UNIVERSIDADE DO VALE DO ITAJAÍ
ENGENHARIA DE COMPUTAÇÃO
NICOLE MIGLIORINI MAGAGNIN

PROCESSAMENTO DIGITAL DE SINAIS – M3

Relatório apresentado como requisito parcial para a obtenção da M3 da disciplina de Processamento digital de Sinais do curso de Engenharia de Computação pela Universidade do Vale do Itajaí da Escola do Mar, Ciência e Tecnologia.

Prof. Walter Antonio Gontijo

Itajaí
2021

1. OBJETIVO

O presente relatório tem como objetivo a implementação de filtros da terceira média da disciplina de Processamento Digital de Sinais do oitavo período de Engenharia de Computação, ministrada pelo professor Walter Gontijo. As implementações são realizadas em *Python*, utilizando a IDE *Visual Studio Code* e na linguagem C, utilizando a IDE *Visual DSP*.

2. DESENVOLVIMENTO

2.1 – EQUALIZADOR

O equalizador foi implementado no Visual DSP assim como feito na M2 e seus testes foram realizados conforme solicitado.

- Valide para diferentes valores de GB, GF e GA. Por exemplo:
- GB = 1; GF = 0 e GA = 0;
- GB = 0; GF = 1 e GA = 0;
- GB = 0; GF = 0 e GA = 1;
- GB = .7; GF = .6 e GA = .5;

