End-to-end OFDM system using KDML

EECS 555

Goals

- Implement an end-to-end OFDM system for Tx and Rx using USRP X310 and GNU Radio
- Use python code to write all components of the system, and build a Deep Learning based complete receiver that does joint channel estimation and symbol detection.
- Built a model to obtain an estimate of the channel response using Super Resolution and Image Restoration Neural Networks.

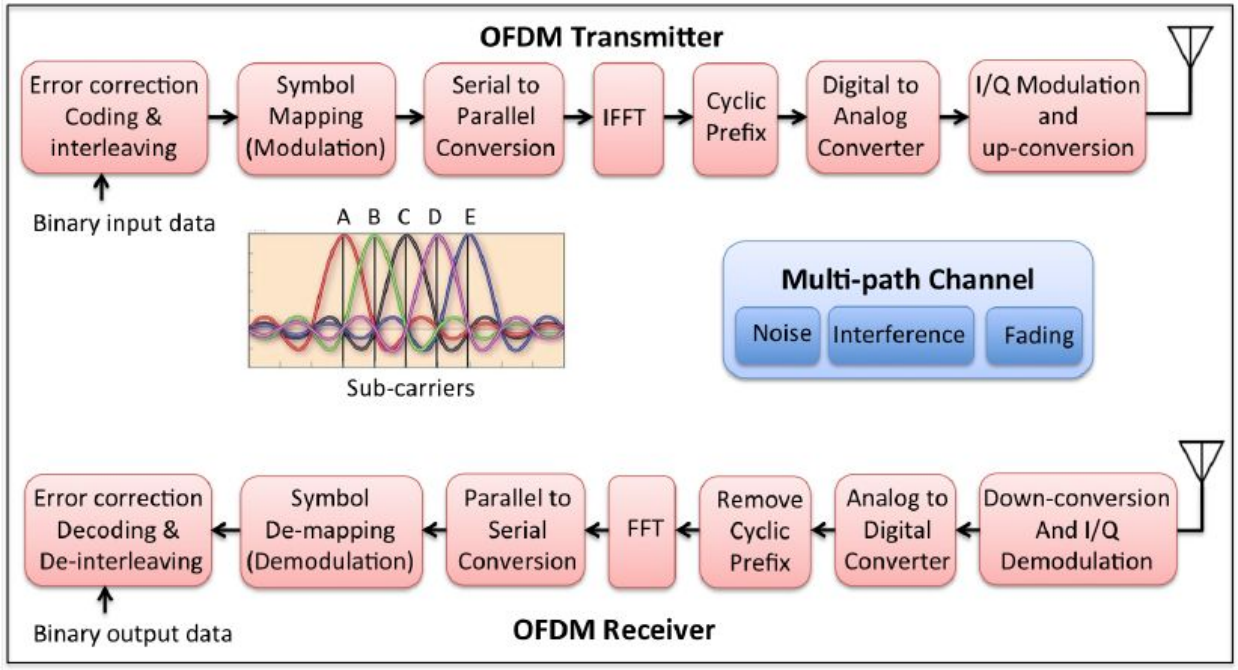
OFDM

- Is a digital multi-carrier modulation scheme that extends the concept of single subcarrier modulation by using multiple subcarriers within the same single channel.

- Multiple orthogonal subcarriers carry the information stream - resistant to frequency selective fading than single carrier systems.

- Each subcarrier is modulated with a conventional digital modulation scheme (such as QPSK, 16QAM, etc.) at low symbol rate.

OFDM


OFDM Transmitter
Error correction Coding & interleaving
Symbol Mapping (Modulation)
Serial to Parallel Conversion
IFFT
Cyclic Prefix
Digital to Analog Converter
I/Q Modulation and up-conversion
Binary input data
A B C D E
Sub-carriers
Multi-path Channel
Noise
Interference
Fading
Down-conversion And I/Q Demodulation
Analog to Digital Converter
Remove Cyclic Prefix
FFT
Parallel to Serial Conversion
Symbol De-mapping (Demodulation)
Error correction Decoding & De-interleaving
Binary output data
OFDM Receiver

Disadvantages of OFDM

- OFDM signals have noise like amplitude variation and has a relatively high large dynamic range, or peak to average power ratio.

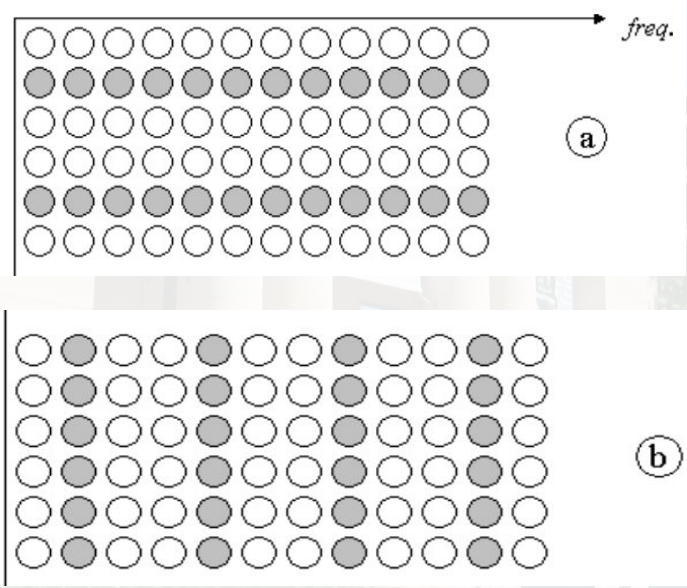
- It is sensitive to carrier frequency offset and drift. Single carrier systems are less sensitive.

