A Deep Reinforcement Learning-based Approach to Beamforming in Cell-Free Networks

A Project Report

submitted by

SHAMEEM AHAMED IBRAHIM

in partial fulfilment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2022

THESIS CERTIFICATE

This is to certify that the thesis titled **A Deep Reinforcement Learning-based Approach to Beamforming in Cell-Free Networks** , submitted by **Shameem Ahamed Ibrahim**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Srikrishna Bhashyam
Project Guide
Associate Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 17th June 2022

ACKNOWLEDGEMENTS

First and foremost, I would like to sincerely thank my project guide Dr. Srikrishna Bhashyam. It is his guidance, swiftness in clearing my small & big doubts, consistency in showing me the way forward, leniency in understanding any delays in the work and dedication to regularly check my progress that has led me to do this project properly.

I would also like to the thank Dr. Balaraman Ravindran for his online lectures on Reinforcement Learning that equipped me to do this project. I would also like to thank all the professors in the Department of Electrical Engineering who have taught me courses crucial to this project, and my career.

I would also like to thank Sanjeev for clearing a difficult roadblock during the project. I would also like to thank Dhruv for sharing his experience working in a related project that helped me in doing my project.

Finally, none of this would have been possible without my invaluable parents giving me an education way above their means, doing their best to support me and stand by me through thick and thin, at home and at campus, and not letting any of the trials and tribulations they were facing affect me.

ABSTRACT

KEYWORDS: Beamforming, Neural Networks, Deep Reinforcement Learning, 6G, Cell-Free Networks

As the carrier frequency of wireless communication keeps increasing from a few GHz in 4G to mmWave in 5G to possibly more than a hundred GHz in 6G, the signal strength decays quickly and gets blocked by more obstacles. In such a case, traditional cellular networks may not be able to operate practically. Thus, we need a cell-free network where there are multiple access points (APs) in an area and all of them communicate with every user equipment (UE) in that area. This can only work when all the APs are connected to a common central processing unit (CPU) using high-speed optical channel, thus keeping them synchronized.

At the CPU, in order to decipher which signal came from which UE, we need beamforming parameters that maximize the signal from the desired UE and minimizes the interference from other UEs. In order to find out the optimal beamforming parameters, we use a Deep Reinforcement Learning technique called Deep Deterministic Policy Gradient (DDPG) implemented using deep neural networks. The performance of the beamforming parameters learned using DDPG is then compared with a benchmark obtained from MATLAB using the fmincon function. We also look at a few preliminaries that help us understand the cell-free network better. This project draws strong inspiration from the work of Hossain *et al.* [4].

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

ABBREVIATIONS

LTE	Long-Term Evolution
MIMO	Multiple-Input Multiple-Output
RV	Random Variable
MGF	Moment-Generating Function
SINR	Signal-to-Interference-plus-Noise Ratio
AP	Access Point
UE	User Equipment
DDPG	Deep Deterministic Policy Gradient
DRL	Deep Reinforcement Learning
Tx	Transmitter
Rx	Receiver
PDF	Probability Density Function
CDF	Cumulative Distribution Function
AWGN	Additive White Gaussian Noise
CSI	Channel State Information
IUI	Inter-User Interference
TD	Temporal-Difference

NOTATION

γ_k	SINR at the k^{th} UE
τ_p	Pilot sequence length (in samples)
ρ_k	Normalized Tx power per symbol of the k^{th} UE
X_{mk}	Parameter X corresponding to the k^{th} UE & the $m^{th} AP$
L_{mk}	Euclidean distance
κ	Path-loss exponent
$\mathcal{F}_{mk}$	Log-normal shadowing factor
h_{mk}	Small-scale fading parameter
$X \sim Y$	RV X is distributed according to the PDF Y

CHAPTER 1

Introduction

Operating frequencies for wireless communications have always been increasing. Carrier frequencies in 3G stayed below 3 GHz [1], while carrier frequencies in 4G (LTE) stayed below 10 GHz [2]. 5G has a frequency range around 30-50 GHz [3] which is in the *mmWave* region. Samsung has a vision for 6G that includes carrier frequencies upto 300 GHz [13]. As carrier frequencies keep increasing, the penetrative power of the signal keeps decreasing. *mmWave* is vulnerable to being stopped by even rain. This calls for *network densification*, i.e. smaller cells thus resulting in there being more APs in a unit area. With network densification comes high levels of Inter-Cell Interference and Inter-User Interference. To solve this, 5G uses *massive MIMO* where multiple antennae with specific beamforming parameters are used to at APs to directly shine signals at an UE giving the network *spatial orthogonality* [10]. Even after using massive MIMO, Inter-Cell Interference (even though reduced) is still the limiting factor in cellular networks [9]. The issue lies at the heart of the concept of cellular communications itself, i.e. in the concept of the cell.

Therefore we need to think beyond cellular communication. Coordinated Multipoint technique with Joint Transmission (CoMP-JT) is a technique where more than one AP transmits the same signal to an UE [11]. The signals have to be synchronized in order for the UE to realize any betterment in performance. This is the problem with CoMP-JT as it needs multiple APs at different distances from an UE to send signals that have to arrive at the UE synchronously.

In this project, we discuss a technique presented by [4]. The network is a cell-free network, i.e. there is no concept of cell. All the APs in an area of consideration send signals to all UE in that area. The APs are all connected to a CPU through high-speed optical links called *front haul*. The CPU processes the signals obtained from the APs and helps the APs send signals in a synchronised manner. All the UEs send signals to all the APs as well. This causes a lot of Inter-User Interference (IUI). In order to solve the issue of Inter-User Interference, we use *Beamforming Parameters*. To discriminate the signal obtained from an particular UE from the signals of other UEs, the CPU uses a set of scalars called beamforming parameters. The optimal beamforming parameters are hard to obtain to brute-forcing as the dimensionality is prohibitively high (this is due to there number of beamforming parameters beings equal to the product of the number of UEs and APs). Due to the multitude of factors influencing the channel gain between an AP and a UE like log-normal shadowing and small-scale fading, it is also impossible to find the optimal beamforming parameters using any hard-coded closed-form expression.

Thus we consider a Deep Reinforcement Learning (DRL) technique called Deep Deterministic Policy Gradient (DDPG) to optimize the beamforming parameters. The DDPG agent runs on the CPU and interacts with signals obtained from the APs in real-time to find the optimal beamforming parameters in an iterative manner. We then benchmark the performance of this DDPG network with the *bisection method* using the MATLAB function `fmincon`.

We also experimentally verify the accuracy of the Welch-Satterthwaite Approximation for the sum of a small to medium number of Gamma RVs. We then study the effects of non-orthogonal pilots in uplink (UL) transmission both theoretically and experimentally. We also study the effects of correlated log-normal shadowing and compare it with the simpler non-correlated log-normal shadowing experimentally. We then experimentally analyse the probability of outage experimentally. The last three experiments are performed in a cell-free environment. For all the experiments performed, the algorithm used is given in a pseudo code format in order to verify for reproducibility and to provide a starting platform to improve on in the future. We use the work of Hossain *et al.* [4] as the basic foundation and to guide the direction of this project.

1.1 Organization of Thesis

The organization of this thesis is as follows,

- Chapter 2 discusses the arrangement of the environment and factors considered in the performance of the cell-free network technique.
- Chapter 3 discusses the accuracy of Welch-Satterthwaite Approximation, the effects of non-orthogonal pilots, the effects of correlated shadowing and the probability of outage in a cell-free network.
- Chapter 4 explains the DDPG network setup with useful background information and how the MATLAB benchmarking setup works.

- Chapter 5 provides the assumptions made, options chosen, numerical values of the parameters used in the experiments along with the results obtained and the inferences drawn from them.
- Chapter 6 provides the conclusion for the thesis and discussed the scope for future work in this subject area.

CHAPTER 2

The Environment

2.1 Arrangement of UEs and APs

In cellular networks, the area under consideration is divided into cells, thus the name. The perfect cell is a circle due to the path-loss (ignoring shadowing and fading) being constant throughout the boundary of the cell. Unfortunately, the circle does not tessellate with itself. Only three polygons (triangle, square, hexagon) tessellate with themselves [8]. Out of these three, hexagon has the highest coverage area for a given cell radius (distance of the farthest point in the cell from the centre). Thus ideally the area under consideration in cellular networks is divided into hexagonal cells, each with one 1 AP in the centre. All UEs in a cell are connected to the central AP of that cell until a hand-off occurs. A hexagonal cell may further be divided into several sectors in order to decrease inter-cell interference. Several cells combine to form a cluster in which every frequency isn't repeated.

In reality, practical considerations force the formation of non-hexagonal cells. This combined with the fact that if there are more APs than UEs, we're not utilizing the full potential of the network when every UE is connected to only one AP. In such situations, cell-free networks become helpful. In a cell-free network, every UE in the area under consideration communicates with every AP in that area.

In this project, uplink data transmission in a cell-free network is analysed. The area of interest is a circular disk in the 2D Euclidean plane. The number of APs is higher than the number of UEs. All the APs are connected to each other and the CPU through high-speed low latency optical fibres. It is assumed that the APs and UEs are stationary. The positions of APs and UEs are chosen uniformly at random.

