Homework 5 submission

ECET 512 — Wireless Systems

Chris Uzokwe
ID: cnu25

February 16, 2021

1 Submitted files

For this assignment, besides this report, the following archives were created:

1.1 SRC Folder

+ *"main.py"*: This **script** simulates the Rayleigh fading signal using the Clark and Gans Fading Model Technique. The script then borrows from homework 3, extending the path loss model with shadowing effects, to include the fading effects. Graphs with each are generated.

1.2 DOC Folder

+ *"fading20fm.png"* Graph of the simulated Rayleigh Fading Signal with max Doppler frequency fm = 20 Hz.

+ *"fading200fm.png"* Graph of the simulated Rayleigh Fading Signal with max Doppler frequency fm = 200 Hz.

+ *"plclean.png"* A graph of the mobile stations received power from both base stations at each second of travel. Shadowing and fading effects neglected.

+ *"plshadow.png"* A graph of the mobile stations received power from both base stations at each second of travel. Shadowing effects included.

+ *"plfade.png"* A graph of the mobile stations received power from both base stations at each second of travel. Fading effects included.

+ *"plshadow+fade.png"* A graph of the mobile stations received power from both base stations at each second of travel. Shadowing and fading effects included.

2 Simulated Rayleigh Fading Signal

Below are graphs of the normalized Rayleigh Fading signal, with maximum Doppler frequencies of 20Hz and 200Hz, respectively. We can see that when the maximum frequency is only 20Hz, the spectrum of the signal we recieved is less varied, resulting in a time series with less fluctuation. This is different than the 200Hz fading signal, which shows more robust movement in the time series, due to the larger acceptance of frequencies from the Doppler spectrum.