Figura 1 – Testes

Teste com os valores de ganho: GB = 0,5, GA = 0,5 e GF = 0,5

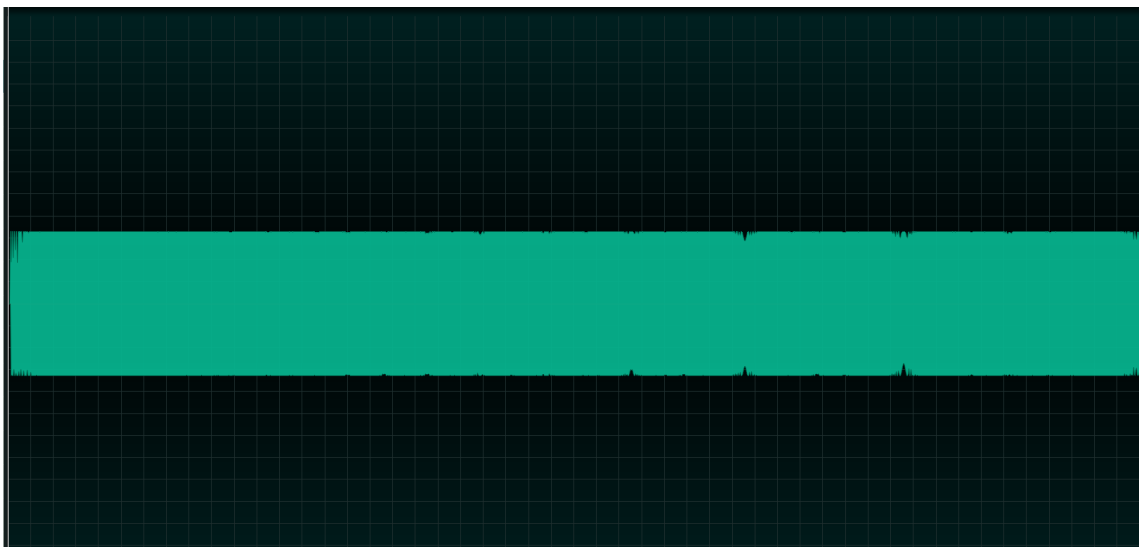

Figura 2 - Teste 1

Teste com os valores de ganho: GB = 1, GA = 0 e GF = 0

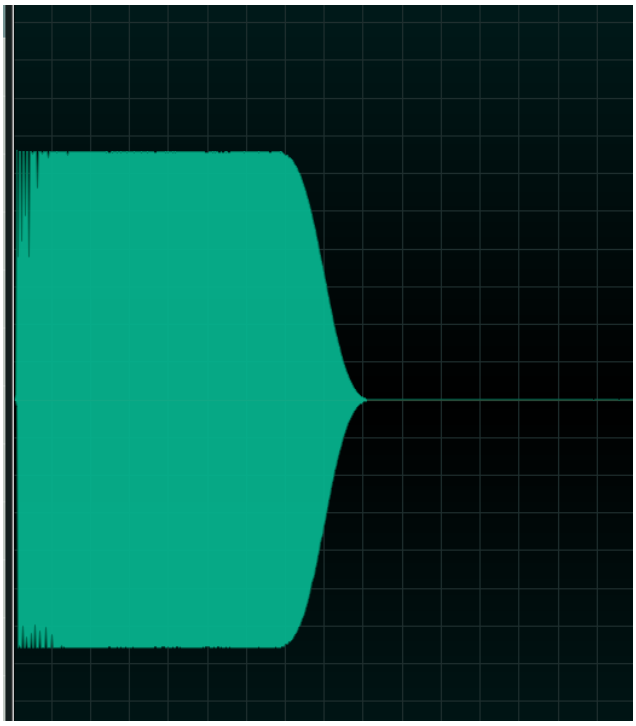

Figura 3 - Teste 2

Teste com os valores de ganho: GB = 0, GA = 1 e GF =0

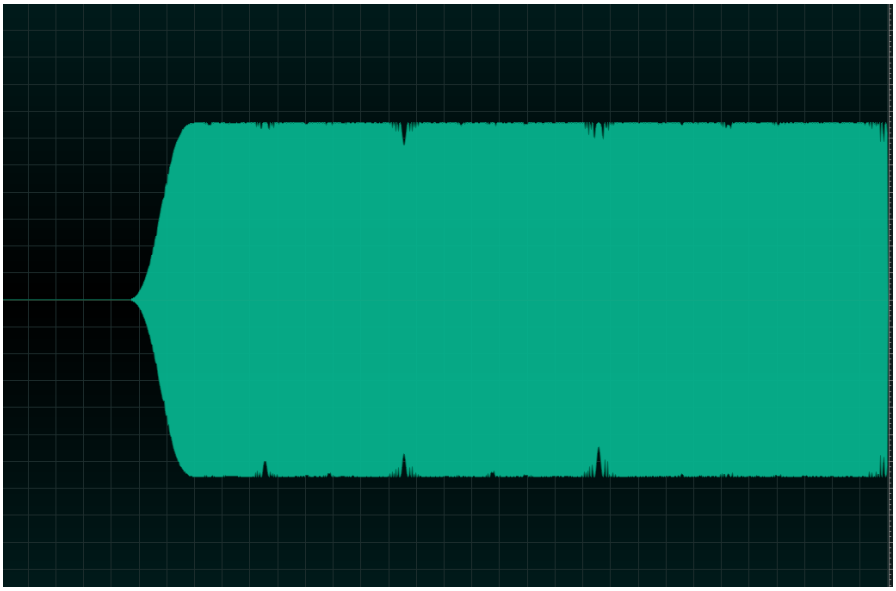

Figura 4 - Teste 3

Teste com os valores de ganho: GB = 0, GA = 0 e GF = 1

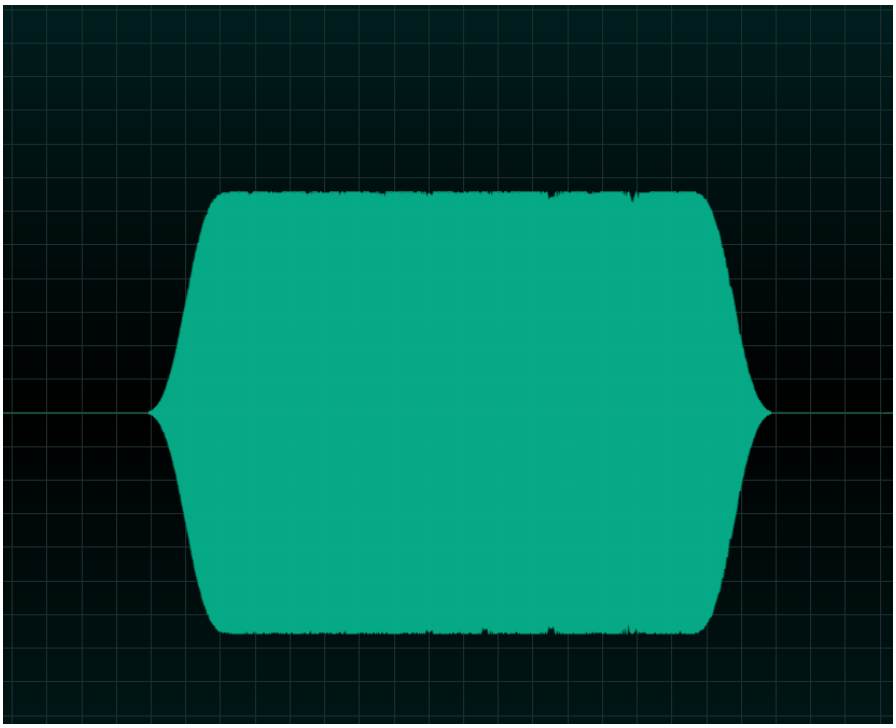

Figura 5 - Teste 4