Types of OFDM Models Designed

Based on the type of Channel we modelled, we tried modelling two different OFDM systems:

1. Block Type Pilot arrangement
   - For Slow Fading Channels
   - Here we modelled the data to be in sequential.
2. Comb Type Pilot arrangement
   - For Fast Fading Channels
   - Here we are sending block streams of data to the Channel.

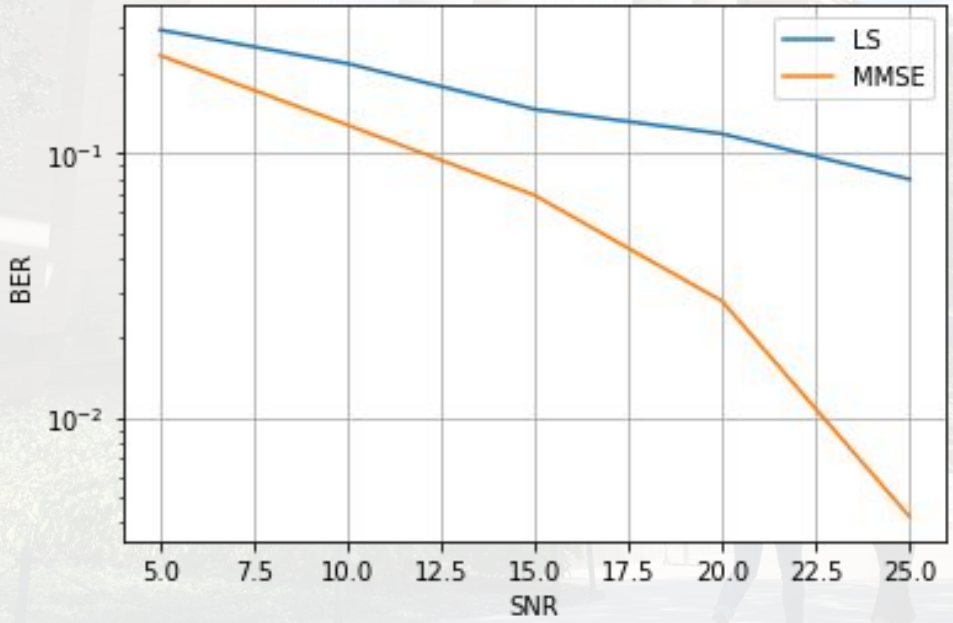
Types of Pilots we used



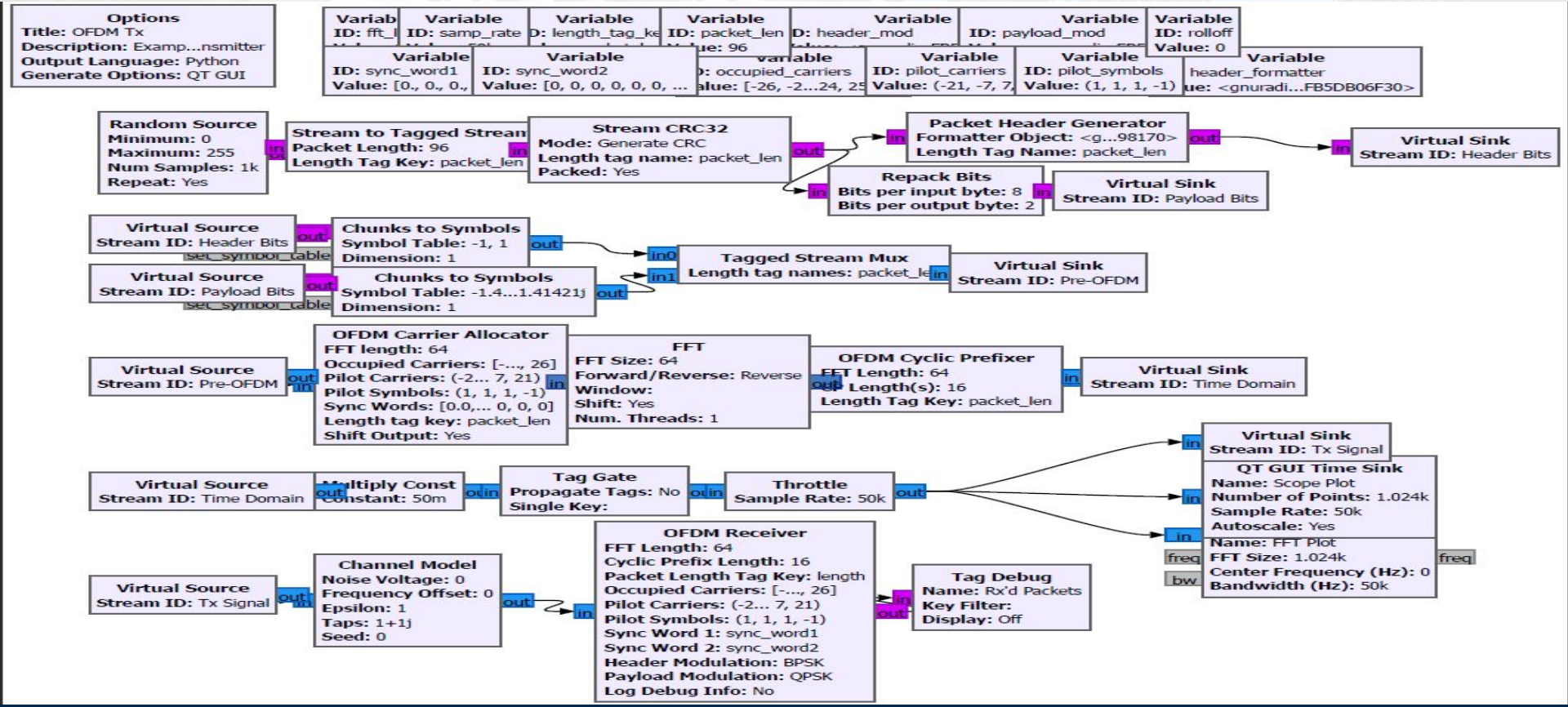
Block Type

Comb Type

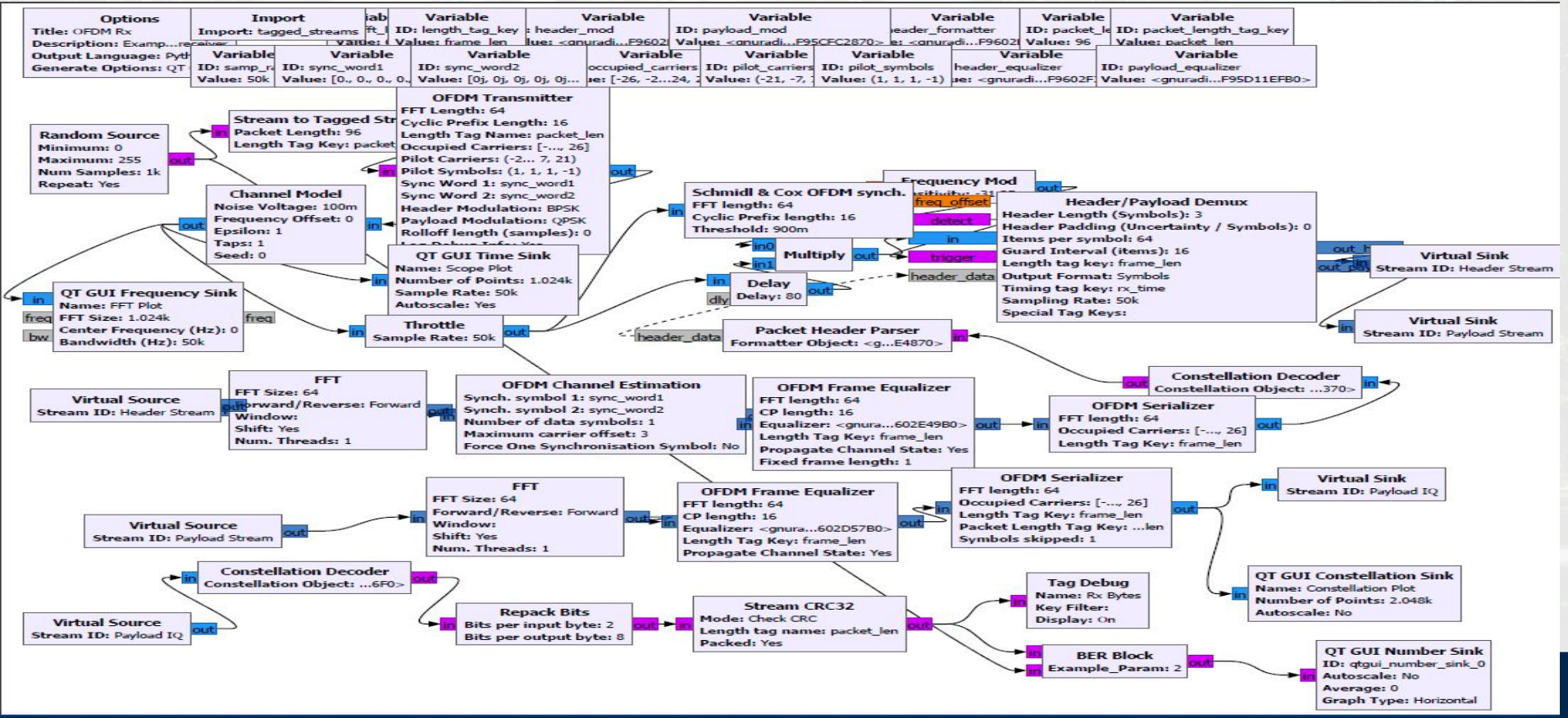
Results for General OFDM System with High Delay Spread



