,0)\,m$ while the UE starts from a point randomly selected inside the dotted rectangle formed by $\mathrm{R}_1 = (0,0)\,m$ and $\mathrm{R}_2 = (160,60)\,m$. The blocker and the UE move in random directions along linear paths with random speeds between 0 to 30(m/s) for the duration of $2s$. Thus, the dataset $\mathscr{D}$ includes a variety of UE and blocker types, including stationary objects, pedestrians, bicycles, vehicles, etc. Paths that locate the UE between the BS and the blocker are considered invalid and are removed from the dataset. The geometric LoS/NLoS transition line for the two BS locations and the blocker at B_2 are depicted with dotted lines connecting the BSs and the blocker.

To train and test the proposed ML-based early warning method, we implement k-fold cross-validation with $k = 5$ where $\mathscr{D}$ is split randomly into 5 groups, $D_1, \ldots, D_5$. In the j^{th} stage, $j = 1, \ldots, 5$, the group D_j is used as the test data while the rest of the dataset is used to train the ML model. Training is done offline, e.g., at the BS, and the trained model is shared with UE to perform EW. Performance is evaluated over the D_j using the trained model, and

Table 3.1: Performance of ML-based EW of blockage method; received signal $|h(t)|$ (3.1), $W = 400(ms)$, $f_s = 1$kHz.

$P(ms)$	$t_1(ms)$	Accuracy	f1 score	AUC
10	$100-250$	99.17%	0.9907	0.9913
10	450	95.74%	0.9317	0.9475
10	550	94.56%	0.9178	0.9289
25	$100-250$	99.51%	0.9920	0.9951
25	450	96.71%	0.9447	0.9506
25	550	95.01%	0.9211	0.9308
≥ 50	$100-250$	99.63%	0.9934	0.9965
≥ 50	450	97.81%	0.9548	0.9609
≥ 50	550	95.74%	0.9247	0.9354

the results are averaged over the 5 stages. Finally, we employ the accuracy, f1 score, and area under the curve (AUC) metrics [13] to evaluate performance of the EW ML method.

3.4 Numerical Results

In this section, we analyze and illustrate performance of the ML-based EW method described in section 3.3. First, we employ the dataset $\mathscr{D}$ with each element representing the direct component (3.1). By fixing the values of f_s and W during training and evaluating the ML performance for each fixed parameter, we found that all performance metrics improve as f_s and W increase and saturate at $f_s = 1$kHz and $W = 400(ms)$. Table 3.1 shows ML performance for these optimized f_s and W vs prediction time. We vary the prediction window P from one to ten 5G frames ([10-100](ms)) and observe that the choice of P does not have a significant impact on the EW performance. The minimum value of t_1 is 100 (ms), or ten 5G frames, thus providing sufficient time for proactive response. Increasing t_1 to 550 (ms), which exceeds the max time reported in the literature [39], slightly lowers the accuracy to

%95. Hence, accurate EW is achievable for a wide range of t_1 values.

To study the effect of multipath fading on the EW performance, we add to each signal $h(t)$ (3.1) in $\mathscr{D}$ an appropriately scaled multipath fading signal with the Jakes Doppler spectrum calculated for the carrier frequency f_c =30GHz and the velocity v of the signal $h(t)$. The resulting Rician fading signal has the K-factor of 11 dB [1]. We employ the modified dataset $\mathscr{D}_f$ to train and test the EW of LoS blockage ML method where $\mathscr{L}_k = 1$ (or 0) still indicates whether the direct component $h(t)$ crosses the LoS/NLoS transition line (or remains in LoS). Simulations using $\mathscr{D}_f$ for similar W, t_1, P values of Table 3.1 for varying f_S values indicate that EW has accuracy above 90% for f_s=1kHz for $P \geq 25(ms)$. For $P = 10(ms)$, the accuracy is reduced (71% to 85%) for f_s=1kHz, but improves significantly (83% to 89%) as f_s increases to 15KHz. One explanation is that higher-rate sampling is able to capture the multipath fading variations, which are on the order of maximum Doppler shift f_{dm} = 3KHz and vary much faster than the diffraction-induced oscillations with rates at most 250Hz in this case. Similar results were obtained for smoothed fading signals. Additional simulation results can be found in Appendix B. The higher-rate samples used in these and following results can be obtained from reference symbols [15] or data-aided approaches, e.g., [37].

Moreover, to test the effect of UE and blocker speed on the EW performance, we split the test sets $\mathscr{D}$ and $\mathscr{D}_f$ into eight bins each with limited speeds starting from $0(m/s)$ with increments of 5 up to $30(m/s)$ and test for each bin separately while fixing the sampling rate f_s in each experiment. Note that training is still performed jointly for all speeds. The results are shown in Table 3.2. When the sampling rate f_s=1kHz, the diverse dilation of the MiniRocket kernels enables it to capture blockage-induced oscillations with various frequencies, hence making the model insensitive to the oscillation frequency variations caused by UE speed (see Fig. 3.4(c)). However, for lower f_s, EW accuracy is reduced in highly mobile conditions since the oscillations caused by higher speeds are undersampled, resulting in performance degradation. We found that smoothing of multipath fading signals

Table 3.2: Accuracy (%) of ML-based EW of blockage method; Multipath fading (F) and non-fading(NF) datasets, $W = 400(ms)$, $t_1 = 250(ms)$, $P = 50(ms)$; Rician Fading, K = 11 dB. For $f_s \leq 200$Hz, fading signals are L1 filtered using sampling rate = 6KHz, window size = 60 [2].

$f_s \rightarrow$	1 KHz		200 Hz		100 Hz		50 Hz	
speed↓	NF	F	NF	F	NF	F	NF	F
0-5	99.93	99.93	97.77	96.74	97.06	96.03	96.39	95.36
5-10	99.78	99.72	96.67	95.43	96.28	95.04	95.87	94.63
10-15	99.67	99.64	95.48	93.91	95.09	93.52	93.99	92.42
15-20	99.53	99.49	93.4	90.49	93.39	90.48	92.23	89.32
20-25	99.41	99.40	86.33	83.20	85.63	82.50	85.27	82.14
25-30	99.37	99.33	78.44	74.86	77.58	74.00	76.77	73.19

improves performance when the sampling rate is 500Hz or lower and speeds are above $20(m/s)$, but is not required for f_s=1kHz as shown in Table 3.2. These findings demonstrate that EW of LoS blockage is feasible for multipath fading signals and diverse mobile scenarios. Additional results (e.g., f1 score and AUC) can be found in Appendix B.

Next, to investigate robustness of the EW method to additive noise, we add independent and identically distributed (iid) zero-mean Gaussian random variables n_k with variance σ^2 to the elements of the multipath fading dataset $\mathscr{D}_f$ sampled at the rate f_s. First we train the ML model for each f_s value in the range of $[1-15]$kHz using a noisy training set with $1/\sigma^2$ values chosen randomly from 0 and 35dB. Next, the test set is created for each variance σ^2 by adding noise samples with this variance to the samples of the signals in the fading test set $\mathscr{D}_f$ using the sampling rate f_s.

Figure 3.6 illustrates accuracy of the EW method vs $1/\sigma^2$ for several f_s values. We observe that increasing the sampling rate improves prediction performance for high-to-moderate noise powers ($1/\sigma^2 \leq 25$dB), but the accuracy saturates at f_s=15kHz. Moreover, $f_s = 1$kHz is sufficient when noise is low ($1/\sigma^2 \geq 25$dB) consistent with the results for the noiseless

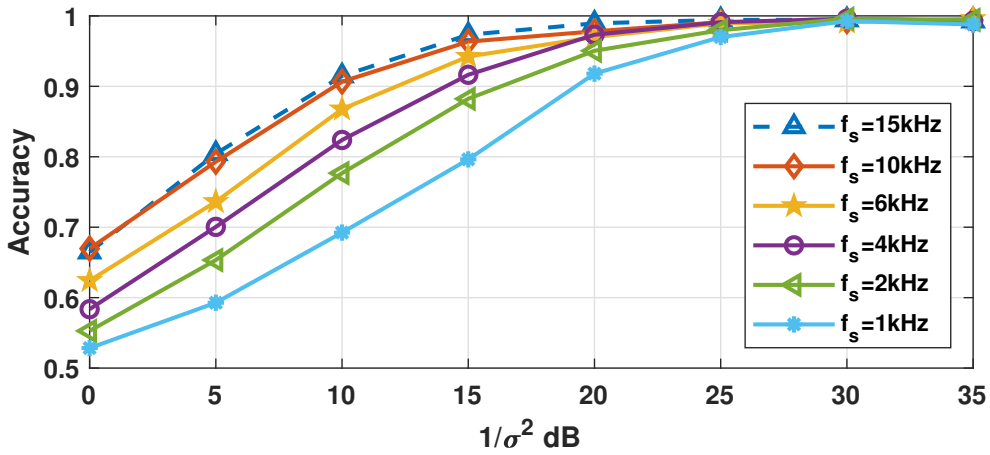


Figure 3.6: Accuracy of mmWave EW of blockage for multipath fading dataset $\mathscr{D}_f$ vs $1/\sigma^2$ for $f_s = \{1, 2, 4, 6, 10, 15\}$ kHz, $W = 400\,ms$, $t_1 = 250\,ms$ and $P = 50\,ms$; Rician fading, K=11dB.

cases discussed above. We conclude that the EW accuracy greater than 90% is achievable when $1/\sigma^2 \geq 10$dB. The f1 score and AUC follow the same trends as is Fig. 3.6 and can be found in Appendix B.

Finally, the proposed method has 39,992 trainable parameters while an LSTM-based time-series classifier has 325,000 trainable parameters. Thus, the training time is much faster and EW of blockage, which requires multiplying the input signal with the trained parameters, is tens of 5G frames sooner with MiniRocket than with LSTM-based models.

3.5 Conclusion

A fast and accurate time-series ML method is developed to provide early warning of mmWave LoS link blockage hundreds of milliseconds, or tens of 5G frames, ahead using in-band RSS measurements, thus allowing sufficient time for the communication system to adapt the data rate, search for a new beam, or perform a handover between base stations. The proposed method relies on physics of diffracted signals rather than on specific

topology or mobility assumptions. Future work will include physical modeling of reflectors and directional antennas and evaluation of their impacts on the EW capability. Finally, we plan to incorporate measurements and utilize the proposed method to improve reliability and efficiency of mmWave and sub-THz systems.

CHAPTER

4

CONCLUSIONS AND FUTURE DIRECTIONS

In this dissertation, we discussed data-driven and communication-efficient optimal control methods for cyber-physical systems and provided solutions to increase reliability of the millimeterwave cellular networks. First, our work focused on online data-driven control for power systems, an example of a networked control system (NCS), using reinforcement learning (RL) and sparsity-constrained optimization to save on communication costs. Moreover, we used received signal strength (RSS) measurements in millimeterwave networks and time-series machine learning models to predict link blockage and increase the robustness

of the network. Our contributions are summarized as follows:

1. Development of communication-efficient RL online control for linear power system models with uncertainties to enable sparse feedback for wide-area control.

2. Development of a data-driven method to predict mmWave signal line-of-sight blockage hundreds of milliseconds ahead using RSS measurements and time-series classifiers.

Chapter 2 of this thesis was published in [23], and Chapter 3 was published in [24]. Possible future research directions are:

Distributed and Decentralized Control Design

In this thesis, the control design process requires a central node to collect data from the rest of the NCS. One interesting problem is the extension of central control design to distributed or decentralized control design. Such control design could further reduce communication costs and make the method more robust to network impairments.

More Complex and Realistic Blockage Scenarios

In chapter 3, we consider omnidirectional antennas and straight-line paths for the user equipment and the blockers. Future work will include modeling the effects of reflectors, curved UE and blockers paths, and directional antennas on the EW capability, incorporating measurements, as well as utilization of the proposed method to improve the reliability and efficiency of mmWave systems.

REFERENCES

[1] 3GPP. Study on channel model for frequencies from 0.5 to 100 GHz. Technical Report (TR) 38.901, 3rd Generation Partnership Project (3GPP), 04 2018. Version 14.2.2.

[2] 3GPP. 5G NR; NR and NG-RAN Overall description; stage-2. Technical specification (TS) 38.300, 3rd Generation Partnership Project (3GPP), 07 2020. Version 16.2.0.

[3] Ziad Ali, Alexandra Duel-Hallen, and Hans Hallen. Early warning of mmwave signal blockage and aoa transition using sub-6 GHz observations. *IEEE Communications Letters*, 24(1):207–211, 2019.

[4] Ahmed Alkhateeb, Iz Beltagy, and Sam Alex. Machine learning for reliable mmwave systems: Blockage prediction and proactive handoff. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1055–1059, 2018.

[5] Muhammad Alrabeiah and Ahmed Alkhateeb. Deep learning for mmwave beam and blockage prediction using sub-6 GHz channels. *IEEE Transactions on Communications*, 68(9):5504–5518, 2020.

[6] Reza Arastoo, MirSaleh Bahavarnia, Mayuresh V Kothare, and Nader Motee. Closed-loop feedback sparsification under parametric uncertainties. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 123–128. IEEE, 2016.

[7] Danish Aziz, Jens Gebert, Anton Ambrosy, Hajo Bakker, and Hardy Halbauer. Architecture approaches for 5G millimetre wave access assisted by 5G low-band using multi-connectivity. In *2016 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, 2016.

[8] Sohail Bahmani, Bhiksha Raj, and Petros T Boufounos. Greedy sparsity-constrained optimization. *Journal of Machine Learning Research*, 14(Mar):807–841, 2013.

[9] Jingchao Bao, Tao Shu, and Husheng Li. Handover prediction based on geometry method in mmwave communications - a sensing approach. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2018.

[10] Amir Beck and Yonina C Eldar. Sparsity constrained nonlinear optimization: Optimality conditions and algorithms. *SIAM Journal on Optimization*, 23(3):1480–1509, 2013.

[11] Giuseppe Belgioioso, Dominic Liao-McPherson, Mathias Hudoba de Badyn, Saverio Bolognani, John Lygeros, and Florian Dörfler. Sampled-data online feedback equilibrium seeking: Stability and tracking. *arXiv preprint arXiv:2103.13988*, 2021.

[12] Tao Bian, Yu Jiang, and Zhong-Ping Jiang. Decentralized adaptive optimal control of large-scale systems with application to power systems. *IEEE Transactions on Industrial Electronics*, 62(4):2439–2447, 2015.

[13] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[14] Andrea Bonfante, Lorenzo Galati Giordano, Irene Macaluso, and Nicola Marchetti. Performance of predictive indoor mmwave networks with dynamic blockers. *IEEE Transactions on Cognitive Communications and Networking*, pages 1–1, 2021.

[15] Ramesh Chandran and Amarpreet Singh Sethi. A novel method for pilot pattern selection in 5g nr systems. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2021.

[16] Gouranga Charan, Muhammad Alrabeiah, and Ahmed Alkhateeb. Vision-aided 6g wireless communications: Blockage prediction and proactive handoff. *IEEE Transactions on Vehicular Technology*, 70(10):10193–10208, 2021.