Teste com os valores de ganho: GB = 0,7, GA = 0,5 e GF = 0,6

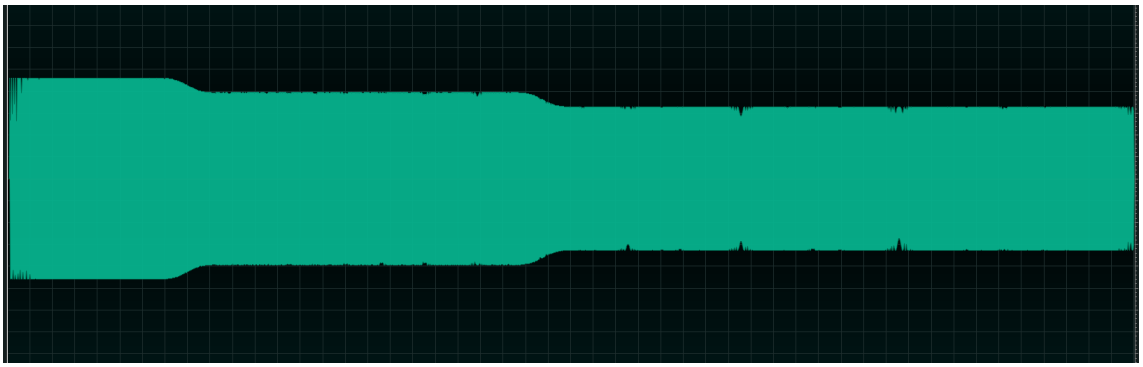

Figura 6 - Teste 5

2.1-2 – COMPARAÇÃO DE DESEMPENHO EQUALIZADOR

2.1-3 – EQUALIZADOR FLOAT

Para realizar a implementação do código do equalizador em short e com melhorias, foi primeiramente executado o código anterior com adição dos contadores de ciclo. Sendo assim, foi necessária a habilitação da contagem utilizando o Preprocessor do projeto e adicionando a instrução **DO_CYCLE_COUNTS**.

