Prática 03

Filipe Augusto Parreira Almeida, RA: 2320622

2 de Novembro - 2023

Projetando o Filtro PASSA-FAIXA

Para o projeto do filtro foi passado as seguintes especificações:

- O sinal **amostrado** deve ser de **44.1 KHz**
- A **banda de passagem** deve ser de **80 Hz a 270 Hz**
- Um filtro **FIR Tipo 1** de no **mínimo 65 taps**

Para a montagem foi utilizado a linguagem de programação Python. Seguindo os seguintes passos:

- Foi definido inicialmente os **parâmetros** do filtro:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from IPython.display import Audio
4 from scipy.signal import firwin, lfilter, freqz
5
6 taps = 65 #Quantidade de taps
7 fs = 44100 #Frequencia de amostragem
8 f1 = 80 #Frequencia de corte inferior
9 f2 = 270 #Frequencia de corte superior
```

- Realizando o calculo dos **coeficientes do filtro** com o auxilio do método firwin e obtendo a **resposta em frequência** e a frequência normalizada do filtro para o plot:

```
# Calculando os coeficientes do filtro
nyquist = fs/2
freq = [f1 / nyquist, f2 / nyquist]
coeficientes = firwin(taps, freq, pass_zero=False)

# Gerando uma resposta de frequ ncia
resRespFreq = 1600
wNorm, respFreq = freqz(coeficientes, worN=resRespFreq, fs=fs)
```

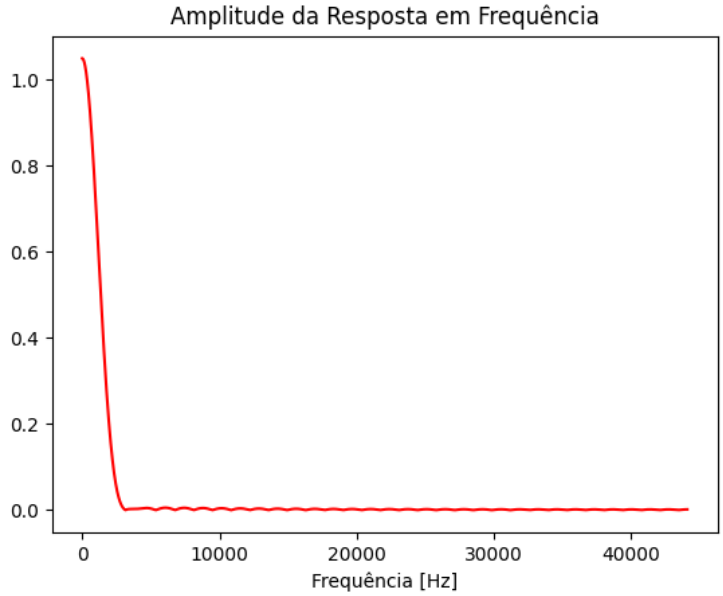
Plot dos Parâmetros

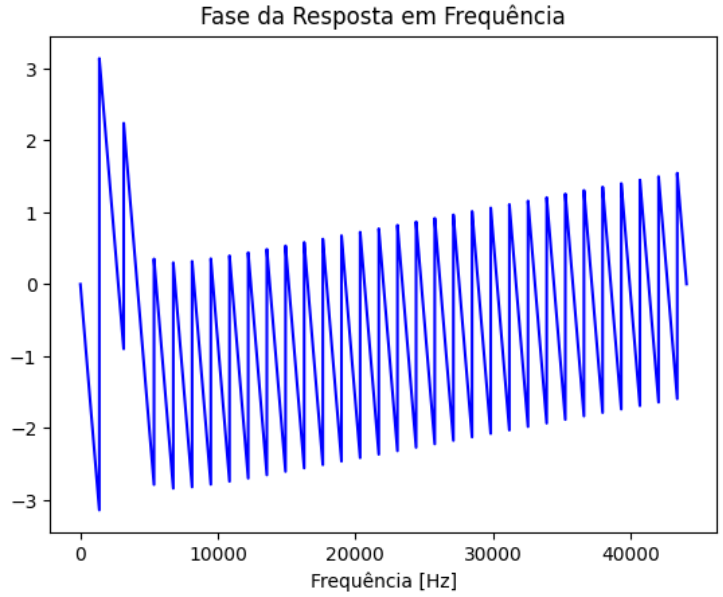
Plot da fase e da amplitude da resposta em frequência do filtro:

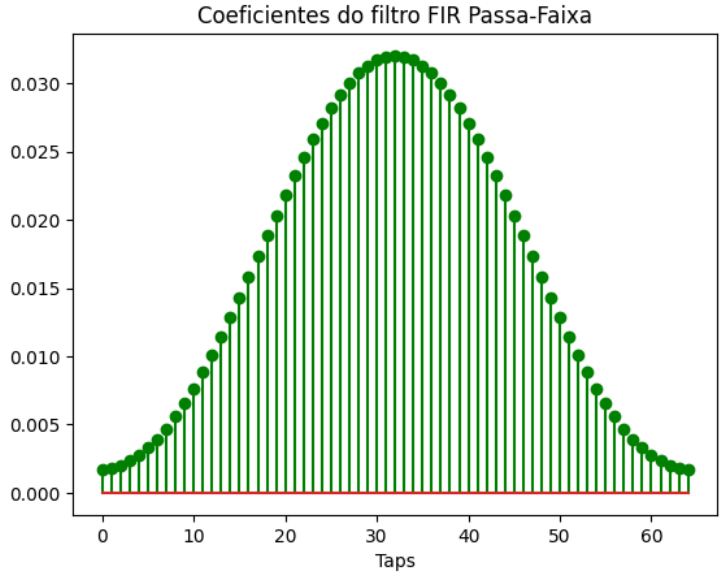
- Código:

```
# Plotando a resposta de frequ ncia
plt.figure()
plt.plot(0.5 * fs * wNorm / np.pi, np.abs(respFreq), 'r')
plt.title("Amplitude da Resposta em Frequ ncia")
plt.xlabel('Frequ ncia [Hz]')

plt.figure()
plt.plot(0.5 * fs * wNorm / np.pi, np.angle(respFreq), 'b')
plt.title("Fase da Resposta em Frequ ncia")
plt.xlabel('Frequ ncia [Hz]')


plt.figure()
plt.stem(coeficientes)
plt.title("Coeficientes do filtro FIR Passa-Faixa")
plt.xlabel('Taps')
```

- Plots:


Amplitude da Resposta em Frequência
1.0
0.8
0.6
0.4
0.2
0.0
0
10000
20000
30000
40000
Frequência [Hz]


Fase da Resposta em Frequência
3
2
1
0
−1
−2
−3
0
10000
20000
30000
40000
Frequência [Hz]



- Análise:

A **resposta em frequência** deste filtro descreve como ele atua em diferentes componentes de frequência do sinal de entrada, dessa forma temos que analisar as suas duas partes: **a amplitude e a fase**. **A amplitude** de um filtro mostra como ele **atenua ou amplifica** diferentes componentes de frequência do sinal, logo, neste caso, é possível perceber através do gráfico de amplitude, que **a amplitude é alta entre a banda de passagem** (80 Hz a 270 Hz) e baixa nas demais. Já **a fase** é possível perceber uma semelhança com a ideia da amplitude, porém ela é mais utilizada para **ajustar a fase do sinal de entrada**, em muitos casos quando a fase do sinal de entrada é **critica**. A resposta em frequência do filtro também é útil para identificar se o filtro está atendendo as expectativas, caso **não** esteja, é possível **alterar os parâmetros** (numero de taps, frequência de corte, etc) para que se chegue no modelo ideal para o uso.

Construção do Filtro em C

A lógica do algoritmo do filtro em C é receber **dois arquivos de entrada** (os coeficientes e o sinal de entrada) e retornar **um arquivo de saída** (o sinal filtrado). O código pode ser executado por meio do terminal onde é necessário passar 3 parâmetros: **arqCoeficientes.txt arqEntrada.txt arqSaida.txt**, o nome dos arquivos de entrada devem ser os **mesmos** que estão no diretório salvo e o nome do arquivo de saída é facultativo.

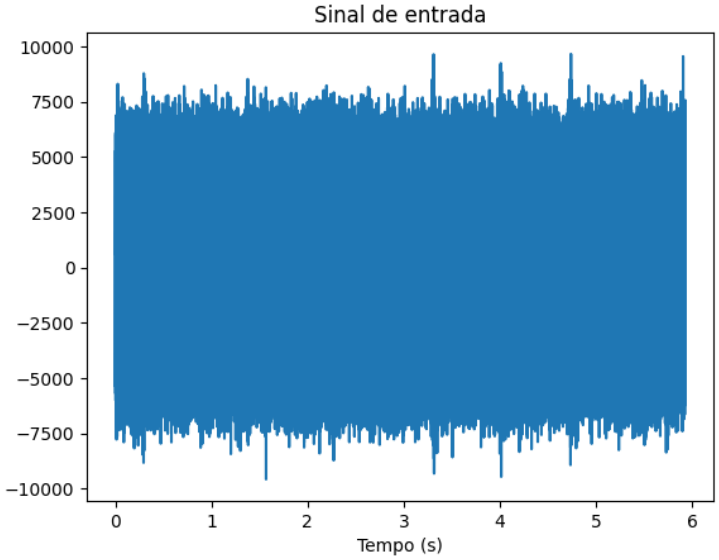
- Código:

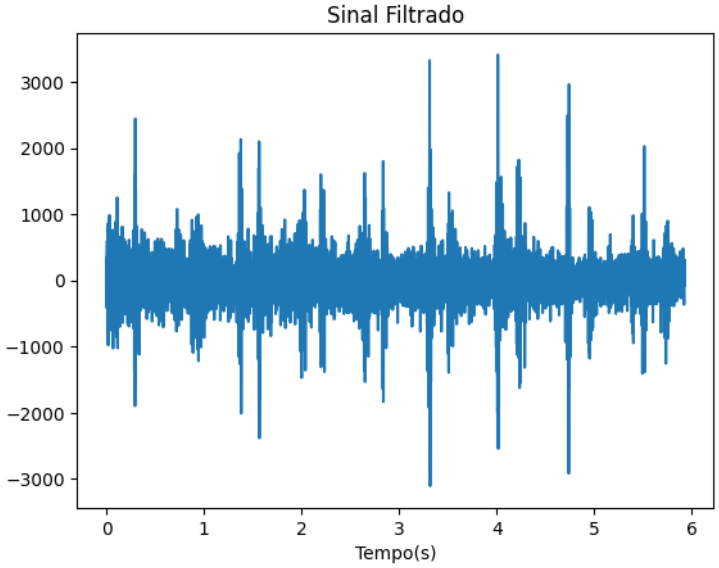
```
#include <stdio.h>
#include <stdlib.h>

