Relatório Atividade M4

Leonardo Santos

Objetivo

O presente trabalho teve como objetivo a reorganização do código do modelo Memory Polynomial (MP), de modo a refletir explicitamente a estrutura apresentada no diagrama de blocos do modelo, conforme a Figura 1, a ser adicionada a este relatório.

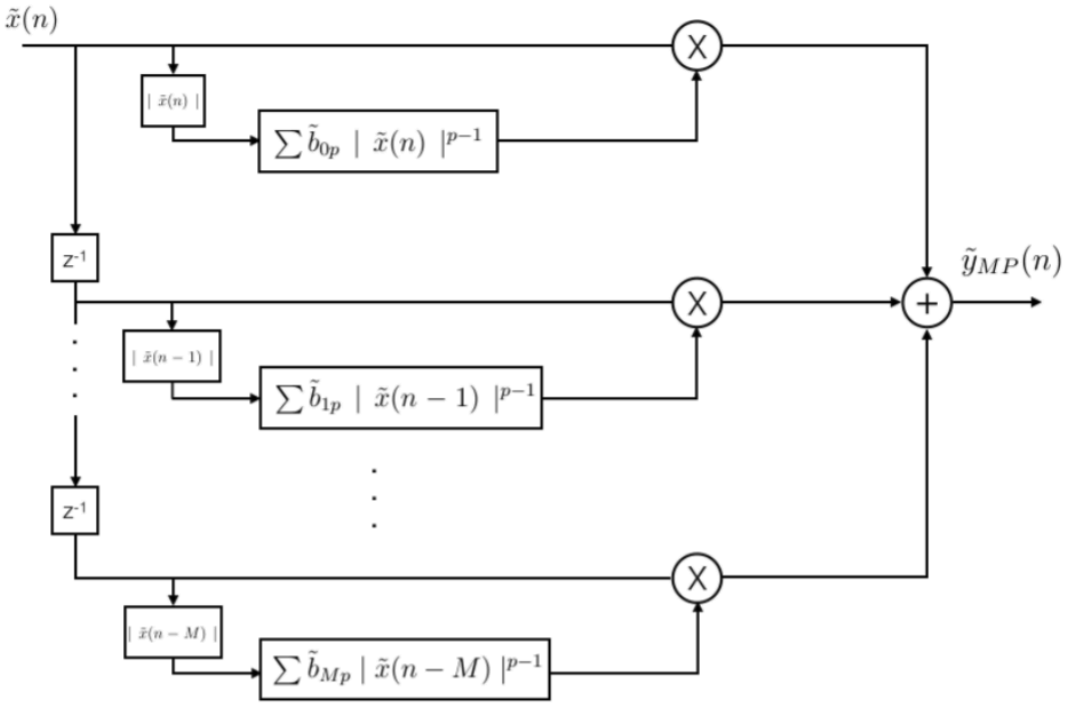

Figura 1: Diagrama de blocos do modelo Memory Polynomial (MP).

A principal modificação consistiu em tornar explícito o laço principal associado à memória do modelo, bem como isolar o somatório dos termos não lineares em funções independentes. Essa abordagem estabelece uma correspondência direta entre a implementação computacional e os ramos do diagrama de blocos do modelo MP, atendendo ao objetivo de modularização proposto.

Metodologia

O modelo Memory Polynomial foi reorganizado de forma modular, de modo que cada termo de memória corresponde a um bloco independente responsável pelo processamento da envoltória complexa do sinal de entrada atrasada. A saída total do modelo é obtida pela soma das contribuições individuais de todos os blocos de memória, em conformidade com a estrutura apresentada no diagrama de blocos.

Os coeficientes complexos do modelo foram estimados por meio do método dos Mínimos Quadrados, utilizando a função `least_squares` da biblioteca SciPy. Para esse procedimento, os coeficientes complexos foram representados por um vetor real, separando-se as partes real e imaginária.