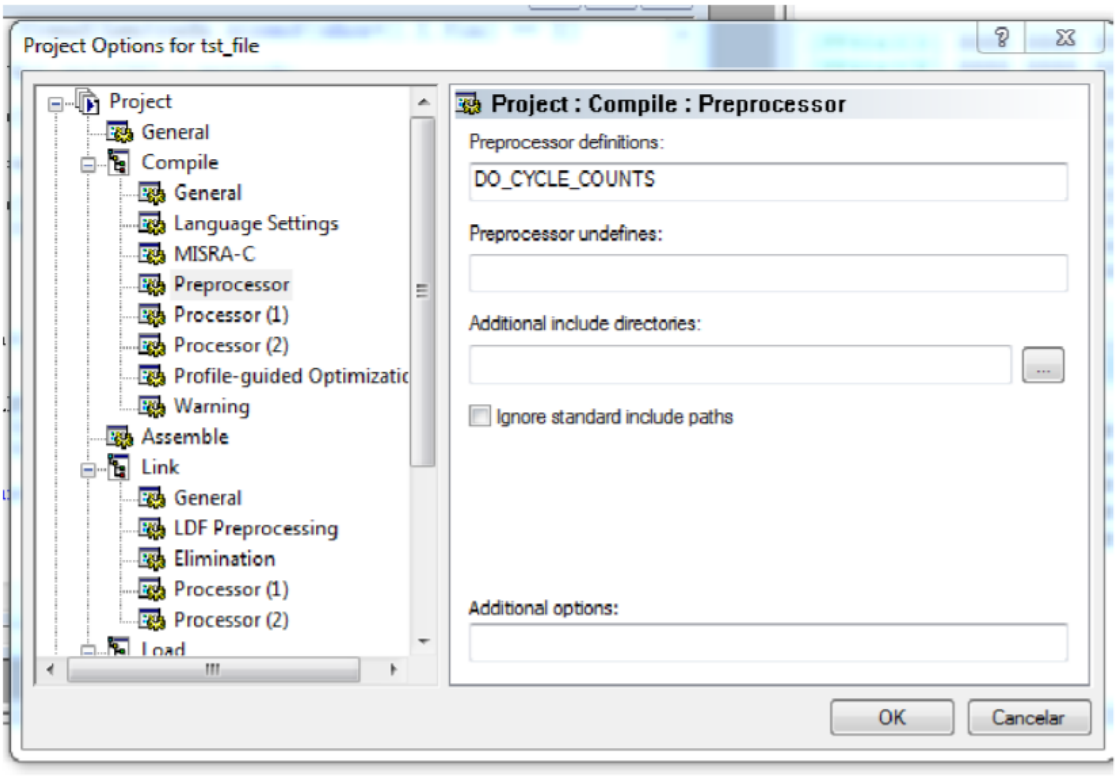

Figura 7 - Habilitada a contagem de ciclos

Após habilitar a opção, são utilizados os comandos para que os status de clock possam ser inicializados, começada e terminada a contagem.

- `cycle_stats_t stats;` // Declarar a estrutura
- `CYCLES_INIT(stats);` // Inicializa
- `CYCLES_START(stats);` // Dispara a contagem
- `saida = proc_alg( Coefs, Vet_entr, NSAMPLES);`
- `CYCLES_STOP(stats);` // Para a contagem
- `CYCLES_PRINT(stats);` // Imprime o resultado

Figura 8 - Comandos de clock

Sendo assim, foi executado o código utilizando o ganho de 1 para filtros passa-alta, passa-baixa e passa-faixa e obtendo-se o seguinte resultado:

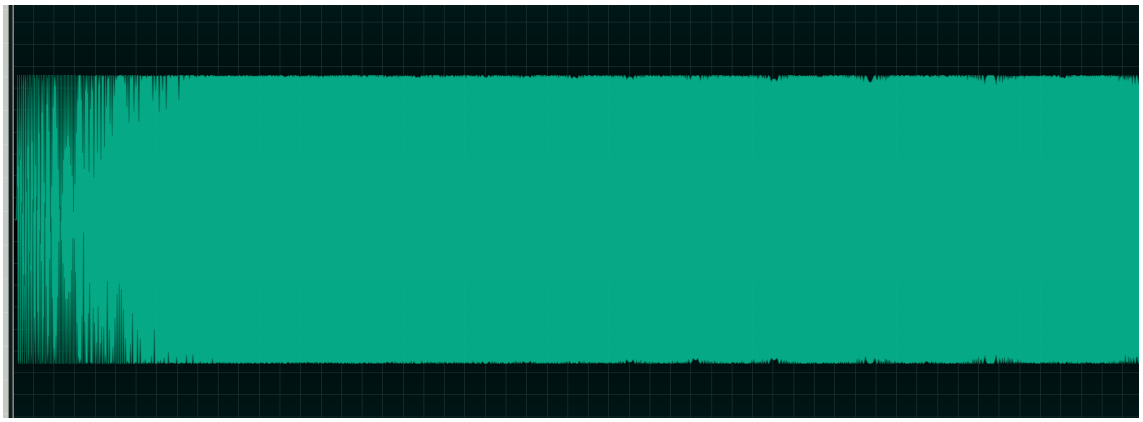

Figura 9 - Resultado equalizador Float

```
terminado!
      AVG    : 290702
      MIN    : 149476
      MAX    : 291798
      CALLS  : 80001
```

Figura 10 - Resultado de tempo

2.1-4 – EQUALIZADOR SHORT

Para a implementação do equalizador em short, foram gerados coeficientes *short* como os do exemplo a seguir, estes são coeficientes PA, PB e PF multiplicados por 32.768, ou 2^{15}, o que transforma os números em 16 bits, o equivalente a um *short*.

