Atividade 4

Leonardo Santos - GRR20196154

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

```
import many as no
import many
```

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:

```
# plotar sinais de entrada e saída
2    x_ext = range(len(in_data_ext))
3    x_val = range(len(in_data_val))
4
5    fig, axs = plt.subplots(2, sharex=True, figsize=(8, 6))
6    fig.suptitle('Sinais de entrada e saída')
7    axs[0].plot(x_ext, np.real(in_ext), label='Entrada')
8    axs[0].plot(x_ext, np.real(out_ext), label='Saída')
9    axs[0].legend()
10    axs[1].plot(x_val, np.real(in_val), label='Entrada')
11    axs[1].plot(x_val, np.real(out_val), label='Saída')
12    axs[1].legend()
13    plt.show()
```

Figura 2: Código pra plotar gráficos de entrada e saída

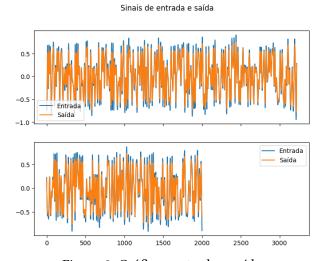


Figura 3: Gráficos entrada e saída

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

```
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*N, 2*j*] = in_data[i]
    for j in range(1, P):
    XX[i*N, 2*j*] = in_data[i*j].real ** j
    XX[i*N, 2*j] = in_data[i*j].imag ** j
    return XX

10
11    P = 5
12    M = 1
13
14    XX_ext = MatrixCoeficientes(in_ext, M, P)
15    XX_val = MatrixCoeficientes(in_val, M, P)
16
17    coefficients, _, _, _ = np.linalg.lstsq(XX_ext, out_ext[M:], rcond=None)
```

Figura 4: Calculo das matrizes de Coeficientes

Em seguida foram feitos os calulos utilizando os valores do extraidos e normalizados que correspoderam a -21dB.

Em seguida foi realizados os passos do item 3, na etapa do item a) observou-se que o maior valor, depois da conversão para um numero inteiro foi de 15, enquanto para a proxima etapa do item b) na etapa de readequação o maior valor foi arendondado para 0. A figura Figura 5 ilustra a o codigo utilizado:

```
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_mag = np.floor(in_val_fix.imag/multiplicador)
in_val_fix_mag = np.floor(out_val_fix.eal/multiplicador)
out_val_fix_imag = np.floor(out_val_fix.mag/multiplicador)
out_val_fix_mag = np.floor(out_val_fix.real/multiplicador)
out_val_fix_real = np.floor(out_val_fix.real/multiplicador)
in_val_imag_float = in_val_fix_real / multiplicador
in_val_imag_float = in_val_fix_imag / multiplicador
out_val_mag_float = out_val_fix_imag / multiplicador
out_val_imag_float = out_val_fix_imag / multiplicador
out_val_imag_float = out_val_fix_imag / multiplicador
out_val_imag_float = out_val_fix_imag / multiplicador
in_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

Estou com dificuldade nessa etapa!