[17] Nilanjan Ray Chaudhuri, Debraj Chakraborty, and Balarko Chaudhuri. Damping control in power systems under constrained communication bandwidth: A predictor corrector strategy. *IEEE Transactions on Control Systems Technology*, 20(1):223–231, 2012.

[18] Tianyi Chen, Kaiqing Zhang, Georgios B Giannakis, and Tamer Basar. Communication-efficient policy gradient methods for distributed reinforcement learning. *IEEE Transactions on Control of Network Systems*, 2021.

[19] Joe H Chow and Kwok W Cheung. A toolbox for power system dynamics and control engineering education and research. *IEEE transactions on Power Systems*, 7(4):1559–1564, 1992.

[20] Joe H Chow and Scott G Ghiocel. An adaptive wide-area power system damping controller using synchrophasor data. In *Control and Optimization Methods for Electric Smart Grids*, pages 327–342. Springer, 2012.

[21] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

[22] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 248–257, 2021.

[23] Amirhassan Fallah Dizche, Aranya Chakrabortty, and Alexandra Duel-Hallen. Sparse wide-area control of power systems using data-driven reinforcement learning. In *2019 American Control Conference (ACC)*, pages 2867–2872. IEEE, 2019.

[24] Amirhassan Fallah Dizche, Alexandra Duel-Hallen, and Hans Hallen. Early warning of mmwave signal blockage using diffraction properties and machine learning. *IEEE Communications Letters*, pages 1–5, 2022.

[25] Florian Dörfler, Mihailo R Jovanović, Michael Chertkov, and Francesco Bullo. Sparsity-promoting optimal wide-area control of power networks. *IEEE Transactions on Power Systems*, 29(5):2281–2291, 2014.

[26] A. Duel-Hallen, Shengquan Hu, and H. Hallen. Long-range prediction of fading signals. *IEEE Signal Processing Magazine*, 17(3):62–75, 2000.

[27] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[28] Marco Giordani, Marco Mezzavilla, Sundeep Rangan, and Michele Zorzi. Multi-connectivity in 5G mmwave cellular networks. In *2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pages 1–7, 2016.

[29] Guenther, Robert D. *Modern Optics*. Oxford University Press, Oxford, New York, 2nd edition edition, June 2018.

[30] Hans Hallen, Ziad Ali, and Alexandra Duel-Hallen. Physics-based wireless channel model with fresnel diffraction. `https://www.codeocean.com/`, 1 2021.

[31] Hans Hallen, Alexandra Duel-Hallen, Shengquan Hu, Tung-Shen Yang, and Ming Lei. A physical model for wireless channels to provide insights for long range prediction. In *MILCOM 2002. Proceedings*, volume 1, pages 627–631. IEEE, 2002.

[32] Yousuf Hashmy, Zhe Yu, Di Shi, and Yang Weng. Wide-area measurement system-based low frequency oscillation damping control through reinforcement learning. *IEEE Transactions on Smart Grid*, 11(6):5072–5083, 2020.

[33] Ian A Hiskens and Jassim Alseddiqui. Sensitivity, approximation, and uncertainty in power system dynamic simulation. *IEEE Transactions on Power Systems*, 21(4):1808–1820, 2006.

[34] Tomoki Hoshiya and Tomonori Sadamoto. Fast online reinforcement learning of distributed optimal controller for large-scale network systems. In *2021 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1135–1141, 2021.

[35] Rabih A Jabr, Bikash C Pal, and Nelson Martins. A sequential conic programming approach for the coordinated and robust design of power system stabilizers. *IEEE Transactions on Power Systems*, 25(3):1627–1637, 2010.

[36] Abhishek Jain, Aranya Chakrabortty, and Emrah Biyik. An online structurally constrained LQR design for damping oscillations in power system networks. In *American Control Conference (ACC), 2017*, pages 2093–2098. IEEE, 2017.

[37] Tao Jia, Alexandra Duel-Hallen, and Hans Hallen. Data-aided noise reduction for long-range fading prediction in adaptive modulation systems. *IEEE Transactions on Vehicular Technology*, 62(5):2358–2362, 2013.

[38] Yu Jiang and Zhong-Ping Jiang. Robust adaptive dynamic programming with an application to power systems. *IEEE Transactions on Neural Networks and Learning Systems*, 24(7):1150–1156, 2013.

[39] Anders E. Kalør, Osvaldo Simeone, and Petar Popovski. Prediction of mmwave/thz link blockages through meta-learning and recurrent neural networks. *IEEE Wireless Communications Letters*, 10(12):2815–2819, 2021.

[40] Bahare Kiumarsi, Kyriakos G Vamvoudakis, Hamidreza Modares, and Frank L Lewis. Optimal and autonomous control using reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[41] Robert G Kouyoumjian and Prabhakar H Pathak. A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface. *Proceedings of the IEEE*, 62(11):1448–1461, 1974.

[42] Prabha Kundur, Neal J Balu, and Mark G Lauby. *Power system stability and control*, volume 7. McGraw-hill New York, 1994.

[43] Jae Young Lee, Jin Bae Park, and Yoon Ho Choi. Integral Q-learning and explorized policy iteration for adaptive optimal control of continuous-time linear systems. *Automatica*, 48(11):2850–2859, 2012.

[44] Frank L Lewis and Derong Liu. *Reinforcement learning and approximate dynamic programming for feedback control*, volume 17. John Wiley & Sons, 2013.

[45] Frank L Lewis, Draguna Vrabie, and Vassilis L Syrmos. *Optimal control*. John Wiley & Sons, 2012.

[46] Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3834–3839. IEEE, 2017.

[47] Feier Lian, Aranya Chakrabortty, and Alexandra Duel-Hallen. Game-theoretic multi-agent control and network cost allocation under communication constraints. *IEEE Journal on Selected Areas in Communications*, 35(2):330–340, 2017.

[48] Feier Lian, Aranya Chakrabortty, Fen Wu, and Alexandra Duel-Hallen. Sparsity-constrained mixed H_2/H_∞ control. In *2018 American Control Conference (ACC)*, pages 6253–6258. IEEE, 2018.

[49] Carolina Lidström and Anders Rantzer. Optimal H_{inf} state feedback for systems with symmetric and Hurwitz state matrix. In *American Control Conference (ACC), 2016*, pages 3366–3371. IEEE, 2016.

[50] Fu Lin, Makan Fardad, and Mihailo R Jovanović. Design of optimal sparse feedback gains via the alternating direction method of multipliers. *IEEE Transactions on Automatic Control*, 58(9):2426–2431, 2013.

[51] Yuanjie Liu, Xiongping Yang, Wenkun Wen, and Minghua Xia. Smarter grid in the 5g era: A framework integrating power internet of things with a cyber physical system. *Frontiers in Communications and Networks*, page 23, 2021.

[52] Michael Lutter, Shie Mannor, Jan Peters, Dieter Fox, and Animesh Garg. Value iteration in continuous actions, states and time. In *International Conference on Machine Learning*, pages 7224–7234. PMLR, 2021.

[53] Ioannis Mavromatis, Andrea Tassi, Giovanni Rigazzi, Robert J Piechocki, and Andrew Nix. Multi-radio 5g architecture for connected and autonomous vehicles: application and design insights. *arXiv preprint arXiv:1801.09510*, 2018.

[54] Neil Mehta, Alexandra Duel-Hallen, and Hans Hallen. Template design and propagation gain for multipath UWB channels with per-path frequency-dependent distortion. In *MILCOM 2009-2009 IEEE Military Communications Conference*, pages 1–7. IEEE, 2009.

[55] Zakaria Mhammedi, Dylan J Foster, Max Simchowitz, Dipendra Misra, Wen Sun, Akshay Krishnamurthy, Alexander Rakhlin, and John Langford. Learning the linear quadratic regulator from nonlinear observations. *Advances in Neural Information Processing Systems*, 33:14532–14543, 2020.

[56] Faris B. Mismar, Ahmad Alammouri, Ahmed Alkhateeb, Jeffrey G. Andrews, and Brian L. Evans. Deep learning predictive band switching in wireless networks. *IEEE Transactions on Wireless Communications*, 20(1):96–109, 2021.

[57] COMSOL Multiphysics. Introduction to comsol multiphysics®. *COMSOL Multiphysics, Burlington, MA, accessed Feb*, 9:2018, 1998.

[58] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

[59] Takayuki Nishio, Hironao Okamoto, Kota Nakashima, Yusuke Koda, Koji Yamamoto, Masahiro Morikura, Yusuke Asai, and Ryo Miyatake. Proactive received power prediction using machine learning and depth images for mmwave networks. *IEEE Journal on Selected Areas in Communications*, 37(11):2413–2427, 2019.

[60] Anup Parikh, Rushikesh Kamalapurkar, and Warren E Dixon. Integral concurrent learning: Adaptive control with parameter convergence without pe or state derivatives. *arXiv preprint arXiv:1512.03464*, 2015.

[61] Michele Polese, Marco Giordani, Marco Mezzavilla, Sundeep Rangan, and Michele Zorzi. Improved handover through dual connectivity in 5G mmwave mobile networks. *IEEE Journal on Selected Areas in Communications*, 35(9):2069–2084, 2017.

[62] Theodore S Rappaport, George R MacCartney, Mathew K Samimi, and Shu Sun. Wideband millimeter-wave propagation measurements and channel models for future wireless communication system design. *IEEE transactions on Communications*, 63(9):3029–3056, 2015.

[63] Ivan D Jimenez Rodriguez, Ugo Rosolia, Aaron D Ames, and Yisong Yue. Learning unstable dynamics with one minute of data: A differentiation-based gaussian process approach. *arXiv preprint arXiv:2103.04548*, 2021.

[64] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2):401–449, 2021.

[65] Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty with probabilistic signal temporal logic. 2016.

[66] Abhinav Kumar Singh and Bikash C Pal. Decentralized dynamic state estimation in power systems using unscented transformation. *IEEE Transactions on Power Systems*, 29(2):794–804, 2014.

[67] Baptiste Sinquin and Michel Verhaegen. K4sid: Large-scale subspace identification with kronecker modeling. *IEEE Transactions on Automatic Control*, 64(3):960–975, 2018.

[68] Sanjib Sur, Xinyu Zhang, Parmesh Ramanathan, and Ranveer Chandra. Beamspy: Enabling robust 60 GHz links under blockage. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 193–206, 2016.

[69] Yu Tsukamoto, Haruhisa Hirayama, Seung Il Moon, Shinobu Nanba, and Hiroyuki Shinbo. Feedback control for adaptive function placement in uncertain traffic changes on an advanced 5g system. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2021.

[70] Kyriakos G Vamvoudakis. Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach. *Systems & Control Letters*, 100:14–20, 2017.

[71] Peter Van Overschee and Bart De Moor. N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75–93, 1994.

[72] Qingqing Wu, Jie Xu, Yong Zeng, Derrick Wing Kwan Ng, Naofal Al-Dhahir, Robert Schober, and A. Lee Swindlehurst. A comprehensive overview on 5g-and-beyond networks with uavs: From communications to sensing and intelligence. *IEEE Journal on Selected Areas in Communications*, 39(10):2912–2945, 2021.

[73] Zhengqing Yun and Magdy F Iskander. Ray tracing for radio propagation modeling: Principles and applications. *IEEE Access*, 3:1089–1100, 2015.

[74] Ali Zaidi, Fredrik Athley, Jonas Medbo, Ulf Gustavsson, Giuseppe Durisi, and Xiaoming Chen. *5G Physical Layer: principles, models and technology components*. Academic Press, 2018.

[75] Song Zhang and Vijay Vittal. Design of wide-area power system damping controllers resilient to communication failures. *IEEE Transactions on Power Systems*, 28(4):4292–4300, 2013.

[76] Xiangnan Zhong, Haibo He, Ding Wang, and Zhen Ni. Model-free adaptive control for unknown nonlinear zero-sum differential game. *IEEE transactions on Cybernetics*, 2017.

APPENDICES

APPENDIX

A

CODE FOR "SPARSE WIDE-AREA CONTROL OF POWER SYSTEMS USING DATA-DRIVEN REINFORCEMENT LEARNING"

This Appendix provides the Matlab code implementing the numerical simulations of Chapter 2, Section 2.4.

