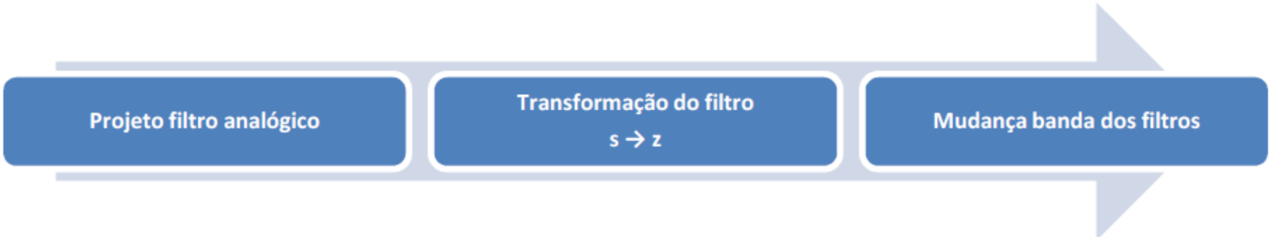
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA BIOMÉDICA

PROJETO 03 - FILTROS EM MATLAB - Filtro IIR	
DISCIPLINA:	DEB0804 - Processamento Digital de Sinais
PROFESSOR:	Alan Cássio Queiroz Bezerra Leite
ALUNOS(A):	....
	....

PROJETO 03

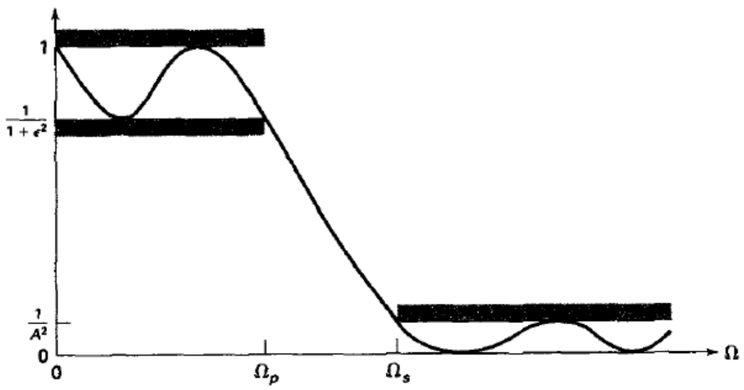
Parte 04

Nesta prática pretende-se demonstrar o projeto de filtros IIR a partir de transformações de filtros analógicos. Este projeto é realizado por etapas sequenciais. Deste modo, os pontos subsequentes representam as etapas necessárias para um projeto de um filtro IIR e sua implementação em cada um dos pontos abordados aqui. Destaca-se que este material é complementar as aulas teóricas e por isto não tem como objetivo ensinar todo o conteúdo de filtros IIR. Seu propósito é reforçar os conteúdos ensinados em sala e exemplificar como projetar um filtro em Matlab usando funções específicas.



Ponto 1: Implementação de um filtro analógico passa-baixas Butterworth

Este ponto se propõem a descrever o projeto básico de um filtro Butterworth analógico passa-baixas usando o Matlab.
Um filtro analógico é especificado no domínio de Laplace (especificado pela variável complexa s). O primeiro passo é descobrir a ordem de um filtro que deve ser usado para satisfazer a determinados requerimentos. Estas especificações são novamente mostradas no gráfico abaixo.



Um filtro analógico é especificado no domínio de Laplace (representado pela variável complexa s) e há várias maneiras de especificar um filtro. Geralmente elas são feitas por duas formas: (i) especificando a frequência de corte e ordem do filtro ou (ii) especificando a banda de transição, ganho na banda de passagem e atenuação na banda de rejeição. A primeira forma deve ser feita usando o código 5.1 onde se projeta um filtro com frequência corte 0.5 (lembrando que a máxima frequência é 1 que equivale a Nyquist) de ordem 3. Note a utilização da função *butter* do Matlab. O termo *low* indica um filtro passa-baixas e o termo *s* indica que o filtro é feito no domínio do analógico de Laplace. Além de *low* podemos usar *high* e *stop* sendo este um rejeita-banda.

