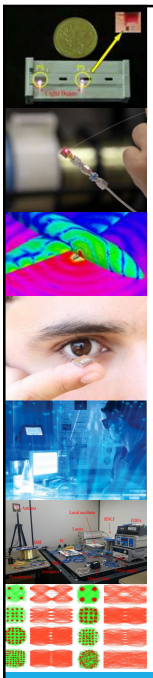


Inatel
Instituto Nacional de Telecomunicações

WOCA
Inatel Labs

Estimação da Rotação de Fase em Sinais QAM Utilizando ML

Msc. Eduardo Saia Lima
Eng. Gustavo Marengo

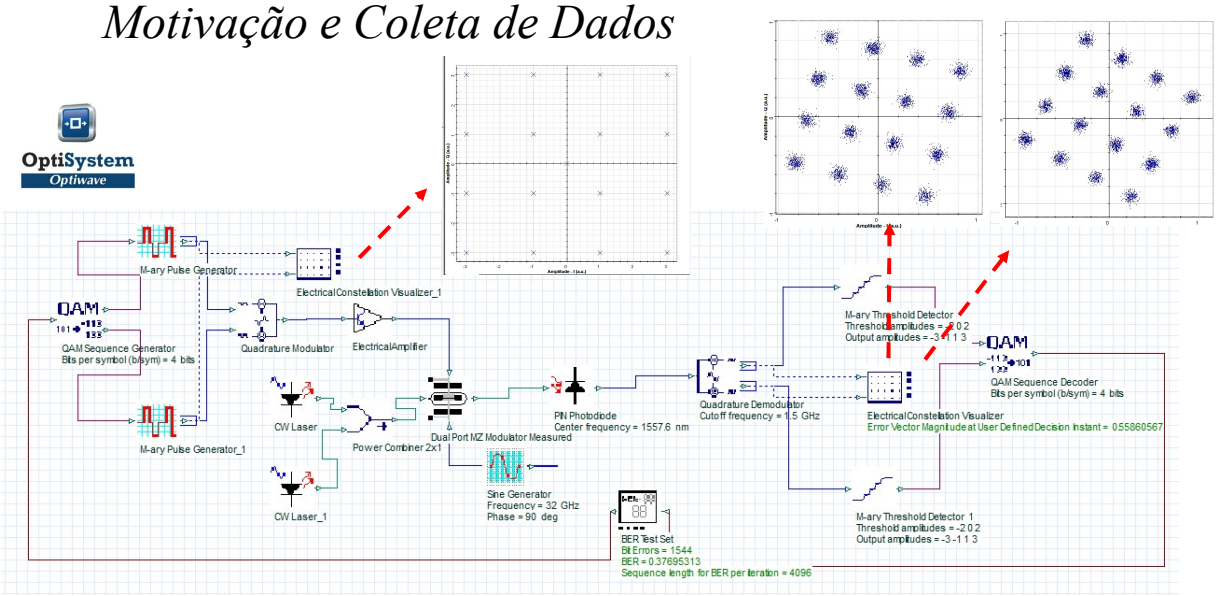
Prof. Dr. Felipe Augusto Pereira de Figueiredo

23 de Junho de 2020

AGENDA

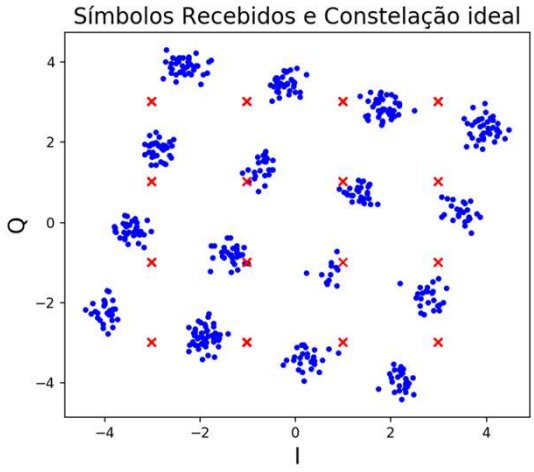
- Motivação e Coleta de Dados
- Estimação da Rotação de Fase Utilizando Regressão Linear
- Estimação da Rotação de Fase Utilizando Multi-layer Perceptron (MLPRegressor)
- Conclusões e Trabalhos Futuros

