PROJETO 02 - FILTROS EM MATLAB - Filtro FIR

DISCIPLINA: DEB0804 - Processamento Digital de Sinais

PROFESSOR: Alan Cássio Queiroz Bezerra Leite

ALUNOS(A): .....

.....

PROJETO 02

Parte 03

Nesta prática pretende-se demonstrar (i) o projeto de filtros de resposta finita (FIR - Finite Impulse Response) através do cálculo de seus coeficientes e (ii) a implementação do filtro. Pra isto usamos a técnica de ”janelas”.

Resumo pontos abordados nesta parte	
Ponto 1	*Execução de uma filtragem usando função “conv”*
Ponto 2	*Calculando coeficientes FIR fase a fase*
Ponto 3	*Calculando coeficientes FIR usando funções fir1 e fir2*
Ponto 4	*Calculando coeficientes FIR usando Parks-McClellan*

Ponto 1: Implementação de filtros FIR

Para implementar um filtro FIR é necessário que se tenha em mãos os coeficientes do filtro. Por isto, antes é necessário aprender a projetar um filtro que é o que faremos nos próximos tópicos. Por enquanto, vamos imaginar que já temos os coeficientes prontos e nossa intenção se resume a implementar o filtro considerando que o projeto (que é a designação dos coeficientes do filtro) já esteja pronta. O Código abaixo mostra como é feita a implementação de um filtro FIR passa baixas com coeficientes já dados (linha 2 e 3). Note que a execução do filtro é feita por uma convolução da entrada com a resposta impulsiva $h(n)$ do filtro na linha 17.

Uma outra forma muito usual de se executar uma filtragem no Matlab é usando a função *filter*. Para isto, basta substituir a linha código *saida = conv(entrada_discretizada, h);* do exemplo anterior por: *saida = filter(h,1,entrada_discretizada)*

Código 4.1 - Implementação de um filtro FIR dados seus coeficientes.

```
1  clc; clear;
2  h = [0.4375, 0.3142, 0.0625, -0.0935, -0.0625, 0.0418, 0.0625, -0.0124, -0.0625, -0.0124,
3  0.0625, 0.0418, -0.0625, -0.0935, 0.0625, 0.3142];
4  n_h = 0:15;
5  %% PARTE 1: Gera um sinal analógico (entrada_analogica) e sua versão discretizada
6  (entrada_discretizada)
7  Fs = 20000;             % taxa de amostragem: 20mil amostras/seg
8  t = 0:(1/Fs):0.02;      % amostragem de 0,02seg
9  N = max(t)/(1/Fs);
10 n = 0:N;    % quantidade de amostras da entrada para filtrar
11 %entrada_analogica = sin(2*pi* 200*t) + sin(2*pi*25*t) + sin(2*pi* 5000*t) + sin(2*pi*
12 9000*t);
13 entrada_discretizada = sin(2*pi*200.*n/Fs) + sin(2*pi*25.*n/Fs) + sin(2*pi*5000.*n/Fs) +
14 sin(2*pi*9000.*n/Fs);
15
16 %% PARTE 2: convolui entrada com h
17 saida = conv(entrada_discretizada, h);
18
19 %% PARTE 3: PLOTA RESULTADOS
20 % calcula espectro entrada
21 fft_sinal_entrada = fft(entrada_discretizada)/N;
22 f_entrada = n.*(Fs/N);
23 % calcula espectro saida
24 N3 = size(saida,2);
25 fft_sinal_saida =   fft(saida)/N3;
26 n3 = 0:size(fft_sinal_saida,2)-1;
27 f_saida = n3.*(Fs/N3);
28 % calcula espectro filtro
29 fft_resp_filtro = fft(h);
30 n4 = 0:size(fft_resp_filtro,2)-1;
31 N4 = size(n4,2);
32 f_h = n4.*(Fs/N4);
33 %plota espectros
34 subplot(3,1,1); stem(n_h,h); xlabel('n'); title('Coef. filtro');
35 subplot(3,1,2); plot(f_entrada(1:N/2), abs(fft_sinal_entrada(1:N/2)));
36                 hold on;
37                 plot(f_saida(1:N3/2), abs(fft_sinal_saida(1:N3/2)),'r');
38                 plot(f_h(1:N4/2), abs(fft_resp_filtro(1:N4/2)),'g');
39                 legend('entrada','saída','resp. impulsiva');
40                 xlabel('Freq (Hz)');
41                 title('Espectros dos sinais e filtro')
42 %plota sinais
43 subplot(3,1,3); plot(n, entrada_discretizada);
44                 hold on;
45                 plot(n3, saida,'r');
46                 legend('entrada','saída');
47                 xlabel('n');
48                 title('Sinais discretos do filtro');
```

