Homework 8 submission

ECET 512 — Wireless Systems

Chris Uzokwe
ID: cnu25

March 9, 2021

1 Submitted files

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

1.1 SRC Folder

+ *"main.py"*: This **script** demonstrates the fading statistics of a Rayleigh fading envelope, as well as a Bit Error Rate Simulation using the generated fading envelope. BER and SNR simulations are extended to include Spatial Diversity, Maximum Ratio Combining, and Spatial Multiplexing. A user is simulated walking through a base station to visualise performance of the system.

1.2 DOC Folder

+ *"miso_sd.png"* A plot of bit error rate vs SNR for spatially diverse system.
+ *"miso_src.png"* Plot of how the BER vs. SNR for .
+ *"mimo_smux.png"* Plot of how the SNR changes as a user moves through a cell.
+ *"mu_sd.png"* A plot of mobile user distance vs BER for spatially diverse system.
+ *"mu_mrc.png"* Mobile user distance vs. BER for Maximum ratio combined system.
+ *"mu_smux.png"* Mobile user distance vs. BER for spatial multiplexed system.

2 BER vs. SNR

Below are demonstrations of the BER vs. SNR for the channels with varying selection configuration.

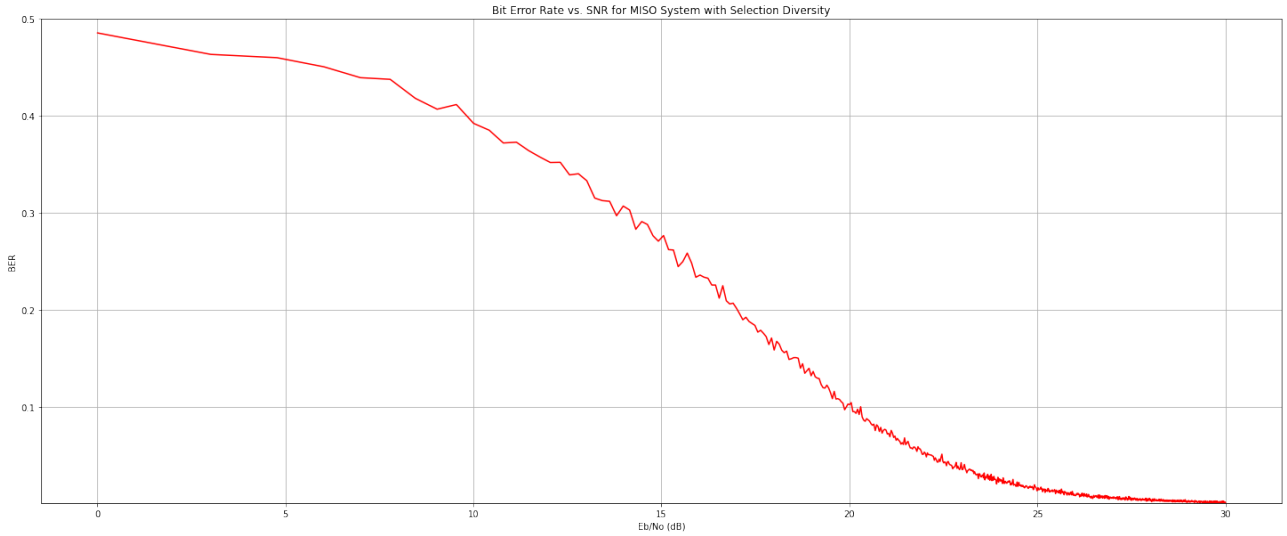

Figure 1: Basestation using Selection Diversity.