Motivação e Coleta de Dados

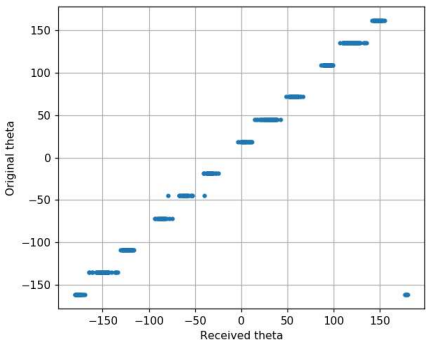


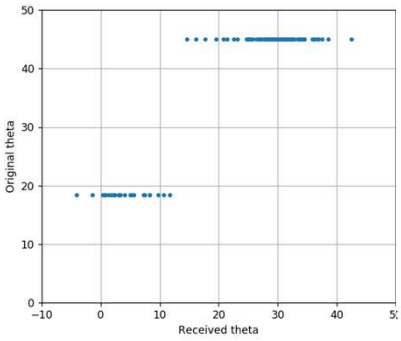
Estimação da Rotação de Fase Utilizando Regressão Linear

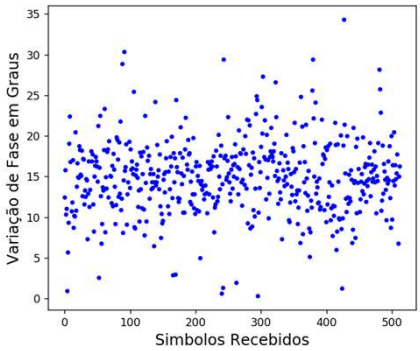
```
df = pd.read_csv('recp15.csv', header=None)
x1 = df[0].to_numpy()
y1 = 1j*df[1].to_numpy()

y = np.array(len(x1))
vetordefasesrx = list(range(len(x1)))
variacao = list(range(len(x1)))
phaserx = list(range(len(x1)))

for i in range(0,len(x1)):
    y = x1 +y1
    phaserx [i] = cmath.phase(y[i])

    vetordefasesrx [i] = numpy.degrees(phaserx[i])

    if (vetordefasestx[i] > vetordefasesrx[i]):
        variacao [i] = (vetordefasestx[i]-vetordefasesrx[i])
    else:
        variacao [i] = (vetordefasesrx[i] - vetordefasestx[i])
    if  (variacao[i] > 270):
            variacao [i] = 360 - variacao [i]

fig = plt.figure(figsize=(6,5))
plt.plot(y.real,y.imag, 'b.') #símbolos recebidos
plt.plot(x.real,x.imag, 'rx') #símbolos ideais
plt.xlabel('I',fontsize=16)
plt.ylabel('Q',fontsize=16)
plt.title('Símbolos Recebidos e Constelação ideal',fontsize=16)
plt.show()
```