Ponto 2: Projetar um filtro FIR pelo método de janelas

Projetar um filtro FIR usando o método de janelas é uma tarefa relativamente fácil. No exemplo do código abaixo dividimos o código em oito partes para facilitar seu entendimento. Na parte 1 (linhas 3 a 7) especificamos as características do filtro. São elas:

- F_s:indica a taxa de aquisição máxima do sinal de entrada do filtro;
- F_c: indica a frequência de corte (em Hertz);
- N: número de coeficientes do filtro que pretendemos projetar;
- Num_Coef_filtro: dos N coeficientes calculados, devemos usar apenas parte deles que será designado por esta variável Num_Coef_filtro.

Na parte 2 (linhas 10 a 16) especificamos o espectro ideal para o filtro baseado nos valores coletados na parte 1. A partir deste espectro idealizado encontramos na parte 3 (linha 19) quais devem ser os coeficientes ideais do filtro para implementação deste espectro idealizado. Note que se o filtro é passa alta (linha 20) devemos multiplicar os coeficientes encontrados por $cos(2\pi n(F_s/2)/Fs) = cos(\pi n) = [1, -1, 1, -1, ...]$. Isto transforma o filtro passa-baixas em passa-altas. Já na linha 25 o código testa se foi escolhido um filtro passa-faixas para transformar o filtro passa-baixas em um passa-faixas através do deslocamento de frequência feito pela multiplicação dos coeficientes do filtro passa-baixa por $cos(2\pi n f_{cc}/F_s)$ onde f_{cc} indica a frequência de centro da banda de passagem.