A.1 Sparse RL control

This section provides the main body of the code, which includes implementing uncertainty in the LTI dynamic system model, response of the open-loop dynamic system to disturbances, response of the mismatched LQR controller to disturbances, and implementation of Alg. 1 and its response to disrurbances.

```
clear all
clc
close all
% define global variables
global S A B2 Q R Klqr sWa TT freq
% load the IEEE 39 bust power system model
load('neData.mat')
ir = 1;

countt = 0;

% perturb A to find the uncertain CPS model
cols = [1:3, 8:11, 16:19, 24:27, 32:35, 40:43, 48:51, 56:59, 64:67,
    72:75];
Ap = A; % Ap = perturbed A inertia and reactance (admitance)
for i = 1:75
    for j = cols(1:length(cols))
        if Ap(i,j) ~= 0 && Ap(i,j) ~= 1
            Ap(i,j) =  A(i,j) + 0.7*Ap(i,j)*rand-0.35*Ap(i,j);
            countt = countt+1
        end
    end
end

% load a previsously generated uncertain model
load('Ap75')

% generate the frequencies for exporation noise as sum of sinusoids
freq = 40.*rand(1,675) -20;
% load predefined frequiecies for the exploration noise
load('freq');
% for the dense case, s = 75*9 = 675
s = 675;
%time step of applying the control input u(t) to the dynamic system
T = 0.05;

% for 0 initial condition use below
%x0 = [0.01*ones(1,75) 0];
% use the random impulse as initial condition to simulate
   disturbance
```

```
Xini = [0.013*randn(1,75) 0];  % correcct X0, created once and
   saved for

% load a previously saved initial condition to regenerate the
   results
load('Xini4')
% set the initial condition to the loaded value
x0 = Xini;

%Simulate the response of the plant (Ax + Bu) to the initial
   condition
% for the open-loop system
[t,x]= ode23('loopNE1',[0 T],x0); % dynamic simulation
% t shows time and x is the vector of state variables
[N1, M1] = size(x); % get the dimension of state variables vector
xt = x(N1,1:75)';
% open-loop system simulated for 10 seconds
[t1,x1]= ode23('loopNE1',[0 10],x0);

figure
%subplot(4,1,1)
% the selected state variables indicate the angular velocity of
   gens
% see PST struct discription for the naming of state variables
plot(t1,x1(:,[2,9,17,25,33,41,49,57,65,73]),'linewidth',1.5)
xlim([0 7])
ylim([-0.3 0.3])
ylabel('a) f (Hz)')
xlabel('time (sec)')


% Calculate the ideal LQR controller K_{lqr} for the known model (A
   , B)
Klqr = lqr(A,B2,Q,R);
% Use Klqr to damp the oscillations in the known model case
% Klqr is used to calculate the control input in 'loopNE2.m'
[t2,x2]= ode23('loopNE2',[0 10],x0);

figure
%subplot(4,1,2)
% the selected state variables indicate the angular velocity of
   gens
plot(t2,x2(:,[2,9,17,25,33,41,49,57,65,73]),'linewidth',1.5)
ylabel('b) f (Hz)')
xlabel('time (sec)')
xlim([0 7])
ylim([-0.3 0.3])


% to compare with Mihailo's ADMM sparsity promoting uncomment
   following
```

```
% % apply admm sparse LQR
% load('K_admm');
% Klqr = K_admm;
% [t5,x5]= ode23('loopNE2',[0 10],x0);
% %figure(5)
% subplot(2,3,5)
%
% plot(t5,x5(:,[2,9,17,25,33,41,49,57,65,73]),'linewidth',1.5)
% xlim([0 7])
% title('E) Sparsity promoting')
% % gamma  = 0.019
% ylabel('frequnecy disturbance')
% xlabel('time (sec)')
% ylim([-0.3 0.3])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the mismatched controller
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% form the initial critic vector
Wc0 = 0.01*ones(1,3570);
%Wa0 = ones(75,9);
% form initial actor as the LQR controller for the uncertain model
   (A0, B0)
[Kini,Pini,Eini] = lqr(Ap,B2,Q,R);
Wa0 = -1*Kini';
% calculate the control input u = -Kx(t)
u =  Wa0' *xt;
% form the basis vector U = [x, u]
Ut = [0.01*ones(1,75)'; u];
Utt = [xt;  u];

% apply K obtained by uncertain Ap on the real system
[Kini,Pini,Eini] = lqr(Ap,B2,Q,R);
Klqr = Kini;
% Simulate the dynamic response to the mismatched controller
[t4,x4]= ode23('loopNE2',[0 7],x0);

figure
%subplot(4,1,3)
% the selected state variables indicate the angular velocity of
   gens
plot(t4,x4(:,[2,9,17,25,33,41,49,57,65,73]),'linewidth',1.5)
xlim([0 7])
ylabel('c) f (Hz)')
xlabel('time (sec)')
ylim([-0.3 0.3])

% % better Wc0
% Q_0 = [Pini + Q + Pini*Ap + Ap'*Pini Pini*B2;
%     B2'*Pini R];
```

```
% ccc = 1;
% for iwc = 1:length(Q_0)
%     for jwc = iwc:length(Q_0)
%         if iwc == jwc
%             Wc0 (ccc) = 0.025 * Q_0(iwc, jwc);
%             ccc = ccc+1;
%         else
%             Wc0 (ccc) =  0.05 *Q_0(iwc, jwc);
%             ccc = ccc+1;
%         end
%     end
% end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the ACTOR-CRITIC RL model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% this is the Sigma in the paper formed by kron product of U(t) and
    U(t-1)
S = KRON(Utt) - KRON(Ut);

% Extract actor from the critic
X(1,:) = [x0 Wc0 Wa0(:,1)' Wa0(:,2)' Wa0(:,3)' ...
    Wa0(:,4)' Wa0(:,5)' Wa0(:,6)' Wa0(:,7)' Wa0(:,8)' Wa0(:,9)'];

xx = [];
tt = [];
Tim = 0;
uu = [];
Ie = [x(N1,76)];
X0 = X(1,:);
%s = 200;
sWa = Wa0';
tic;
% actor-critic update loop
for i = 1:10/0.05 % simulate for 10 seconds with steps of 0.05
    % fomr the basis U
    Ut = [X(i,1:75)'; u];
    % simulate the dynamic system using the actor and the
       exploration
    % noise
    [t,x]= ode23('loopNE_PE',[0 T],X0);
    % get dimensiones of the state vector
    [N,M] = size(x);
    X(i+1,:) = x(N,:);

    % the time step converted to time
    TT = i*0.05;
    % collext the simulated state variables
    xx = [xx; x];
```

```
    % calculate the associated time vector
    tt = [tt; t+Tim*ones(size(t))];
    Ie = [Ie, x(N,76)+Ie(i)];
    % the initial condition as the disturbance
    X0 = [x(N,1:75) 0 x(N,77:4321)];

    % the actor update term Wdot extracted
    Wadot = x(N,3647:4321);
    % find the largest entries of Ks to use in GraSP
    [WaS , Iw] = sort(abs(Wadot),'descend');
    L = Iw(1:s); % keep the largest s entries

    % set rest of entries to zero
    Wpa = zeros(675,1);
    for cc = 1:length(L);
        Wpa(L(cc),1) =  Wadot(L(cc));
    end

    Wa = X(i,3647:4321);
    Wadot = Wpa;
     % GRaSP
     % sort g ########## step2
    [Sorted,In] = sort(abs(Wadot),'descend');
    % take min of 2s and size of K matrix
    [m,n] = size(Wa);
    % take min of 2s and size of K matrix
    UB = min((m*n),2*s);
    % keep the indices of 2s largest of g ########## step3
    Z = In(1:UB);
    Waz = (abs(Wadot) > 1e-5).*Wa;
    % non zero elements of current Wa
    [row,col,v] = find(Waz);
     % ############## step 4
    Tau = [Z ;row];

    % restricted newton descend step 5
    DE = zeros(675,1);
    for cc = 1:length(Tau);
        DE(Tau(cc),1) =  Wadot(Tau(cc));
    end

    Wa_temp = Wa + DE; %%%%%%%%% step 5
    % prune extra links, keep s largest elements of Wa

    %%%%%%%%%%%%%%%%%%%%%%%%% step 6
    [WaS , Iw] = sort(abs(Wa_temp),'descend');
    L = Iw(1:s);

    Wp = zeros(675,1);
    for cc = 1:length(L);
        Wp(L(cc),1) =  Wa_temp(L(cc));
```

```
232      end
233
234      % update the actor
235      Wa = Wp;
236      % extract actor for next iteration
237      X(i,3647:4321) = Wa;
238
239      % reshape actor vector from 1*nm to n*m
240      Waold = Wa;
241      count = 1;
242      Wa = zeros(9,75);
243      for iw = 1:9
244          Wa(iw,:) = Waold(count:count+74);
245          count = count + 75;
246      end
247      % calculate control input as u = -Kx
248      u = Wa * x(N,1:75)';
249      % collect the calcualted u for duration of simulation
250      uu = [uu u];
251      % update time step
252      Tim = Tim + t(N);
253
254      % calculate control input as u = -Kx
255      u = Wa * x(N,1:75)';
256      % form the basis U based on x , u
257      Utt = [X(i+1,1:75)'; u];
258      % calculate updated Signal for next iteration
259      S = KRON(Utt) - KRON(Ut);
260
261  end
262  toc;
263  tim = toc; % calcualte simulation time
264
265  % the following figure illustrates state variables of the SRL Alg.
266  figure
267  %subplot(4,1,4)
268  % the selected state variables indicate the angular velocity of
       gens
269  plot(tt,xx(:,[2,9,17,25,33,41,49,57,65]),'linewidth',1.5)
270  xlim([0 15])
271  ylabel('d) f (Hz)')
272  xlabel('time (sec)')
273  ylim([-0.3 0.3])
274  grid on
275  ylabel('\omega (rad/s)', 'FontWeight','bold')
276  xlabel('time (s)', 'FontWeight','bold')
277  xticks([0:15])
278  set(gca,'FontWeight','bold')
279
280  % grid on
281  % plot([3.211 3.211],[-0.4,0.5],'k')
```

```
% text(3.211,0.15,'Phase 2')
% text(0.2,0.23,'Phase 1')
% plot([2 2],[-0.4,0.5],'r')
% text(2,-0.2,'End exploration noise')
% plot([4.644 4.644],[-0.4,0.5],'b')
% text(4.644,-0.1,'End of phase 2')


% to calculate the area under the state variables = LQR cost
% for each simulated case: Klqr, Kmiss, Kopen, Ksp
Int = trapz(0:0.05:10,X);
Jsparse = sum(abs(Int(1:75)));

Int = trapz(t4,x4);
Jpert = sum(abs(Int(1:75)));

Int = trapz(t2,x2);
Jlqr = sum(abs(Int(1:75)));

Int = trapz(t1,x1);
Jopen = sum(abs(Int(1:75)));

% calculate the porportional LQR cost
% for each controller vs ideal LQR
perf1 = 100*(Jsparse-Jlqr)/Jlqr;
perf2 = 100*(Jpert-Jlqr)/Jlqr;
perf3 = 100*(Jopen-Jlqr)/Jlqr;
perf4 = 100*(Jadmm-Jlqr)/Jlqr;
```

The open-loop system dynamics simulated in line 48 of the above block of code is implemented with the following function.

```
function xout=loopNE1(t,x)
% define global variables
global A B2 Q R
% create the state-variables vector
X = x(1:75);
% LQR cost as an extra state
V = x(76);
% form the actor to calculate the cotnrol input
Wa = zeros(9,75)';
u = Wa'*X; % calcualte the control inpuut u = -Kx
% update the differential equations
xdot = A*X + B2*u;
Vdot = X'*Q*X + u'*R*u;
% output the results
xout=[xdot; Vdot ];
```

Moreover, the differential equations for the the closed-loop system of the ideal LQR

controller and the mismatched LQR controller, used in lines 70 and 118 of the main body of the code are implemented with the following function.

```
function xout=loopNE2(t,x)
% define global variables
global S A B2 Q R Klqr
% get the LQR feedback control gain matrix from main body of code
K = Klqr;
% create the state-variables vector
X = x(1:75);
% calculate control input as u = -Kx
u = -K*X;
% update the differential equations
xdot = A*X + B2*u;
Vdot = X'*Q*X + u'*R*u;
% output the results
xout=[xdot; Vdot ];
```

Finally, the dynamic system implementing the sparse RL control simulations using from line 172 of the main body of the code for Alg. 1 is defined in the following function.

```
function xout=loopNE_PE(t,x)
global S A B2 Q R TT freq
% define global variables
% create the state-variables vector
X = x(1:75);
% from the critic from the state vector
Wc = x(77:3646);
count = 3647;
% extract actor from the critix
for iw = 1:9
    Wa(iw,:) = x(count:count+74);
    count = count + 75;
end
Wa = Wa';
% LQR cost as an extra state variable
V = x(76);
countQ = 2851;
% update critic based on x
for iq = 1:9
    Qux(iq,:) = Wc(countQ:countQ+74);
    countQ = countQ + 75;
end
Qux = Qux';
% calcualate control input u = -Kx
u = Wa'*X;
% add exploration noise for t = [0, T_{pe}]
if TT < 2
    u = u + ones(9,1).*(0.01 * exp(-0.7*TT)*sum(sin(5*freq*TT)));
```

```
end
% update the differential equations
xdot = A*X + B2*u;
Vdot = X'*Q*X + u'*R*u;
% update critic
Wcdot = -30 * S./((1+S'*S)^2)*(Wc'*S+0.5*V);
% update actor
Wadot = -1 * X * (Wa'*X + Qux'*X)';
% output the results
xout=[xdot; Vdot; Wcdot; Wadot(:,1); Wadot(:,2); Wadot(:,3);...
    Wadot(:,4); Wadot(:,5); Wadot(:,6); Wadot(:,7); Wadot(:,8);
       Wadot(:,9) ];
```