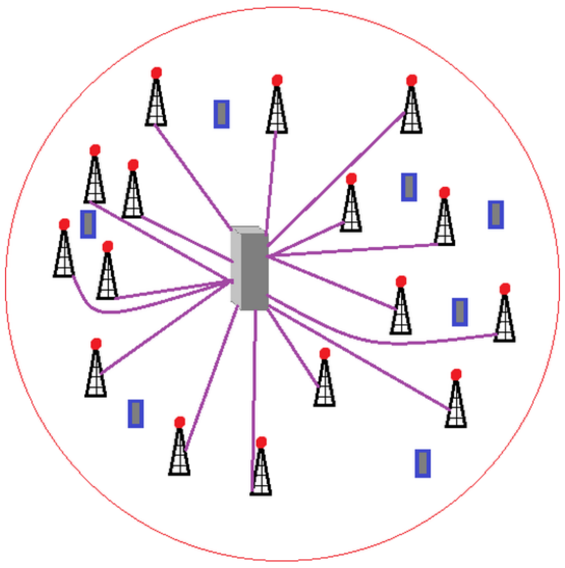


Figure 2.1: A possible network configuration for $M = 15$ and $K = 7$. Notice the random placement of APs and UEs

Fig. 2.1 shows such a possible configuration. This leads to configurations where one UE in very close to an AP or where several UEs don't have an AP nearby. Thus any protocol must be evaluated for a variety of configurations before deciding on its efficacy.

2.2 The Channel

All APs are assumed to be isotropic radiators of the signal. All UEs are assumed to be isotropic receivers as well. The channel is assumed to be flat and single-tap. The number of APs is denoted by M and the number of UEs is denoted by K. The gain between the k^{th} UE and the m^{th} AP is assumed to be the following,

$$g_{mk} = L_{mk}^{\kappa/2} \mathcal{F}_{mk}^{1/2} h_{mk} \tag{2.1}$$

The path-loss exponent κ is always greater than or equal to 2. h is distributed as a Nakagami-$\mathcal{M}$ RV. This is the small-scale fading and its value changes at every time step. A Nakagami-$\mathcal{M}$ RV is distributed according to the following PDF,

$$f_X(x) = \frac{2\mathcal{M}^{\mathcal{M}}}{\Gamma(\mathcal{M})\Omega^{\mathcal{M}}} x^{2\mathcal{M}-1} e^{-\frac{\mathcal{M}}{\Omega}x^2} \tag{2.2}$$

where $\mathcal{M}$ is the *shape parameter* ($\mathcal{M} \geq 1/2$) and Ω is the *spread parameter* ($\Omega > 0$). $|h|^2$ is required for calculating the SINR and it is distributed as a Gamma RV with the following PDF,

$$f_X(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} \tag{2.3}$$

where α is the *shape parameter* ($\alpha = \mathcal{M}$) and β is the *rate parameter* ($\beta = \mathcal{M}/\Omega$).
$\mathcal{F}$ is the log-normal shadowing factor. It is defined as,

$$\mathcal{F}_{mk} = 10^{\frac{\sigma_{mk} z_{mk}}{10}} \tag{2.4}$$

σ_{mk} is the standard deviation of the log-normal shadowing in decibels. If simplicity is the aim, z_{mk} is taken to be distributed following the standard normal distribution, i.e. $z_{mk} \sim \mathcal{N}(0, 1)$. In reality, the shadow fading between various UEs and APs are not independent of one another. So for a general case,

$$z_{mk} = \sqrt{\delta} a_m + \sqrt{1-\delta} b_k \tag{2.5}$$

where a_m ($m = 1, ..., M$) and b_k ($k = 1, ..., K$) are distributed according to the standard normal distribution. δ is the *transmitter-receiver shadow fading correlation coefficient*. This correlation happens because nearby links tend to be shadowed by the same objects or structures.
In this project, the problem of uplink beamforming is the main goal. The problem is tackled with the help of a DRL technique called DDPG. In this technique, a network is trained over several episodes in the environment and each episode contains a fixed number of time steps. $L_{mk}^{-\kappa}$ and $\mathcal{F}_{mk}$ don't change throughout the training process and are assumed to be known to the CPU for simplicity. h_{mk} on the other hand is changing every time step, unknown and has to be estimated using pilot sequences. The effects of shadow fading correlation coefficient (δ) and the effects of non-orthogonal pilots are studied in further chapters.

2.3 The SINR

Assume that the CSI matrix (g_{mk}) is known to the CPU after training with pilot sequences and that during the pilot training each UE had a mutually orthogonal pilot sequence allotted to it, thus leading to no pilot estimation errors. Let x_k be the signal sent by the k^{th} UE with power p_k uniformly in all directions. x_k has an average power of 1, since the term p_k is the actual transmit power. The CPU tries to find out what was the signal sent by each UE using a set of *beamforming parameters* ω_{mk} such that $0 \leq \omega_{mk} \leq 1, \; \forall \, m, k$. The CPU's estimate of the signal received from the k^{th} UE is as follows,

$$y_k = \sum_{m=1}^{M} \omega_{mk} \left[\sum_{l=1}^{K} \sqrt{p_l} g_{ml} x_l + \eta_m \right] \tag{2.6}$$

where η_m is the AWGN added at the receiver of the m^{th} AP. Expanding the above equation,

$$y_k = \underbrace{\sum_{m=1}^{M} \omega_{mk} \sqrt{p_k} g_{mk} x_k}_{\text{Desired Signal}} + \overbrace{\sum_{m=1}^{M} \omega_{mk} \sum_{l=1, l\neq k}^{K} \sqrt{p_l} g_{ml} x_l}^{\text{Inter-User Interference}} + \underbrace{\sum_{m=1}^{M} \omega_{mk} \eta_m}_{\text{Noise}} \tag{2.7}$$

The neural network that will learn the beamforming parameters should ideally maximize the power of the first term and minimize the power of the second term. From equation 2.7, the SINR of the signal from the k^{th} UE at the CPU γ_k after applying the beamforming parameters is as follows,

$$\gamma_k = \frac{\sum_{m=1}^{M} \omega_{mk}^2 g_{mk}^2 p_k}{\sum_{m=1}^{M} \omega_{mk}^2 \sum_{l=1, l\neq k}^{K} g_{ml}^2 p_l + \sum_{m=1}^{M} \sigma_m^2 \omega_{mk}^2} \tag{2.8}$$

where σ_m^2 is the noise power at the m^{th} AP. The network is constrained by the inter-user interference part and not the noise part in the denominator, otherwise there wouldn't be a problem to solve here. Equation 2.8 is of utmost importance to the neural network as will be seen in coming chapters. The Per-User Transmission Rate (R) can be calculated by using the total Transmission Rate divided by number of UEs. The following equations illustrate the way to calculate R,

$$R = \frac{1}{K} \sum_{k=1}^{K} \gamma_k \tag{2.9}$$

CHAPTER 3

The Preliminaries

3.1 Welch-Satterthwaite Approximation

3.1.1 Motivation

The calculation of SINR of each UE is used in finding out the Per-User Transmission Rate and outage probability, and in training the neural network. This calculation requires us to sum a number of non-identically distributed Gamma RVs. The number of independent non-identically distributed Gamma RVs depends on whether the network is large-scale, medium-scale or small-scale. In order to simplify this calculation and most importantly to obtain an upper limit for the outage probability, a way to accurately approximate the sum of Gamma RVs is needed.

Since it is not guaranteed that the number of RVs to be summed will be large, *Central Limit Theorem* is ruled out. It is also the case that the Gamma RVs to be summed may not necessarily have the same rate parameters. This rules out the possibility of using the MGF method to derive the aforementioned sum. This leads us to using the *Welch-Satterthwaite Approximation* for approximately finding out a close-enough upper bound for the sum of non-identical Gamma RVs [14].

3.1.2 The Approximation

Let $X_1, ..., X_n$ be i.n.d (independent non-identically distributed) Gamma RVs for some $n > 1,\ n \in \mathbb{N}$. Let $\alpha_1, ..., \alpha_n$ be their respective shape parameters and $\beta_1, ...\beta_n$ be their respective rate parameters. This can be written as $X_i \sim \mathcal{G}(\alpha_i, \beta_i),\ i = 1, ..., n$. Let Y be the sum of all the aforementioned i.n.d. Gamma RVs. i.e. $Y = X_1 + ... + X_n$. Y is the RV whose PDF we have to approximate. The *Welch-Satterthwaite Approximation* states that the PDF of Y can be approximated by an upper bound which is also a Gamma

RV. Let the PDF of Y be denoted by $f_Y(y)$. Then,

$$f_Y(y) \approx \frac{\beta^\alpha}{\Gamma(\alpha)} y^{\alpha-1} e^{-\beta y} \tag{3.1}$$

where α and β are defined as,

$$\alpha = \frac{(\frac{\alpha_1}{\beta_1} + ... + \frac{\alpha_n}{\beta_n})^2}{\frac{\alpha_1}{\beta_1^2} + ... + \frac{\alpha_n}{\beta_n^2}} \tag{3.2}$$

$$\beta = \frac{\frac{\alpha_1}{\beta_1} + ... + \frac{\alpha_n}{\beta_n}}{\frac{\alpha_1}{\beta_1^2} + ... + \frac{\alpha_n}{\beta_n^2}} \tag{3.3}$$

An important fact to be observed is that if all the β_is $(i = 1, , , n)$ are equal, we get $\alpha = \alpha_1 + ... + \alpha_n$ and $\beta = \beta_1 = ... = \beta_n$. This is what we get from the MGF method as well as thus the *Welch-Satterthwaite Approximation* gives the exact PDF of Y if all the X_is had equal rate parameters.

3.1.3 Experimental Verification

The following method, given here in pseudo code format, was used to experimentally verify the accuracy of the Welch-Satterthwaite Approximation for a few different number of i.n.d. Gamma RVs summed.