Código 4.2 - Projeto de um filtro FIR passa-baixas, passa-latas e faixas.

```
clc; clear;
%% PARTE 1: dados do filtro
Fs = 20000;            %taxa de amostragem
fc = 2000;             %frequencia de corte do filtro
N =  64;               %Quantidade total de coeficientes filtro
Num_Coef_Filtro = 64;  %Qtos coeficientes vou usar do total
tipo_filtro = input('\nTipo filtro(1=passa-baixas; 2=passa-altas; 3=passa-faixa) =');

%% PARTE 2: gera o espectro ideal do filtro
H = zeros(1,N);            %resposta em freq ideal do filtro
f_resol = Fs/N;            %calcula resolução frequencia
m_corte = round(fc/f_resol)+1;   %estima indice "m" de fc
H = [ones(1, m_corte + 1) zeros(1, (N/2-m_corte))  zeros(1, (N/2-m_corte))  ones(1, m_corte
- 1)]; %gera espectro ideal do filtro
m = 0:N-1;
f = m.*f_resol;            %define o eixo de frequencias do grafico 1

%% PARTE 3: gera todos coef. ideais do filtro
h_ideal = ifft(H);
if (tipo_filtro==2) %se passa-alta, multiplicar coef por [1,-1,1,-1,...]
    n = 0:N-1;
    deslocamento_f = cos(2*pi*n.*(Fs/2)/Fs);
    h_ideal = h_ideal.*deslocamento_f;
end
if (tipo_filtro==3) %se passa-faixa, multiplicar coef por modulacao
    n = 0:N-1;
    fcc = input('\nDigite em Hz frequência central =');
    deslocamento_f = cos(2*pi*fcc.*n/Fs);
    h_ideal = h_ideal.*deslocamento_f;
end

%% PARTE 4: GERA SIMETRIA PAR NA FUNCAO SYNC DOS COEFICIENTES
h_ideal(N/2:N) = h_ideal(1:N/2+1);
for i=2:(N/2)
    h_ideal(i-1)= h_ideal(N-i+1);
end
inicio = N/2 - Num_Coef_Filtro/2 + 1;
fim    = N/2 + Num_Coef_Filtro/2;
h = real(h_ideal(inicio:fim)); %pega so parte dos coeficientes ideias do filtro

%% PARTE 5 (opcional): aplica janela
resposta = input('\nDeseja aplicar janela aos coef (1=sim; 2=nao)? = ');
if (resposta == 1)
    jan = window(@blackman,(fim-inicio)+1);
    h = h.*jan';
end

%% PARTE 6: testa a implementação filtro com sinal sintetico
N_sinal_sintetico = 400;     % quantidade de amostras do sinal sintetico
n1 = 0:N_sinal_sintetico-1;
entrada_discretizada = sin(2*pi*500.*n1/Fs) + sin(2*pi*2500.*n1/Fs) + sin(2*pi*5000.*n1/Fs)
+  sin(2*pi*8000.*n1/Fs);
saida = conv(entrada_discretizada, h);
N_sinal_saida = size(saida,2);
N_resp_impulsitva = Num_Coef_Filtro;

%% PARTE 7: calcula espectros sinal entrada, saida e h(n)
fft_sinal_entrada = fft(entrada_discretizada)/N_sinal_sintetico;
fft_sinal_saida   = fft(saida)/N_sinal_saida;
fft_resp_filtro   = fft(h);
f_entrada = n1.*(Fs/N_sinal_sintetico);
n3 = 0:size(fft_sinal_saida,2)-1;
f_saida   = n3.*(Fs/N_sinal_saida);
n2 = 0:size(fft_resp_filtro,2)-1;
f_h = n2.*(Fs/N_resp_impulsitva);

%% PARTE 8: plota
subplot(2,2,1); stem(f,H); title('H(f) idealizado'); xlabel('f(Hz)')
subplot(2,2,2); stem(real(h_ideal)); xlabel('n'); hold on; stem([inicio:fim],h,'-.r');
title('Coeficientes h(n) do filtro'); ylabel('Amplitude'); xlabel('n');
legend('Todos','Selecionados');
subplot(2,2,3); plot(f_entrada(1:N_sinal_sintetico/2),
abs(fft_sinal_entrada(1:N_sinal_sintetico/2)));
                hold on;
                plot(f_saida(1:N_sinal_saida/2),
abs(fft_sinal_saida(1:N_sinal_saida/2)),'r');
                plot(f_h(1:N_resp_impulsitva/2),
abs(fft_resp_filtro(1:N_resp_impulsitva/2)),'g');
                legend('entrada','saída','resp. impulsiva');
                xlabel('Freq (Hz)');
                title('Espectros dos sinais e filtro')
subplot(2,2,4); plot(n1, entrada_discretizada);
                hold on;
                plot(n3, saida,'r');
                legend('entrada','saída');
                xlabel('n');
                title('Sinais discretos do filtro');
```

Na parte 4 cuidados para gerar um gráfico de coeficientes simétrico em relação ao eixo vertical pois a $ifft$ só estima metade da função *sync*. A outra metade deve ser preenchida através de espelhamento. Na parte 5 temos a opção de usar uma janela aos coeficientes para atenuar o *ripple* da banda passante. Aplicar uma janela é importante quando Num_Coef_filtro < N pois nesta situação é gerado um *ripple* que pode ser atenuado por algumas funções de janelamento.

Finalmente, nas partes 6 a 8 geramos um sinal sintético com 4 componentes de frequência para testar o filtro e calculamos os espectros do sinal sintético de entrada, o espectro da resposta impulsiva do filtro e o espectro do sinal de saída. Faça testes para ver os efeitos de filtragem.

Uma forma muito interessante de se avaliar a resposta impulsiva de um filtro é usando a função $freqz$ do Matlab. O argumento desta função são os coeficientes $h(n)$ do filtro e o resultado é a magnitude da resposta em frequência do filtro (designado como *mag* no código 4.3) e as correspondentes frequências normalizadas por π. Assim, a maior frequência tem valor 1 e indica a frequência de Nyquist que em nosso exemplo é $F_s/2 = 20.000/2 = 10.000Hz$.

Código 4.3 - Visualização da resposta em frequência de um filtro

```
1  [mag,f] = freqz(h);
2  plot(f/pi,abs(mag))
```

Ponto 3: Projetar um filtro FIR usando a função do Matlab FIR1 e FIR2.

