Modelo matemático de um polinômio com memória

André Corso Pozzan
Atividade **3** - Iniciação Científica **GICS**

Orientador: Prof. Ph.D. Eduardo Gonçalves de Lima

23 de março de 2025

1 Atividade

Modificar o código da Atividade 2, de tal forma a obter o modelo conhecido como MP, descrito pela equação:

$$\tilde{y}(n) = \sum_{p=1}^{P} \sum_{m=0}^{M} \tilde{b}_{2p-1,m} \left|\tilde{x}(n-m)\right|^{2p-2} \tilde{x}(n-m) \tag{1}$$

Observando que agora os sinais são complexos.

Para esse modelo MP, use o conjunto de dados de entrada e saída da seguinte forma: utilize os sinais `"in_extraction"` e `"out_extraction"` para obter os coeficientes. Utilize os sinais `"in_validation"` e `"out_validation"` para obter a saída estimada do modelo.

Uma vez tendo a saída estimada, calcular a métrica NMSE descrita pela equação:

$$NMSE = 10 \log_{10} \left[\frac{\sum_{n=1}^{N} |e(n)|^2}{\sum_{n=1}^{N} |y_{\text{ref}}(n)|^2} \right] \tag{2}$$

Além disso, obter os gráficos: amplitude de saída em função da amplitude de entrada (AM-AM) e diferença entre fase de saída e de entrada em função da amplitude de entrada (AM-PM), para dados medidos e estimados.

Dica: o comando ()' calcula o transposto conjugado. O comando ().' calcula apenas o transposto. Normalmente você quer apenas o transposto, ou seja, você quer apenas adequar sua matriz para a dimensão correta.

2 Resolução

Inicialmente, realizei algumas anotações para observar a diferença entre o modelo da atividade passada e o novo. Assim, foi possível uma melhor interpretação da modificação do código em Python.

2.1 Cálculos no papel

$$\tilde{y}(n) = \sum_{p=1}^{P} \sum_{m=0}^{M} \tilde{h}_{2p-1,m} \cdot |\tilde{x}(n-m)|^{2p-2} \cdot \tilde{x}(n-m)$$

$\tilde{x} \in$ entrada dados
$\tilde{y} \in$ saída "
$\tilde{h} \in$ coeficientes

Se $P=1$ e $M=1$

$$\tilde{h}_{2-1,0} \cdot |\tilde{x}(n)|^{0} \cdot \tilde{x}(n) \qquad \begin{bmatrix} x(n) & x(n-1) \end{bmatrix} \cdot \begin{bmatrix} h_{10} & h_{11} \end{bmatrix}^T = out(n)$$

$$\tilde{h}_{1,0} \cdot \tilde{x}(n) + \tilde{h}_{1,1} \cdot \tilde{x}(n-1)$$

Se $P=2$ e $M=1$

$$\tilde{h}_{3,0} \cdot |\tilde{x}(n)|^2 \cdot \tilde{x}(n) + \tilde{h}_{3,1} \cdot |\tilde{x}(n-1)|^2 \cdot \tilde{x}(n-1)$$

$$out(1) = \tilde{h}_{1,0} \cdot \tilde{x}(1) + \tilde{h}_{1,1} \cdot \tilde{x}(0) + \tilde{h}_{3,0} |\tilde{x}(1)|^2 \cdot \tilde{x}(1) + \tilde{h}_{3,1} \cdot |\tilde{x}(0)|^2 \cdot \tilde{x}(0)$$

$$\begin{bmatrix} \tilde{x}(1) & \tilde{x}(0) \\ |\tilde{x}(1)|^2 \cdot \tilde{x}(1) & |\tilde{x}(0)|^2 \cdot \tilde{x}(0) \end{bmatrix} \cdot \begin{bmatrix} \tilde{h}_{1,0} & \tilde{h}_{1,1} \\ \tilde{h}_{3,0} & \tilde{h}_{3,1} \end{bmatrix}^T = \begin{bmatrix} out(1) \end{bmatrix}$$

2.2 Código em Python

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

data = loadmat('in_out_SBRT2_direto.mat')

in_extraction = data['in_extraction']
out_extraction = data['out_extraction']

in_validation = data['in_validation']
out_validation = data['out_validation']