Algorithm 1 Verifying Welch-Satterthwaite Approximation

```
for $n \in$ List of number of RVs do
    Initialize array $y$
    for $i = 1, ..., n$ do
        Generate unique $\alpha_i$ & $\beta_i$
        Generate a large number of random numbers from $\mathcal{G}(\alpha_i, \beta_i)$
        Add the random numbers to $y$
    end for
    Calculate $\alpha$ & $\beta$ according to 3.2 & 3.3
    Generate PDF of $\mathcal{G}(\alpha, \beta)$ using scipy.stats
    Plot the PDF of $\mathcal{G}(\alpha, \beta)$ and histogram of $y$ together
end for
```

3.2 Effect of Non-orthogonal Pilots

3.2.1 Motivation

Pilot sequences are used in *coherent detection* communication to estimate the channel gain. They are signals known to both the transmitter and receiver before the communication. Every channel has a coherence time (τ_c), which is decided by the Doppler frequency (f_D), over which the channel can be assumed to be constant, which means that the small-scale fading is highly correlated with itself for this duration. This time is utilized to both send the pilot sequences (to estimate the channel gain) and data (which is decoded using the estimated channel gain).
The number of symbols in the pilot sequence is controlled by both the coherence time and the bandwidth available. The number of symbols in the pilot sequence equals the maximum number of unique mutually orthogonal pilot sequences. This is due to the fact that there can't be $n+1$ mutually orthogonal vectors (or even linearly independent vectors) in an n-dimensional vector space. As the number of UEs in a network increases, non-orthogonal pilot sequences have to be used and this causes additional errors in the estimation of channel gain (called *Pilot Contamination*). This causes additional errors in decoding the data resulting from a decrease in SINR. In this section, we will study the effects of non-orthogonal pilots in cell-free networks.

3.2.2 Theoretical Background

Each UE is given a unique pilot sequence $\phi_k,\ k = 1, ..., K$ of length τ_p symbols by the CPU, such that $||\phi_k||^2 = 1$. The duration of the pilot sequence should be less than the coherence time (τ_c) of the channel. The UEs then send this pilot sequence of theirs to all the APs uniformly (i.e. without directing it towards any AP or any group of APs). The corresponding received signal at the m^{th} AP is,

$$y_m = \sum_{k=1}^{K} \sqrt{\tau_p \rho_k} g_{mk} \phi_k + \eta_m \tag{3.4}$$

where ρ_k is the *normalized SNR* of each pilot symbol. η_ms are the AWGN i.i.d. RVs with zero mean and power 1 (the noise power is 1 because the pilot symbols are normalized, so the calculation of ρ_k includes the actual noise power). In order to estimate the signal obtained from the k^{th} UE, we multiply 3.4 with ϕ_k^H,

$$\dot{y}_{mk} = \phi_k^H y_m = \sqrt{\tau_p \rho_k} g_{mk} + \sum_{k'=1, k' \neq k}^{K} \sqrt{\tau_p \rho_{k'}} g_{mk'} \phi_k^H \phi_{k'} + \phi_k^H \eta_m \tag{3.5}$$

The first term is our desired signal. The second term is inter-user interference (IUI) and the third term is noise. The second term vanishes if all the UEs are allotted mutually orthogonal pilots (i.e. $\phi_k^H \phi_{k'} = 0, \forall\, k \neq k'$). This may not be possible in reality due to the constraint posed by coherence time (τ_c).

The total pilot signal obtained by adding the received pilots from all the APs and then multiplying them with ϕ_k^H is,

$$\dot{y}_k = \phi_k^H \left(\sum_{m=1}^{M} y_m \right) = \sum_{m=1}^{M} \dot{y}_{mk} \tag{3.6}$$

Using equation 3.5 to expand,

$$\dot{y}_k = \underbrace{\sum_{m=1}^{M} \sqrt{\tau_p \rho_k} g_{mk}}_{\text{Desired Signal}} + \overbrace{\sum_{m=1}^{M} \sum_{k'=1, k' \neq k}^{K} \sqrt{\tau_p \rho_{k'}} g_{mk'} \phi_k^H \phi_{k'}}^{\text{Pilot Contamination}} + \underbrace{\sum_{m=1}^{M} \phi_k^H \eta_m}_{\text{Noise}} \tag{3.7}$$

The SINR of the signal in 3.7 $\dot{\gamma}_k$ is,

$$\dot{\gamma}_k = \frac{\sum_{m=1}^{M} \tau_p \rho_k g_{mk}^2}{\sum_{m=1}^{M} \sum_{k'=1, k' \neq k}^{K} \tau_p \rho_{k'} g_{mk'}^2 |\phi_k^H \phi_{k'}|^2 + M} \tag{3.8}$$

The average Per-User Transmission Rate is,

$$\dot{R} = \frac{1}{K} \sum_{k=1}^{K} log_2(1 + \dot{\gamma}_k) \tag{3.9}$$

3.2.3 Experimental Analysis

The following method, given here in pseudo code format, was used to experimentally analyse the effect of non-orthogonal pilots on the Per-User Transmission Rate of transmitting pilots in a cell-free network.

Algorithm 2 Analysing the effects of Non-orthogonal Pilots

```
for $\tau_p \in$ List of $\tau_p$s do
    Initialize $\dot{R}\_array$
    for $m \in$ List of number of APs do
        Distribute the orthogonal pilots to all the UEs, repeating when necessary
        Calculate $\dot{\gamma}_k$s using 3.8
        Calculate $\dot{R}$ using 3.9
        Store $\dot{R}$ in the $\dot{R}\_array$
    end for
    Plot $\dot{R}$ vs $m$ graph for this value of $\tau_p$
end for
```

3.3 Effect of Correlated Shadowing

3.3.1 Motivation

Log-normal shadowing is a large-scale fading phenomenon that occurs due to the geography of the channel in which the signal is propagating. It is assumed to not change with time if the APs and UEs are stationery (as is the case in this project). In reality, the different links between the UEs and APs have shadow fading parameters that are correlated [6]. In this section, we will discuss the differences between correlated and uncorrelated log-normal shadowing for different standard deviations of the log-normal distribution. We will also device experiments that simulate the environment and bring out the differences discussed. The results will be presented in later chapters.

3.3.2 Explanation

The expression for log-normal shadow fading is given by equation 2.4, where z_{mk} is given by equation 2.5. This model assumes that the shadow fading experienced by a link is explained exhaustively by a contribution from the UE side and a contribution from the AP side. The *transmitter-receiver shadow fading correlation coefficient* (δ) determines how much weight is given to each of those contributions. Whatever is the effect of z_{mk}, is amplified or attenuated by σ_{mk}.

In this experiment, we will vary the number of APs, keeping the number of UEs fixed, and see how the uncorrelated and the correlated case compare for a few different values of σ_{mk} when it comes to Per-User Transmission Rate in a cell-free network. One important difference to notice between the correlated and uncorrelated cases is that, in the uncorrelated case there are MN independent RVs (one for each combination of M APs and K UEs) and in the correlated case there are only $M+N$ independent RVs (M a_ms and K b_ks).

In this experiment, there is no technique (like beamforming or multiplying with ϕ_k^H) to enhance the power of the desired signal. So IUI deals a heavy blow to the SINR. The expression for SINR is simply,

$$\gamma_{sh,k} = \frac{\sum_{m=1}^{M} g_{mk}^2}{\sum_{m=1}^{M} \sum_{k'=1,k'\neq k} g_{mk'}^2 + \sum_{m=1}^{M} \sigma_m^2} \tag{3.10}$$

The Per-User Transmission Rate R_{sh} is therefore,

$$R_{sh} = \frac{1}{K} \sum_{k=1}^{K} log_2(1+\gamma_{sh,k}) \tag{3.11}$$

3.3.3 Experimental Analysis

The following method given there in pseudo code format, was used to experimentally analyse the effect of varying σ_{mk} in both correlated and uncorrelated shadow fading environments on Per-User Transmission Rate in a cell-free network.

Algorithm 3 Analysing the effects of Correlated Shadowing

```
for $\sigma \in$ List of $\sigma$s do
    for Correlated & Uncorrelated case do
        Initialize array of $R_{sh}$s
        for $m \in$ List of number of APs do
            Calculate the gain matrix $g_{mk}$
            Calculate $\gamma_{sh,k}$s using 3.10
            Calculate $R_{sh}$ using 3.11
            Store $R_{sh}$ in the array of $R_{sh}$s
        end for
        Plot $R_{sh}$ vs $m$ graph for this case
    end for
end for
```

3.4 Probability of Outage

3.4.1 Motivation

Receiver sensitivity dictates how low the power of the received signal in a link can get before the packet is lost or the call is dropped. Therefore maintaining the received signal power above this threshold is of utmost importance in designing communication networks. Probability of Outage is the fraction of time for which the received signal power of the link is below the threshold set by receiver sensitivity. It is an important performance measure of the quality of a link.
In this section, we will look the outage performance in a cell-free network without the use of beamforming parameters using *Monte-Carlo simulations.*

3.4.2 Theoretical Background

Since the path-loss and log-normal shadow fading are fixed, probability of outage depends on how often the small-scale fading dips below acceptable levels. The small-scale fading (h_{mk}) in our model is distributed as a Nakagami-$\mathcal{M}$ RV. The determining factor for probability of outage is SINR falling below a threshold. Let this threshold be γ_{th}. There is no beamforming involved in this experiment, so in equation 2.8, substitute $\omega_{mk} = 1,\ \forall\ m, k$. Therefore the simplified equation for SINR at the k^{th} UE is,

$$\gamma_k = \frac{\sum_{m=1}^{M} g_{mk}^2 p_k}{\sum_{m=1}^{M} \sum_{l=1,l\neq k}^{K} g_{ml}^2 p_l + M\sigma_m^2} \tag{3.12}$$

Let X_k and Y_k be the numerator and denominator of 3.12. Let's also ignore the effect of AWGN as the IUI is the deciding factor of SINR. Note that X_k and Y_k can be written as sum of Gamma RVs as follows,

$$X_k = \sum_{m=1}^{M} X_{mk},\ X_{mk} \sim \mathcal{G}\left(\alpha_{mk}, \frac{\beta_{mk} L_{mk}^{2\kappa}}{p_k \mathcal{F}_{mk}}\right) \tag{3.13}$$

$$Y_k = \sum_{m=1}^{M} \sum_{l=1,l\neq k}^{K} Y_{ml},\ Y_{ml} \sim \mathcal{G}\left(\alpha_{ml}, \frac{\beta_{ml} L_{ml}^{2\kappa}}{p_l \mathcal{F}_{ml}}\right) \tag{3.14}$$

