Relatório Atividade M1

Leonardo Santos

Objetivo

O objetivo desta atividade foi identificar os coeficientes de um Polinômio de Memória (MP) a partir de um conjunto de dados reais contendo 99 amostras, utilizando **otimização não linear**.

Na Atividade 2 os coeficientes haviam sido obtidos por meio de **otimização linear** (resolução direta de um sistema de equações). Já na Atividade M1, o problema foi reformulado como um processo de **mínimos quadrados não lineares**.

Metodologia

O trabalho foi dividido em duas partes:

1. Construção de uma função de erro
   - A função recebe como argumentos os coeficientes do polinômio de memória.
   - Internamente, a função gera a saída estimada pelo modelo MP e calcula o vetor de erro como:

   $e(n) = y_{\text{out}(n)} - y_{\text{est}(n)}$
   - O retorno da função é o vetor de resíduos.
2. Otimização não linear
   - Em Python, foi utilizado o comando `least_squares` da biblioteca SciPy.
   - Em Octave, utilizou-se o comando `fminunc`, minimizando a soma dos quadrados do erro.
   - Foram testadas diferentes condições iniciais (vetor de zeros, de uns ou aleatório).
   - O critério de parada baseou-se na redução do gradiente do erro.

Implementação

Jupyter Notebook (Python)

No Jupyter Notebook, o procedimento foi:

- Carregar os sinais de entrada (`in`) e saída (`out`).
- Implementar a função de erro.
- Executar o `least_squares`, que ajusta todos os coeficientes de uma só vez.
- Calcular o MSE verdadeiro com:

```
mse = np.mean(res.fun**2)
```

O resultado mostrou convergência rápida, com coeficientes próximos aos da Atividade 2 e MSE $\approx$ 0.0048.

Octave

No Octave, a implementação foi feita em um script `.m`:

Carregou-se o arquivo `.mat` contendo `in` e `out`. Criou-se a função `erro_mp` para calcular o vetor de resíduos. Definiu-se a função objetivo como a soma dos quadrados do erro:

```
f = @(c) sumsq(erro_mp(c, x_in, y_out, ordem, memoria));
```

Utilizou-se o comando:

```
[coef_otimo, fval] = fminunc(f, x0, options);
```

O valor final do MSE foi calculado como `fval / length(y_out)`.

Resultados

Os coeficientes obtidos em Python e Octave foram muito semelhantes, confirmando a consistência do método. O valor do MSE coincidiu em ambos os ambientes quando calculado de forma correta, em torno de 0.0048. A pequena diferença inicial entre os resultados se deveu ao fato de que o `least_squares` em Python retorna `cost = SSE/2`, enquanto no Octave foi calculado diretamente o **MSE**.

Conclusão

A atividade demonstrou como o mesmo problema de identificação de coeficientes pode ser resolvido tanto em Python quanto em Octave, utilizando abordagens equivalentes de otimização não linear. Os resultados foram consistentes e validaram a implementação em ambos os ambientes.