A.2 Power System Model Data

The power system model used for simulations is generated using the Matlab Power Systems Toolbox (PST) [19] and the following data file.

```
% data for New England Test case

% to be used with the MatNetEig toolbox and
% Power System toolbox available at http://www.ecse.rpi.edu/pst/PST
   .html

disp('New England data')

% -----------
% bus data
%(bus#)( volt )( ang  )(  pgen )(  qgen )( pload )( qload )(gshunt)
   (bshunt)(bus type)
bus=[
   1   1.048  -9.43   0.0000   0.0000   0.0000   0.0000 0.00000
      0.00000 3 99  -99 1.0   0     0;
   2   1.0505 -6.89   0.0000   0.0000   0.0000   0.0000 0.00000
      0.00000 3 99  -99 1.0   0     0;
   3   1.0341 -9.73   0.0000   0.0000   3.2200   0.0240 0.00000
      0.00000 3 99  -99 1.0   0     0;
   4   1.0116 -10.53  0.0000   0.0000   5.0000   1.8400 0.00000
      1.00000 3 99  -99 1.0   0     0;
   5   1.0165 -9.38   0.0000   0.0000   0.0000   0.0000 0.00000
      2.00000 3 99  -99 1.0   0     0;
   6   1.0172 -8.68   0.0000   0.0000   0.0000   0.0000 0.00000
      0.00000 3 99  -99 1.0   0     0;
   7   1.0067 -10.84  0.0000   0.0000   2.3380   8.4000 0.00000
      0.00000 3 99  -99 1.0   0     0;
```

```
 8   1.0057 -11.34  0.0000   0.0000   5.2200   1.7600 0.00000
    0.00000 3 99  -99 1.0   0     0;
 9   1.0322 -11.15  0.0000   0.0000   0.0000   0.0000 0.00000
    0.00000 3 99  -99 1.0   0     0;
10   1.0235 -6.31   0.0000   0.0000   0.0000   0.0000 0.00000
   0.00000 3 99  -99 1.0   0     0;
11   1.0201 -7.12   0.0000   0.0000   0.0000   0.0000 0.00000
   0.00000 3 99  -99 1.0   0     0;
12   1.0072 -7.14   0.0000   0.0000   0.0850   0.8800 0.00000
   0.00000 3 99  -99 1.0   0     0;
13   1.0207 -7.02   0.0000   0.0000   0.0000   0.0000 0.00000
   0.00000 3 99  -99 1.0   0     0;
14   1.0181 -8.66   0.0000   0.0000   0.0000   0.0000 0.00000
   0.00000 3 99  -99 1.0   0     0;
15   1.0194 -9.06   0.0000   0.0000   3.2000   1.5300 0.00000
   0.00000 3 99  -99 1.0   0     0;
16   1.0346 -7.66   0.0000   0.0000   3.2940   3.2300 0.00000
   0.00000 3 99  -99 1.0   0     0;
17   1.0365 -8.65   0.0000   0.0000   0.0000   0.0000 0.00000
   0.00000 3 99  -99 1.0   0     0;
18   1.0343 -9.49   0.0000   0.0000   1.5800   0.3000 0.00000
   0.00000 3 99  -99 1.0   0     0;
19   1.0509 -3.04   0.0000   0.0000   0.0000   0.0000 0.00000
   0.00000 3 99  -99 1.0   0     0;
20   0.9914 -4.45   0.0000   0.0000   6.8000   1.0300 0.00000
   0.00000 3 99  -99 1.0   0     0;
21   1.0337 -5.26   0.0000   0.0000   2.7400   1.1500 0.00000
   0.00000 3 99  -99 1.0   0     0;
22   1.0509 -0.82   0.0000   0.0000   0.0000   0.0000 0.00000
   0.00000 3 99  -99 1.0   0     0;
23   1.0459 -1.02   0.0000   0.0000   2.4750   0.8460 0.00000
   0.00000 3 99  -99 1.0   0     0;
24   1.0399 -7.54   0.0000   0.0000   3.0860   -0.922 0.00000
   0.00000 3 99  -99 1.0   0     0;
25   1.0587 -5.51   0.0000   0.0000   2.2400   0.4720 0.00000
   0.00000 3 99  -99 1.0   0     0;
26   1.0536 -6.77   0.0000   0.0000   1.3900   0.1700 0.00000
   0.00000 3 99  -99 1.0   0     0;
27   1.0399 -8.78   0.0000   0.0000   2.8100   0.7550 0.00000
   0.00000 3 99  -99 1.0   0     0;
28   1.0509 -3.27   0.0000   0.0000   2.0600   0.2760 0.00000
   0.00000 3 99  -99 1.0   0     0;
29   1.0505 -0.51   0.0000   0.0000   2.8350   1.2690 0.00000
   1.00000 2 99  -99 1.0   0     0;
30   1.0475 -4.47   2.5000   1.3621   0.0000   0.0000 0.00000
   0.00000 2  8   -5 1.0   0     0;
31   1.0400  0.00   5.7293   1.7036   0.0920   0.0460 0.00000
   0.00000 2  8   -5 1.0   0     0;
32   0.9831  1.63   6.5000   1.7590   0.0000   0.0000 0.00000
   0.00000 2  8   -5 1.0   0     0;
33   0.9972  2.18   6.3200   1.0335   0.0000   0.0000 0.00000
```

```
      0.00000 2  8    -5 1.0    0      0;
  34    1.0123  0.74    5.0800    1.6400    0.0000    0.0000 0.00000
      0.00000 2  4    -3 1.0    0      0;
  35    1.0493  4.14    6.5000    2.0884    0.0000    0.0000 0.00000
      0.00000 2  8    -5 1.0    0      0;
  36    1.0635  6.83    5.6000    0.9688    0.0000    0.0000 0.00000
      0.00000 2  8    -5 1.0    0      0;
  37    1.0278  1.26    5.4000   -0.0444    0.0000    0.0000 0.00000
      0.00000 2  8    -5 1.0    0      0;
  38    1.0265  6.55    8.3000    0.1939    0.0000    0.0000 0.00000
      0.00000 2  8    -5 1.0    0      0;
  39    1.0300 -10.96   10.000    0.6846    11.040    2.5000 0.00000
      0.00000 1  15 -10 1.0    0      0];

% -----------
% line data
%b#1 b#2( res  )(  rea  )(charg )(tap )(phase)
line = [
   1    2 0.00350   0.04110 0.69870 0.0000 0    0     0     0;
   1   39 0.00100   0.02500 0.75000 0.0000 0    0     0     0;
   2    3 0.00130   0.01510 0.25720 0.0000 0    0     0     0;
   2   25 0.00700   0.00860 0.14600 0.0000 0    0     0     0;
   2   30 0.00000   0.01810 0.00000 1.0250 0    0     0     0;
   3    4 0.00130   0.02130 0.22140 0.0000 0    0     0     0;
   3   18 0.00110   0.01330 0.21380 0.0000 0    0     0     0;
   4    5 0.00080   0.01280 0.13420 0.0000 0    0     0     0;
   4   14 0.00080   0.01290 0.13820 0.0000 0    0     0     0;
   5    8 0.00080   0.01120 0.14760 0.0000 0    0     0     0;
   6    5 0.00020   0.00260 0.04340 0.0000 0    0     0     0;
   6    7 0.00060   0.00920 0.11300 0.0000 0    0     0     0;
   6   11 0.00070   0.00820 0.13890 0.0000 0    0     0     0;
   7    8 0.00040   0.00460 0.07800 0.0000 0    0     0     0;
   8    9 0.00230   0.03630 0.38040 0.0000 0    0     0     0;
   9   39 0.00100   0.02500 1.20000 0.0000 0    0     0     0;
  10   11 0.00040   0.00430 0.07290 0.0000 0    0     0     0;
  10   13 0.00040   0.00430 0.07290 0.0000 0    0     0     0;
  10   32 0.00000   0.02000 0.00000 1.0700 0    0     0     0;
  12   11 0.00160   0.04350 0.00000 1.0060 0    0     0     0;
  12   13 0.00160   0.04350 0.00000 1.0060 0    0     0     0;
  13   14 0.00090   0.01010 0.17230 0.0000 0    0     0     0;
  14   15 0.00180   0.02170 0.36600 0.0000 0    0     0     0;
  15   16 0.00090   0.00940 0.17100 0.0000 0    0     0     0;
  16   17 0.00070   0.00890 0.13420 0.0000 0    0     0     0;
  16   19 0.00160   0.01950 0.30400 0.0000 0    0     0     0;
  16   21 0.00080   0.01350 0.25480 0.0000 0    0     0     0;
  16   24 0.00030   0.00590 0.06800 0.0000 0    0     0     0;
  17   18 0.00070   0.00820 0.13190 0.0000 0    0     0     0;
  17   27 0.00130   0.01730 0.32160 0.0000 0    0     0     0;
  19   33 0.00070   0.01420 0.00000 1.0700 0    0     0     0;
  19   20 0.00070   0.01380 0.00000 1.0600 0    0     0     0;
  20   34 0.00090   0.01800 0.00000 1.0090 0    0     0     0;
```

```
  21   22 0.00080   0.01400 0.25650 0.0000 0   0    0    0;
  22   23 0.00060   0.00960 0.18460 0.0000 0   0    0    0;
  22   35 0.00000   0.01430 0.00000 1.0250 0   0    0    0;
  23   24 0.00220   0.03500 0.36100 0.0000 0   0    0    0;
  23   36 0.00050   0.02720 0.00000 1.0000 0   0    0    0;
  25   26 0.00320   0.03230 0.51300 0.0000 0   0    0    0;
  25   37 0.00060   0.02320 0.00000 1.0250 0   0    0    0;
  26   27 0.00140   0.01470 0.23960 0.0000 0   0    0    0;
  26   28 0.00430   0.04740 0.78020 0.0000 0   0    0    0;
  26   29 0.00570   0.06250 1.02900 0.0000 0   0    0    0;
  28   29 0.00140   0.01510 0.24900 0.0000 0   0    0    0;
  29   38 0.00080   0.01560 0.00000 1.0250 0   0    0    0;
  31    6 0.00000   0.02500 0.00000 1.0000 0   0    0    0];

% -----------
% machine data
% num bus b-mva x_1  r_a  x_d  x'_d  x"_d T'_do T"_do
%                        x_q  x'_q  x"_q T'_qo T"_qo
%                        H    d_o   d_1 type s s apf rpf
mac_con = [
  1  30  1000.0 0.130 0  1.000 0.31  0.31 10.20 0.00 ...
                        0.690 0.69  0.69  0    0 ...
                        4.200 0     0.00 2 0 0;
  2  31  1000.0 0.350 0  2.950 0.7   0.7 6.560 0.00 ...
                        2.820 1.7   1.7 1.500 0.00 ...
                        3.030 0     0.00 3 0 0;
  3  32  1000.0 0.30  0  2.500 0.530 0.530 5.700 0.0 ...
                        2.370 0.88  0.88   1.500 0.0 ...
                        3.580 0      0.00 3 0 0;
  4  33  1000.0 0.295 0  2.620 0.440 0.440 5.690 0.0 ...
                        2.580 1.66  1.66  1.500 0.0 ...
                        2.860 0     0.00 3 0 0;
  5  34  1000.0 0.540 0  6.700 1.320 1.32 5.400 0.0 ...
                        6.200 1.660 1.66 0.440 0.0 ...
                        2.600 0     0.00 3 0 0;
  6  35  1000.0 0.224 0  2.540 0.500  0.5  7.300 0.0 ...
                        2.410 0.810 0.81  0.400 0.0 ...
                        3.480 0     0.00 3 0 0;
  7  36  1000.0 0.322 0  2.900 0.490 0.49 5.660 0.0 ...
                        2.800 1.86  1.86 1.500 0.0 ...
                        2.640 0     0.00 3 0 0;
  8  37  1000.0 0.280 0  2.900 0.570 0.570 6.700 0.0 ...
                        2.800 0.91 0.91 0.410 0.0 ...
                        2.430 0     0.00 3 0 0;
  9  38  1000.0 0.298 0  2.106 0.570 0.570 4.790 0.0 ...
                        2.050 0.590 0.590 1.960 0.0 ...
                        3.450 0     0.00 3 0 0;
  10 39  1000.0 0.030 0  0.200 0.060 0.06  7.000 0 ...
                        0.190 0.080 0.08  0.700 0 ...
                        50.00 50.00 0.00 3 0 0;
];
```

```
140 |
141 | % -----------
142 | % exciter data
143 | % AC exciter (type 7 -- IEEE AC4a)
144 | exc_con = [...
145 |     7   1      0          0                 0.0      200            0.015
                0         0      0       0      0      0      10 ...
146 | -10.0    0      0      5.64    -4.53       0         0       0          0
            0       0      0      0       0 ...
147 |   0      0      0 ;%ac4
148 |     7   2      0          0                 0.0      200            0.015
                0         0      0       0      0      0      10 ...
149 | -10.0    0      0      5.64    -4.53       0         0       0          0
            0       0      0      0       0 ...
150 |   0      0      0 ;%ac4
151 |     7   3      0          0                 0.0      200            0.015
                0         0      0       0      0      0      10 ...
152 | -10.0    0      0      5.64    -4.53       0         0       0          0
            0       0      0      0       0 ...
153 |   0      0      0 ;%ac4
154 |     7   4      0          0                 0.0      200            0.015
                0         0      0       0      0      0      10 ...
155 | -10.0    0      0      5.64    -4.53       0         0       0          0
            0       0      0      0       0 ...
156 |   0      0      0 ;%ac4
157 |     7   5      0          0                 0.0      200            0.015
                0         0      0       0      0      0      10 ...
158 | -10.0    0      0      5.64    -4.53       0         0       0          0
            0       0      0      0       0 ...
159 |   0      0      0 ;%ac4
160 |     7   6      0          0                 0.0      200            0.015
                0         0      0       0      0      0      10 ...
161 | -10.0    0      0      5.64    -4.53       0         0       0          0
            0       0      0      0       0 ...
162 |   0      0      0 ;%ac4
163 |     7   7      0          0                 0.0      200            0.015
                0         0      0       0      0      0      10 ...
164 | -10.0    0      0      5.64    -4.53       0         0       0          0
            0       0      0      0       0 ...
165 |   0      0      0 ;%ac4
166 |     7   8      0          0                 0.0      200            0.015
                0         0      0       0      0      0      10 ...
167 | -10.0    0      0      5.64    -4.53       0         0       0          0
            0       0      0      0       0 ...
168 |   0      0      0 ;%ac4
169 |     7   9      0          0                 0.0      200            0.015
                0         0      0       0      0      0      10 ...
170 | -10.0    0      0      5.64    -4.53       0         0       0          0
            0       0      0      0       0 ...
171 |   0      0      0 ;%ac4
172 |   ];
```

```
173
174
175
176
177 gsib_idx = [];geib_idx = [];
178 load_con = [];
179 hgt_con = [];
180 svc_con = [];tcsc_con = [];upfc_con = [];
181 lmod_con= [];rlmod_con = [];
182 tgt_con = [];
183
184 % -----------
185 % pss data
186 % column 1 type 1for speed input
187 % column 2 generator number
188 % column 3 Gpss pss gain
189 % column 4 Tw
190 % column 5 Tn1
191 % column 6 Td1
192 % column 7 Tn2
193 % column 8 Td2
194 % column 9 ymax
195 % column 10 ymin
196 % columns 11 to 12 define a deltapomega filter if one is fitted
197 % column 11 Twd
198 % column 12 Tnf
199 % column 13 Tdf
200 % column 14 nnf
201 % column 15 ndf
202 % column 16 Gp
203
204
205
206 % Data from "Robust and coordinated tuning of PSS gains using
       sequenetial
207 % linear programming"
208 Tw = 3; Tn1 = 0.1; Td1 = 0.01; Tn2 = 0.1;
209 Td2 = 0.01; ymax = 5; ymin = -5;
210
211
212 % Target: MDR 10 - Table 2
213 pss_con = [...
214    1    1   12   Tw    Tn1   Td1   Tn2   Td2   ymax    ymin   0   0   0   0   0
          0;
215    1    2   12   Tw    Tn1   Td1   Tn2   Td2   ymax    ymin   0   0   0   0   0
          0;
216    1    3   12   Tw    Tn1   Td1   Tn2   Td2   ymax    ymin   0   0   0   0   0
          0;
217    1    4   10   Tw    Tn1   Td1   Tn2   Td2   ymax    ymin   0   0   0   0   0
          0;
218    1    5   12   Tw    Tn1   Td1   Tn2   Td2   ymax    ymin   0   0   0   0   0
```

```
        0;
219     1   6   12   Tw    Tn1   Td1   Tn2   Td2   ymax    ymin   0   0   0   0   0
        0;
220     1   7   11.03   Tw    Tn1   Td1   Tn2   Td2   ymax    ymin   0   0   0   0
        0   0;
221     1   8   9.51   Tw    Tn1   Td1   Tn2   Td2   ymax    ymin   0   0   0   0   0
          0;
222     1   9   12   Tw    Tn1   Td1   Tn2   Td2   ymax    ymin   0   0   0   0   0
        0
223 ];
224
225
226 dcl_con = [];dcr_con = [];dci_con = [];
227 dcld_con = [];dcrc_con = [];dcic_con = [];
```

APPENDIX

B

SUPPLEMENTARY MATERIAL FOR “EARLY WARNING OF MMWAVE SIGNAL BLOCKAGE USING DIFFRACTION PROPERTIES AND MACHINE LEARNING”

In this appendix, we provide additional numerical results and derivations that are referred to in the chapetr 3.

B.1 Simple Model Derivation

In this section, we present the derivation of the simple model mentioned at the end of Section 3.2 of the chapter 3. This simple model is used to provide insights into the Fresnel calculation results.

The oscillation patterns are obtained using the full Fresnel diffraction calculation [29] in the physical model with little insight as to their source. We can provide a rough, qualitative understanding of their origin by noting that finite element [57] calculations near a metal edge, which show the electric field distribution adjacent to the edge and thus indicate near-field sources for the Fresnel diffraction, exhibit a large electric field amplitude at the very edge. Thus, a qualitative approximation for the diffraction can be made by assuming that it generates a cylindrical passband wave $h_{edge}(t) = A_e/\sqrt{r(t)}\cos(\omega t - k\,r(t))$, with $r(t)$ the time-dependent distance from the blocker edge to the UE as in Fig. 1 of the main text, given by $r(t) = \sqrt{d^2 + v^2(t-t_t)^2 - 2d\,v(t-t_t)\sin(\alpha)}$ where $v = |\mathbf{v}|$, t_t is the time when the UE crosses the geometric transition, angular frequency $\omega = 2\pi f_c$, and $k = 2\pi/\lambda_c$. This wave is not a reflection, but an approximate near-field source for the Fresnel diffraction. It will interfere with the unscattered passband plane wave from the transmitter $h_T(t) = A\cos(\omega t - k\,u(t))$ traveling along the u-direction as shown in Fig. 1 of the main text, where $u(t) = d - v(t-t_t)\sin(\alpha)$. Thus, our simple model uses two plane waves propagating at an angle θ degrees with respect to each other as shown in Fig. 1 of chapter 3. In general, two waves interfere to create a pattern with oscillation spatial period that ranges from $\lambda/2$ at $\theta = 180°$ (opposite directions) to ∞ at $\theta = 0°$ (co-propagating). Here, the angle θ between the two waves, hence the oscillation frequency, decreases as the geometric transition is approached. Along the geometric transition line, both plane waves propagate in the same direction. Beyond the transition, the plane wave from the BS is blocked, so only the cylindrical wave remains and thus interference (hence oscillation) stops.