Estimação da Rotação de Fase Utilizando Regressão Linear

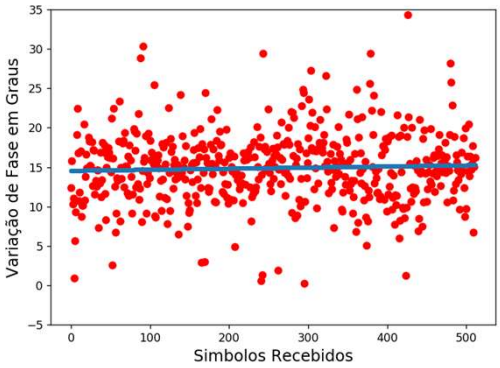


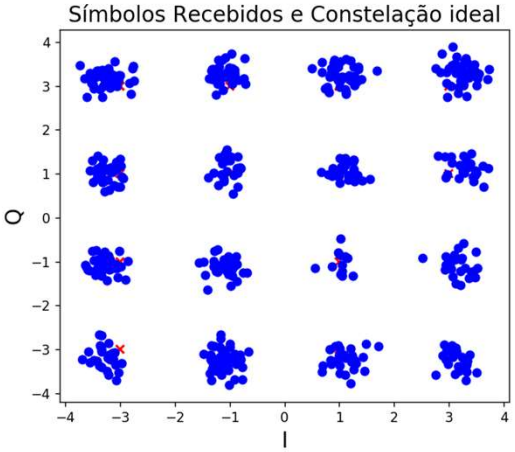




Estimação da Rotação de Fase Utilizando Regressão Linear

Gradiente descendente por batelada

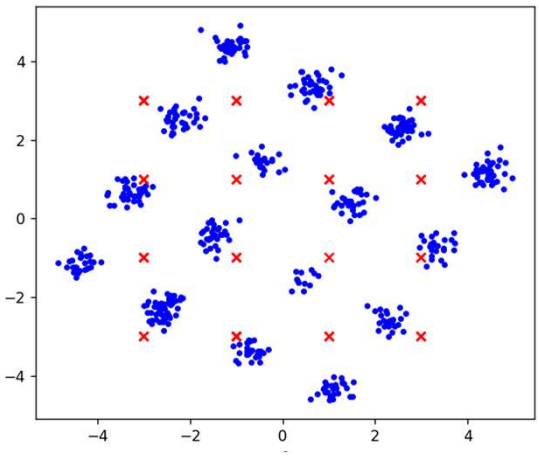


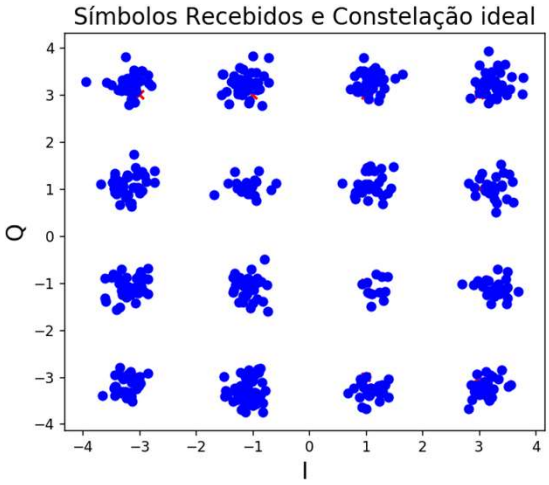


a = sum(yhat)/len(yhat) anguloderotacaoemrad= a*3.1415/180

[14.92161289] [0.26042359]

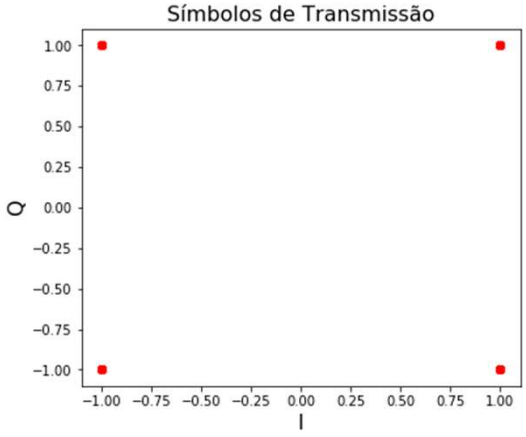
Estimação da Rotação de Fase Utilizando Regressão Linear