Código 5.1 - Projeto de um filtro Butterworth passa-baixas analógico a partir da frequência de corte e número de polos.

```
N = 3;
OmegaC = 2500/5000;  %freq de corte 2.500Hz com taxa de aquisição de 2x5.000Hz (normalizado)
[b,a] = butter(N,OmegaC,'low','s')
```

O resultado é: $b = 0.1250$ e $a = [1.01.00.500.1250]$ que matematicamente equivale a:

$$\frac{0.1250}{1.0s^3 + 1.0s^2 + 0.50s + 0.1250}$$

A segunda forma de especificação deve ser determinada a banda de transição, o ganho na banda de passagem e atenuação na banda de rejeição. Este processo de projeto é mostrado no Código 5.2. Neste caso, usamos a biblioteca empregada nesta disciplina e a partir destas especificações determina-se a ordem do filtro e sua função matemática. Neste exemplo usamos uma banda de transição de 0.2π a 0.3π (lembrando que 1π é a frequência de Nyquist neste tipo de representação). O ripple da banda de passagem é de 7dB e a atenuação na banda de rejeição é 16dB neste exemplo.

Código 5.2 - Projeto de um filtro Butterworth passa-baixas analógico a partir da banda de transição, ripple e atenuação.

```
wp = 0.2*pi; % 0 a 0.2pi = banda passagem
ws = 0.3*pi; % 0.3pi a 1pi = banda rejeição
rp = 7;      %ripple banda passagem em dB
as = 16;     %atenuação mínima na banda rejeição em dB
ripple = 10^(-rp/20);    %descobre valor do ripple sem escala log
atenuacao = 10^(-as/20); %descobre valor da atenuação sem escala log
[b, a] = afd_butt(wp, ws, rp, as)
w = 0:pi/1000:pi;  %determina um eixo para freq
h = freqs(b,a,w);  %calcula a resposta em freq do filtro
plot(w/pi,abs(h));
```

O resultado é: $b = 0.1238$ e $a = [1.00.99690.49690.1238]$ que matematicamente equivale a:

$$\frac{0.1238}{1.0s^3 + 0.9969s^2 + 0.4969s + 0.1238}$$

Ponto 2: Transformações de frequência de analógico (s) para digital (z)

Há várias formas de converter um filtro analógico especificado matematicamente no domínio s para um filtro no formato digital que deve ser especificado matematicamente no domínio z. Uma destas técnicas de conversão é a (i) transformada de invariância ao impulso (que será ilustrada no código 5.3) e outra é a (ii) transformada bilinear. A transformação por invariância ao impulso utiliza a função *impinvar* do Matlab é ilustrada no código 5.3. Consideramos neste exemplo T=1 lembrando que $\omega = \Omega T$.

Código 5.3 - Exemplo de conversão de uma equação no domínio "s" para o "z" usando o método de invariância ao impulso

```
%% PARTE 1: especificações projeto
wp = 0.2*pi; % 0 a 0.2pi = banda passagem
ws = 0.3*pi; % 0.3pi a 1pi = banda rejeição
rp = 7;      %ripple banda passagem em dB
as = 16;     %atenuação mínima na banda rejeição em dB
T = 1;       % taxa de amostragem da frequencia

%% PARTE 2: desenha filtro analógico
[b, a] = afd_butt(wp, ws, rp, as);
w = 0:pi/1000:pi;  %determina um eixo para freq

h = freqs(b,a,w);  %calcula a resposta em freq do filtro
subplot(2,1,1); plot(w/pi,abs(h)); title('resp freq analogico');

%% PARTE 3: digitaliza filtro usando inv ao impulso
[bz, az] = impinvar(b,a,T);
hz = freqz(bz,az,w);
subplot(2,1,2); plot(w/pi,abs(hz)); title('resp freq digital');
```

O resultado é: $bz = [-0.00.04380.0314]$ e $az = [1.0 - 2.02331.4675 - 0.3690]$ que matematicamente equivale a transformação:

$$\frac{0.1238}{1.0s^3 + 0.9969s^2 + 0.4969s + 0.1238} \Rightarrow \frac{0.0438z^{-1} + 0.0314z^{-2}}{1.0 - 2.0233z^{-1} + 1.4675z^{-2} - 0.3690z^{-3}}$$