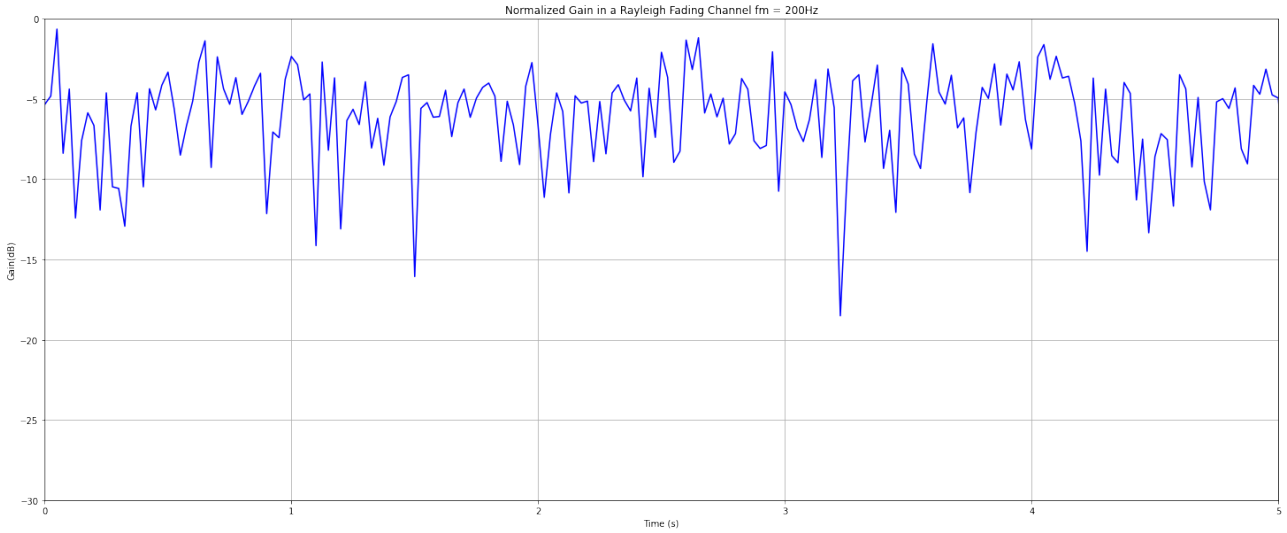

Figure 1: Rayleigh Fading Signal with Max Doppler Frequency fm = 20Hz

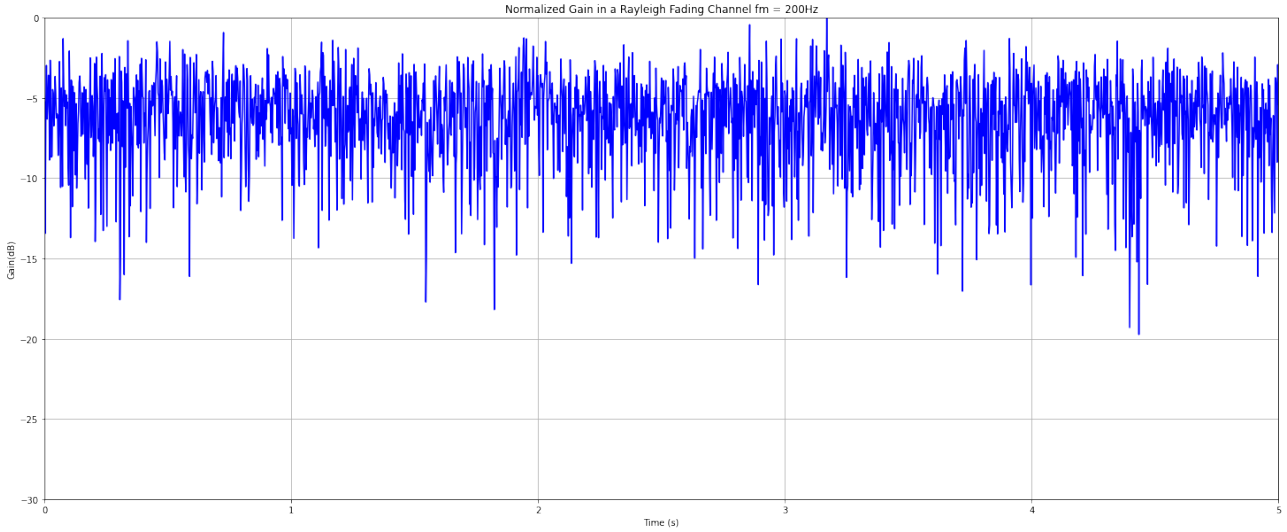

Figure 2: Rayleigh Fading Signal with Max Doppler Frequency fm = 200Hz.

3 Downlink Power vs. Time with Shadowing+Fading

The following plots depict the effects of multipath fading and shadowing on our path loss model of mobile user movement. The graphs show that shadowing is really the dominant factor in this larger scale depiction. The fading model is similar to the original, but the reality is the changes are hard to recognize because they are happening so fast. One might model the distribution of both fading and shadowing using the Nakagami-lognormal distribution.

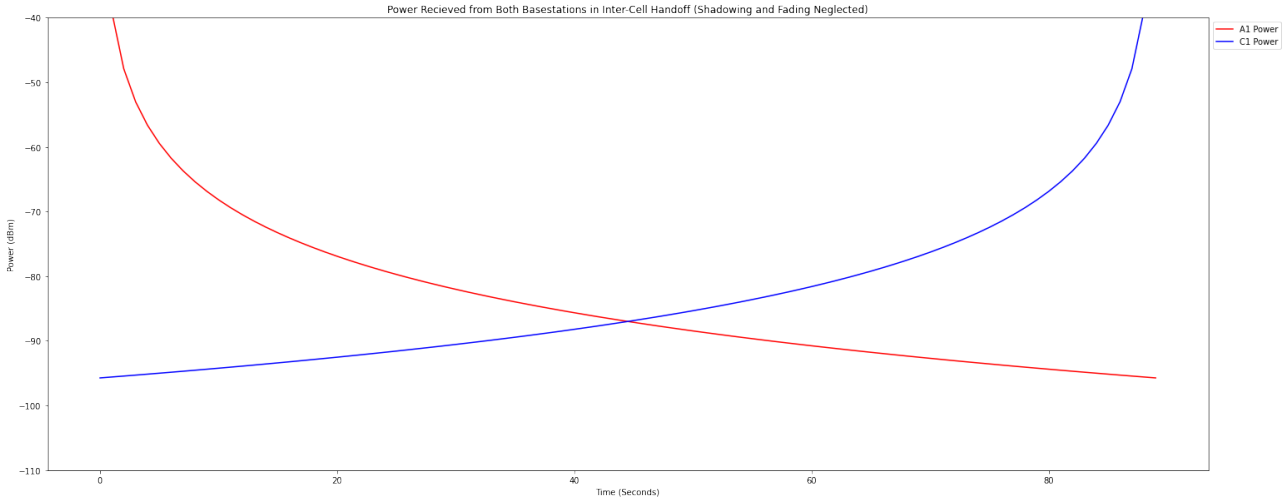

Figure 3: Mobile stations received power from both base stations at each second of travel. Shadowing and fading effects neglected.