#define MIN(X, Y) (((X) < (Y)) ? (X) : (Y))
#define MAX(X, Y) (((X) < (Y)) ? (Y) : (X))

long double* convolve(long double* h, long double* x, int lenH, int lenX, int* lenY)
{
    int nconv = lenH+lenX-1;
    (*lenY) = nconv;
    int i,j,h_start,x_start,x_end;

    long double *y = (long double*) calloc(nconv, sizeof(long double));

    for (i=0; i<nconv; i++)
    {
        x_start = MAX(0,i-lenH+1);
        x_end   = MIN(i+1,lenX);
        h_start = MIN(i,lenH-1);
        for(j=x_start; j<x_end; j++)
        {
        y[i] += h[h_start--]* x[j];
        }
    }
    return y;
}

int quantidadeLinhas(char *path){
    char letra;
    int countLinhas = -1;
    FILE *file = fopen(path, "r");

    while((letra = fgetc(file)) != EOF){
        if(letra == '\n'){
        countLinhas++;
        }
    }

    rewind(file);

    return countLinhas;
}

long double* lerDados(char *path){
    FILE *file = fopen(path, "r");
    int tamArq = quantidadeLinhas(path);
    long double *data = (long double*)malloc(tamArq * sizeof(long double));

    for (int i = 0; i ≤ tamArq; i++){
        fscanf(file, "%Lf", &data[i]);
    }
```