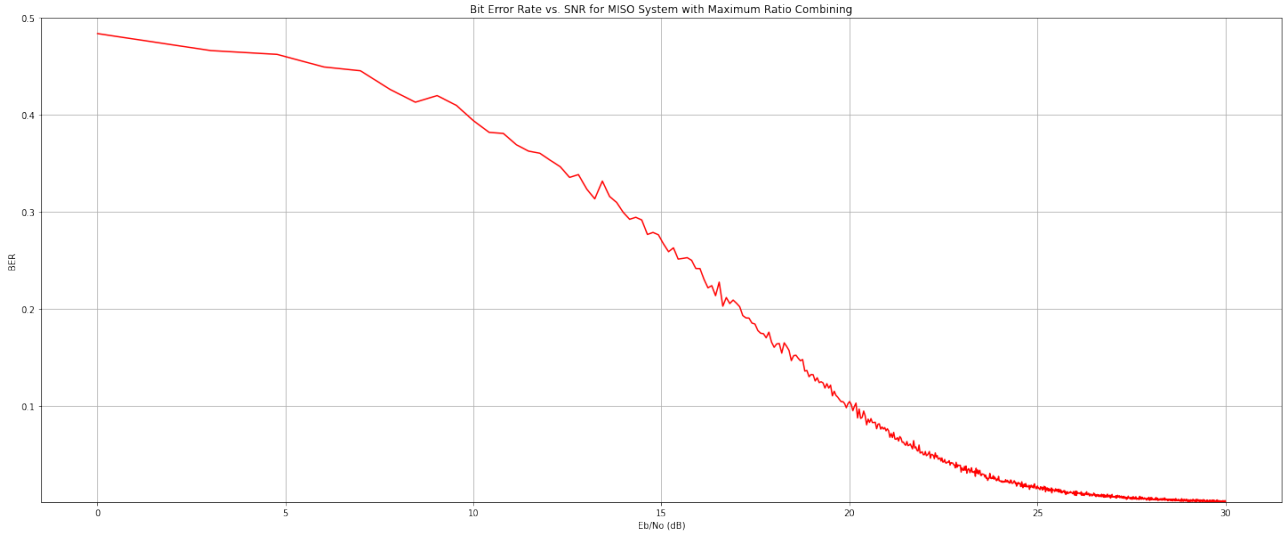

Figure 2: Basestation using Maximum Ratio Combining.

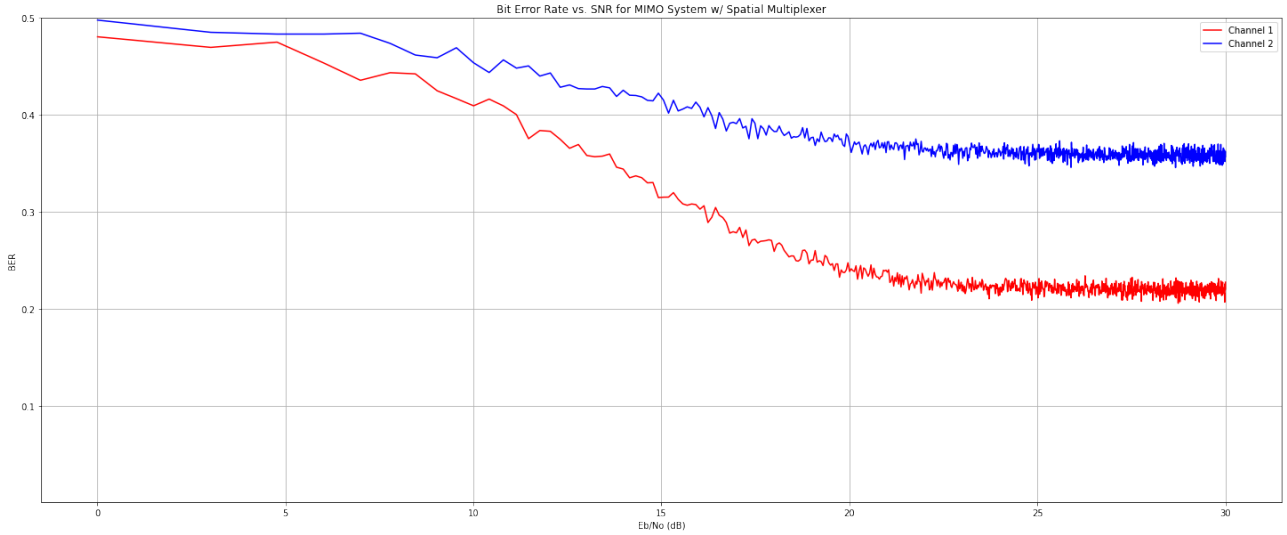

Figure 3: Basestation using Spatial Multiplexing.