P_VOLTERRA = 10
M_VOLTERRA = 2

in_extraction = np.array(in_extraction)

def criaMatrizDeRegressao(in_data, P_VOLTERRA, M_VOLTERRA):
    matrizDeRegressao = []
```

```
    for n in range(len(in_data)):
        linhaDaMatrizDeRegressao = []

        for p in range(1, P_VOLTERRA + 1):
            for m in range(M_VOLTERRA + 1):
                dataIndex = max(0, n - m)
                power = 2*p-2

                numeroCalculado = (abs(in_data[dataIndex]**power)) * in_data[dataIndex]
                numeroCalculado = numeroCalculado.item()

                linhaDaMatrizDeRegressao.append(numeroCalculado)
        matrizDeRegressao.append(linhaDaMatrizDeRegressao)

    return np.array(matrizDeRegressao);



matrizDeRegressaoExtraction = criaMatrizDeRegressao(in_extraction, P_VOLTERRA, M_VOLTERRA)
resultado = np.linalg.lstsq(matrizDeRegressaoExtraction, out_extraction, rcond=None)
coeficientes = resultado[0]
print("Coeficientes:",coeficientes)



matrizDeRegressaoValidation = criaMatrizDeRegressao(in_validation, P_VOLTERRA, M_VOLTERRA)
saidaEstimadaPolinomio = np.dot(matrizDeRegressaoValidation, coeficientes)


nmse = 10*np.log10(np.sum((out_validation - saidaEstimadaPolinomio)**2)/np.sum(out_validation**2))
print(f"NMSE: {nmse} dB")

# Primeiro gráfico
plt.plot(in_validation, label='Entrada (in_validation)')
plt.plot(out_validation, label='Saída (out_validation)')
plt.plot(saidaEstimadaPolinomio, label='Estimado (saidaEstimadaPolinomio)', linestyle='--')
plt.xlabel('Amostras')
plt.ylabel('Amplitude')
plt.title(f"Comparação dos Dados de Entrada, Saída e Estimado para P = {P_VOLTERRA} M = {M_VOLTERRA}")
plt.legend()
plt.grid(True)


plt.tight_layout()
plt.show()

plt.scatter(np.abs(in_validation), np.abs(out_validation), label="Dados Medidos")
plt.scatter(np.abs(in_validation), np.abs(saidaEstimadaPolinomio), label="Estimado", alpha=0.7)
plt.xlabel("Amplitude de Entrada")
plt.ylabel("Amplitude de Saída")
plt.title("Gráfico AM-AM")
plt.legend()
plt.grid()
plt.show()


#Variação da fase de entrada para a de saída
fase_out = np.angle(out_validation)
fase_in = np.angle(in_validation)
delta_fase = np.degrees(fase_out - fase_in)  # Converte para graus

fase_out_estimada = np.angle(saidaEstimadaPolinomio)
delta_fase_estimada = np.degrees(fase_out_estimada - fase_in)  # Converte para graus

plt.scatter(np.abs(in_validation), delta_fase, label="Diferença de Fase (AM-PM)")
plt.scatter(np.abs(in_validation), delta_fase_estimada, label="Estimado", alpha=0.7)
```