Antes da multiplicação, os coeficientes apenas são separados, retirando as vírgulas presentes anteriormente e evitando duplicação das mesmas.

```
with open('Coef_PA.dat', 'r') as f:
    coefs_PA = [line.strip().replace(',', '') for line in f]

coefs_short_PA = numpy.zeros(len(coefs_PA))
for i in range(len(coefs_PA)):
    coefs_short_PA[i] = int(float(coefs_PA[i]) * 32768)

with open("coefs_short_PA.dat", "w") as f:
    for s in coefs_short_PA:
        f.write(str(s) +",\n")
```

Figura 11 - Código para gerar coeficientes

Outras diferenças nas implementações são que todas as variáveis float tornam-se short e a saída recebe um shift de 15, para que possa se tornar de int para short. Os resultados para ganhos em PA, PB e PF de 1 foram:

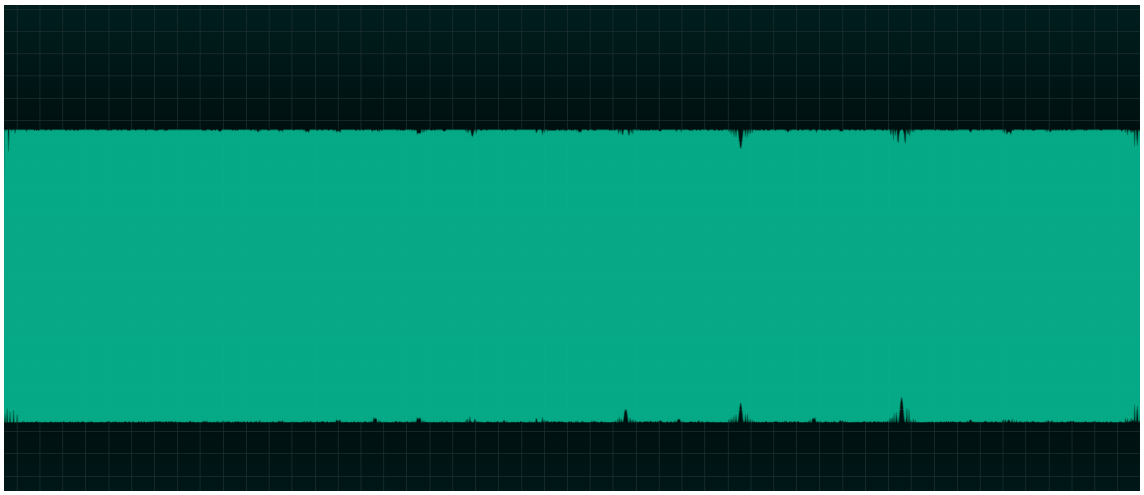

Figura 12 - Resultado filtro short

```
AVG    : 23692
MIN    : 23692
MAX    : 23692
CALLS  : 80001
```

Figura 13 - Resultado filtro short

TABELA COMPARATIVA

Implementação	AVG	MIN	MAX	CALLS
Equalizador	290702	149476	291798	80001
Equalizador_otimizado(short)	23692	23692	23692	80001

2.2 – FILTRO ADAPTATIVO

2.2-1 – FILTRO ADAPTATIVO EM PYTHON

Os filtros adaptativos são principalmente usados no processamento de imagem para aprimorar ou restaurar dados, removendo o ruído sem borrar significativamente as estruturas na imagem. O algoritmo consiste em através de uma entrada de Sistema desconhecido e uma entrada de sinal sonoro, é gerada uma saída tentando se aproximar cada vez mais da saída desejada a partir dos dados de entrada.