3 Bit Error Rate vs. Mobile User Position

Below are the results of the Bit Rate vs Mobile User Distance. SNR values remain the same as in homework 7.

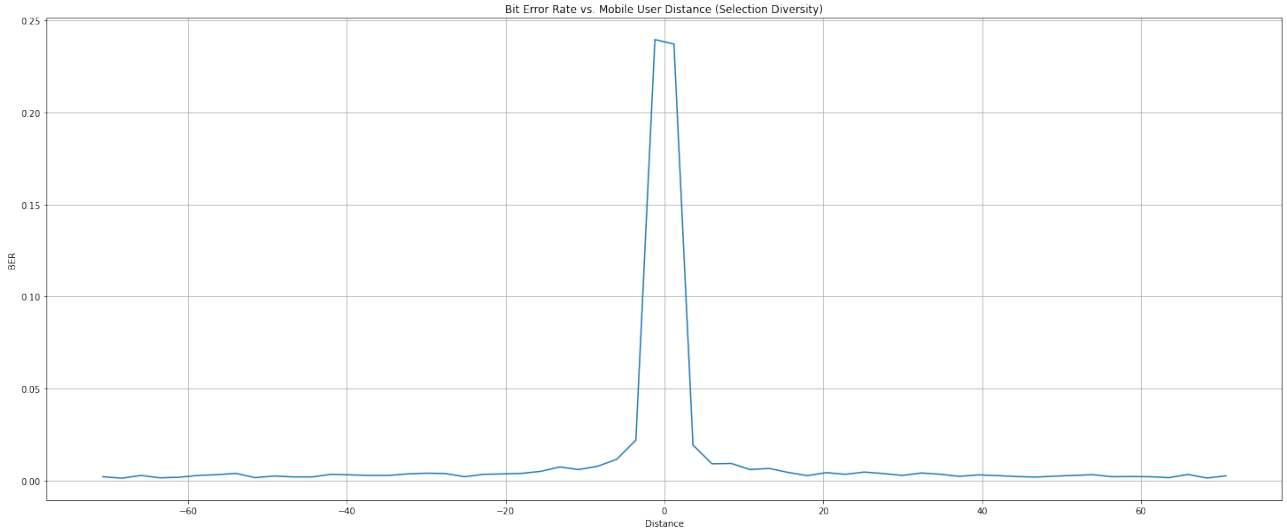

Figure 4: Mobile user BER when using spatial diversity.