We now derive an expression for this simple approximation for the oscillation prior to blockage to obtain an approximate RSS squared. Define the distance between the blocker edge and the receiver path where the path crosses the geometric LoS/NLoS transition (see Fig. 3.1 of chapter 3) to be d. The angle α away from normal of the path crossing is also shown in Fig. 1 of the main paper. Let s denote a scalar distance from the UE to the geometric transition point along the path from LoS (x_0, y_0) to NLoS (x_1, y_1); it is negative in the LoS region, positive in NLoS.

The realistic physical model is based upon the coordinates (x, y) as shown in Fig. 3.1 of chapter 3, but these are not optimal for deriving the simple model. Instead, we create a set of coordinates (u, q) by translating (x, y) so that the origin is at the blocker edge $\mathbf{r}_{bl}$, then rotate so that the coordinate u is along the geometric transition from the blocker as shown in Fig. 3.1 of chapter 3. The coordinate q is perpendicular to u and also measured from the blocker but towards LoS. The distance from the (u, q) origin, $\mathbf{r}_{bl}$, to the mobile is $r = \sqrt{u^2 + q^2}$ (see Fig. 1 of the main paper). The interference between the two waves h_T and h_{edge} is given by the sum of the direct signal traveling along u and the cylindrical wave traveling along r. The cylindrical wave amplitude scales as $1/\sqrt{r}$. We also include a phase factor ϕ to account for the response of the scattering. The direct plane wave has amplitude A, and the amplitude of the cylindrical wave is thus $A_e = A\sqrt{d}$. Then the passband received signal corresponding to (3.1) in chapter 3 is given by

$$\begin{aligned}
\tilde{h}(u,v) =& A\,cos(\omega t - ku) + A\sqrt{d/r}\,cos(\omega t - kr + \phi) \\
=& A[cos(\omega t)\,cos(ku) + sin(\omega t)\,sin(ku) + \\
&+ \sqrt{d/r}\,cos(\omega t)\,cos(ku + \phi) + \\
&+ \sqrt{d/r}\,sin(\omega t)\,sin(ku + \phi)] \\
=& A\,cos(\omega t)[cos(ku) + \sqrt{d/r}\,cos(kr + \phi)] + \\
&+ A\,sin(\omega t)[sin(ku) + \sqrt{d/r}\,sin(kr + \phi)].
\end{aligned}$$

From the above, the squared envelope of the equivalent lowpass signal is given by

$$\begin{aligned}
h^2 =& A^2/2[cos^2(ku) + d/r\,cos^2(kr + \phi) + \\
&+ 2\sqrt{d/r}\,cos(ku)cos(kr + \phi) + sin^2(ku) + \\
&+ d/r\,sin^2(kr + \phi) + 2\sqrt{d/r}\,sin(ku)sin(kr + \phi)] \\
=& A^2/2[1 + d/r + \sqrt{d/r}\,cos(ku - kr - \phi)] \\
=& A^2/2[1 + d/r - \sqrt{d/r}\,cos(ku - kr)],
\end{aligned}$$

where the last line assumes that $\phi = 180°$ as expected and is consistent with the known result of $h^2 = A^2/2$ along the geometric transition where $u = d = r$. This result is placed onto the path of the mobile in the LoS region (where s is negative),

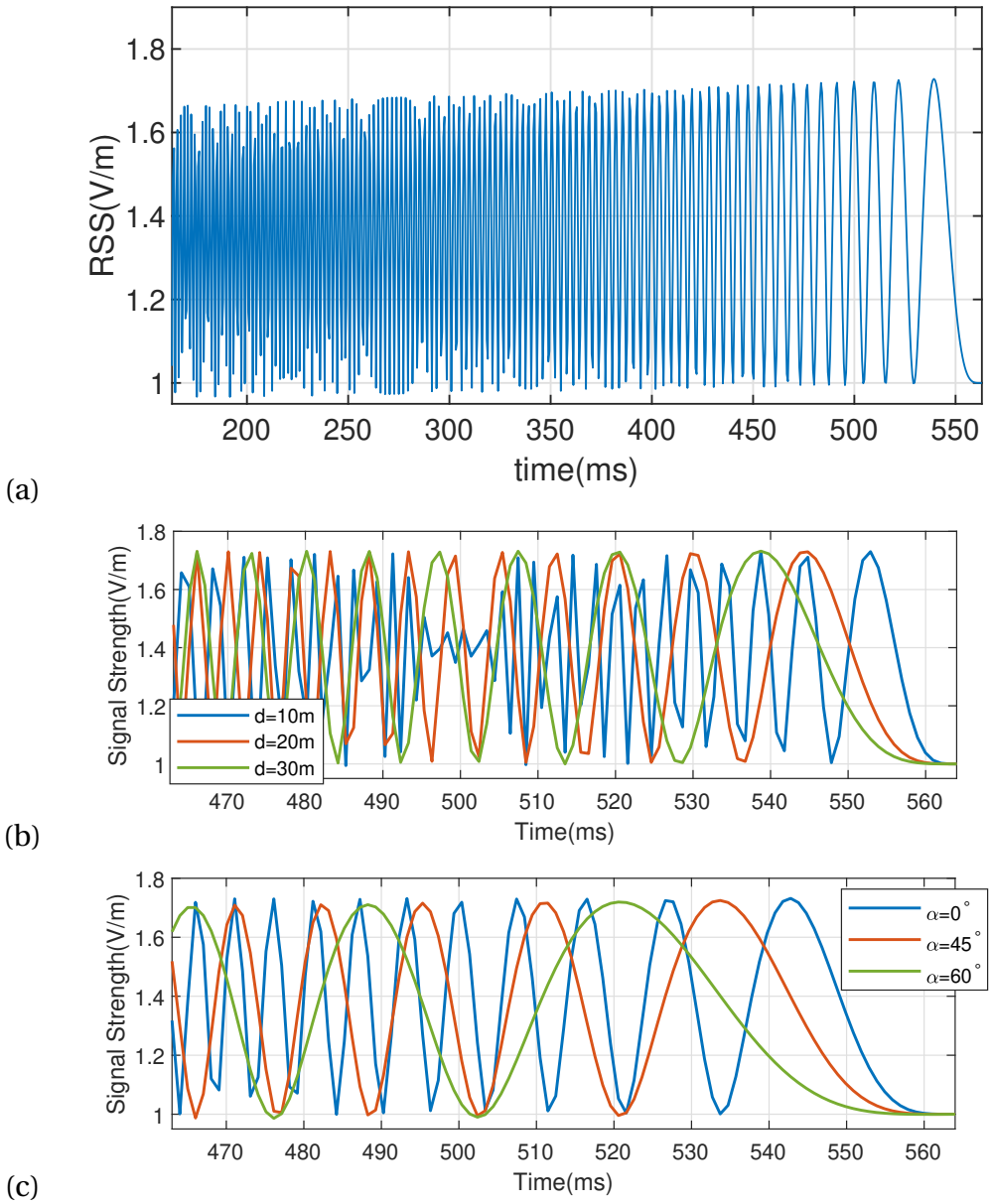
$$\begin{aligned}
u &= d - s\,sin(\alpha);\ q = -s\,cos(\alpha) \\
r &= \sqrt{d^2 - 2ds\,sin(\alpha) + s^2 sin^2(\alpha) + s^2 cos^2(\alpha)} \\
&= \sqrt{d^2 + s^2 - 2ds\,sin(\alpha)}.
\end{aligned}$$

To convert to the parameters of the main paper, we rewrite the mobile path parameter s as $s = -v(t - t_t)$ where here the speed is v and referenced to a time t_t when the UE crosses the geometric transition. Inserting this into the above, we obtain

$$|h(t)|^2 = A^2/2(1 + d/r(t) - \sqrt{d/r(t)}\cos(k\,u(t) - k\,r(t)),$$

where we emphasize that u and r are time-dependent via substituting for s: $u = d - v(t - t_t)\,sin(\alpha)$ and $r = \sqrt{d^2 + v^2(t - t_t)^2 - 2d\,v(t - t_t)\,sin(\alpha)}$.

Figure B.1(a) shows the result for the scenario of Figs. 3.1 and 3.2 of chapter 3. The simple model explains most of the qualitative features of the oscillation pattern, including frequency change and the variation with angle α and distance d. In Fig. B.1(a), we observe the reduction in the oscillation pattern envelope with time (distance) from $t_t = 563(ms)$ (geometric transition crossing) as t decreases, i.e., $t_t - t$ grows. While the $1/\sqrt{r(t)}$ dependence of the cylindrical wave $h_{edge}(t)$ could explain this decrease, the latter effect is much less pronounced than in the pattern of Fig. 3.2 of chapter 3, which is generated using a full diffraction calculation. This discrepancy is likely caused by the approximations of the simple model (B.1). In fact, the simple model also fails in the NLoS region. Contributions from regions away from the edge (in the full Fresnel formalism of our realistic physical model) are needed to properly match the actual signal. Thus, we do not use the simple model for any training or testing scenario creation (the realistic physical model is used for those cases). Results in Fig. B.1 can be compared to the Figs. 3.4(a-c) in chapter 3. The simple model is useful for insights into the origins of several key properties of the pattern used for early warning.



(a)

(b)

(c)

Figure B.1: Parameter variation calculations of the simple model for a 5 m path length ending at the geometric transition $s = 0$ with speed 28 m/s, the T_x at (x, y) = (-10, 0) m and the blocker at (0, 0) m so the (x, y) and (u, q) axes are the same. (a) Simple two-wave model. (b) Change d: $\alpha_1 = \alpha_2 = \alpha_3 = 0°; d_1 = 50m, d_2 = 30m, d_3 = 10m$. Compare to Fig. 4(a). (c) Change α: $\alpha_1 = 0°, \alpha_2 = 45°, \alpha_3 = 60°; d_1 = d_2 = d_3 = 40m$. Compare to Fig. 4(b).

B.2 ML Evaluation Metrics and Complexity

The four possible outcomes for the prediction problem are: true positive (TP), true negative (TN), false positive (FP), and false negative (FN) where the symbols TP, TN, FP, and FN represent the total number of corresponding outcomes for the test dataset (D_j). We use the following performance metrics for this analysis: accuracy= $\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$, f1 score= $\frac{\text{TP}}{\text{TP+0.5(FP+FN)}}$, and area under the curve (AUC), defined as the area under the receiver operating curve (ROC), which plots TPR = $\frac{\text{TP}}{\text{TP+FN}}$ vs FPR = $\frac{\text{FP}}{\text{FP+TN}}$ [13].

To study the effect of size of the dataset $\mathscr{D}$ on EW performance, in section 3.3.2, second paragraph, we vary N from 1000 to 100,000 and report the EW accuracy, f1 score, and AUC for each case. Table B.1 summarizes the simulation results. We observe that for $N < 10,000$ the EW performance degrades while increasing N over 10,000 does not change the EW performance significantly. Therefore, $N = 10,000$ is a suitable size for dataset $\mathscr{D}$. Similar conclusions were reached for other prediction times t_s, P, as well as multipath fading and noisy data.

Table B.1: EW performance vs dataset size N, $W = 400(ms)$, $f_s = 1$kHz, $t_1 = 250(ms)$, $P = 50(ms)$, dataset $\mathscr{D}$

N	Accuracy	f1 score	AUC
1000	89.74%	0.8259	0.8712
3000	97.87%	0.9641	0.9641
6000	98.86%	0.9793	0.9839
10000	99.53%	0.9919	0.9955
20000	99.60%	0.9928	0.9972
100000	99.61%	0.9928	0.9972

Finally, in our experiments for $N = 10,000$, physical model dataset generation, offline training, and testing take $1179(s)$, $62(s)$, and $0.69(s)$, respectively, for each EW result in Table

I of Chapter 3, demonstrating low computational complexity of the proposed ML method. Moreover, it takes 0.34(ms) to compute the EW result in real time, which is negligible compared to the prediction range of hundreds of milliseconds. The experiments are run using Python on a PC with 3.4 GHz Intel core-i7 processor and 16 GB of memory.

B.3 Performance for Fading Signals

The results of the experiment discussed in Section 3.4, paragraph 2 and Table 3.1 for fading dataset $\mathscr{D}_f$ is reported in Table B.2. Moreover, an example of a smoothed fading signal from dataset $\mathscr{D}_f$, sampled at 6KHz and passed through an L1 moving average filter with window size of 60 is shown in Fig B.2. ML performance for the resulting smoothed fading signal dataset $\mathscr{D}_s$ are reported in Table B.3.

Table B.2: Accuracy (%) of ML-based EW of blockage method; Fading received signal, $W = 400(ms)$, Rician fading, K=11dB, sampling rate f_s is shown in the top row.

