Atividade 4

Leonardo Santos - GRR20196154

Primeiramente foi feito a normalização dos dados, utilizando esse trecho de código ilustrado pela Figura 1 a seguir:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import loadmat


# carregar arquivo .mat
mat = loadmat('in_out_SBRT2_direto.mat')

in_data_ext = mat['in_extraction']
out_data_ext = mat['out_extraction']
in_data_val = mat['in_validation']
out_data_val = mat['out_validation']

max_value = np.concatenate((max(abs(in_data_ext)), max(abs(out_data_ext)),max(abs(in_data_val)),max(abs(out_data_val)))).max()


in_ext = in_data_ext/max_value
out_ext = out_data_ext/max_value
in_val = in_data_val/max_value
out_val = out_data_val/max_value

```

Figura 1: Código de normalização dos dados

Em seguida foi utilizado o trecho de código ilustrado pela Figura 2, para avaliar se osvalores se encotravam entre os valores de -1 e 1. O resultado esta ilustrado pela Figura 3 a seguir:

```python
# plotar sinais de entrada e saída
x_ext = range(len(in_data_ext))
x_val = range(len(in_data_val))

fig, axs = plt.subplots(2, sharex=True, figsize=(8, 6))
fig.suptitle('Sinais de entrada e saída')
axs[0].plot(x_ext, np.real(in_ext), label='Entrada')
axs[0].plot(x_ext, np.real(out_ext), label='Saída')
axs[0].legend()
axs[1].plot(x_val, np.real(in_val), label='Entrada')
axs[1].plot(x_val, np.real(out_val), label='Saída')
axs[1].legend()
plt.show()
```

Figura 2: Código pra plotar gráficos de entrada e saída

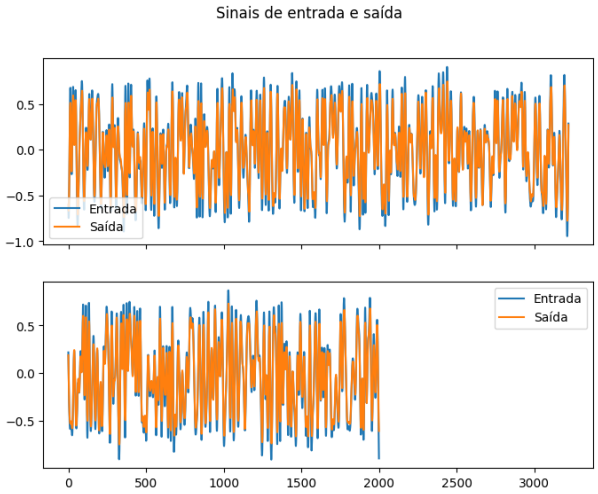

Figura 3: Gráficos entrada e saída

Com os dados normalizados foi feito a matriz de coeficientes, utilizando o seguinte trecho de código ilustrado pela Figura 4 a seguir:

```python
def MatrixCoeficientes(in_data, M, P):
    n = len(in_data)
    XX = np.zeros((n - M, 2*P), dtype=np.complex128)
    for i in range(M, n):
        XX[i-M, 0] = in_data[i]
        for j in range(1, P):
            XX[i-M, 2*j-1] = in_data[i-j].real ** j
            XX[i-M, 2*j] = in_data[i-j].imag ** j
    return  XX

P = 5
M = 1

XX_ext = MatrixCoeficientes(in_ext, M, P)
XX_val = MatrixCoeficientes(in_val, M, P)

coefficients, _, _, _ = np.linalg.lstsq(XX_ext, out_ext[M:], rcond=None)
```

Figura 4: Calculo das matrizes de Coeficientes

Em seguida foi realizados os passos do item 3 porem os valores de maximo me mino deram menores que 1. O codigo utilizado esta ilustrado pela Figura 5 a seguir:

```python
bits = 4
multiplicador = 2**bits
precisão = 1/multiplicador
in_val_fix = np.round(in_val * multiplicador)
out_val_fix = np.round(out_val * multiplicador)
in_val_fix_imag = np.floor(in_val_fix.imag/multiplicador)
in_val_fix_real = np.floor(in_val_fix.real/multiplicador)
out_val_fix_imag = np.floor(out_val_fix.imag/multiplicador)
out_val_fix_real = np.floor(out_val_fix.real/multiplicador)


in_val_real_float = in_val_fix_real / multiplicador
in_val_imag_float = in_val_fix_imag / multiplicador


out_val_real_float = out_val_fix_real / multiplicador
out_val_imag_float = out_val_fix_imag / multiplicador

out_val = out_val_real_float + 1j*out_val_imag_float

in_val = in_val_real_float + 1j*in_val_imag_float
```

Figura 5: Calculo da precisão em bits