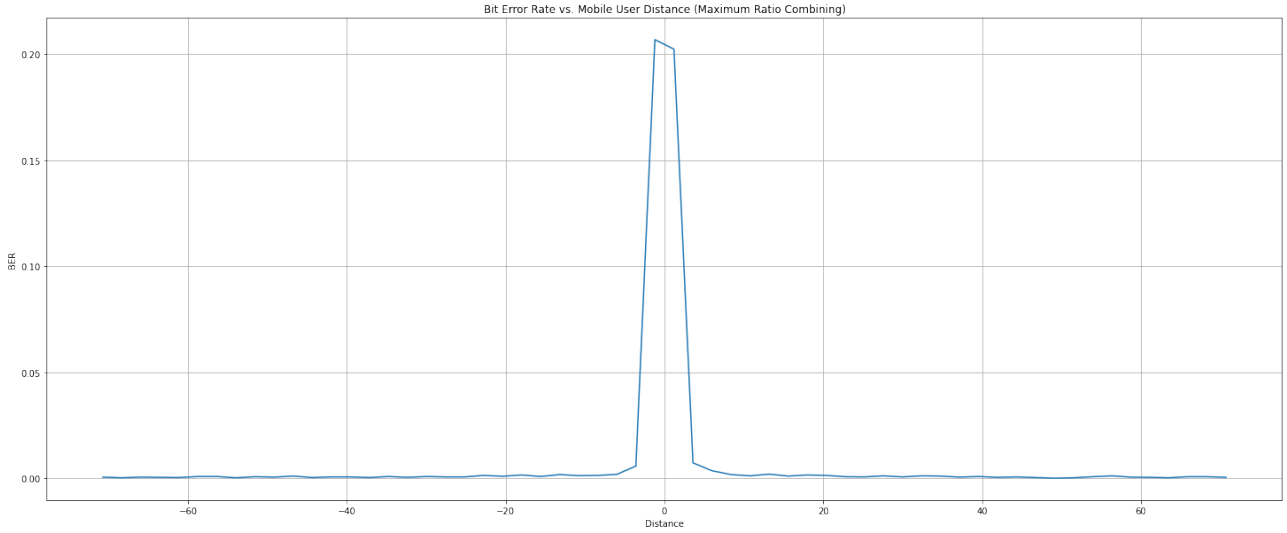

Figure 5: Mobile user BER when using Maximum Ratio Combining.

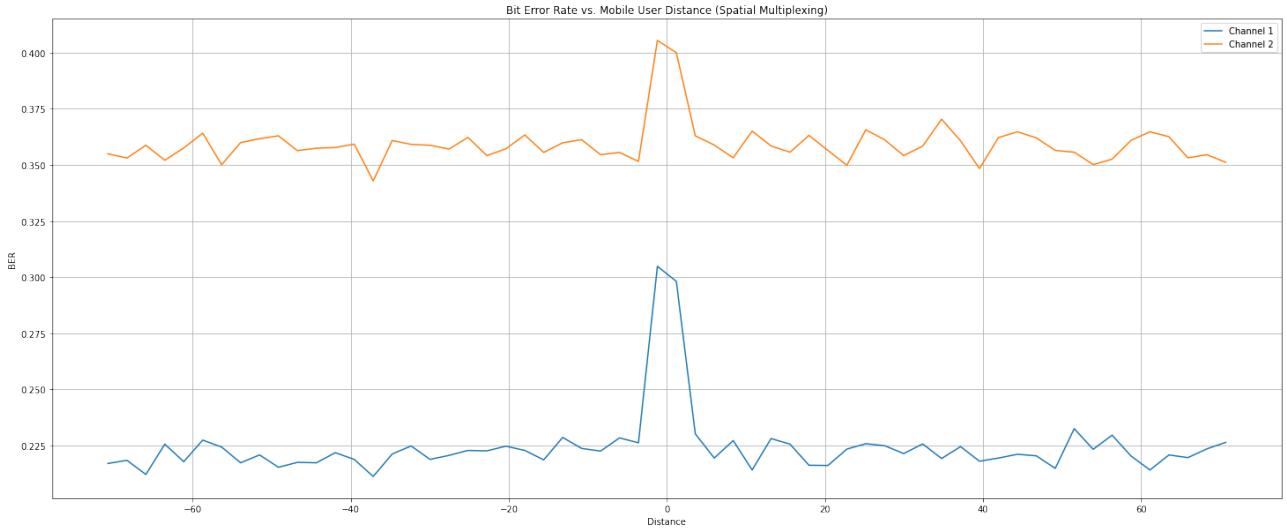

Figure 6: Mobile User BER when using a spatially multiplexed channel.

4 Python code

```
"""# MISO w/ Maximum Ratio Combining SNR vs. BER"""

def miso_simulator_mrc(No = 1000):

  k=4
  n_samples= 5000
  Eb = 1

  noise = Eb/No

  X = np.array([])
  bits, x = encoded_bit_gen(n_samples)

  for i in range(k):
    X = np.hstack((X, x))

  X = X.reshape((k, n_samples))

  N = np.array([])


  for i in range(k):
    N = np.hstack((N ,np.random.normal(0, 1, n_samples)*noise))

  N = N.reshape((k, n_samples))

  H = np.array([])

  for i in range(k):
    r, T = rayleighfade(n_samples, 20)
    theta = np.exp(-1j*np.random.uniform(0, 2*np.pi, n_samples))
    h = r*theta

    H = np.hstack((H, h))

  H= H.reshape((k, n_samples))
  Y = H*X + N
```

Figure 7: Snippet of main.py.