Welch-Satterthwaite Approximation tells us that sums of Gamma RVs can be approximated with a single effective Gamma RV. Therefore it can approximately be written that,

$$X_k \sim \mathcal{G}(\alpha_{xk}, \beta_{xk}),\ Y_k \sim \mathcal{G}(\alpha_{yk}, \beta_{yk}) \tag{3.15}$$

where $\alpha_{xk},\ \beta_{xk},\ \alpha_{yk},\ \beta_{yk}$ can be calculated using equations 3.2 and 3.3. The probability of outage can then be written as,

$$\mathbb{P}(\gamma_k \leq \gamma_{th}) = \mathbb{P}(X_k \leq \gamma_{th} Y_k) \tag{3.16}$$

Let F_{Xk} & f_{Yk} be the CDF of X_k and PDF of Y_k respectively. Then the above equation can be written as,

$$\mathbb{P}(\gamma_k \leq \gamma_{th}) = \int_0^\infty F_{Xk}(\gamma_{th} y) f_{Yk}(y) dy \tag{3.17}$$

The CDF of a Gamma RV $X \sim \mathcal{G}(\alpha, \beta)$ is given by,

$$F_X(x) = \frac{1}{\Gamma(\alpha)} \gamma(\alpha, \beta x) \tag{3.18}$$

where $\gamma(.,.)$ is the *upper incomplete gamma function.* Combining the above equation with the expression for the formula of the PDF of a Gamma RV, the expression for probability of outage can be written as,

$$\mathbb{P}(\gamma_k \leq \gamma_{th}) = \frac{\beta_{yk}^{\alpha_{yk}}}{\Gamma(\alpha_{xk})\Gamma(\alpha_{yk})} \int_0^\infty y^{\alpha_{yk}-1} e^{-\beta_{yk} y} \gamma(\alpha_{xk}, \beta_{xk}\gamma_{th} y) dy \tag{3.19}$$

Equation [7] tells us that the above integral can be calculated and using that the above equation can be written as,

$$\mathbb{P}(\gamma_k \leq \gamma_{th}) = \frac{\beta_{yk}^{\alpha_{yk}}}{\Gamma(\alpha_{xk})\Gamma(\alpha_{yk})} \frac{(\beta_{xk}\gamma_{th})^{\alpha_{xk}} \Gamma(\alpha_{xk} + \alpha_{yk})}{\alpha_{xk}(\beta_{xk}\gamma_{th} + \beta_{yk})^{\alpha_{xk}+\alpha_{yk}}} {}_2F_1(1, \alpha_{xk}+\alpha_{yk}; \alpha_{xk}+1; \frac{\beta_{xk}\gamma_{th}}{\beta_{xk}\gamma_{th} + \beta_{yk}}) \tag{3.20}$$

where ${}_2F_1(.,.;.;.)$ is the *Gaussian hypergeometric function.* This is the theoretical formula for probability of outage in a cell-free network. In the following subsection, the probability of outage is generated experimentally.

3.4.3 Experimental Analysis

The following method given here in pseudo code format, was used to experimentally analyse the probability of outage for various numbers of APs and UEs using *Monte-Carlo simulations* in a cell-free network.

Algorithm 4 Analysing the Probability of Outage

```
for $k \in$ List of number of UEs do
    Initialize $probability\_outage\_array$
    for $m \in$ List of number of APs do
        Generate large-scale fading values
        Initialize outage counter
        for $i = 1, ..., No.\ of\ simulations$ do
            Generate small-scale fading values
            Calculate SINR using 3.12
            if SINR $\leq \gamma_{th}$ then
                Increment outage counter by 1
            end if
        end for
        Append $\frac{outage\ counter}{No.\ of\ simulations}$ to $probability\_outage\_array$
    end for
    Plot Probability of Outage vs $m$ graph for this value of $k$
end for
```

CHAPTER 4

Deep Learning Network

4.1 Reinforcement Learning

In machine learning, we have the training data beforehand and feed it to our neural network or perceptron to learn the pattern. In real life, it may not be possible or advisable to generate data for training beforehand and then train on it. In such situations, we use reinforcement learning, where our agent interacts with an environment and learns the correct actions using states and rewards. Reinforcement learning agents are useful in solving problems like making a robotic arm pick an object, making a 4-legged robot walk and playing video games. One famous reinforcement learning agent is Google DeepMind's *AlphaGo*, which defeated the world's best human player in the board game Go in 2016 [5].

Reinforcement learning uses the ground work laid by *Multi-arm Bandit Problems* and extends it by introducing multiple states [17]. In Multi-arm Bandit, the environment gives a *reward* as feedback for performing an action. In reinforcement learning, the rewards are combined to form a *return*, which is the expected discounted sum of all future rewards for performing a particular action in a state. Deep Neural Networks can be used in reinforcement learning to learn a function that ideally outputs the best action given a state, after training. This is called Deep Reinforcement Learning (DRL).

A Deep Neural Network has multiple layers of *neurons*, each of which takes an input vector, multiplies it with a weight matrix, adds a bias vector, passes it through an activation function and gives the resulting vector as output. In an equation form, the function of a layer of neurons can be written as,

$$v_{output} = f_{activation}(M_{weights} v_{input} + v_{bias}) \tag{4.1}$$

The elements of both the bias vector and weight matrix are trainable.

4.2 Deep Deterministic Policy Gradient

DDPG is a DRL algorithm proposed by Lillicrap *et al.* [12]. The following are the features of the DDPG algorithm,

- It is a *policy gradient* algorithm, i.e. it uses gradient descent to learn a policy of that outputs actions when given states as opposed to outputting a value function based for all actions from which the best action is then chosen.
- It is *deterministic*, i.e. the agent outputs a single action when given a state as opposed to giving an action drawn from a probability distribution depending on the state.
- It is an *off-policy* algorithm, i.e. the policy learned is different from the policy that is used to generate the data for learning.
- It stores it's interactions with environment as *Markov Decision Process* experiences of the format `[State, Action, Reward, Next_State]` in a *Replay Buffer*, from which batches of past experiences are sampled to learn from.
- It is an *online-learning* algorithm, i.e. the policy is being updated as the agent is interacting with the environment and gathering experiences as opposed to the the policy network being trained after the agent has gathered all the experiences it needs from interacting with the environment.
- It is an *Actor-Critic* algorithm, i.e. it has an actor network and a critic network both working together. The actor network outputs an action for a given state and the critic network gives a value for the action committed for the given state by the actor. The actor network is trained to maximize the critic value.
- Since the agent outputs a deterministic action, a noise is added to the said action for the purpose of *exploration*. Usually, *Ornstein-Uhlenbeck Noise* is added (due to its self-correlating property) but a *Normal Noise* of appropriate distribution is also found to work [insert paper reference].
- In order for there to be stability in the learning process, two *target networks* (one for the actor, one for the critic) are used. The target networks are used for gradient descent in the critic network. The target networks are then updated regularly. At the end of the training process, the weights of the target networks converge with their corresponding actual networks due to the *Polyak averaging method* [reference paper].

4.3 Combining Environment and Network

In order for the DDPG algorithm to work properly, we need to connect in properly to the cell-free environment. The following table shows how the environment is connected to the DDPG agent.

Table 4.1: Relation between the Environment and the DDPG agent

State	SINR values (γ_k)
Reward	Total Transmission Rate (KR)
Action	Beamforming parameters (ω_{mk})

Table 4.2: Network Notations

Network Component	**Notation**
Episode number	e
Timestep number	t
State	s_t
Action	a_t
Reward	r_t
Discount factor	ζ
Polyak averaging factor	τ
Network weights	θ
Batch Size	$\mathcal{L}$
Actor	$\mu(s_t\|\theta^{\mu})$
Target actor	$\mu'(s_t\|\theta^{\mu'})$
Critic	$Q(s_t, a_t\|\theta^{Q})$
Target critic	$Q'(s_t, a_t\|\theta^{Q'})$

The algorithm is run for E episodes with each episode consisting of T timesteps. At the beginning of each episode, the initial state is generated uniformly at random from a range of possible SINR values. Per-User Transmission Rate (R) is Total Transmission Rate divided by number of UEs. For g_{mk}, the path-loss and log-normal shadowing are fixed at the beginning and don't change throughout the training process. Small-scale fading is generated anew every timestep. The reward for any action performed is directly dependent only on the next state.

Table 4.2 shows the notation used for this DDPG network. For updating the critic network, the *Temporal-Difference*(TD) [16] target is first generated using the target networks as follows,

$$y_t = r_t + \zeta Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'}) \tag{4.2}$$

The loss used for updating the critic network is as follows,

$$L = \frac{1}{\mathcal{L}}\sum_{i=1}^{\mathcal{L}}(y_i - Q(s_i, a_i|\theta^{Q}))^2 \tag{4.3}$$

The loss used for updating the actor network is the batch average of the negative of the critic values in a batch. The target networks are updated using the *Polyak averaging factor* according to the following method,

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{4.4}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \tag{4.5}$$

4.3.1 Algorithm

The following method given here in pseudo code format, was used to optimize the beamforming parameters for a cell-free network with Nakagami-$\mathcal{M}$ fading, using a DDPG agent. This algorithm is only a modified version of *Algorithm 1* in [main paper].