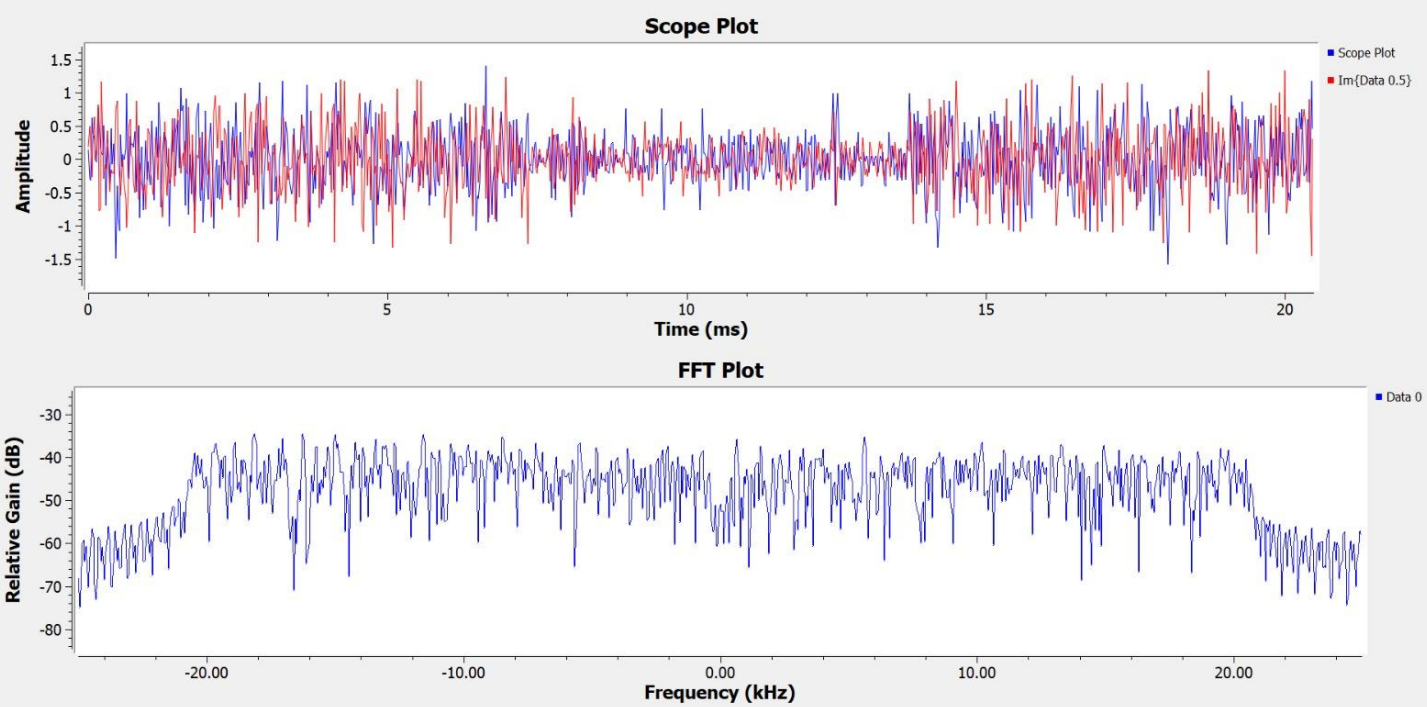
End to end OFDM system implementation in GNU radio



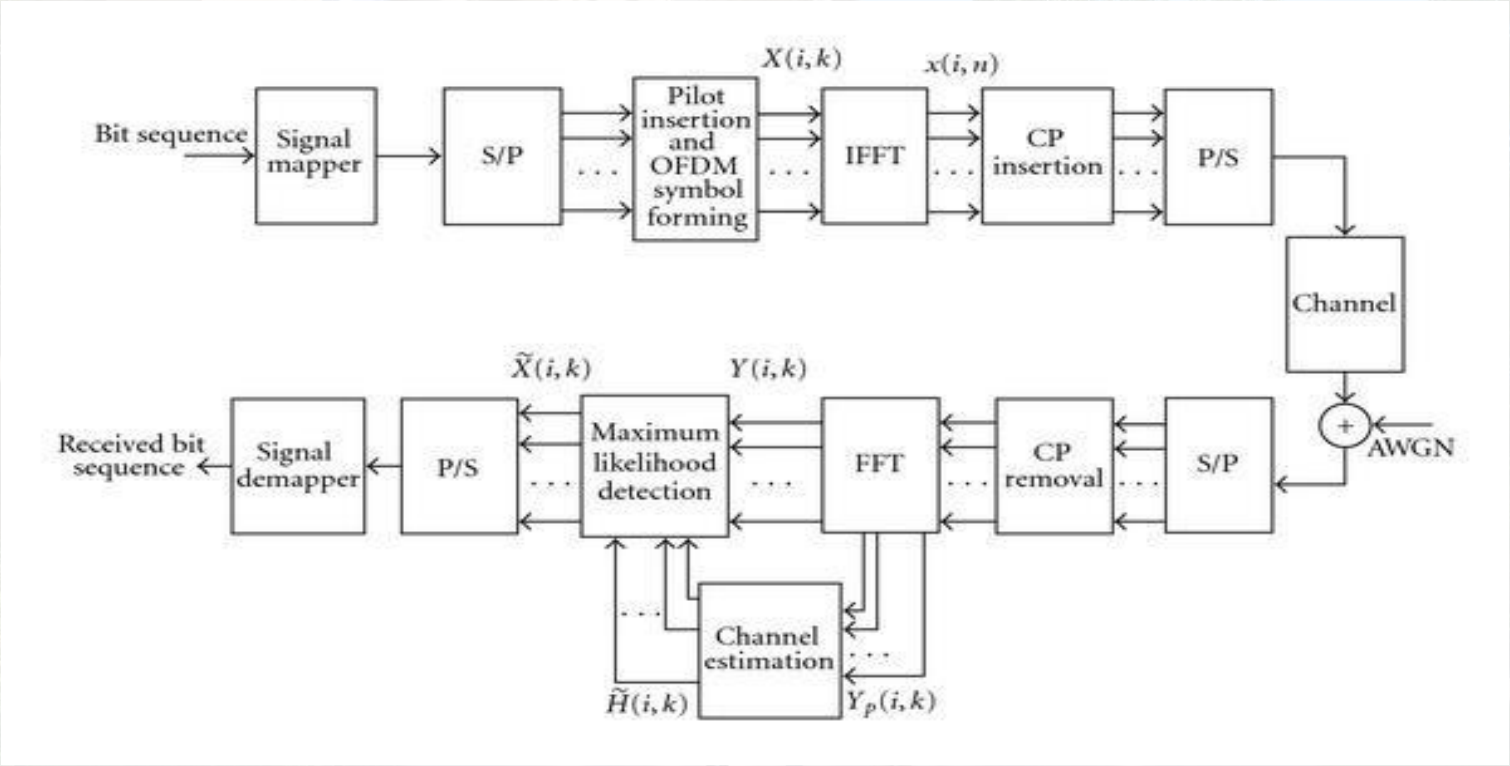
End to end OFDM system implementation in GNU radio



OFDM system using USRP/ GNU Radio



OFDM System in Python



Channel Model

We chose to model our channel with Rician fading (LOS included).