```
52
53      rewind(file);
54
55      return data;
56  }
57
58  int main(int argc, char *argv[])
59  {
60      //Primeiro o arquivo dos coeficientes, segundo o sinal de ...
            entrada e terceiro
61      // o arquivo de saida
62      char *coeficientes = argv[1];
63      char *signalIN = argv[2];
64      char *arqOut = argv[3];
65
66      long double *h = lerDados(coeficientes);
67      long double *x = lerDados(signalIN);
68      int tamH = quantidadeLinhas(coeficientes);
69      int tamX = quantidadeLinhas(signalIN);
70
71
72      int lenY;
73      long double *y = convolve(h,x,tamH,tamX,&lenY);
74
75      FILE *arqSaida = fopen(arqOut, "w");
76
77      if(arqSaida == NULL){
78          printf("Error!");
79          exit(1);
80      }
81
82      for(int i=0;i<lenY;i++) {
83          fprintf(arqSaida, "%Lf\n",y[i]);
84      }
85      puts("");
86      free(y);
87      free(h);
88      free(x);
89      return 0;
90  }
```

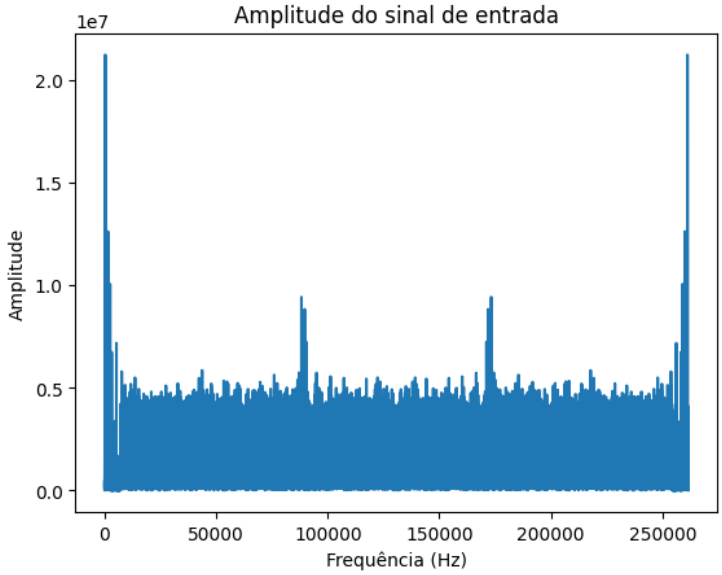
Análise dos Resultados

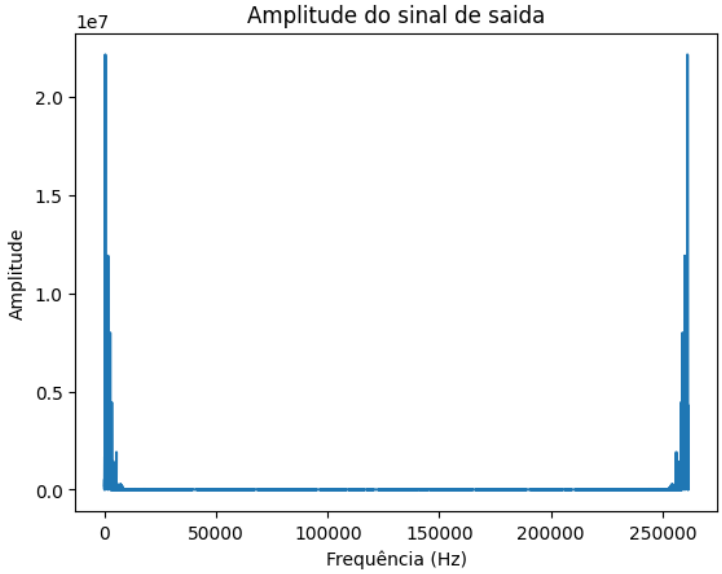
- Plot dos sinais de entrada e saída no tempo:





- Plot dos sinais de entrada e saída na frequência:





- Análise:

Analisando os gráficos dos sinais no domínio do tempo é bastante perceptível que **o sinal de saída é bem mais definido** do que o sinal antes de ser filtrado, também, através da analise dos gráficos no domínio da frequência é possível perceber que **as altas frequências foram praticamente cortadas**.

Escutando o sinal de saída é notável a diferença com relação ao sinal de entrada, sendo possível uma **maior percepção dos batimentos cardíacos**. Escutando o áudio filtrado e analisando o gráfico da saída no domínio do tempo **é possível identificar a taxa de batimentos**, foi identificado **aproximadamente** 9 **batimentos** durante a duração do áudio (aproximadamente **5 segundos**), logo, realizando a seguinte equação: $Fbatimentos = Nbatimentos * 12$, onde o numero de batimentos foi dado em **5 segundos**, para saber a frequência de batimentos por minuto basta multiplicar por 12. Logo, temos que $Fbatimentos = 9 * 12$, $Fbatimentos = 108/min$, sendo assim, **108 batimentos por minuto**.