Algorithm 5 Optimizing the Beamforming Parameters

```
Initialize the actor and critic networks with random weights
Initialize both the target networks with the weights from their corresponding actual networks
Initialize the Replay Buffer
for $e = 1, ..., E$ do
    Generate initial state ($s_0$)
    for $t = 1, ..., T$ do
        Get the beamforming parameters ($a_t$) from the actor network using the current state ($s_t$)
        Implement $a_t$ and observe the new state ($s_{t+1}$) and the reward ($r_t$)
        Store the experience ($s_t, a_t, r_t, s_{t+1}$) in the Replay Buffer
        Sample a batch of $\mathcal{L}$ experiences from the Replay Buffer
        Calculate TD target using equation 4.2
        Obtain the critic loss using equation 4.3
        Update the critic $Q(s_t, a_t|\theta^Q)$ by minimizing the critic loss
        Update the actor by minimizing the actor loss (i.e. the negative of the batch average of the critic values)
        Update target networks using the assignment operations given in 4.4 and 4.5
    end for
end for
```

Table 4.3: Relation between the Environment and the DDPG agent

Parameter	**Notation**	**Assignment**
Input Vector	x	Action(a)
Objective Function	$f(x)$	-Per-User Transmission Rate($-R$)
Lower Bound	$lb \leq x$	All zeros vector
Upper Bound	$x \leq ub$	All ones vector
Initial Input	x_0	Random permissible vector

4.4 Benchmarking using MATLAB

The performance of the DDPG network has to be evaluated against a reliable benchmark. For this purpose, we use the function '`fmincon`' from MATLAB, which used the *bisection method* [18] to solve multi-dimensional optimization problems under linear and non-linear constraints and is guaranteed to find the optimal solution obeying the given constraints. '`fmincon`' minimizes an *objective function* that outputs a scalar and can take in a vector as input.
The values of the product of path-loss and log-normal shadowing are passed to MATLAB from Python in a `.txt` file. This ensures the same configuration is being optimized for in both DDPG and MATLAB. The value of small-scale fading is generated in MATLAB itself. Table 4.3 shows the parameters fed into `fmincon` function. Since $f(x)$ has to be a deterministic function, we use the average value of the small-scale fading (i.e. expected value of a Gamma RV) in the optimization phase. While testing to find out the performance of the optimized beamforming parameter obtained from `fmincon`, the average reward over a large number of *Monte-Carlo simulations* is calculated.

4.4.1 Algorithm

The following method given here in pseudo code format, was used to find the benchmark for the Per-User Transmission Rate of a cell-free network using the `fmincon` function.

Algorithm 6 Benchmarking using MATLAB

```
Obtain the values of $M, K, \sigma_m$, path-loss and shadow fading
Generate initial input vector $x_0$
Set the options to plot 'optimplotstepsize'
Feed the parameters and deterministic objective function into 'fmincon'
Run the 'fmincon' optimizer
Retrieve the optimized beamforming parameter from 'fmincon'
Initialize sum_rewards to zero
for i=1,...,No. of Simulations do
    Calculate the actual reward for the optimized beamforming parameter
    Add the reward obtained to sum_rewards
end for
Export $\frac{sum\_reward}{No.\ of\ Simulations}$ to Python
```

CHAPTER 5

Simulation

5.1 Preliminaries

5.1.1 Welch-Satterthwaite Approximation

Table 5.1 shows the numerical values used for experimentally verifying the accuracy of Welch-Satterthwaite Approximation.

Table 5.1: Values for Welch-Satterthwaite Approximation

Parameter	**Value**
Numbers of RVs (Ks)	`[2,4,6,8]`
Number of Samples per RV	`30000`
Shape Parameter of i^{th} RV (α_i)	$i+1$
Rate Parameter of i^{th} RV (β_i)	$\frac{1}{i+2}$

Figure 5.1 shows the experimental results comparing the Welch-Satterthwaite Approximation with PDFs created from a large number of samples.

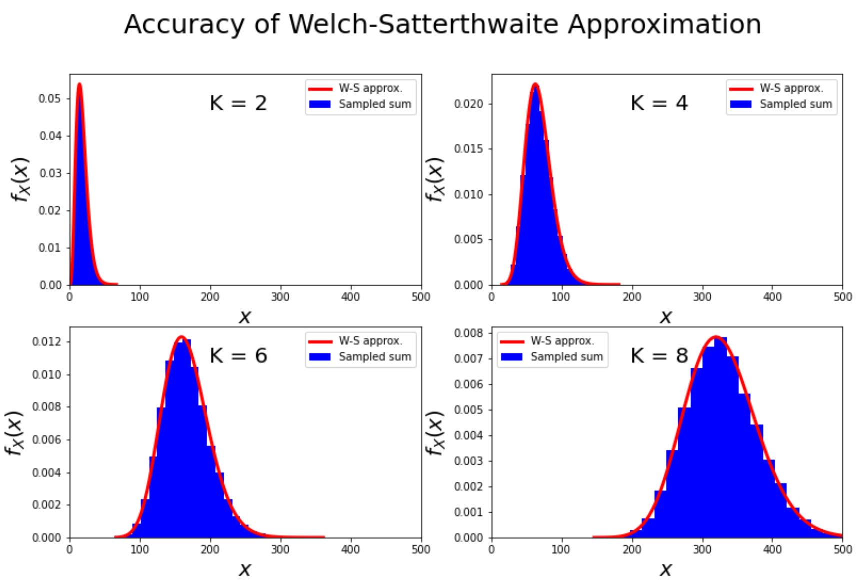


Figure 5.1: Accuracy of Welch-Satterthwaite Approximation

It can be observed that the Welch-Satterthwaite Approximation is accurate for the sum of a small to medium number of Gamma RVs.

5.1.2 Effect of Non-Orthogonal Pilots

Table 5.2 shows the numerical values used for experimentally studying the effect of non-orthogonal pilots in a cell-free network. In order to remove the effect of random processes such as position of UEs and APs, log-normal shadowing and small-scale fading, we have assumed all gains to be 1, i.e. $g_{mk} = 1, \; \forall \, m, k$.

Table 5.2: Values for Effect of Non-Orthogonal Pilots

Parameter	**Value**
Number of UEs (K)	`10`
Numbers of APs (Ms)	`[10,15,20,25,30]`
Lengths of Pilot Sequence (τ_ps)	`[4,5,7,9,10]`
Normalized Symbol Power (ρ_k)	100 mW $\forall$ UEs

Figure 5.2 shows the effect non-orthogonal pilots for various lengths of pilot sequence and various numbers of APs.

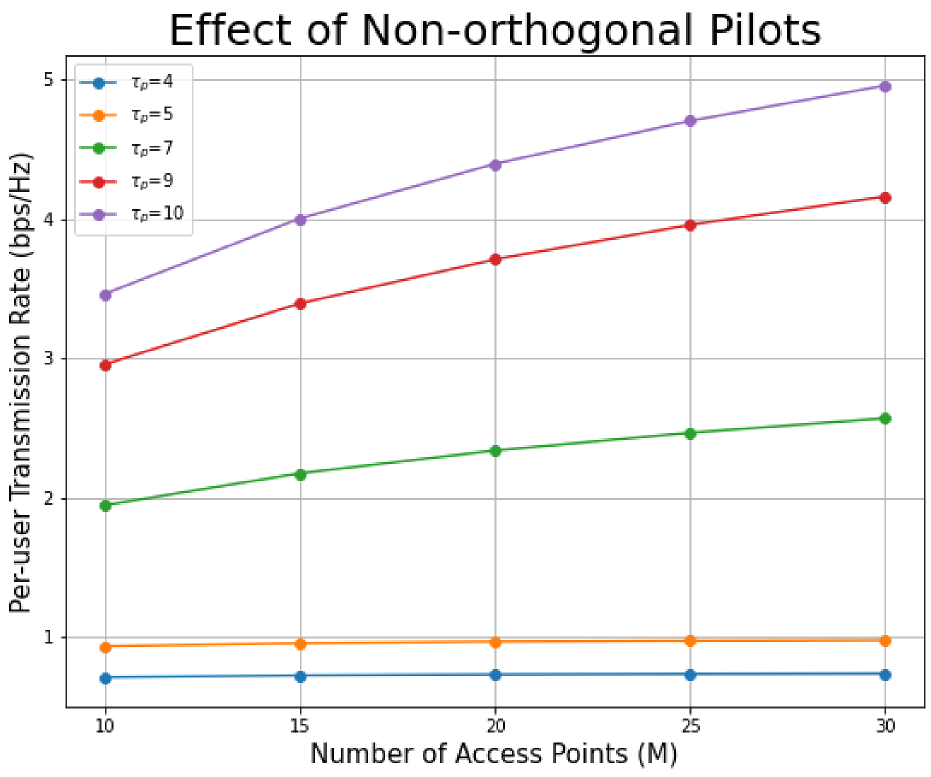


Figure 5.2: Effect of Non-Orthogonal Pilots

It can be observed that for $\tau_p \leq \frac{K}{2}$, increasing the number of APs has little effect on Per-User Transmission Rate. For $\tau_p > \frac{K}{2}$, increasing the number of APs increases the Per-User Transmission Rate.

5.1.3 Effect of Correlated Shadowing

Table 5.3 shows the numerical values used for experimentally studying the effects of correlated shadowing. The effects of path-loss and small-scale fading have been ignored (i.e. $L_{mk} = h_{mk} = 1, \ \forall \ m, k$) in order to remove the effects of random independent processes.

Table 5.3: Values for Effect of Correlated Shadowing

Parameter	**Value**
Number of UEs (K)	`10`
Numbers of APs (Ms)	`[10,13,17,20]`
Standard Deviation ($\sigma_{mk} \ \forall \ m, k$)	`[4,6]` in dB
Total Noise Power ($\sum \sigma_m^2$)	-30 dBm
Correlation Coefficient (δ)	0.5

Figure 5.3 shows the difference between correlated and uncorrelated shadowing for two different values of σ_{mk} for various numbers of APs.