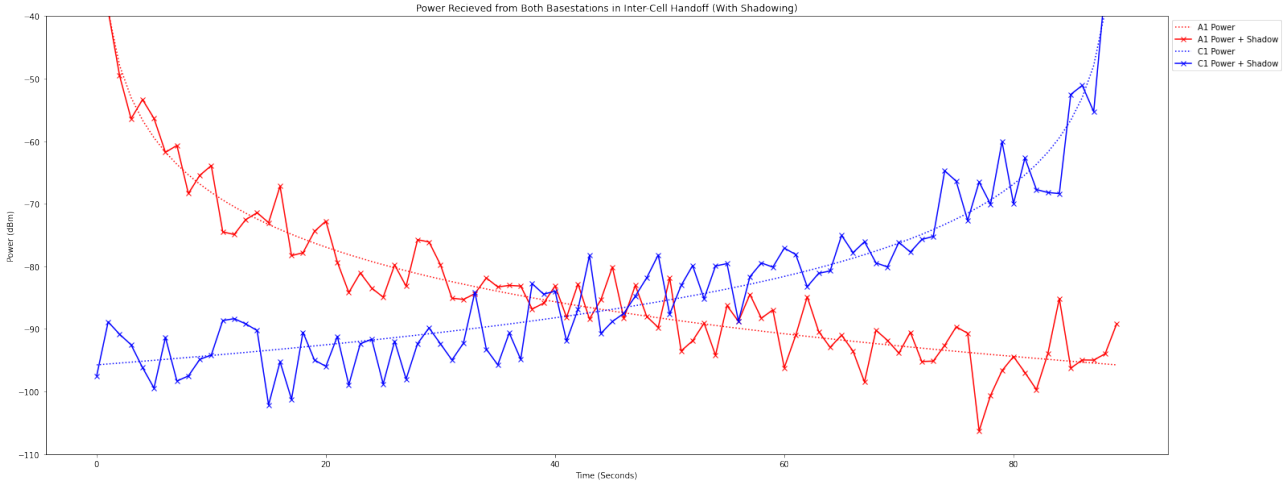

Figure 4: Mobile stations received power from both base stations at each second of travel. Shadowing effects included.

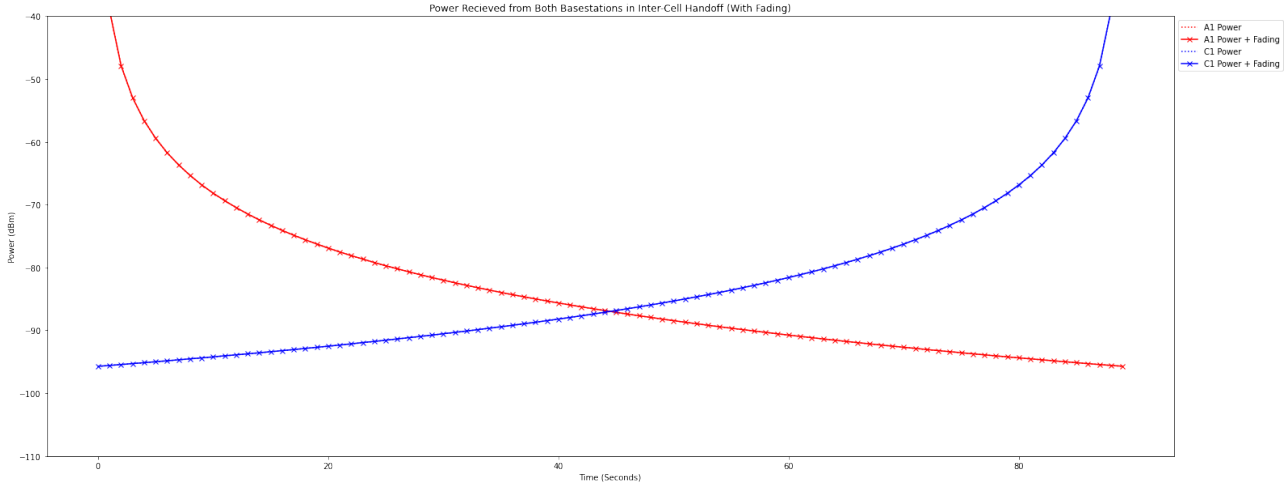

Figure 5: Mobile stations received power from both base stations at each second of travel. Fading effects included.