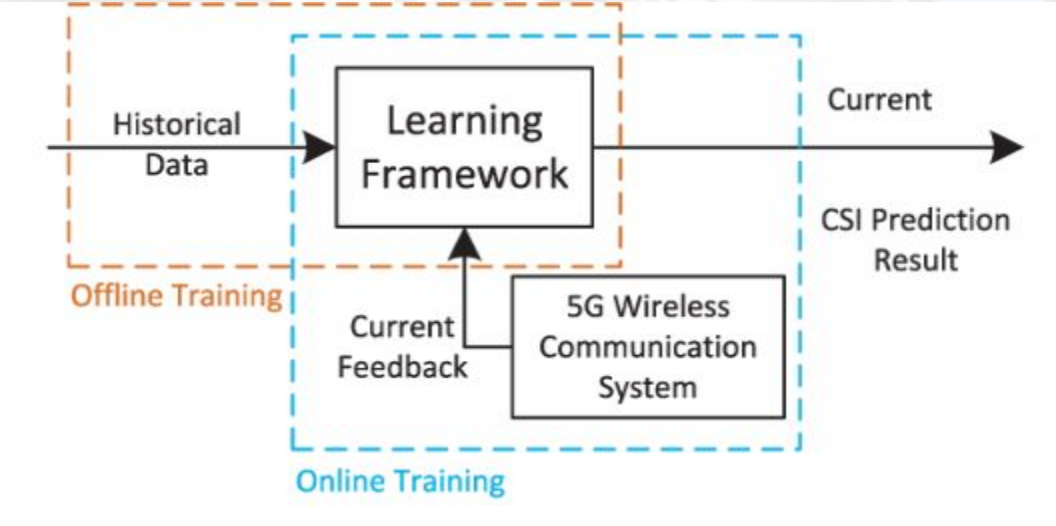
$$\tilde{a}_n(t) = a_n(t) \exp\left[-j2\pi f_c \tau_n(t)\right]$$

$$h(t, \tau) = \sum_n \tilde{a}_n(t)\, \delta(t - \tau_n(t))$$

Channel Model

```
def generate_random_taps(self, nsymbol):
    for i in range(self.no_taps):
        list_rx = [np.random.uniform(0,1) for i in range(self.no_taps-1)]
        list_rx=np.sort(list_rx)

        tou= np.hstack((0,list_rx))

        ray= np.hstack((1,np.random.rayleigh(0.5, self.no_taps-1)))
        self.channelResponse[nsymbol, i] = ray[i]*np.exp(-1j*2*self.center_freq*tou[i])
    self.H_exact[nsymbol, :] = np.fft.fft(self.channelResponse[nsymbol,:], self.K)
```

Using Machine Learning



Our problem statement : Due to increasing complexity of wireless communication scenarios, tracking precise channel state information (CSI) is crucial.

Conventional Channel Estimation

OFDM uses the reference signal based estimation courtesy of pilot symbols and performs channel estimation.

Primarily two conventional estimation methods:

- LS algorithm
- MMSE

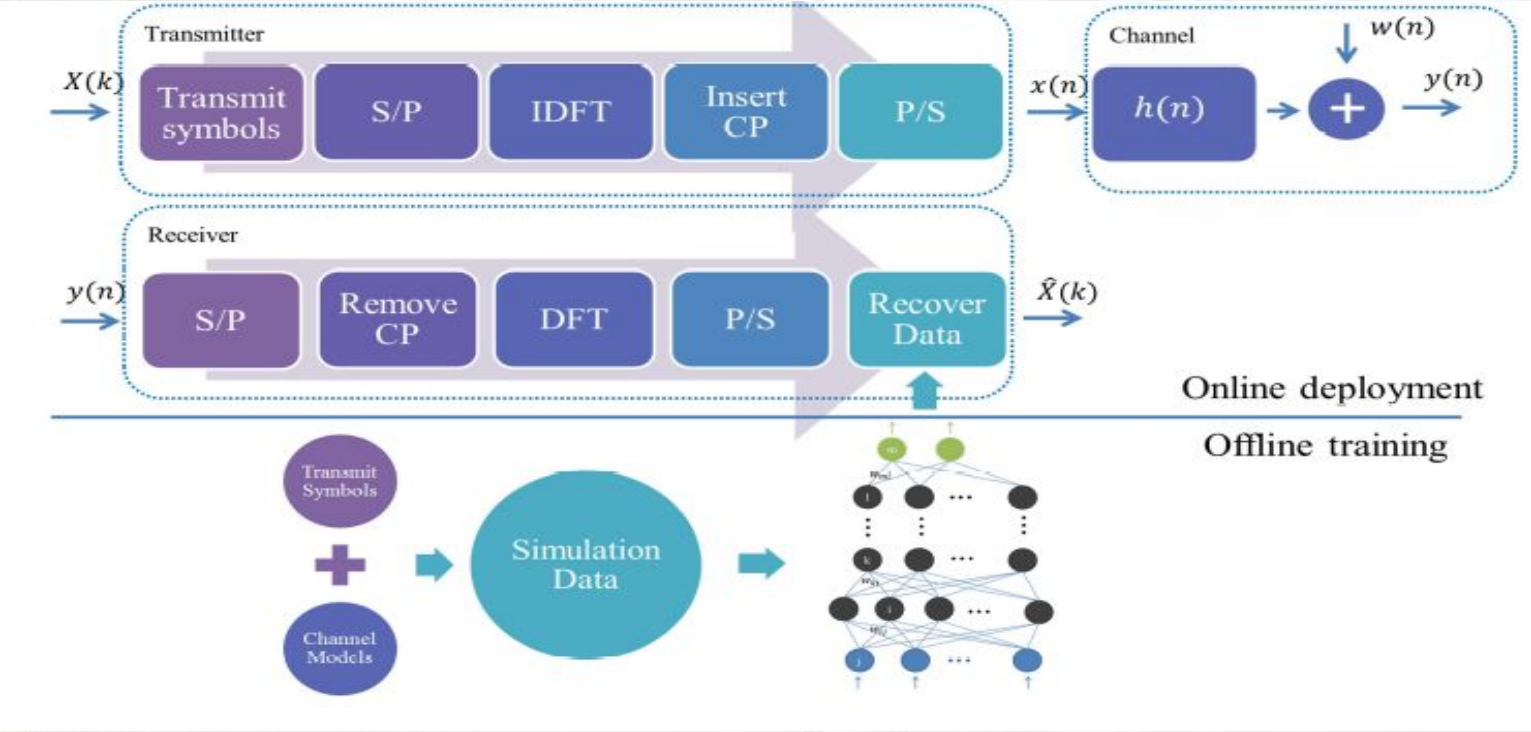
LS has simple structure and low complexity characteristics but its performance is not satisfactory. MMSE uses the prior knowledge to improve its performance but the complexity is too high.

LS and MMSE Estimation

```python
def channelEstimate(self, OFDM_demod):
    pilots = OFDM_demod[self.pilotCarriers]
    # extract the pilot values from the RX signal
    Hest_at_pilots = pilots / self.pilotValue
    Hest_abs = scipy.interpolate.interp1d(self.pilotCarriers,
    abs(Hest_at_pilots), kind='linear')(self.allCarriers)
    Hest_phase = scipy.interpolate.interp1d(self.pilotCarriers,
    np.angle(Hest_at_pilots), kind='linear')(self.allCarriers)
    Hest = Hest_abs * np.exp(1j*Hest_phase)

    return Hest
```

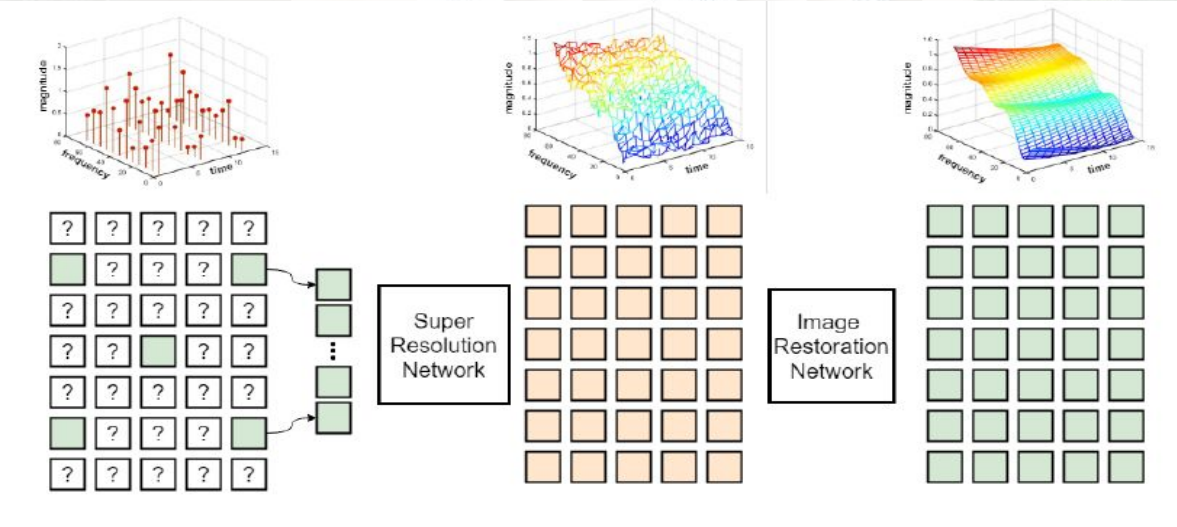
```python
def MMSE_channelEstimate(self, OFDM_demod, nsymbol):
    # import pdb;pdb.set_trace()
    C_response = np.array(self.H_exact[nsymbol,:]).reshape(-1,1)
    C_response_H = np.conj(C_response).T
    R_HH = np.matmul(C_response, C_response_H)
    snr = 10**(self.SNRdb/10)
    # W = R_HH/(R_HH+(self.beta/snr)*np.eye(self.K))
    W = np.matmul(R_HH,np.linalg.inv((R_HH+(self.beta/snr)*np.eye(self.K))))
    HhatLS = self.channelEstimate(OFDM_demod)
    HhatLS = HhatLS.reshape(-1,1)
    HhatLMMSE = np.matmul(W, HhatLS)

    return HhatLMMSE.squeeze()
```