O processo de identificação foi realizado considerando uma ordem polinomial igual a 3 e uma profundidade de memória igual a 2, totalizando $P * (M + 1)$ coeficientes complexos.

```
from scipy.optimize import least_squares

ordem = 3
memoria = 2
num_coef = ordem * (memoria + 1)

x0 = np.zeros(2 * num_coef)

res = least_squares(
    erro_mp_complex_blocos,
    x0,
    args=(in_data_ext, out_data_ext, ordem, memoria),
    verbose=2
)

coef_otimo = res.x[:num_coef] + 1j * res.x[num_coef:]
```

Modularização do Modelo Memory Polynomial

Na implementação original do modelo MP, os efeitos da memória e da não linearidade eram computados de forma conjunta por meio de laços aninhados, fazendo com que cada termo contribuísse diretamente para a saída final em um único bloco de cálculo. Essa abordagem dificultava a associação direta entre o código e os ramos do diagrama de blocos do modelo.

Com o objetivo de resolver a Issue 1 — **Modularização do Modelo Memory Polynomial** — o código foi reorganizado de forma que cada ramo de atraso e processamento do diagrama de blocos passou a ser representado por uma função independente. Cada função calcula a contribuição polinomial individual de um termo de memória m, antes da soma final das contribuições.

O bloco responsável pelo processamento de um único termo de memória é implementado pela função `bloco_memoria_mp`:

```
def bloco_memoria_mp(x_delay, coef_m, ordem):
    y_m = np.zeros_like(x_delay, dtype=complex)

    for p in range(1, ordem + 1):
        y_m += coef_m[p-1] * x_delay * np.abs(x_delay)**(2*(p-1))

    return y_m
```

A nova forma de cálculo da saída do modelo consiste na execução paralela dessas funções para cada termo de memória, seguida da soma das contribuições individuais:

```
def mp_model(x_in, coef, ordem, memoria):
    y_est = np.zeros(len(x_in), dtype=complex)
    idx = 0

    for m in range(memoria + 1):
        x_delay = np.roll(x_in, m)
        coef_m = coef[idx:idx + ordem]
        idx += ordem

        y_est += bloco_memoria_mp(x_delay, coef_m, ordem)

    return y_est
```

Essa reorganização torna explícito o laço principal associado à memória do modelo e estabelece uma correspondência direta entre cada função implementada e os ramos do diagrama de blocos do Memory Polynomial.

Avaliação de Desempenho

O desempenho do modelo identificado foi avaliado por meio da métrica Normalized Mean Squared Error (NMSE), calculada entre o sinal de saída estimado pelo modelo e o sinal de saída real no conjunto de validação.

```
nmse_calculo = lambda predicted_val, data_out: (
    10 * np.log10(
        np.mean(np.abs(data_out - predicted_val) ** 2) /
        np.mean(np.abs(data_out) ** 2)
    )
)
```

O valor obtido para o NMSE foi:

NMSE = −26.583997587165697 dB

Resultados Gráficos

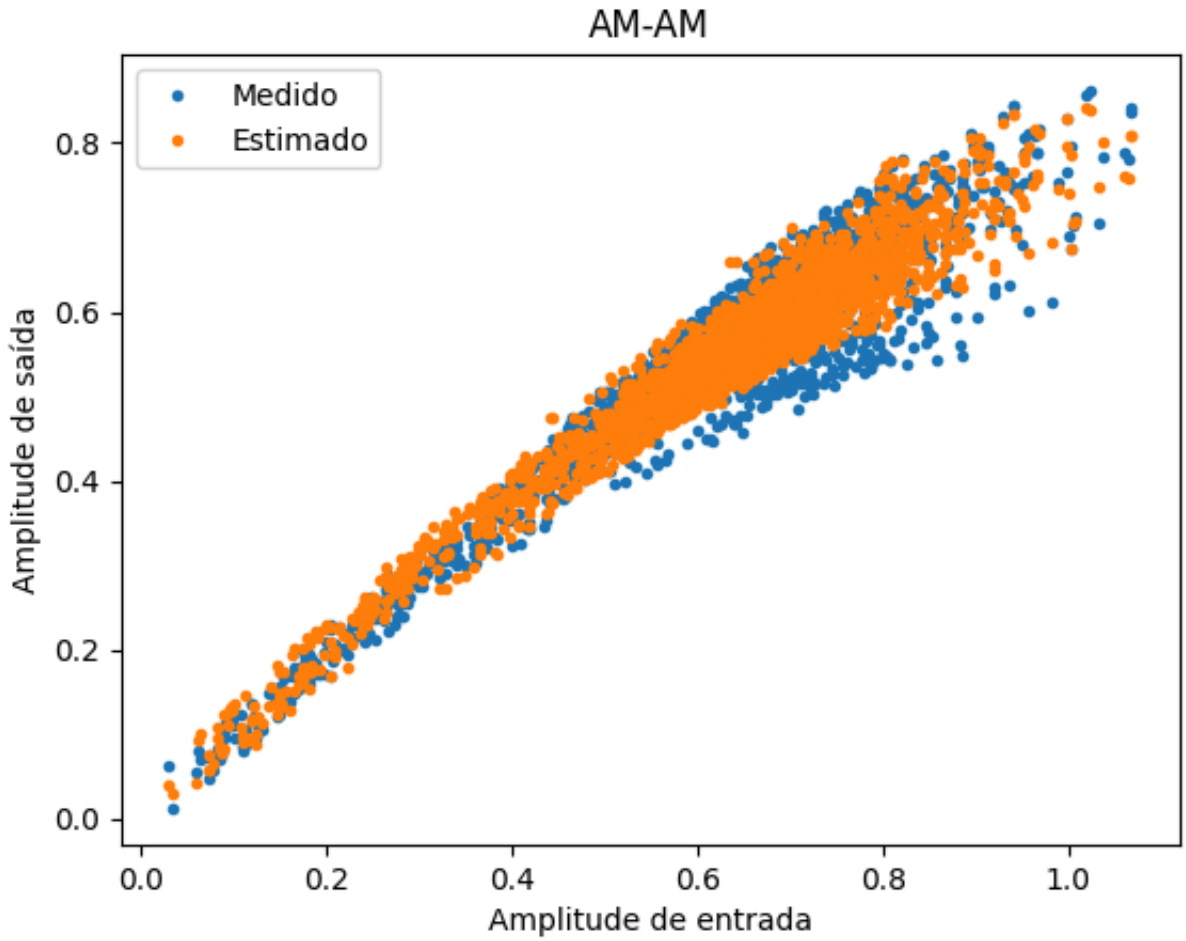

Figura 2: Sinal de saída real e sinal de saída estimado pelo modelo Memory Polynomial.

Próxima Etapa

Próxima Atividade

A próxima etapa do trabalho consiste na implementação da **Geração e Extração de Tabelas de Busca (Lookup Tables – LUTs) Unidimensionais**, conforme descrito na Issue 2 do projeto.

O objetivo dessa atividade é substituir a avaliação direta de polinômios de alta ordem do modelo Memory Polynomial por tabelas de consulta, reduzindo a complexidade computacional do processamento. Essa abordagem segue a técnica apresentada na Figura 14 e na Tabela 2 das fontes de referência, nas quais a não linearidade do modelo é representada por valores complexos previamente calculados.

A atividade consiste em extrair um conjunto de Q valores complexos pré-calculados a partir das funções polinomiais obtidas na etapa anterior. Esses valores serão organizados em tabelas unidimensionais que realizam o mapeamento entre a amplitude da envoltória complexa do sinal de entrada e a respectiva saída complexa estimada do modelo.

Cada LUT será associada a um termo de memória do modelo, permitindo que o processamento não linear seja realizado por meio de operações de indexação e interpolação, em substituição ao cálculo explícito dos termos polinomiais. Essa estratégia visa facilitar a implementação em arquiteturas com recursos limitados, como DSPs e FPGAs, além de possibilitar análises comparativas entre precisão e custo computacional.

Mais detalhes sobre a especificação da atividade, bem como seu acompanhamento e status de desenvolvimento, podem ser consultados no seguinte link:

https://sharing.clickup.com/9011407186/t/h/868h2qyya/BJBVBNEXBF67EJL