```
plt.xlabel("Amplitude de Entrada")
plt.ylabel("Diferença de Fase (graus)")
plt.title("Gráfico AM-PM")
plt.legend()
plt.grid()
plt.show()
```

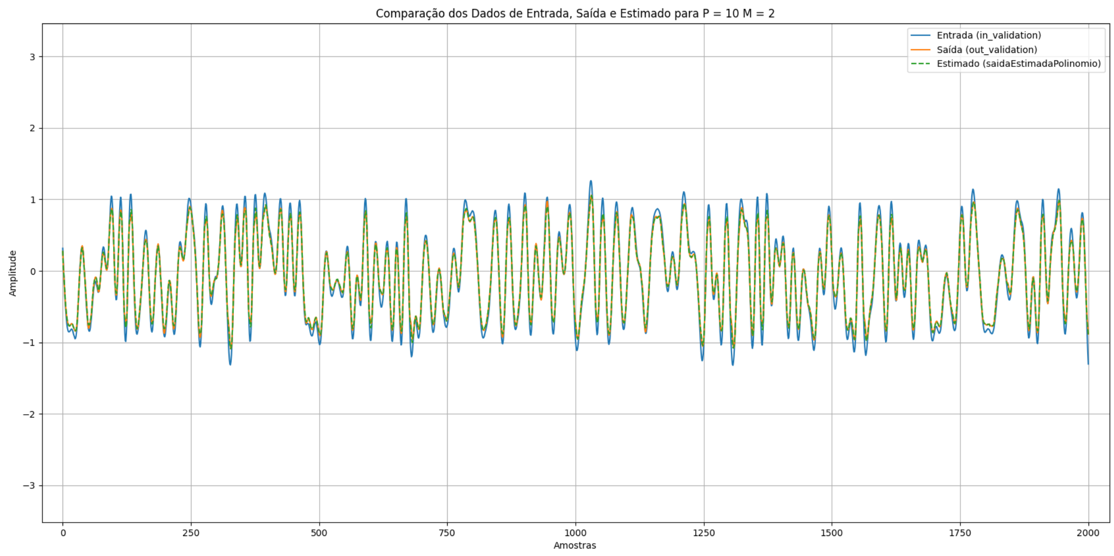

Figura 1: Gráfico da comparação in validation, out validation e a saída do polinômio

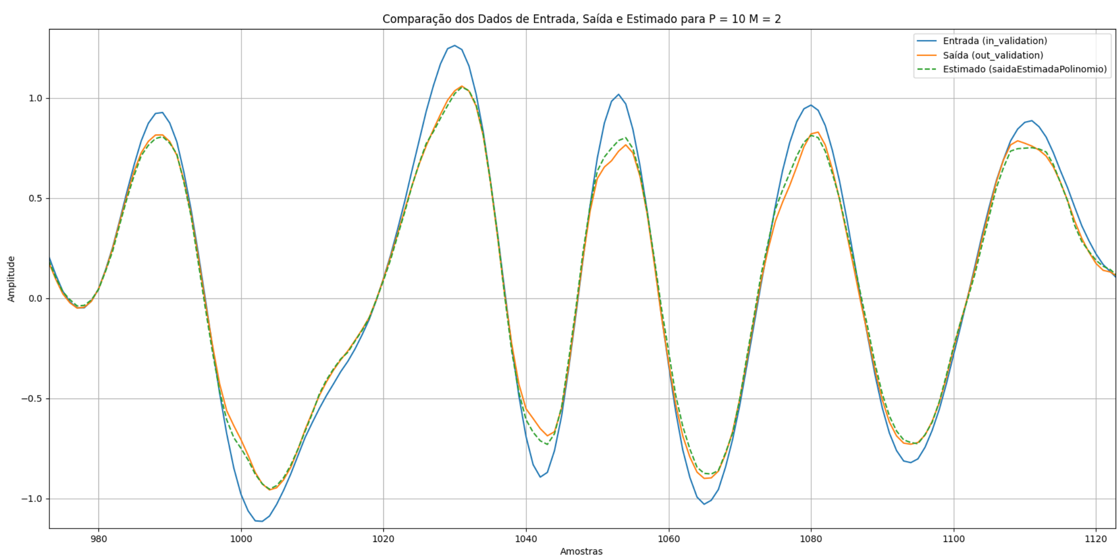

Figura 2: Mesmo gráfico anterior porém com ampliação na região central

2.2.1 Calculo da métrica NMSE

Para calcular a métrica NMSE(Normalized Mean Square Error) demostrada na equação 2, é necessário calcular a soma dos quadrados dos erros entre os valores medidos e os valores estimados pelo modelo e dividir pela soma dos quadrados dos valores medidos e depois aplicar ao $10\log_{10}$

$$NMSE = 10\log_{10}\left(\frac{\sum(\text{quadrados dos erros})}{\sum(\text{quadrados dos valores reais})}\right) \tag{3}$$

O NMSE é frequentemente expresso em decibéis (dB). Um valor de NMSE menor (mais negativo em dB) indica uma melhor precisão do modelo, ou seja, uma menor diferença entre os sinais medidos e estimados.

Saída no console para P = 10 e M = 2:

```
"NMSE: (-25.89104963164262+0.6260551485472379j) dB"
```

Saída no console para P = 5 e M = 1:

```
"NMSE: (-23.435754390886782+4.315957714595194j) dB"
```

Ao diminuir os valores de P e M que configuram o polinômio observamos um aumento no valor do NMSE o que significa uma menor precisão em relação ao amplificador.

2.2.2 Gráfico AM-AM

Este gráfico mostra a relação entre a amplitude instantânea do sinal de saída de um amplificador e a amplitude instantânea do sinal de entrada, demonstrando os pontos medidos do amplificador.

Para casos como esse é comum a utilização do gráfico de dispersão em que é possível exibir as leituras feitas no amplificador e seus correspondentes do polinômio desenvolvido.

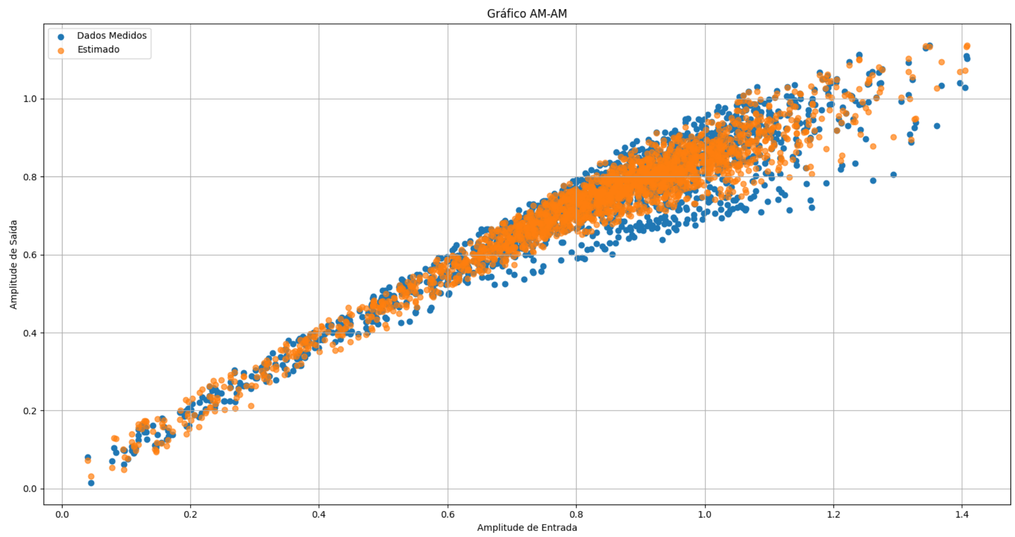

Figura 3: Gráfico AM-AM

Os pontos em azul representam o comportamento real do amplificador, indicando como a amplitude do sinal de saída varia em resposta às mudanças na amplitude do sinal de entrada. Em um amplificador idealmente linear, essa relação seria diretamente proporcional.

Do mesmo modo, os pontos em laranja representam o resultado gerado pelo polinômio aplicado nas mesmas entradas que o amplificador, assim, é possível verificar se o modelo matemático está representado fielmente e analisar as regiões de cobertura.

2.2.3 Gráfico AM-PM