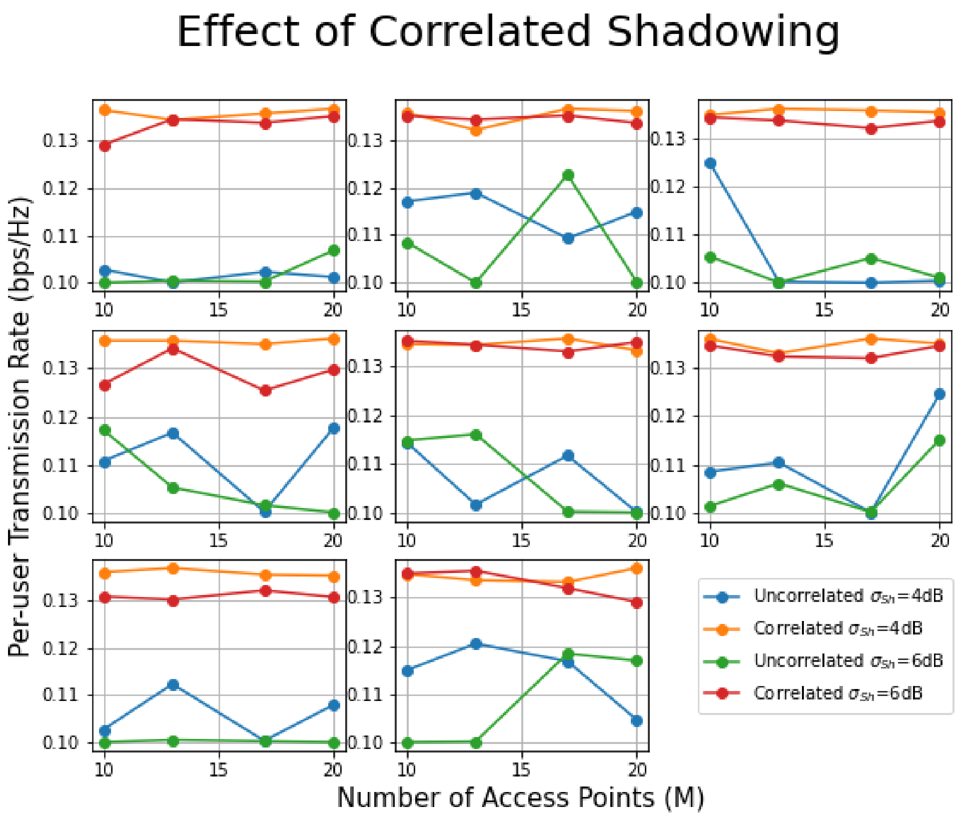


Figure 5.3: Effect of Correlated Shadowing

It can be observed that changing the standard deviation of the log-normal shadowing from $4dB$ to $6dB$ has little effect on the Per-User Transmission Rate. It can also be observed that correlated shadowing has a better performance and varies less. This is because there are less independent RVs involved in correlated shadowing ($M+K$ compared to MK for uncorrelated case). There appears to be no clear effect of changing the number of APs.

5.1.4 Probability of Outage

Table 5.4 shows the numerical values for studying the probability of outage in a cell-free network without any optimized beamforming parameters. In order to remove the effect of unrelated random processes, the effects of path-loss and shadow fading are ignored (i.e. $L_{mk} = \mathcal{F}_{mk} = 1,\ \forall\, m, k$).

Table 5.4: Values for Probability of Outage

Parameter	Value
Numbers of UEs (Ks)	`[25,30,35,40]`
Numbers of APs (Ms)	`[10,32,55,77,100]`
Number of Simulations	`50000`
Transmit Power (p_k)	37 dBm $\forall$ UEs
SINR threshold (γ_{th})	-17 dB
Total Noise Power ($\sum \sigma_m^2$)	-80 dBm

Figure 5.4 shows the probability of outage for various numbers of APs and UEs in a Nakagami-$\mathcal{M}$ fading cell-free network.

It can be observed that the probability of outage decreases approximately in a logarithmic fashion when the number of APs is increased for a given number of UEs. It can also be observed that the probability of outage increases as the number of UEs increases. This is because an increase in the number of UEs increases the IUI interference.

5.2 Environmental Configuration

Table 5.5 shows the numerical values of parameters in the environment for optimizing the beamforming parameters using DDPG in a cell-free networks. We assume the instantaneous CSI is available at the CPU at every time step, i.e. g_{mk} is known, so there is no need to estimate it using pilot sequences. In 2.8, the total noise power is $\sum_{m=1}^{M} \sigma_m^2 \omega_{mk}^2$. This is substituted with its upper bound $M\sigma_m^2$ for simplicity.

Table 5.6 shows the values of parameters and options chosen for the DDPG network

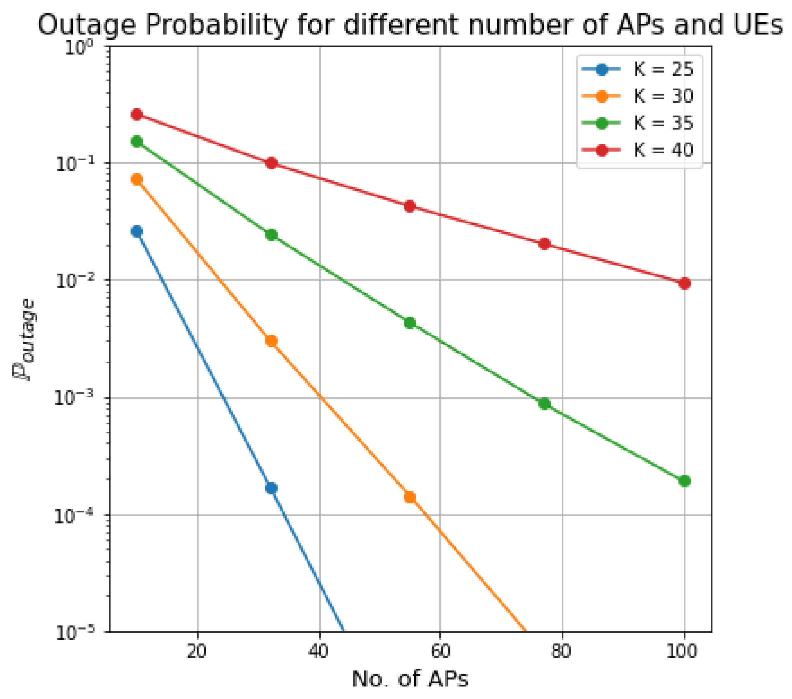


Figure 5.4: Probability of Outage under Nakagami-$\mathcal{M}$ Fading

Table 5.5: Values of Parameters in the Environment

Parameter	**Value**
Radius of Area	18 m
Number of UEs (K)	$[7, 10]$
Number of APs (M)	$[15, 50]$
Path-loss Exponent (κ)	2
Shape Parameter (α_{mk})	$1 \; \forall \; m, k$
Rate Parameter (β_{mk})	$1 \; \forall \; m, k$
Transmit Power (p_k)	37 dBm $\forall$ UEs
Standard Deviation of Shadow Fading (σ_{mk})	8 dB $\forall \; m, k$
Noise Power (σ_m^2)	-114 dBm $\forall \; m$

that optimizes the beamforming parameters in a cell-free network.

Image 5.5 shows the placement of APs and UEs for $K = 10$ and $M = 50$. Image 5.6 shows the placement of APs and UEs for $K = 7$ and $M = 15$.

Table 5.6: Values of Parameters for the Network

Name	**Choice**
Neuron Type	`keras.Dense`
Discount Factor (ζ)	0.99
Polyak Averaging Factor (τ)	0.001
Critic Optimizer	`Adam`
Actor Optimizer	`Adam`
Number of Episodes (E)	400
Number of Timesteps per Episode (T)	200
Replay Buffer Size	100000
Batch Size ($\mathcal{L}$)	64

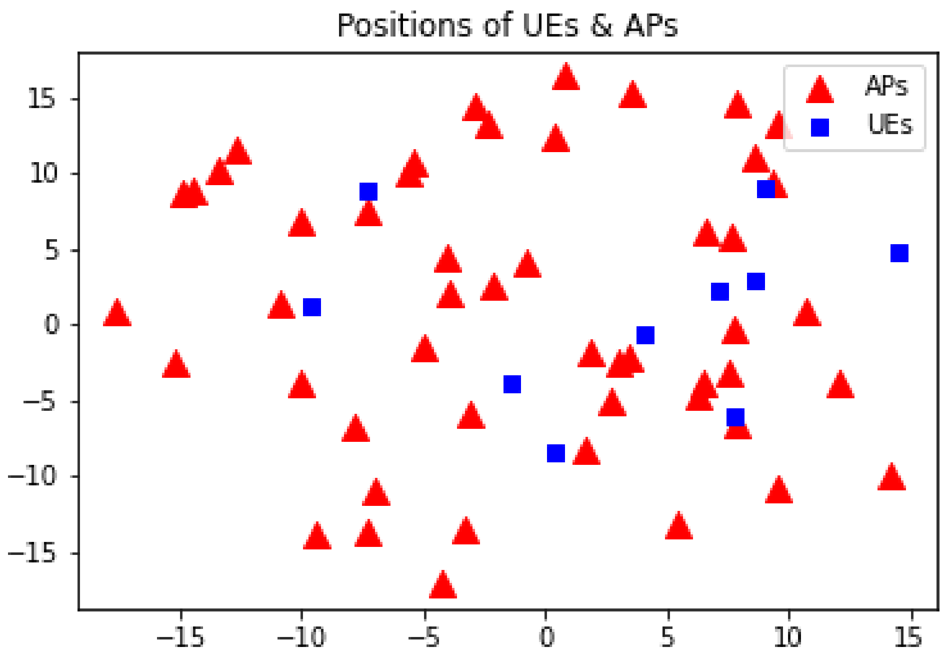


Figure 5.5: Arrangement of APs and UEs for $K = 10$ and $M = 50$

5.3 MATLAB Benchmark

For the arrangement shown in figure 5.5, figure 5.7 shows the normalized square root `constant_matrix` consisting of the path-loss and log-normal shadowing in grayscale image format along with the optimized beamforming parameters obtained using the `fmincon` function also displayed as a grayscale image. Figure 5.9 shows the step sizes of the objective function during the process of optimization of the `fmincon` function for all iterations.