P(ms)	t1(ms)	1kHz	2kHz	4kHz	6kHz	10kHz	15kHz
10	$100-250$	85.79	86.01	86.48	87.45	88.94	89.09
10	450	80.33	81.58	82.69	84.52	85.06	85.79
10	550	71.57	73.33	76.35	81.09	83.12	83.27
25	$100-250$	97.31	97.49	97.96	98.27	98.54	98.94
25	450	95.24	95.27	95.28	95.30	95.56	95.73
25	550	91.97	91.98	91.97	91.97	91.97	92.28
>50	$100-250$	99.53	99.54	99.56	99.59	99.61	99.63
>50	450	97.07	97.07	97.07	97.07	97.08	97.22
>50	550	94.39	94.39	94.41	94.43	94.88	94.90

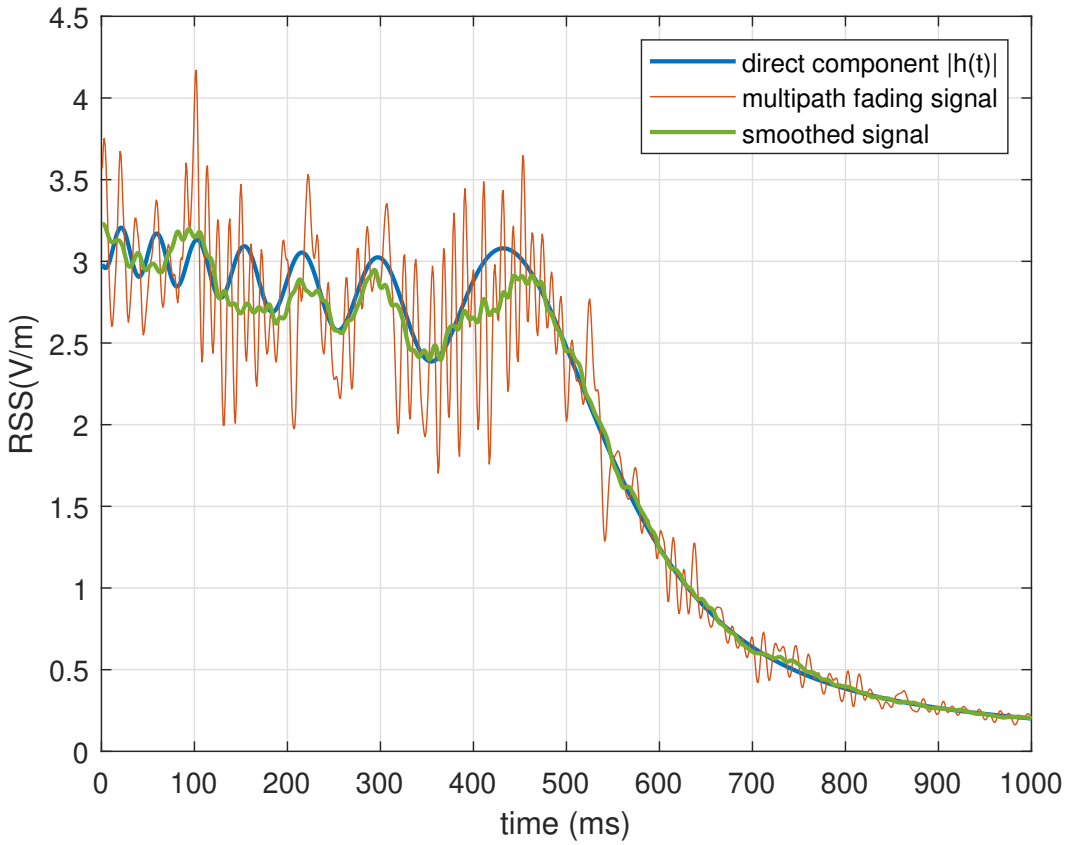


Figure B.2: RSS $|h(t)|$ for noiseless signal, multipath fading signal, and smoothed signal, transitioning from LoS to NLoS.

B.4 Effect of Speed on Early Warning

Additional statistics including the f1-score and AUC of the experiments described and presented in Table 3.2 are shown in Table B.4. We observe that the f1-score and AUC follow the same trends as the accuracy reported in Table 3.2 of the paper.

Table B.3: Accuracy (%) of ML-based EW of blockage method; Fading received signal first sampled at 6KHz and then passed through an L1 moving average filter with window size of 60, $W = 400(ms)$, Rician fading, K=11dB. Sub-sampling rate f_s is shown in the top row.

P(ms)	t1(ms)	1kHz	2kHz	4kHz	6kHz	10kHz	15kHz
10	$100-250$	84.22	85.47	86.22	86.93	87.21	87.99
10	450	79.75	80.58	81.29	93.67	84.26	84.93
10	550	70.89	72.59	75.74	80.56	82.81	82.54
25	$100-250$	96.67	96.57	97.07	97.97	98.42	98.76
25	450	94.68	94.57	95.13	95.27	95.44	95.67
25	550	90.73	90.98	91.08	91.57	91.66	92.02
>50	$100-250$	99.09	99.14	99.28	99.33	99.41	99.44
>50	450	96.67	96.99	97.01	97.01	97.02	97.03
>50	550	93.93	93.98	94.01	94.01	94.02	94.02

Table B.4: f1-score and AUC of ML-based EW of blockage method; Multipath fading (F) and non-fading(NF) datasets, $W = 400(ms)$, $t_1 = 250(ms)$, $P = 50(ms)$; Rician Fading, K = 11 dB. For $f_s \leq 200Hz$, fading signals are L1 filtered using sampling rate = 6KHz, window size = 60.

	$f_s \rightarrow$	1 KHz		200 Hz		100 Hz		50 Hz	
	speed $\downarrow$	NF	F	NF	F	NF	F	NF	F
f1-score	0-5	0.9989	0.9981	0.9707	0.9595	0.9685	0.9573	0.9621	0.9509
	5-10	0.9940	0.9925	0.9624	0.9490	0.9579	0.9445	0.9509	0.9375
	10-15	0.9933	0.9915	0.9457	0.9300	0.9422	0.9265	0.9311	0.9154
	15-20	0.9901	0.9887	0.9149	0.8858	0.9108	0.8817	0.9092	0.8801
	20-25	0.9883	0.9879	0.8579	0.8266	0.8503	0.8190	0.8404	0.8091
	25-30	0.9855	0.9849	0.7768	0.7410	0.7605	0.7247	0.7520	0.7162
AUC	0-5	0.9947	0.9932	0.9786	0.9674	0.9711	0.9599	0.9618	0.9506
	5-10	0.9918	0.9909	0.9664	0.9530	0.9526	0.9392	0.9499	0.9365
	10-15	0.9901	0.9898	0.9526	0.9369	0.9505	0.9348	0.9325	0.9168
	15-20	0.9854	0.9854	0.9297	0.9006	0.9258	0.8967	0.9155	0.8864
	20-25	0.9834	0.9833	0.8651	0.8338	0.8557	0.8244	0.8547	0.8234
	25-30	0.9820	0.9815	0.7802	0.7444	0.7729	0.7371	0.7601	0.7243

B.5 Performance metrics for noisy and fading measurements

Additional simulation for the f1-score and AUC of the experiments described in the last paragraph of Section 3.4 are provided in Fig. B.3

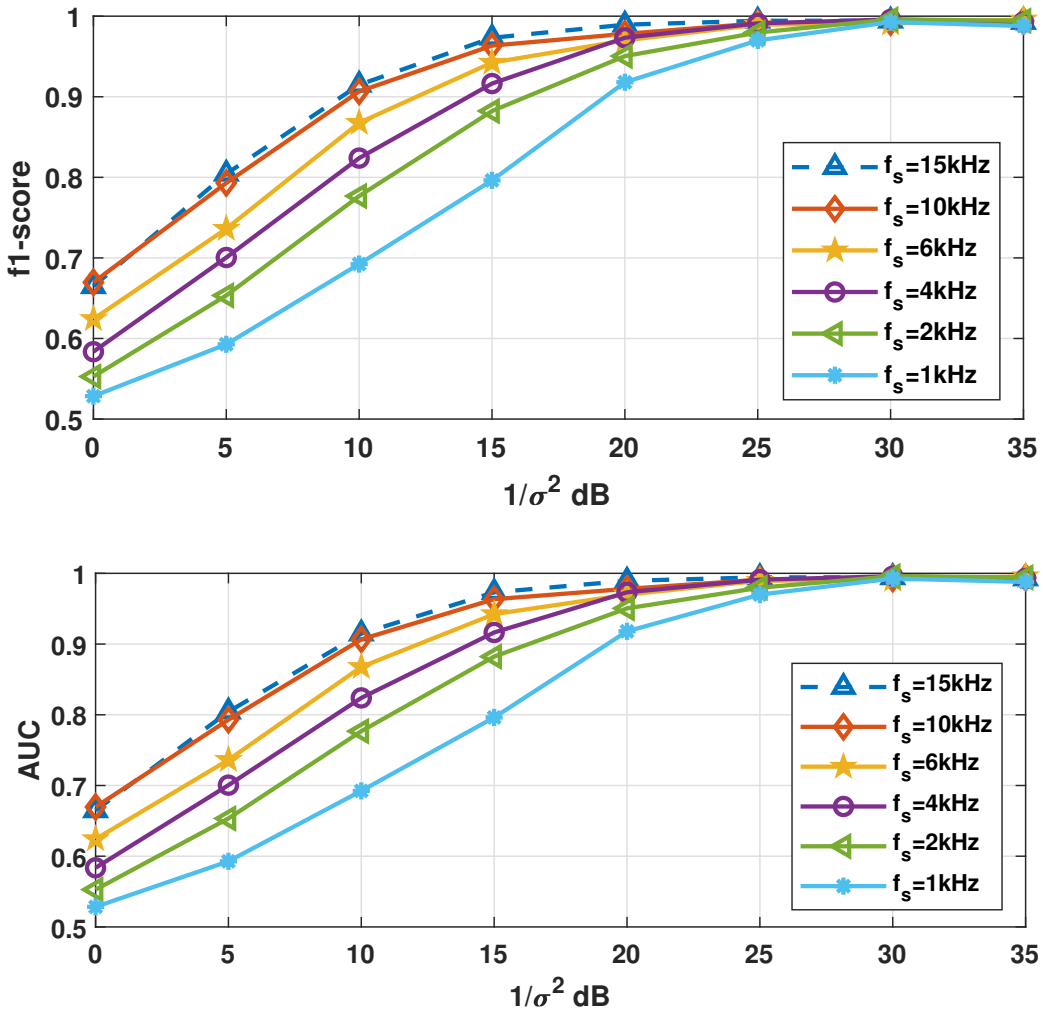


Figure B.3: F1 score and AUC of mmWave EW of blockage of multipath fading signal dataset $\mathscr{D}_f$ vs $1/\sigma^2$ for $f_s = \{1,2,4,6,10,15\}$ kHz, $W = 400(ms)$, $t_1 = 250(ms)$ and $P = 50(ms)$. See Fig. 6 in the paper.

APPENDIX

C

CODE FOR “EARLY WARNING OF MMWAVE SIGNAL BLOCKAGE USING DIFFRACTION PROPERTIES AND MACHINE LEARNING”