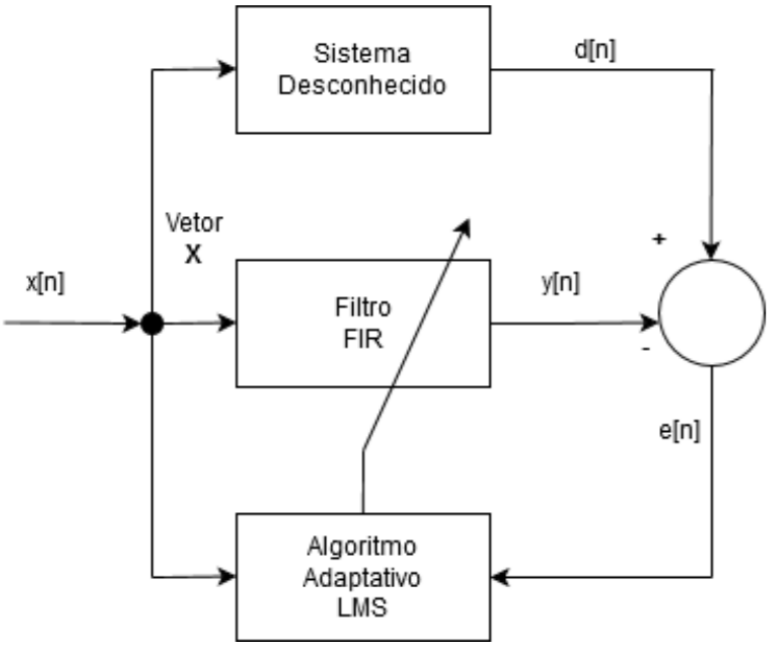

Figura 14 - Diagrama de bloco do filtro adaptativo

Condições para implementar o algoritmo LMS:

- O filtro deve ter como entrada:
  - Vetor de coeficientes, dado por $w[n]$.
  - Vetor de entrada, dado por $x[n]$.
  - Saída desejada, dada por $d[n]$.
- O filtro deve ter como saída:
  - Saída do filtro, dada por $y[n]$.
  - Vetor de coeficientes atualizado, dado por $w[n+1]$.

Passos para implementação do algoritmo LMS:

1. Filtragem:

$$y[n] = w^T[n] * x[n]$$

Figura 15 - Descrição da fórmula do algoritmo adaptativo

$$e[n] = d[n] - y[n]$$

Figura 16 - Estimativa do erro

$$w[n+1] = w[n] + 2 * \mu * e[n] * x[n]$$

Figura 17 - Adaptação do vetor de coeficientes

O filtro adaptativo tem como entrada um ruído branco e um Sistema desconhecido, além de uma saída desejada. Para o filtro adaptativo implementado, foram solicitadas duas validações do filtro, a primeira delas com uma média móvel, a qual foi usada uma média móvel de coeficiente 8.