Model 1


Transmitter
$X(k)$
Transmit symbols
S/P
IDFT
Insert CP
P/S
$x(n)$
Channel
$h(n)$
$w(n)$
$y(n)$
Receiver
$y(n)$
S/P
Remove CP
DFT
P/S
Recover Data
$\hat{X}(k)$
Online deployment
Offline training
Transmit Symbols
Channel Models
Simulation Data

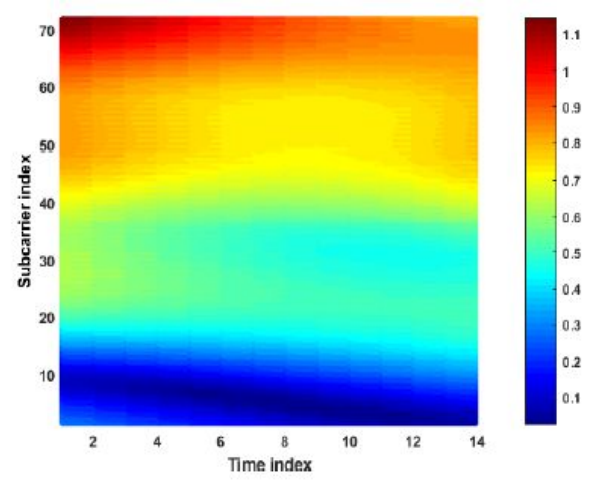
Model 2

This scheme considers the pilot values, altogether, as a low-resolution image and uses an SR network cascaded with a denoising IR network to estimate the channel.



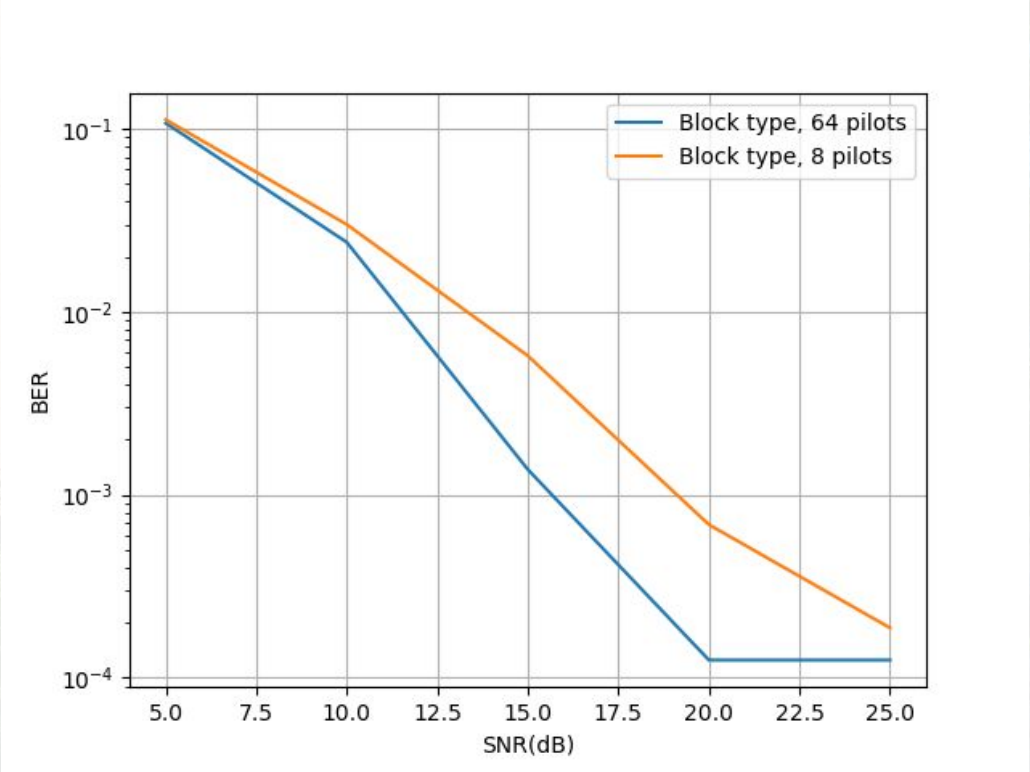
Model 2

For this link, the channel time-frequency response matrix H (of size N_S x N_D) between Tx and Rx which has complex values, are as two 2D-images (one for real values and another one for imaginary values).



Application : 2D-image for a sample channel time-frequency grid with N_D= 14 time slots and N_S = 72 subcarriers (based on LTE)

Results for DNN model 1



Model 2 Training

Our code for both the ML models

Deep Learning Model For Channel Estimation on PyTorch:
https://github.com/rohsequ/Deep-Learning-Model-for-Channel-Estimation-PyTorch

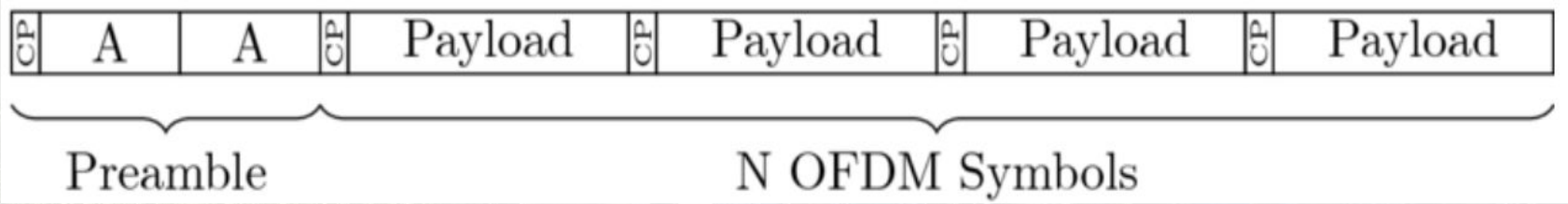
Deep Learning Model to estimate the output Symbols:
https://github.com/rohsequ/Deep-Learning-Model-For-Symbol-detection-from-received-symbols

General OFDM System Implementation on Python:
https://github.com/rohsequ/OFDM-System-Implementation-in-Python

Smith and Cox Synchronization

Schmidl and Cox synchronization algorithm uses two repeated training halves, referred to as S&C method.