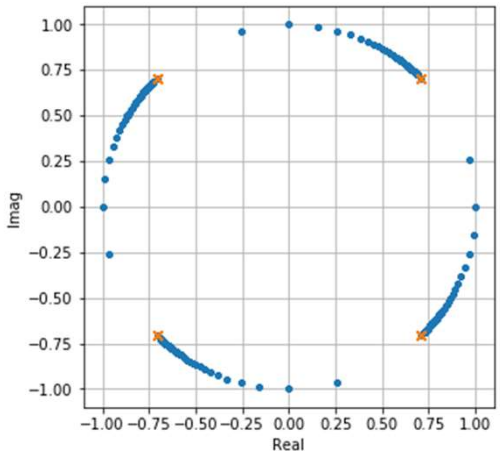


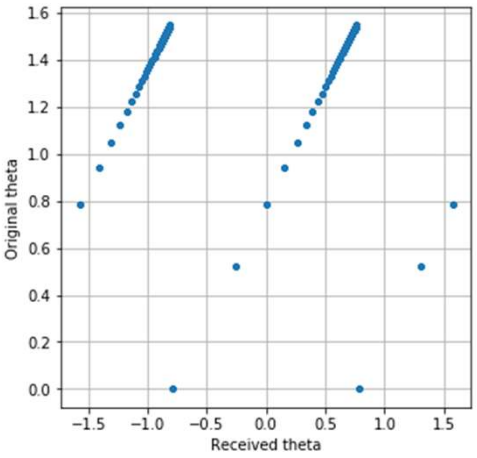
Estimação da Rotação de Fase Utilizando Multi-layer Perceptron (MLPRegressor)

```
def mod(V):
    ip = np.zeros((len(V),1),dtype=complex)
    inc = 0
    for v in V:
        if(v==0):
            ip[inc] = -1.0 - 1j*1.0
        elif(v==1):
            ip[inc] = -1.0 + 1j*1.0
        elif(v==2):
            ip[inc] = 1.0 - 1j*1.0
        else:
            ip[inc] = 1.0 + 1j*1.0
        inc += 1
    # Normalization of energy to 1.
    s = (1/np.sqrt(2))*ip;
    return s

def demod(c):
    c_seq = mod(np.array([0,1,2,3]))
    c_seq = c_seq[:,0]
    e = []
    for i in range(0,len(c_seq)):
        e.append(np.abs(c_seq[i] - c)**2)
    return e.index(np.min(e))
```



Estimação da Rotação de Fase Utilizando Multi-layer Perceptron (MLPRegressor)





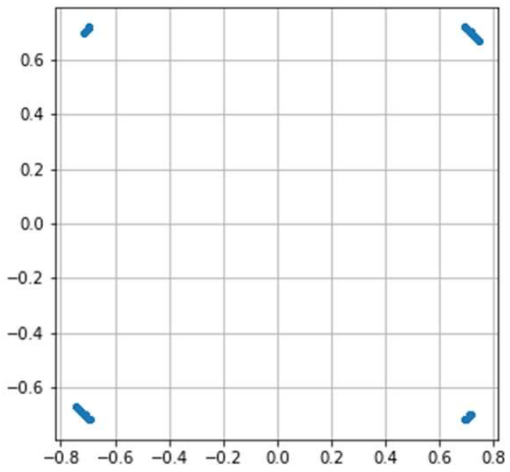
Estimação da Rotação de Fase Utilizando Multi-layer Perceptron (MLPRegressor)

```
reg.score(theta_test, theta_orig_test)
```