C.1 Dataset Generation

This section presents the Python code used in Chapter 3, Section 3.3.2 to generate the training and testing datasets. Note that the `mmWave` package and the `MMWaveRoutines` codes can be found online [30].

```
# -*- coding: utf-8 -*-
"""
Created on Tue Aug 10 18:27:58 2021

The speed of Tx and blocker are both randomized + various sampling rates

Results are saved in the GenData folder

@author: afallah
"""
from __future__ import division
import numpy as np
import subprocess as terminal
import math
import csv
from random import *
from scipy import signal
import MMWaveRoutines as mmWave
import random
import matplotlib.pyplot as plt
import time


'''----------Geometric Helper Functions--------'''
def slope(x1, y1, x2, y2): # Line slope given two points
    return (y2-y1)/((x2-x1)+ 0.00000001)

def angle(s1, s2): # angle given slopes
    return math.degrees(math.atan((s2-s1)/(1+(s2*s1))))


def rectsample(A, D): # sample uniformly randomly from inside a rectangle
    x  = random.uniform(A[0], D[0])
    y = random.uniform(A[1], D[1])
    return (x,y)


def line(p1, p2): # find lines given points
    A = (p1[1] - p2[1])
    B = (p2[0] - p1[0])
```

```
    C = (p1[0]*p2[1] - p2[0]*p1[1])
    return A, B, -C

def intersection(L1, L2): # find intersection of lines
    D  = L1[0] * L2[1] - L1[1] * L2[0]
    Dx = L1[2] * L2[1] - L1[1] * L2[2]
    Dy = L1[0] * L2[2] - L1[2] * L2[0]
    if D != 0:
        x = Dx / D
        y = Dy / D
        return x,y
    else:
        return False

def Intri(x1,y1,x2,y2,x3,y3,xp,yp): # this one works correct #
    check to see inside triangle
    c1 = (x2-x1)*(yp-y1)-(y2-y1)*(xp-x1)
    c2 = (x3-x2)*(yp-y2)-(y3-y2)*(xp-x2)
    c3 = (x1-x3)*(yp-y3)-(y1-y3)*(xp-x3)
    if (c1<0 and c2<0 and c3<0) or (c1>0 and c2>0 and c3>0):
        return True
    else:
        return False




### select a Transmitter point (fixed for now, then fixed y, the
    move 2D)
transmitterX1 = -10
transmitterX2 = -10
transmitterY1 = 0
transmitterY2 = 0

# Transmitter aperture [x1, y1, x2, y2, theta0_1,theta0_2, aperture
    size,  radius of sphere,  reflectivity, initial phase] # see
    slides
# the aperture size is set to be large to cover the desired area (
    see red line)
Ax1 = 40
Ax2 = 40
Ay1 = 100000
Ay2 = 100000
Asize = 200000
transmitterAperture = [Ax1, Ay1, Ax2, Ay2, 0, 0, Asize, 0, 1, 0]

# no reflectors so used the zero reflector setup
# zeroReflector = [1, 0, 1, 0, 0, 0, 0, 0, 0, 0]
#inputReflectors = [zeroReflector]

# for fading simulation
```

```
paperReflector1 = [90, 20, 90, 20, 90, 90, 2.5, 0, -1, 0]
paperReflector2 = [60.5, 30, 60.5, 30, 110, 110, 0.5, 0, -1, 0]
paperReflector3 = [20, 50, 20, 50, 135, 135, 2, 0, -1, 0]
paperReflector4 = [30, 10, 30, 10, 95, 95, 1.5, 0, -1, 0]
paperReflector5 = [130, 40, 130, 40, 45, 45, 2.2, 0, -1, 0]
paperReflector6 = [40, 25, 40, 25, 80, 80, 5, 0, -1, 0]
paperReflector7 = [110, 35, 110, 35, 120, 120, 6, 0, -1, 0]
#inputReflectors = [zeroReflector]
inputReflectors = [paperReflector1, paperReflector2,
    paperReflector3, paperReflector4, paperReflector5,
    paperReflector6, paperReflector7]


# Low and High frequency of simulation - here both 30 GHz
inputFreqLow = 30
inputFreqHigh = 30


# mobile moving path
#inputCalcLineX1 = 20
#inputCalcLineY1 = 10
#inputCalcLineX2 = 0
#inputCalcLineY2 = 10

#inputPoints = 1000


inputFrequencies = 1 # number of frequencies to simulate - we only
    have 30 GHz so 1
pathLength = 20 # to determine the path lenth in meters - also
    consider speed

NumSamp = 10 # number of cases in the dataset (N in the paper, e.g.
     10,000)

## this is the sampling rate or fs in the paper
## select single value or a list
RateList = [1000]# [1000, 2000, 4000, 6000]
for inputPoints in RateList:
    # initialize transmitter
    mmWave.addTransmitter(transmitterX1, transmitterY1)
    plt.axis('scaled')
    plt.xlim([-40, 100])
    plt.ylim([-40, 40])
    plt.grid()
    mmWave.showTransmitterRegion(transmitterAperture, transmitterX1
    , transmitterY1) # plot transmitter

    # add some mobile start points in between
    transmitterLength = transmitterAperture[6]
    apertureY1 = transmitterAperture[1] + transmitterLength/2
```

```
132     apertureY2 = transmitterAperture[1] - transmitterLength/2
133     apertureX = transmitterAperture[0]
134     # slopes
135     s1 = (apertureY1-transmitterY1)/(transmitterAperture[0]-
        transmitterX1)
136     s2 = (apertureY2-transmitterY1)/(transmitterAperture[0]-
        transmitterX1)
137     # intercepts
138     b1 = transmitterY1 - s1*transmitterX1
139     b2 = transmitterY1 - s2*transmitterX1
140     # move the transmitter aperture (see blue line commented)
141     Bx = transmitterX1+22
142     By = s1*(Bx) + b1
143     Cx = transmitterX1+22
144     Cy = s2*(Cx) + b2
145
146     # myline = plt.Line2D((Bx,Cx),(By,Cy),color='b') # data coords
147     # plt.gca().add_artist(myline)
148
149     myline = plt.Line2D((Ax1 ,Ax2),(Ay1+Asize/2,Ay2-Asize/2),color=
        'r') # data coords
150     plt.gca().add_artist(myline)
151
152     # sample RX starting points uniformly randomly from inside the
        rectangle
153     points= []
154     for pcount in range(NumSamp):
155         point = rectsample((Ax1+5, Asize/2 - Ay1 + 5), (100, 15)) #
        samples from rectangle
156         points.append(point)
157
158     xx, yy = zip(*points)
159     plt.scatter(xx, yy, s=1)
160
161
162     print('number of points', len(points))
163
164     endpoints = []
165     ValidStartPoints = []
166     transitions = []
167     trans_idx = []
168     LowF =[]
169     HighF =[]
170     Ang = []
171     TxPoints = []
172     ApPoints =[]
173     end1 = []
174     thetas = np.arange(-math.pi/2, math.pi/2+0.1, math.pi/10)
175
176     start_time = time.time() # calculate computation time
177
```

```
178     for i in range(len(points)):
179         # random Rx path
180         theta = random.uniform(-math.pi, math.pi+0.1) # randim
    direction
181         pathLength = random.randint(0,30) # random speed between 0
    and 30 m/s
182         # select a start point for Rx and find end point
183         endx1 = pathLength*math.cos(theta) + points[i][0] #
    calculate Rx endpoit based on direction and speed
184         endy1 = pathLength*math.sin(theta) + points[i][1] #
    calculate Rx endpoit based on direction and speed
185         # randomize Tx to blocker distance
186         transmitterX1 = random.uniform(-10, 30) # Tx location
    randomize on the x-axis
187         transmitterY1 = 0 # y-axis is fixed for the Tx location
188         # move Tx same as Rx (relative motion to simulate blocker
    movements)
189         transmitterX2 = transmitterX1 + pathLength*math.cos(theta)
190         transmitterY2 = transmitterY1 + pathLength*math.sin(theta)
191         # find blocker velocity vector
192         thetaAp = random.uniform(-math.pi/2, math.pi/2+0.1)
193         # random blocker speed
194         BlockerPathLen = random.randint(0,30)# random speed between
     0 and 30 m/s
195         endx = endx1 + BlockerPathLen*math.cos(thetaAp) # calculate
     Blocker endpoit based on direction and speed
196         endy = endy1 + BlockerPathLen*math.sin(thetaAp) # calculate
     Blocker endpoit based on direction and speed
197
198         # check if points are valid - should not go behind aperture
199         if endx > Ax2+2 and transmitterX2 < Ax2-2:
200             end1.append((endx1, endy1))
201             endpoints.append((endx, endy))
202             ValidStartPoints.append((points[i][0], points[i][1]))
203             # plot the mobile trajectory
204             # myline = plt.Line2D((points[i][0],endx),(points[i
    ][1], endy),color='r') # data coords
205             # plt.gca().add_artist(myline)
206             # check if transition from LOS to NLOS happens and add
    label to dataset
207             yupper = s1*(endx) + b1
208             ylower = s2*(endx) + b2
209             # print(yupper)
210             # print(ylower)
211             # print(s1, s2, b1, b2)
212             # plt.scatter(endx, ylower, s=15)
213             # plt.scatter(endx, yupper, s=15)
214
215
216             # create the LoS triangle
```

```
217             # find lower and upper sides of the trianle (line
    equation)
218             # between endpoint of the trainsmitter and the aperture
     end points
219             lower_side = line([transmitterX2, transmitterY2],[
    apertureX, apertureY2])
220             upper_side = line([transmitterX2, transmitterY2],[
    apertureX, apertureY1])
221             x_tri = 5000
222             y_lower = (lower_side[2] - lower_side[0]*5000)/
    lower_side[1]
223             y_upper = (upper_side[2] - upper_side[0]*5000)/
    upper_side[1]
224
225             Los_lable = Intri(transmitterX2, transmitterY2, x_tri,
    y_lower, x_tri, y_upper, endx1, endy1)
226             print(Los_lable)
227
228
229             if Los_lable:  # check if LoS to NLoS transition
    happened (upper edge)
230                 transitions.append(1)
231                 # find the intersection points
232                 L1 = line([points[i][0],points[i][1]],[endx, endy])
233                 L2 = line([transmitterX1, transmitterY1],[Bx, By])
234                 Isec = intersection(L1, L2)
235                 # print(Isec)
236                 # find the index where the transition happens
237                 idex_trans = np.floor(((Isec[0] - points[i][0])
    *1000)/(endx-points[i][0]))
238                 # print(idex_trans)
239                 trans_idx.append(idex_trans)
240
241             else: # if no transition happened add 0 as label
242                 transitions.append(0)
243                 idex_trans = -1
244                 # print(idex_trans)
245                 trans_idx.append(idex_trans)
246
247
248
249             # get signal magnitudes using model
250             # select point in the rectangle
251             inputCalcLineX1 = points[i][0]
252             inputCalcLineX2 = endx
253             inputCalcLineY1 = points[i][1]
254             inputCalcLineY2 = endy
255             # run simulation
256             freqMags, frequencies = mmWave.runSimulation(
    inputCalcLineX1, inputCalcLineX2, inputCalcLineY1,
    inputCalcLineY2, inputFreqHigh, inputFreqLow, transmitterX1,
```

```
    transmitterX2, transmitterY1, transmitterY2, inputPoints,
    inputFrequencies, inputReflectors, transmitterAperture)
257             # mmWave.plotData(freqMags, frequencies,inputCalcLineX1
    , inputCalcLineX2, inputCalcLineY1, inputCalcLineY2, inputPoints
    )
258             plt.figure()
259             plt.plot(freqMags[0])
260             plt.show()
261             LowF.append(freqMags[0])
262             # HighF.append(freqMags[1])
263             # calculate the angle
264             sl1 = slope(inputCalcLineX1, inputCalcLineY1,
    inputCalcLineX2, inputCalcLineY2)
265             sl2 = slope(transmitterX1, transmitterY1,
    transmitterAperture[0], transmitterAperture[6]/2 -
    transmitterAperture[1])
266             ang = angle(sl1, sl2)
267             Ang.append(ang)
268
269             # collect Tx and aperture location info
270             TxPoints.append((transmitterX1, transmitterY1,
    transmitterX2, transmitterY2))
271             ApPoints.append((transmitterAperture[0],
    transmitterAperture[1], transmitterAperture[2],
    transmitterAperture[3], transmitterAperture[6]))
272
273
274     print('Computation time', time.time()-start_time)
275
276     print('number of valid points', len(ValidStartPoints))
277     # plt.xlim([4,8])
278     # plt.ylim([-8,8])
279     # count = 0
280     # for point in points:
281     #     if Intri(20,0,80,30,80,-30, point[0], point[1]) == True:
282     #         count = count+1
283     # print(count)
284     # xx2, yy2 = zip(*endpoints)
285     # plt.scatter(xx2, yy2, s=1, color='y')
286
287     # xx3, yy3 = zip(*end1)
288     # plt.scatter(xx3, yy3, s=1, color='g')
289
290
291     # PlotTx = np.array(TxPoints)
292     # plt.scatter(PlotTx[:,0], PlotTx[:,1], s=1, color='r')
293     # plt.scatter(PlotTx[:,2], PlotTx[:,3], s=1, color='r')
294
295
296     # plt.xlim([-20,30])
297     # plt.ylim([-30,30])
```

```
298     # plt.show()
299
300
301     # print(transitions)
302     # print(trans_idx)
303
304     # # select point in the triangle
305     # inputCalcLineX1 = points[0][0]
306     # inputCalcLineX2 = endpoints[0][0]
307
308     # inputCalcLineY1 = points[0][1]
309     # inputCalcLineY2 = endpoints[0][1]
310
311     # freqMags, frequencies = mmWave.runSimulation(inputCalcLineX1,
        inputCalcLineX2, inputCalcLineY1, inputCalcLineY2,
        inputFreqHigh, inputFreqLow, transmitterX1, transmitterX2,
        transmitterY1, transmitterY2, inputPoints, inputFrequencies,
        inputReflectors, transmitterAperture)
312     # # Run the program with frequency averaging, frequency range
        in globals
313     # # freqMags, frequencies = mmWave.averageOverRange(5,
        inputCalcLineX1, inputCalcLineX2, inputCalcLineY1,
        inputCalcLineY2, inputFreqHigh, inputFreqLow, transmitterX1,
        transmitterX2, transmitterY1, transmitterY2, inputPoints,
        inputFrequencies, inputReflectors, transmitterAperture)
314
315     # # Plot, also can calculate delays and show them, largest
        contributors, correlations
316     # mmWave.plotData(freqMags, frequencies,inputCalcLineX1,
        inputCalcLineX2, inputCalcLineY1, inputCalcLineY2, inputPoints)
317     # plt.show()
318
319
320     # remove incomple rows from dataset
321     LowL = [len(l) for l in LowF]
322     Lowt = np.where(np.array(LowL) != inputPoints)
323     for Lenidx in sorted(list(Lowt[0]), reverse=True):
324         del LowF[Lenidx]
325         del trans_idx[Lenidx]
326         del transitions[Lenidx]
327         del ValidStartPoints[Lenidx]
328         del endpoints[Lenidx]
329         del Ang[Lenidx]
330         del TxPoints[Lenidx]
331         del ApPoints[Lenidx]
332
333     # collect all data
334     DataTable = np.column_stack((np.array(LowF),np.array(trans_idx)
        , np.array(transitions), np.array(ValidStartPoints), np.array(
        endpoints), np.array(Ang), np.array(TxPoints), np.array(ApPoints
        )))
```

```
    #np.savetxt('mmW_new_ds4.csv', DataTable, delimiter=",")
    import os.path as pth
    filename = "fading/mmW_fad_" # set 1st part of file name to
   save datasets
    filenum = 1
    while pth.exists("GenData/" +filename+str(filenum)+".csv"): #
   to avoid saving over available files
        filenum+=1

    np.savetxt("GenData/" + filename+str(filenum)+".csv", DataTable
   , delimiter=",")
```

C.2 Training and Testing

This section provides the Python code used to generate the numerical results of Section 3.4 and Table 3.1.

```
# -*- coding: utf-8 -*-
"""
Created on 2/22/2022

@author: afallah

with P=[t1, t2] implemented
to test various W, t1, and P values
"""
# ====================================================
# Load the mmW_ahf_a data file and perform data selection and
   cleaning
# ====================================================

import numpy as np
import subprocess as terminal
import math
import csv
from random import *
from scipy import signal
import random
import pandas as pd
#from sktime.utils.data_processing import *
from sktime.datatypes._panel._convert import
   from_2d_array_to_nested
from sktime.datatypes._panel._check import are_columns_nested,
   is_nested_dataframe
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, roc_auc_score
```

```
27 import matplotlib.pyplot as plt
28
29 # load dataset generated previously
30 df = pd.read_csv('GenData/mmW_mb6_1.csv', index_col=None, header=
       None)
31
32
33 df = df.iloc[:,:]
34
35 Nsample = len(df)
36 Xdf = df.iloc[0:Nsample,0:1000].values.tolist()
37 Ydf = df.iloc[0:Nsample,1001].values.tolist()
38 Tdf = df.iloc[0:Nsample,1000]
39
40
41 Y_new = []
42 T_new = []
43 Index_new = []
44 for i in range(len(Xdf)):
45     if Xdf[i][-1] < 0.37*Xdf[i][0]:
46         Y_new.append(1)
47         Index_new.append(min(range(len(Xdf[i])), key=lambda j: abs(
       Xdf[i][j]-0.37*Xdf[i][0]))) # find the index based on new
       transition definition
48     else:
49         Y_new.append(0)
50         Index_new.append(-1)
51 # print("relabel Accuracy :", f1_score(Y_new, Ydf, average=None)) #
       print the test accuracy score
52 df2 = pd.concat([pd.DataFrame(Xdf), pd.DataFrame(Index_new), pd.
       DataFrame(Y_new)], axis = 1, ignore_index=True)
53
54
55 def extract_rows(dfl, starts, L, fillval=np.nan):
56     a = dfl.values
57
58     idx = np.asarray(starts)[:,None] + range(L)
59     valid_mask = idx < dfl.shape[1]
60     idx[~valid_mask] = 0
61
62     val = a[np.arange(len(idx))[:,None],idx]
63     return np.where(valid_mask, val, fillval)
64
65
66
67 # observation window
68 W = 400
69 # prediction t1
70 t1 = 250
71 # prediction window width
72 P = 50 # P=t2-t1
```

```
73
74 ddf = df2.iloc[0:Nsample,0:1002]
75 #NLos = ddf.loc[ddf[1000]>=W+t1+P] # select rows with transition
       index>W+t1 (Los to NLos)
76 NLos = ddf.loc[ddf[1000]>=max(400+t1+P,W+t1+P)] # fix W so all
       datasets have same number of cases
77 Los = ddf.loc[ddf[1000]==-1] # select rows with no transition (Los
       tp Los)
78
79 tran_idx = list(NLos.iloc[:,-2].astype(int)) # transition index of
       the Los to Nlos
80 r = np.random.randint(0,P,size=len(NLos)) # randomly set the
       transition index within P
81
82 # extraxt observation window W
83 TLow = extract_rows(NLos, starts=tran_idx-r-W-t1, L=W , fillval=np.
       nan)
84
85 # add label 1 for transition Los to NLos
86 clippedNLos = np.column_stack((TLow, np.ones(len(TLow))))
87
88 # For Los
89 # Select a point randomly in the signal as starting point
90 ntran_idx = list(np.random.randint(0,df.shape[1]-17-W, size=len(Los
       )))
91 NTLow = extract_rows(Los, starts=ntran_idx, L=W , fillval=np.nan)
92
93 # add label 0 for no transition
94 clippedLos = np.column_stack((NTLow, np.zeros(len(NTLow))))
95
96 # stack data
97 ClippedData = np.vstack((clippedNLos, clippedLos))
98 # randomly shuffle rows of data
99 np.take(ClippedData,np.random.permutation(ClippedData.shape[0]),
       axis=0,out=ClippedData)
100
101 # create X and Y for dataset
102 X = ClippedData[:,:W ]
103 Y = ClippedData[:,-1]
104
105
106 # print('before balance: \n', sum(Y), len(Y))
107 # # balance the dataset
108 # from imblearn.under_sampling import RandomUnderSampler
109 # under_sampler = RandomUnderSampler(sampling_strategy=0.5,
       random_state=42)
110 # X_res, y_res = under_sampler.fit_resample(X, Y)
111 # X, Y = X_res, y_res
112 # print('after balance: \n', sum(y_res), len(y_res))
113
114 y_nested = Y.reshape((-1,))
```

```
115
116
117
118
119
120 #X_window = ClippedData[:,Wi:]
121
122 X_nested = from_2d_array_to_nested(X)
123 print(f"X_nested is a nested DataFrame: {is_nested_dataframe(
      X_nested)}")
124 print(f"The cell contains a {type(X_nested.iloc[0,0])}.")
125 print(f"The nested DataFrame has shape {X_nested.shape}")
126
127
128
129
130
131 '''Train/test split percentage'''
132 Ntrain= 0.25 # split of the test set
133 ''' split the dataset to train/test split'''
134 # to change the percentage, change Ntrain at top of the code
135 X_train, X_test, y_train, y_test = train_test_split(X_nested,
      y_nested, test_size=Ntrain, random_state=123)
136 print(X_train.shape, y_train.shape, X_test.shape, y_test.shape) #
      print the shape of train/test sets
137
138
139
140
141
142 '''------------------------------------------------'''
143 '''------------------miniROCKET------------------'''
144 '''------------------------------------------------'''
145 from sktime.transformations.panel.rocket import MiniRocket,
      MiniRocketMultivariate
146 from sklearn.linear_model import RidgeClassifierCV
147 # define miniROKCET transformer onject
148 minirocket_multi = MiniRocket(num_features=20000)
149 # fit the miniROCKET transformer on the training data
150 minirocket_multi.fit(X_train)
151 X_train_transform = minirocket_multi.transform(X_train) # extract
      features from the train data
152 # define the Ride Classifier object and fit in the the extracted
      features
153 classifier = RidgeClassifierCV(alphas=np.logspace(-3, 3, 10),
      normalize=True)
154 classifier.fit(X_train_transform.values, y_train)# train the
      classifier on the extracted features
155
156 X_test_transform = minirocket_multi.transform(X_test) # extract
      features from the test set
```

```
Accu = classifier.score(X_test_transform.values, y_test)
print("Mini-Rocket Accuracy :", Accu) # print the test accuracy
    score
y_pred = classifier.predict(X_test_transform.values)
F1 = f1_score(y_test, y_pred, average=None)
F1_mean = np.mean(F1)
print('f1-score: ', F1, F1_mean)
AucScore = roc_auc_score(y_test, y_pred)
print('AUC: ', AucScore)
```