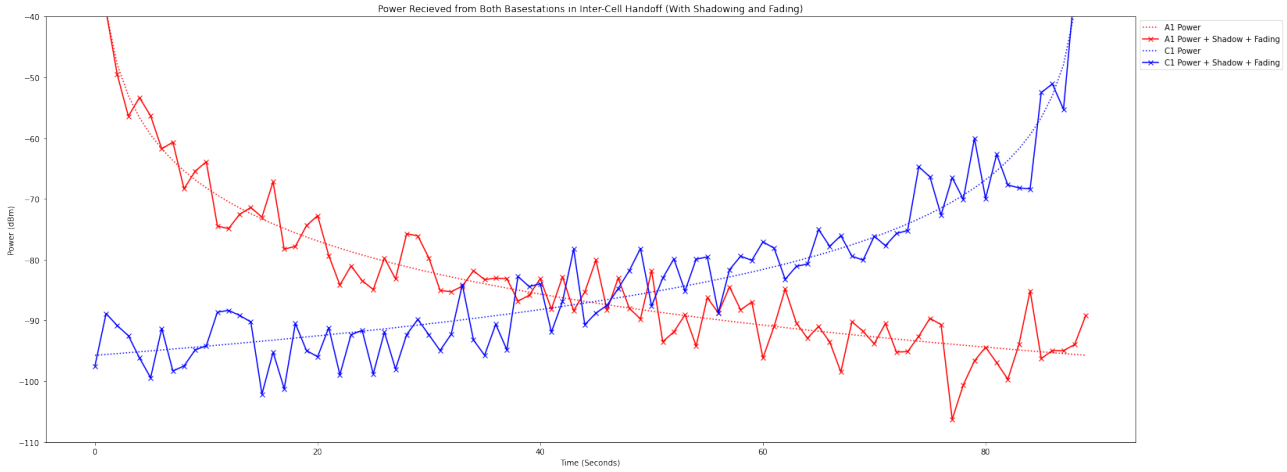

Figure 6: Mobile stations received power from both base stations at each second of travel. Shadowing and fading effects included.

4 Python code

```
# -*- coding: utf-8 -*-
"""ECET 512 HW 5

# Simulated Rayleigh Fading Signal
"""

import numpy as np
import matplotlib.pyplot as plt

# simulation parameters
N = 8192
fm20 = 20
fm200 = 200
fm = fm20

df = (2*fm)/(N-1)
T = 1/df
print(T)
# generating doppler spectrum
spectrum = np.arange(-fm, fm+(df*1/2), df)

dspectrum = np.sqrt([1.5/(np.pi*fm*np.sqrt(1-(f/fm)**2)) for f in spectrum])
dspectrum[0]=2*dspectrum[1]-dspectrum[2]
dspectrum[-1]=2*dspectrum[-2]-dspectrum[-3]

# Gaussian RVs and Frequency Spectrum
pos = np.array([])
pos2 = np.array([])

for i in range(int(N/2)):
  pos = np.append(pos, [np.random.normal(0, 1) + 1j*np.random.normal(0, 1)])
  pos2 = np.append(pos2, [np.random.normal(0, 1) + 1j*np.random.normal(0, 1)])

neg = np.flip(np.conj(pos))
neg2 = np.flip(np.conj(pos2))

gaus1 = np.concatenate((neg, pos))
gaus2 = np.concatenate((neg2, pos2))

# Shaping Gaussians by Dspectrum
X = gaus1*np.sqrt(dspectrum)
Y = gaus2*np.sqrt(dspectrum)

#tX = np.insert(X, 4096, 0)
#tY = np.insert(Y, 4096, 0)

tX = np.abs(np.fft.ifft(X))**2
tY = np.abs(np.fft.ifft(Y))**2
```

Figure 7: Snippet of main.py.