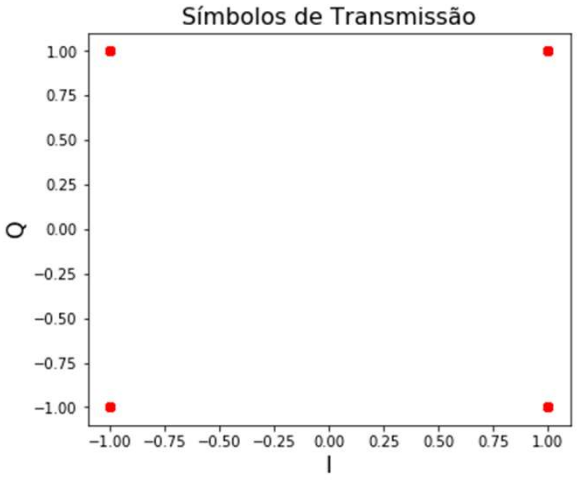
```
0.9986423491462224
```

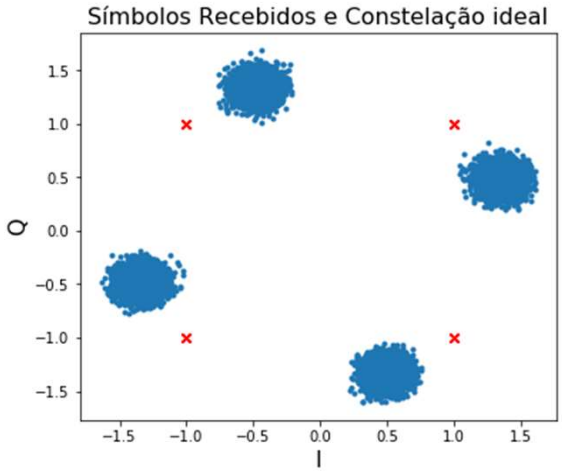
```
theta_pred = reg.predict(theta_test).reshape(len(theta_test), 1)

y_rec = np.exp(1j*theta_pred)*y_test
```

```
plt.figure(figsize=(5,5))
plt.plot(y_rec.real,y_rec.imag,'.')
plt.grid()
plt.show()
```

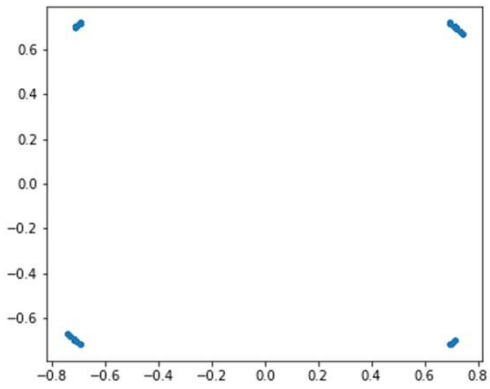


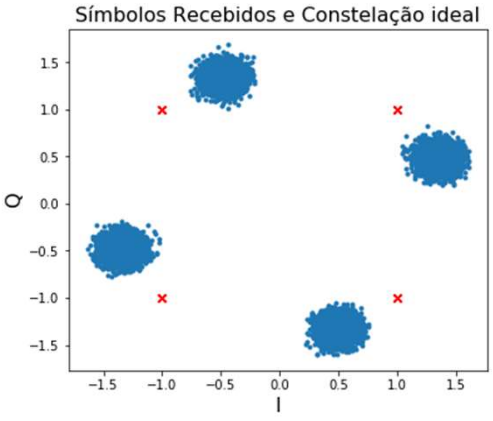
Estimação da Rotação de Fase Utilizando Regressão Linear





Estimação da Rotação de Fase Utilizando Multi-layer Perceptron (MLPRegressor)






```
theta_pred = reg.predict(theta).reshape(len(theta), 1)

y_rec = np.exp(1j*theta_pred)*y
```

```
fig = plt.figure(figsize=(6,5))
plt.plot(y_rec.real,y_rec.imag,'.')
plt.show()
```

Conclusões e Trabalhos Futuros

. *Vislumbra-se como trabalhos futuros a otimização dos códigos utilizados e aplicação em sistemas ópticos reais.*

. *Outra proposta seria adequar o código para realizar estimação do canal RoF e wireless sem a utilização de portadoras piloto, otimizando a taxa útil do sistema.*