For the arrangement shown in figure 5.6, figure 5.8 shows the normalized square root `constant_matrix` consisting of the path-loss and log-normal shadowing in grayscale

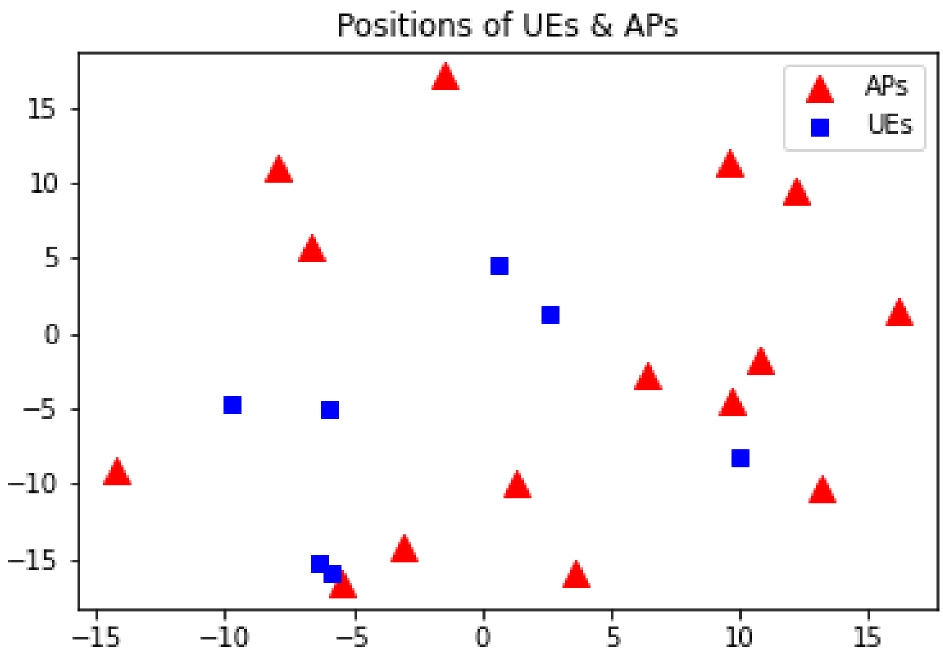


Figure 5.6: Arrangement of APs and UEs for $K = 7$ and $M = 15$

image format along with the optimized beamforming parameters obtained using the `fmincon` function and the DDPG network also displayed as grayscale images. Figure 5.10 shows the step sizes of the objective function during the process of optimization of the `fmincon` function for all iterations.
Figure 5.11 and 5.12 show the Total Transmission Rate (i.e. total throughput) for 4 different sets of beamforming parameters average over 10000 trials.

(i) $\omega_{mk} = 1, \forall\, m, k$

(ii) Optimized beamforming parameters obtained using `fmincon` function.

(iii) For each AP, the UE with the highest entry in the `constant_matrix` gets 1, the rest get 0.

(iv) For each UE, the AP with the highest entry in the `constant_matrix` gets 1, the rest get 0.

It can be observed that the beamforming parameters obtained from MATLAB combine each UE with only one AP (i.e. the beamforming parameters for a UE is almost zero for all APs but one AP has almost one).

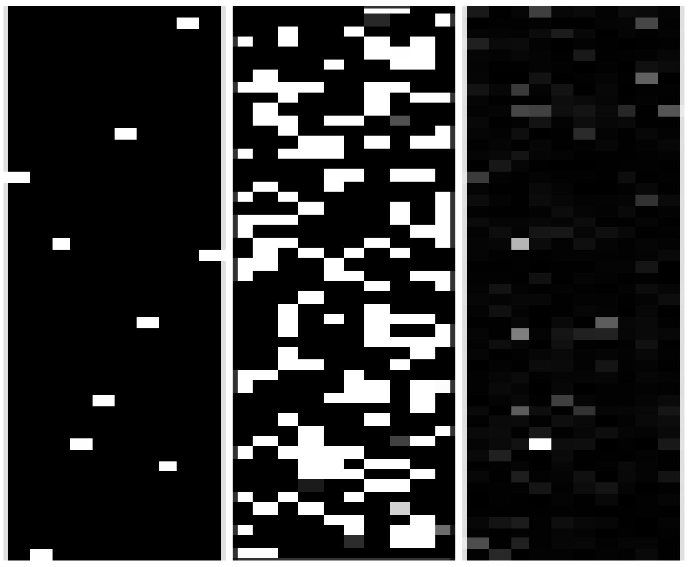


Figure 5.7: Left to Right: (a) Optimized Beamforming parameters from MATLAB. (b) Optimized Beamforming Parameters from DDPG. (c) Normalized Square Root of the $constant\_matrix$, $M - 50,\ K = 10$

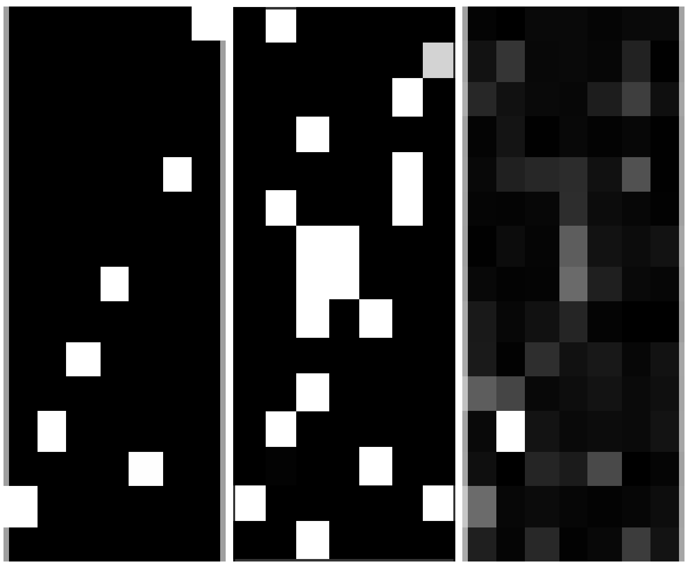


Figure 5.8: Left to Right: (a) Optimized Beamforming parameters from MATLAB. (b) Optimized Beamforming Parameters from DDPG. (c) Normalized Square Root of the $constant\_matrix$, $M - 15,\ K = 7$

5.4 Deep Learning Network

Figure 5.13 and 5.14 shows the training episodic returns averaged over the past 40 episodes for the DDPG network. The episodic reward is the average reward over all the timesteps in that episode. In order to get the Per-User Transmission Rate, the episodic return has to be divided by the number of UEs. It can be observed that the episodic reward doesn't increase significantly after the first few episodes.

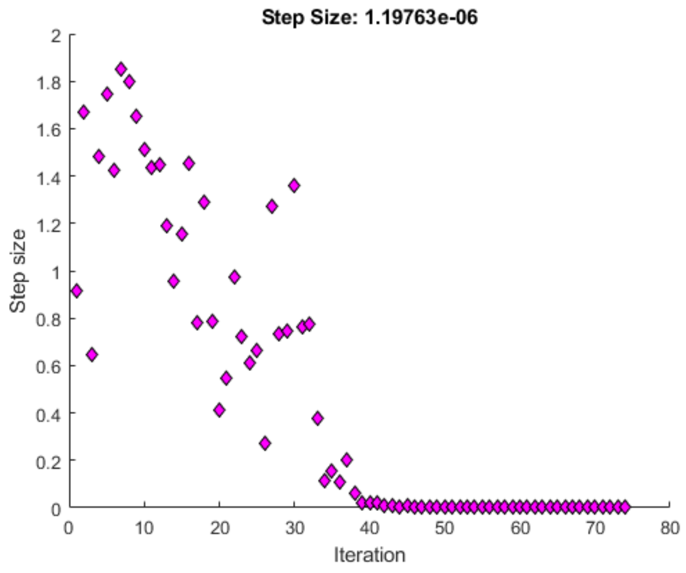


Figure 5.9: Step Sizes during process of Optimization in MATLAB, $M = 50,\ K = 10$

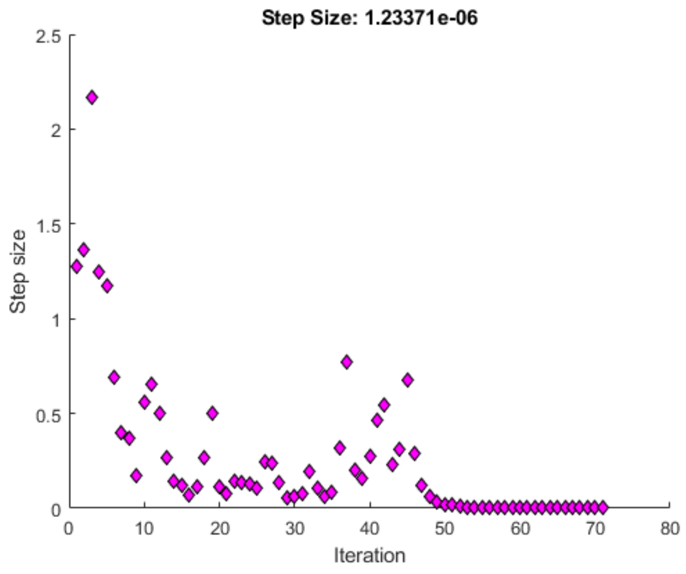
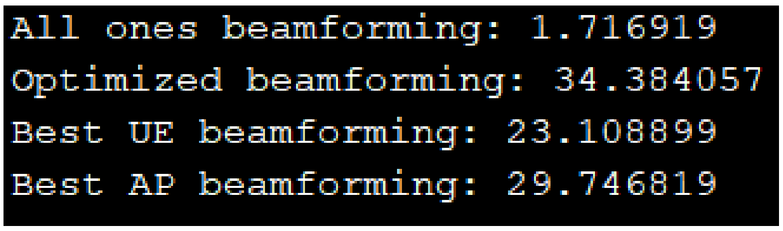
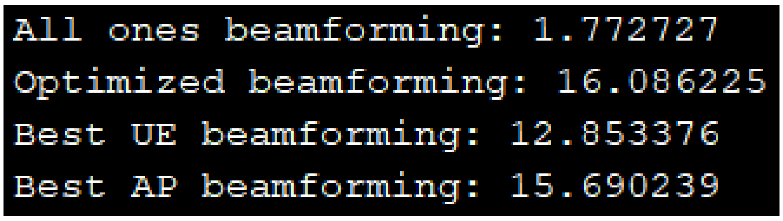