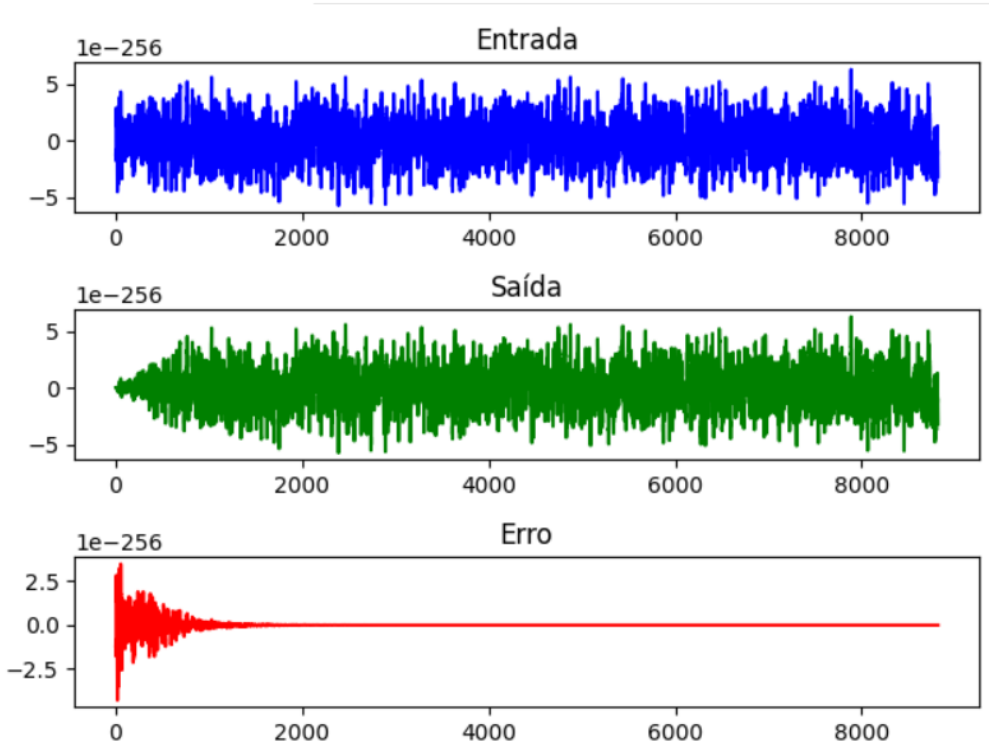

Figura 18 - Filtro adaptativo utilizando os coeficientes de média móvel

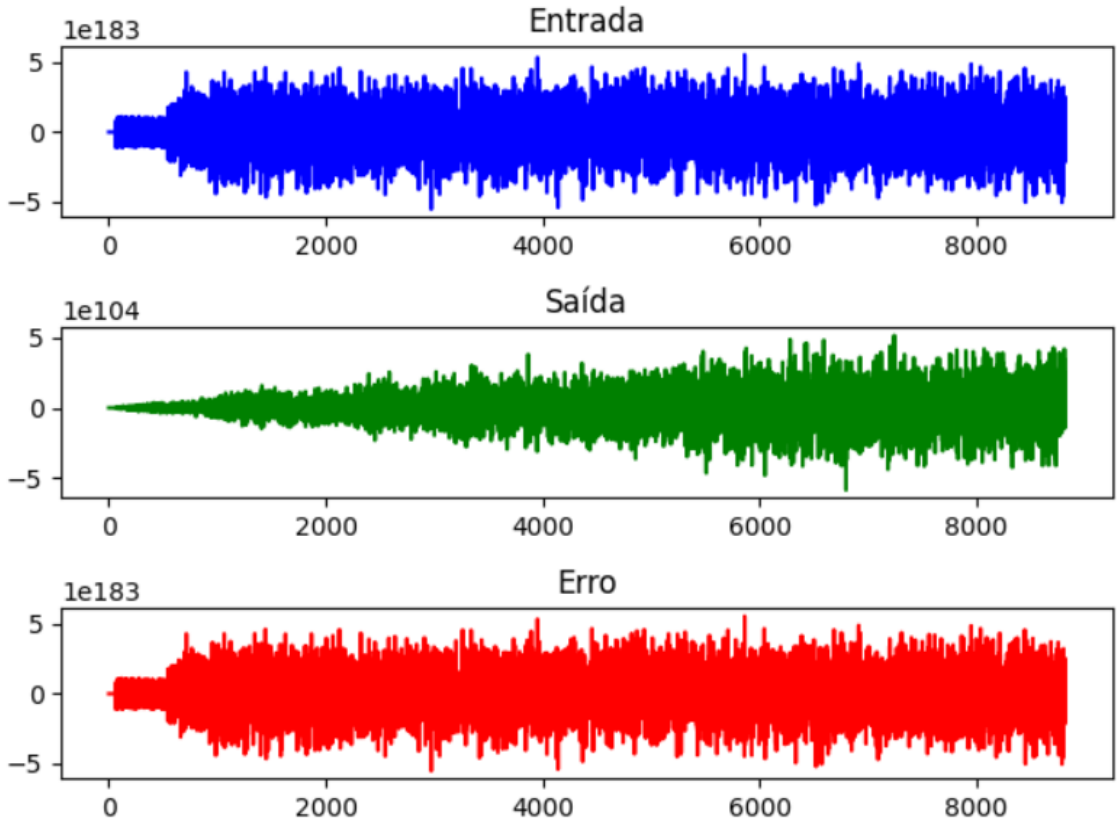

Figura 19 - Filtro adaptativo com validação para coeficientes Passa-baixa

3. REFERÊNCIAS

KIKINIS, Carl-Fredrik Westin Ron; KNUTSSON, Hans. **Adaptive Image Filtering**. 2021. Disponível em: https://www.sciencedirect.com/topics/computer-science/adaptive-filter. Acesso em: 13 dez. 2021.