C.3 Test the effect of Noise and Multipath Fading

This section provides the Python code used to generate the numerical results of Section 3.4, Table 3.2 and Fig. 3.6 of chapter 3.

```
# -*- coding: utf-8 -*-
"""
Created on 1/3/2022 add noise to moving blocker & Rx case
Also study various sampling rate vs noise

This is the main file for fig 5 of draft

@author: afallah
"""

import numpy as np
import subprocess as terminal
import math
import csv
from random import *
from scipy import signal
import random
import pandas as pd
#from sktime.utils.data_processing import *
from sktime.datatypes._panel._convert import
    from_2d_array_to_nested
from sktime.datatypes._panel._check import are_columns_nested,
    is_nested_dataframe
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, roc_auc_score
import matplotlib.pyplot as plt
from numpy import sum,isrealobj,sqrt
from numpy.random import standard_normal


```

```
30
31 # ============================================================
32 # Adding the AWGN noise
33 # ============================================================
34
35 def awgn(s,SNRdB,P,L=1):
36     """
37     AWGN channel
38     Add AWGN noise to input signal. The function adds AWGN noise
    vector to signal 's' to generate a resulting signal vector 'r'
    of specified SNR in dB. It also
39     returns the noise vector 'n' that is added to the signal 's'
    and the power spectral density N0 of noise added
40     Parameters:
41         s : input/transmitted signal vector
42         SNRdB : desired signal to noise ratio (expressed in dB) for
    the received signal
43         L : oversampling factor (applicable for waveform simulation
    ) default L = 1.
44     Returns:
45         r : received signal vector (r=s+n)
46 """
47     gamma = 10**(SNRdB/10) #SNR to linear scale
48     # if s.ndim==1:# if s is single dimensional vector
49     #     P=L*sum(abs(s)**2)/len(s) #Actual power in the vector
50     # else: # multi-dimensional signals like MFSK
51     #     P=L*sum(sum(abs(s)**2))/len(s) # if s is a matrix [MxN]
52     N0=P/gamma # Find the noise spectral density
53     if isrealobj(s):# check if input is real/complex object type
54         n = sqrt(N0/2)*standard_normal(s.shape) # computed noise
55     else:
56         n = sqrt(N0/2)*(standard_normal(s.shape)+1j*standard_normal
    (s.shape))
57     r = s + n # received signal
58     return r
59
60
61
62 NumDatasets = 6 # number of datasets to loop over e.g. 14 in
    GenData/fs
63
64 ResAcc = np.zeros([NumDatasets,10])
65 Resf1 = np.zeros([NumDatasets,10])
66 ResAuc = np.zeros([NumDatasets,10])
67 icount = 0
68 jcount = 0
69
70 # loop over datasets
71 for i in range(1,NumDatasets+1):
72     dataFile = 'C:/Users/Amirhassan/Desktop/fs-fading/MatlabFading/
    fad_'+str(i)+'.csv' # define folder path and fixed name part
```

```
73      # dataFile = 'GenData/fs/mmW_mb6_15k.csv'
74
75
76      df = pd.read_csv(dataFile, index_col=None, header=None) # read
        csv file
77
78
79      df2 = df.iloc[0:10000,:] # Clip to N=10k if larger
80
81      #extracts samples per case from dataset
82      Sr = df2.shape[1] - 16
83
84      Nsample = len(df) # size of N
85      Xdf = df2.iloc[0:Nsample,0:Sr].values.tolist()
86      Ydf = df2.iloc[0:Nsample,Sr+1].values.tolist()
87      Tdf = df2.iloc[0:Nsample,Sr]
88
89      # make sure LoS to NLoS labels are correct
90      Y_new = []
91      Index_new = []
92      for i in range(len(Xdf)):
93          if Xdf[i][-1] < 0.37*Xdf[i][0]:
94              Y_new.append(1)
95              Index_new.append(min(range(len(Xdf[i])), key=lambda j:
        abs(Xdf[i][j]-0.37*Xdf[i][0]))) # find the index based on new
        transition definition
96          else:
97              Y_new.append(0)
98              Index_new.append(-1)
99      # print("relabel Accuracy :", f1_score(Y_new, Ydf, average=None
        )) # print the test accuracy score
100     df2 = pd.concat([pd.DataFrame(Xdf), pd.DataFrame(Index_new), pd
        .DataFrame(Y_new)], axis = 1, ignore_index=True)
101
102
103
104
105
106     # ========================================================
107     # Select the NLOS to LOS with trans_index >= 400
108     # Select rows with no transition
109     # ========================================================
110     L = int(0.4*Sr) #400
111     W = int(0.45*Sr) #450
112
113
114     ddf = df2.iloc[0:Nsample,0:Sr+2]
115     NLos = ddf.loc[ddf[Sr]>=W] # select rows with transition index
        >400 (Los to NLos)
116     Los = ddf.loc[ddf[Sr]==-1] # select rows with no transition (
        Los tp Los)
```

```
117
118     # clip the signals to 400 samples before the transition
119     # or 400 samples from beginning of no transition cases (maybe
        select random interval in the future)
120     def extract_rows(dfl, starts, L, fillval=np.nan):
121         a = dfl.values
122
123         idx = np.asarray(starts)[:,None] + range(L)
124         valid_mask = idx < dfl.shape[1]
125         idx[~valid_mask] = 0
126
127         val = a[np.arange(len(idx))[:,None],idx]
128         return np.where(valid_mask, val, fillval)
129
130
131     # 400 sample window + Ld (50) = 450
132     tran_idx = list(NLos.iloc[:,-2].astype(int)-W) # transition
        index of the Los to Nlos
133     #tran_idx2 = list(NLos.iloc[:,-2].astype(int)+1000-450) #
        transition index of the Los to Nlos
134
135     TLow = extract_rows(NLos, starts=tran_idx, L=L , fillval=np.nan
        )
136     #THigh =  extract_rows(NLos, starts=tran_idx2, L=400, fillval=
        np.nan)
137
138     clippedNLos = np.column_stack((TLow, np.ones(len(TLow))))
139
140     # make this selection random as well
141     ntran_idx = list(np.random.randint(0,int(0.599*Sr),size=len(Los
        )))
142     #ntran_idx2 = list(np.asarray(ntran_idx)+1000)
143     NTLow = extract_rows(Los, starts=ntran_idx, L=L , fillval=np.
        nan)
144     #NTHigh =  extract_rows(Los, starts=ntran_idx2, L=400, fillval=
        np.nan)
145
146     clippedLos = np.column_stack((NTLow, np.zeros(len(NTLow))))
147
148
149     ClippedData = np.vstack((clippedNLos, clippedLos))
150
151     np.take(ClippedData,np.random.permutation(ClippedData.shape[0])
        ,axis=0,out=ClippedData)
152
153
154     X = ClippedData[:,:L ]
155     Y = ClippedData[:,-1]
156
157
158     # print('before balance: \n', sum(Y), len(Y))
```

```
159     # # balance the dataset
160     from imblearn.under_sampling import RandomUnderSampler
161     under_sampler = RandomUnderSampler(sampling_strategy=1,
    random_state=42)
162     X_res, y_res = under_sampler.fit_resample(X, Y)
163     X, Y = X_res, y_res
164     # print('after balance: \n', sum(y_res), len(y_res))
165
166
167     # loop over SNR
168     for SNR in range(0,50,5):
169         ''''----------------------------------------'''
170         '''--------------------Noisy Data for test------------------
    '''
171         '''----------------------------------------'''
172         # calculate the average LoS power for SNR
173         Pn=(sum(sum(abs(X)**2))/len(X))/X.shape[1]
174         # add the AWGN noise
175         gamma = 10**(SNR/10)
176         N0=1/gamma
177         noi = sqrt(N0/2)*standard_normal(X.shape)
178         print(sqrt(N0/2))
179         Xn = X + noi
180
181
182
183         # put to format od sktime nested data
184         X_nested = from_2d_array_to_nested(Xn)
185         print(f"X_nested is a nested DataFrame: {
    is_nested_dataframe(X_nested)}")
186         print(f"The cell contains a {type(X_nested.iloc[0,0])}.")
187         print(f"The nested DataFrame has shape {X_nested.shape}")
188         y_nested = Y.reshape((-1,))
189
190
191
192         '''Train/test split percentage'''
193         Ntrain= 0.25 # split of the test set
194         ''' split the dataset to train/test split'''
195         # to change the percentage, change Ntrain at top of the
    code
196         X_train, X_test, y_train, y_test = train_test_split(
    X_nested, y_nested, test_size=Ntrain, random_state=123)
197         print(X_train.shape, y_train.shape, X_test.shape, y_test.
    shape) # print the shape of train/test sets
198
199
200
201
202         '''----------------------------------------------'''
203         '''--------------------miniROCKET------------------'''
```

```
204         '''-----------------------------------------------'''
205         from sktime.transformations.panel.rocket import MiniRocket,
     MiniRocketMultivariate
206         from sklearn.linear_model import RidgeClassifierCV
207         # define miniROKCET transformer onject
208         minirocket_multi = MiniRocket(num_features=20000)
209         # fit the miniROCKET transformer on the training data
210         minirocket_multi.fit(X_train)
211         X_train_transform = minirocket_multi.transform(X_train) #
    extract features from the train data
212         # define the Ride Classifier object and fit in the the
    extracted features
213         classifier = RidgeClassifierCV(alphas=np.logspace(-3, 3,
    10), normalize=True)
214         classifier.fit(X_train_transform.values, y_train)# train
    the classifier on the extracted features
215
216         X_test_transform = minirocket_multi.transform(X_test) #
    extract features from the test set
217         Accu = classifier.score(X_test_transform.values, y_test)
218         print("Mini-Rocket Accuracy :", Accu) # print the test
    accuracy score
219         y_pred = classifier.predict(X_test_transform.values)
220         F1 = f1_score(y_test, y_pred, average=None)
221         F1_mean = np.mean(F1)
222         print('f1-score: ', F1, F1_mean)
223         AucScore = roc_auc_score(y_test, y_pred)
224         print('AUC: ', AucScore)
225
226         ResAcc[icount, jcount] = Accu
227         Resf1[icount, jcount] =F1_mean
228         ResAuc[icount, jcount] =AucScore
229         jcount +=1
230     icount+=1
231     jcount = 0
232
233
234
235 # plot accuracy redults to review
236 plt.figure()
237 plt.plot(ResAcc.T)
238 plt.grid()
239 plt.show()
240
241 np.savetxt("ResAcc_fad_matlab.csv", ResAcc, delimiter=",")
242 np.savetxt("Resf1_fad_matlab.csv", Resf1, delimiter=",")
243 np.savetxt("ResAuc_fad_matlab.csv", ResAuc, delimiter=",")
```

To add multipath fading to a dataset generate in Section C.1 use the following Matlab

code.

```
clear
clc

% read the dataset D (not noisy)
D = readmatrix("C:\Users\Amirhassan\Desktop\Hz\GenData\mmW_mb6_1.csv");
% keep only the signal part
Ddim = size(D);
Ds = D(:,1:Ddim(2)-16); % the rest are metadata of locations and movements

%plot(Ds(200,:)) % plot a sample to check if import is correct


%% Define the fading channel
fs = Ddim(2)-16;
doppJakes = doppler('Jakes');
ricChan = comm.RicianChannel( ...
    'SampleRate',fs,'MaximumDopplerShift',50,'KFactor',20,...
    'DopplerSpectrum',doppJakes, 'RandomStream','mt19937ar with seed', ...
    'Seed',73, 'FadingTechnique','Sum of sinusoids','InitialTimeSource', 'Input port');

sig = Ds(507,:)';
figure
out = ricChan(sig, 0);
sm = smoothdata(out, 'movmean', 100);
%subplot(1,2,2)
plot(sig)
hold on
plot(abs(out))
plot(abs(sm),'g')

%% loop over dataset Ds and add fading
Dsize = size(Ds);
Dsf = zeros(Dsize); % empty faded matrix
Dss = zeros(Dsize); % smoothed version
reset(ricChan);
for i = 1:Dsize(2)
    outSig = abs(ricChan(Ds(:,i), 0));
    Dsf(:,i) = outSig;
    Dss(:,i) = smoothdata(outSig, 'movmean', 40);
    reset(ricChan);
end
indx = 400;
figure
plot(Dsf(:,indx))
hold on
```

```
plot(Ds(:,indx))
plot(Dss(:,indx))


%% save fading datasets as csv
Dsf = Dsf';
Dss = Dss';
Dsf2 = cat(2, Dsf, D(:,Ddim(2)-15:end));
Dss2 = cat(2, Dss, D(:,Ddim(2)-15:end));

writematrix(Dsf2, 'L1_fad_1k.csv');
writematrix(Dss2, 'L1_fadSmooth_1k.csv');
```