Figure 5.10: Step Sizes during process of Optimization in MATLAB, $M = 15,\ K = 7$



```
All ones beamforming: 1.716919
Optimized beamforming: 34.384057
Best UE beamforming: 23.108899
Best AP beamforming: 29.746819
```

Figure 5.11: Performance of four different sets of Beamforming Parameters, $M = 50,\ K = 10$



```
All ones beamforming: 1.772727
Optimized beamforming: 16.086225
Best UE beamforming: 12.853376
Best AP beamforming: 15.690239
```

Figure 5.12: Performance of four different sets of Beamforming Parameters, $M = 15,\ K = 7$

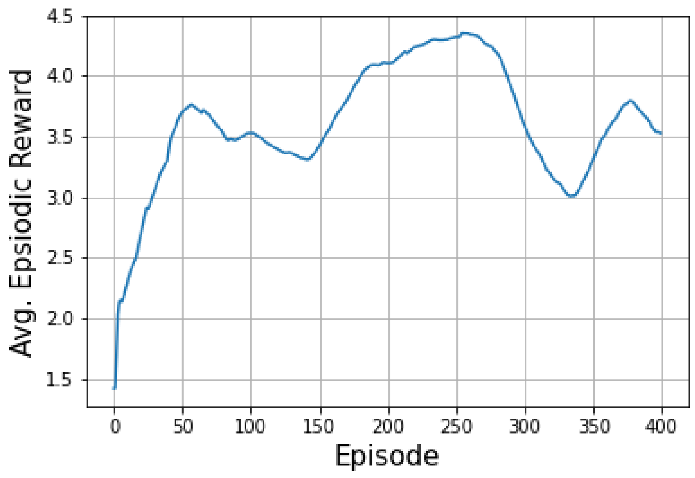


Figure 5.13: Training the DDPG network, $M = 50,\ K = 10$

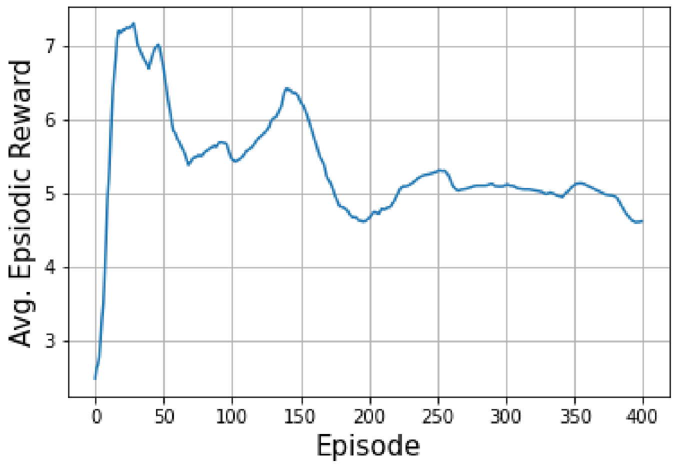


Figure 5.14: Training the DDPG network, $M = 15,\ K = 7$

CHAPTER 6

Conclusion

In this project, the accuracy of Welch-Satterthwaite Approximation has been verified experimentally for small to medium number of Gamma RVs. The effect of non- orthogonal pilots on Per-User Transmission Rate in a cell-free network has been experimentally analysed in isolation from other random effects. The effect of correlated log-normal shadow fading and the standard deviation (σ_{mk}) of the fading on Per-User Transmission Rate in a cell-free network has been experimentally studied. Probability of outage in a cell-free network without an optimized beamforming has been experimentally studied. The theoretical formula for the probability of outage and its derivation has been given. The expression for calculating the SINR of various UEs in a cell-free network has been given with its derivation.
A brief introduction to *Reinforcement Learning* has been given. The nature and salient features of DDPG have been given with explanations for what those features mean. A DDPG network has been implemented to optimize the beamforming parameters to maximize the Per-User Transmission Rate and its performance has been benchmarked using the *bisection method* in the MATLAB function `fmincon`. The nature of the optimized beamforming parameter obtained from MATLAB has been shown visually. Clearly understandable algorithms for all the code used to generate the results have been given in pseudo code form.

6.1 Scope for Future Work

Probability of Outage using the optimized beamforming parameters can be analysed. Other reinforcement learning algorithms like *Proximal Policy Optimization* (PPO) [15] can be used to optimize the beamforming parameters. Algorithms designed to estimate the CSI while optimizing the beamforming parameters can be used to bring cell-free networks closer to being implemented in a real world situation. Algorithms that simultaneously take care of the effect of having non-orthogonal pilots and the estimation errors they bring along, while optimizing the beamforming parameters can be looked into.

REFERENCES

[1] **3GPP** (1999). User Equipment (UE) radio transmission and reception (FDD). Technical Specification (TS) 25.101, 3rd Generation Partnership Project. Pg. 13.

[2] **3GPP** (2007). Evolved universal terrestrial radio access (E-UTRA); user equipment (UE) radio transmission and reception. Technical Specification (TS) 36.101, 3rd Generation Partnership Project. Tables 5.5-1 and 5.6.1-1.

[3] **3GPP** (2019). User equipment (UE) radio transmission and reception; part 1: Range 1 standalone. Technical Specification (TS) 38.101-1, 3rd Generation Partnership Project. Table 5.5-1.

[4] **Al-Eryani, Y.**, **M. Akrout**, and **E. Hossain** (2021). Multiple access in cell-free networks: Outage performance, dynamic clustering, and deep reinforcement learning-based design. *IEEE Journal on Selected Areas in Communications*, **39**(4), 1028–1042.

[5] **DeepMind** (2016). AlphaGo. URL `https://www.deepmind.com/research/highlighted-research/alphago`.

[6] **Goldsmith, A.**, *Wireless Communications*, chapter 2.7. Cambridge University Press, 2005. ISBN 978-0-511-13675-7, 48–51.

[7] **Gradshteyn, I.** and **I. Ryzhik**, *Table of Integrals, Series, and Products*. Academic Press, 2007. ISBN 978-0-12-373637-6. Equation 6.455.2.

[8] **Gulberg, J.**, *Mathematics From the Birth of Numbers*. New York: W.W. Norton, 1997. ISBN 978-0-393-04002-9.

[9] **Interdonato, G.**, **E. Björnson**, **H. Q. Ngo**, **P. K. Frenger**, and **E. G. Larsson** (2019). Ubiquitous cell-free massive MIMO communications. *EURASIP Journal on Wireless Communications and Networking*, **197**. URL `http://arxiv.org/abs/1804.03421`.

[10] **Larsson, E. G.**, **O. Edfors**, **F. Tufvesson**, and **T. L. Marzetta** (2014). Massive MIMO for next generation wireless systems. *IEEE Communications Magazine*, **52**(2), 186–195.

[11] **Lee, D.**, **H. Seo**, **B. Clerckx**, **E. Hardouin**, **D. Mazzarese**, **S. Nagata**, and **K. Sayana** (2012). Coordinated multipoint transmission and reception in lte-advanced: deployment scenarios and operational challenges. *IEEE Communications Magazine*, **50**(2), 148–155.

[12] **Lillicrap, T. P.**, **J. J. Hunt**, **A. Pritzel**, **N. Heess**, **T. Erez**, **Y. Tassa**, **D. Silver**, and **D. Wierstra** (2016). Continuous control with deep reinforcement learning. *International Conference on Learning Representations*. URL `https://arxiv.org/abs/1509.02971`.

[13] **Samsung** (2022). 6G Spectrum: Expanding the Frontier. *Samsung Research*. URL `https://cdn.codeground.org/nsr/downloads/researchareas/2022May_6G_Spectrum.pdf`.

[14] **Satterthwaite, F. E.** (1946). An approximate distribution of estimates of variance components. *Biometrics Bulletin*, **2**(6), 110–114. ISSN 00994987. URL `http://www.jstor.org/stable/3002019`.

[15] **Schulman, J.**, **F. Wolski**, **P. Dhariwal**, **A. Radford**, and **O. Klimov** (2017). Proximal policy optimization algorithms. *Computing Resource Repository*, **abs/1707.06347**. URL `http://arxiv.org/abs/1707.06347`.

[16] **Sutton, R. S.** (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, **3**, 9–44. URL `https://doi.org/10.1007/BF00115009`.

[17] **Sutton, R. S.** and **A. G. Barto**, *Reinforcement Learning: An Introduction*, chapter 2. The MIT Press, 2015, 2nd edition, 31–32.

[18] **Wood, G. R.** (1992). The bisection method in higher dimensions. *Mathematical Programming*, **55**(1–3), 319–337.