A função FIR1 do Matlab tem um dos seguintes formatos:

- Coeficientes = $fir1(n, W_n)$
- Coeficientes = $fir1(n, W_n, 'ftype')$
- Coeficientes = $fir1(n, W_n, window)$
- Coeficientes = $fir1(n, Wn, 'ftype', window)$

Onde W_n indica a faixa de frequências que determinam o filtro, n a ordem, *ftype* o tipo (que pode ser *high* ou *stop*) e *window* os coeficientes da janela que pode ser opcional. Observe os exemplos do código 4.4. A função $FIR1$ usa o método de projeto conhecido como *projeto baseado por resposta ao impulso*.

Código 4.4 – Cálculos coeficientes filtro FIR usando função FIR1

```
ordem = 50;
resol_plot_freq = 512;

%Exemplo 1a: passa-baixas (frequência normalizada)
coef = fir1(ordem, 0.3);
freqz(coef,1,resol_plot_freq)

%Exemplo 1b: passa-baixas (frequência nominal considerando Fs=20k)
fc = 3000;
Fs = 20e3;
coef = fir1(ordem, fc/(Fs/2));
[H , freq] = freqz(coef,1,resol_plot_freq, 20e3);
plot(freq,abs(H));



%Exemplo 2: passa-altas
coef = fir1(ordem, 0.4,'high');
freqz(coef,1,resol_plot_freq)

%Exemplo 3: passa-altas com janela de Hanning
coef = fir1(ordem, 0.6, 'high', hann(ordem+1));
freqz(coef,1,resol_plot_freq)
%Exemplo 4: passabanda
coef = fir1(ordem, [0.3 0.5]);
freqz(coef,1,resol_plot_freq)

%Exemplo 5: rejeita-banda
coef = fir1(ordem, [0.3 0.7], 'stop');
freqz(coef,1,resol_plot_freq)
```

No código 4.4, deve-se escolher frequências normalizaas. Isto quer dizer que a maior frequência que pode ser representada pelo sistema, que equivale a $F_s/2$ (segundo critério de Nyquist) é denominada de 1. Assim, se desejamos projetar um filtro para cortar em $3000Hz$ usando um sistema com $F_s = 20000Hz$, sabe-se que a máxima frequência de $10000Hz$ equivale a 1. Logo, a frequência de $3000Hz$ equivale $3000/10000 = 0,3$. A função *freqz* analisa a resposta em frequência dos coeficientes.

Um outro tipo de projeto dos coeficientes do filtro FIR é conhecido como *amostragem baseada em frequência* e é implementada pela função FIR2 conforme ilustra o código 4.5. A função FIR2 tem geralmente o formato $fir2(n, f, m, window)$ onde n indica o ordem do filtro, f as frequências do filtro, m as respectivas amplitudes que devem ter as frequências especificadas pelo vetor f e *window* o tipo de janela usado para os $n + 1$ coeficientes do filtro (o uso de janelas é opcional e pode ser deixado em branco conforme ilustra alguns dos exemplos do código 4.5).

Código 4.5 – Cálculos coeficientes filtro FIR usando função FIR2

```
ordem = 50;
resol_plot_freq = 512;

%Exemplo 1: passa-baixas
f = [0, 0.3,  0.4,  1]; m = [1, 1, 0, 0];
coef = fir2(ordem, f, m);
freqz(coef,1,resol_plot_freq);

%Exemplo 2: passa-altas
f = [0, 0.5,  0.65,  1]; m = [0, 0, 1, 1];
coef = fir2(ordem, f, m);
freqz(coef,1,resol_plot_freq);

%Exemplo 3: passa-altas com janela
f = [0, 0.5,  0.65,  1]; m = [0, 0, 1, 1];
coef = fir2(ordem, f, m, hann(ordem+1));
freqz(coef,1,resol_plot_freq);

%Exemplo 4: passa-banda
f = [0, 0.4, 0.5, 0.7, 0.8, 1]; m = [0, 0, 1, 1, 0, 0];
coef = fir2(ordem, f, m);
freqz(coef,1,resol_plot_freq);

%Exemplo 5: rejeita-banda
f = [0, 0.4, 0.5, 0.7, 0.8, 1]; m = [1, 1, 0, 0, 1, 1];
coef = fir2(ordem, f, m);
freqz(coef,1,resol_plot_freq);
```