Neste gráfico, é possível observar a diferença de fase entre o sinal de saída e o sinal de entrada em função da amplitude instantânea do sinal de entrada e é muito utilizado na análise. Da mesma forma que o gráfico AM-AM os pontos azuis representam os dados obtidos com `"in_validation"` e `"out_validation"`, e os pontos laranjas a saída calculada com o polinômio:

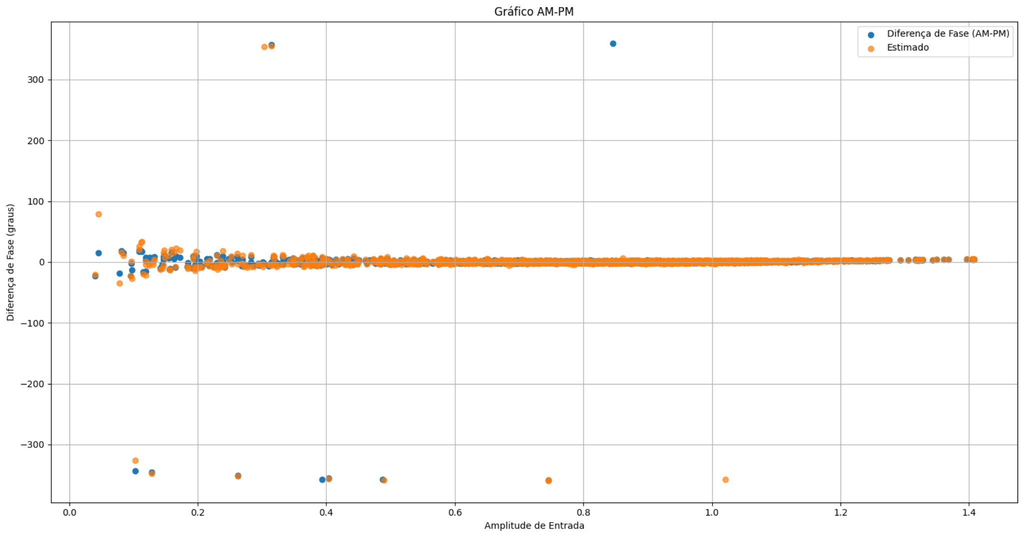

Figura 4: Gráfico AM-PM

Este tipo de análise é fundamental para caracterizar o comportamento não linear de um amplificador de potência (PA), pois um PA ideal deveria apenas amplificar o sinal sem alterar sua fase em função da amplitude.

Também, a utilização do módulo do número complexo do sinal de entrada é essencial para representar corretamente sua amplitude instantânea no eixo horizontal do gráfico.

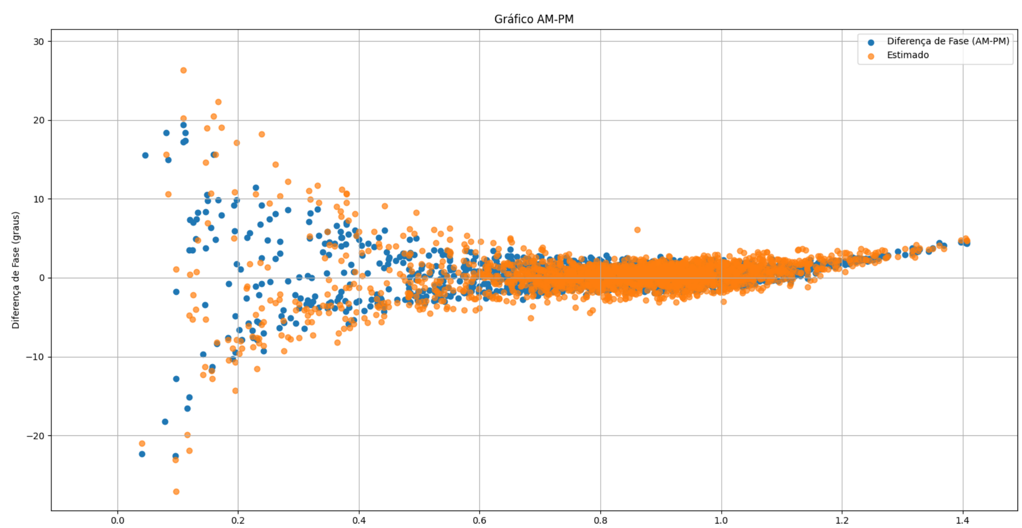

Figura 5: Gráfico AM-PM com zoom na região central

A análise da concordância entre os pontos azuis (dados medidos) e os pontos laranjas (saída modelada) permite avaliar a precisão do modelo em capturar a característica de conversão AM-PM do amplificador. Uma boa sobreposição indica que o modelo representa fielmente como a fase do sinal de saída varia com a amplitude do sinal de entrada.

Referências

LIMA, Eduardo Gonçalves de. **Behavioral modeling and digital base-band predistortion of RF power amplifiers**. Jan. 2009. Tese (Doutorado) – POLITECNICO DI TORINO.