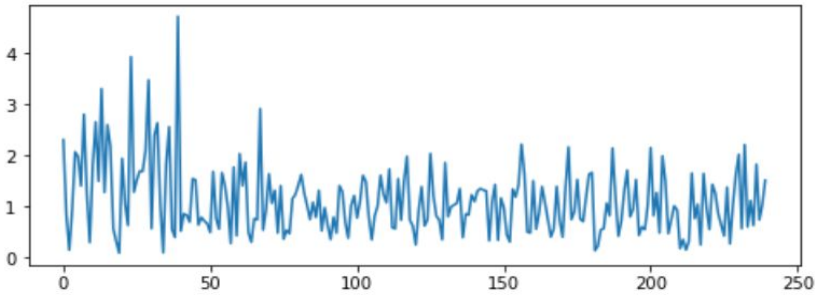
It estimates the start of each symbol.



Smith and Cox Synchronization

```
def addSTO(signal, sto):
    return np.hstack([np.zeros(sto), signal])

def addNoise(signal, sigma2):
    noise = np.sqrt(sigma2/2) * (np.random.randn(len(signal)) + 1j*np.random.randn(len(signal)))
    return signal + noise

def addChannel(signal, h):
    return scipy.signal.lfilter(h, (1,), signal)
```

```
x = createFrame(ofdm, qam_preamble=qam_preamble*2)
sto = ofdm.K//2
r = addNoise(addCFO(x, sto), 0.5)
plt.plot(abs(r));
```

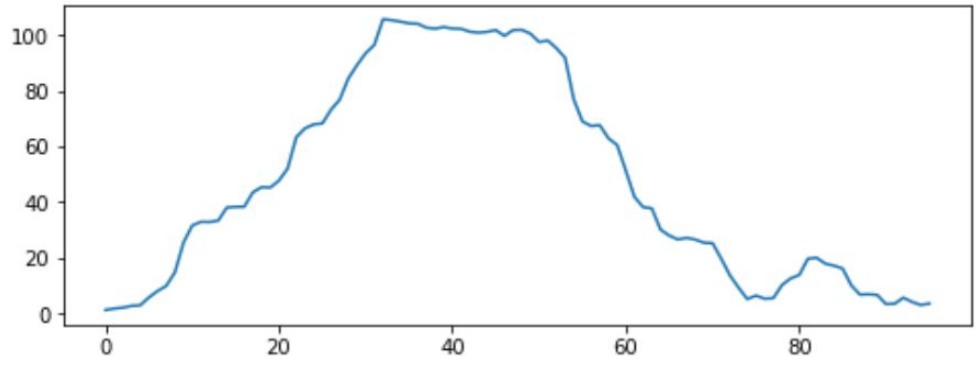


Smith and Cox Synchronization

```
1  # Method 1 to calculate P(d)
2  d_set = np.arange(0, sto + ofdm.K)
3  P1 = np.zeros(len(d_set), dtype=complex)
4  for i, d in enumerate(d_set):
5      P1[i] = sum(r[d+m].conj()*r[d+m+ofdm.L] for m in range(ofdm.L))
```

```
1  plt.plot(d_set, abs(P1))
```

```
[<matplotlib.lines.Line2D at 0x24586172e80>]
```



Conclusion and future work

In order to change the length of the cyclic-prefix adaptively,
we must estimate the T~ normalized RMS delay spread:

$$\hat{\tau}_{\mathrm{RMS}} = \sqrt{\frac{\sum_n \left|\hat{h}(n)\right|^2 n^2}{\sum_n \left|\hat{h}(n)\right|^2} - \left(\frac{\sum_n \left|\hat{h}(n)\right|^2 n}{\sum_n \left|\hat{h}(n)\right|^2}\right)^2}$$

We will change the
length of the CP based on the
estimation result.

References

[1] Soltani, Mehran, et al. "Deep learning-based channel estimation." *IEEE Communications Letters* 23.4 (2019): 652-655.

[2] H. Ye, G. Y. Li and B. -H. Juang, "Power of Deep Learning for Channel Estimation and Signal Detection in OFDM Systems," in IEEE Wireless Communications Letters, vol. 7, no. 1, pp. 114-117, Feb. 2018, doi: 10.1109/LWC.2017.2757490.

[3] P. Sowjanya, P. Satyanarayana, “Real-Time Data Transfer Based on Software Defined Radio Technique using Gnu radio/USRP”, doi: 10.35940/ijeat.A1140.109119

[4] D. Li, K. Deng, M. Zhao, S. Zhang, and J. Zhu. Knowledge-driven machine learning: Concept, model and case study on channel estimation, 12 2020.

[5] D. Wu and W. Fan. A new channel estimation method based on pilot-aided and local adaptive least squares support vector regression in software radio ofdm system. In 2009 International Joint Conference on Artificial Intelligence, pages 349–352, 2009

Thank you

Rishhabh, Rohan, Sudharsan, Vibhuti