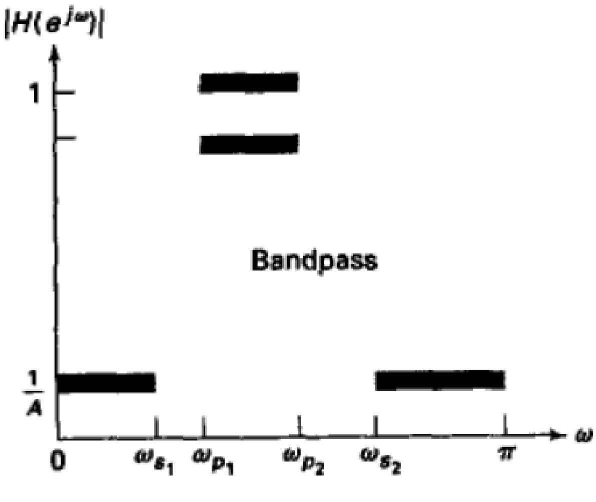
Já a transformação bilinear emprega a função do matlab *bilinear* e para ver seu uso use o código 5.3 substituindo na linha 16 o nome *impinvar* por *bilinear*. Ambas as funções servem a um mesmo propósito. Contudo, a transformação bilinear é geralmente mais usada por geralmente apresentar menos distorções na transformação do plano s para o círculo z.

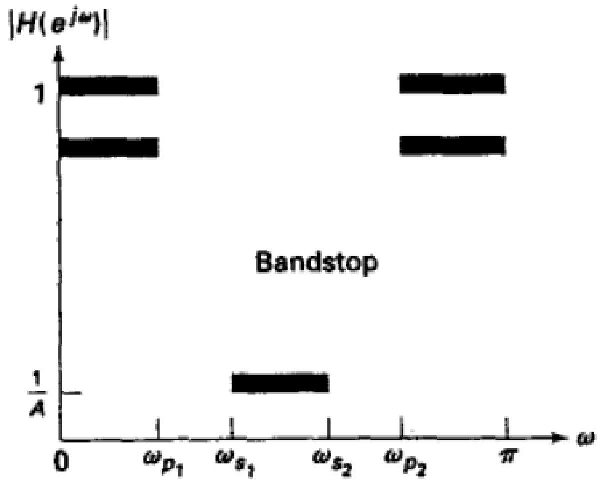
Ponto 3: Projeto de filtros passa-altas, rejeita-banda e passa-banda

Para o projeto de filtros passa-altas, rejeita-bandas e passa-banda. Para projetar estes filtros usaremos a função *butter* do Matlab do seguinte modo:

- $[b, a] = butter(N, w_n,' high')$ projeta um filtro passa-altas com frequência de corte $w_n (w_n = w_c = 3dB)$;
- $[b, a] = butter(N, w_n)$ projeta um filtro passa-bandas de ordem 2N se w_n é um vetor tal que $wn = [w_1 w_2]$ compreende a faixa de frequências que deve passar;
- $[b, a] = butter(N, w_n,' stop')$ projeta um filtro rejeita-bandas de ordem 2N se w_n é um vetor tal que $w_n = [w_1 w_2]$ e compreenda as faixas de frequências que devem ser rejeitadas;

Em todos os casos anteriores, a frequência é dada em função de π indicando que π refere-se a máxima frequência do sinal. Destaca-se também que as frequências especificadas anteriormente devem ter amplitudes próximas de -3dB (0,7 unidades).
Uma outra função importante é a $[N, w_n] = buttord(w_p, w_s, r_p, a_s)$ onde especificamos a banda de transição especificadas pelas frequências $[w_p, w_s]$, o ripple da banda de passagem (r_p - em dB) e a atenuação nas banda de rejeição (a_s - em dB). O resultado da função é a determinação da ordem N do filtro e da frequência de corte wn. No caso desta função, para um filtro passa-bandas, wp e ws são vetores com dois elementos tal que: $wp = [wp1, wp2]$ e $ws = [ws1, ws2]$. Para entender melhor estes parâmetros, veja as figuras abaixo. É importante destacar que esta função considera frequências normalizadas na faixa [01] onde 1 indica a maior frequência.





Código 5.4 - Código de um filtro passa-baixas de 0 a 3.400Hz com transição de 3400 a 4000Hz.

```
Fs = 40000;
wp= 3400/(Fs/2);
ws= 4000/(Fs/2);
[n, wn]=buttord(wp, ws, 1, 20);
[b, a]= butter(n,wn);
num_freq = 512;
freqz(b, a, num_freq, Fs);
```

Código 5.5 - Código de um filtro passa-altas de 4000Hz a 20000Hz sendo esta última a máxima frequência.

```
Fs = 40000;
wp= 4000/(Fs/2);
ws= 3400/(Fs/2);
[n, wn]=buttord(wp, ws, 1, 20);
[b, a]= butter(n,wn,'high');
num_freq = 512;
freqz(b, a, num_freq, Fs);
```

Código 5.6 - Código de um filtro passa-faixas de 4000 a 8000Hz com transições de largura 600Hz.

```
Fs = 40000;
ws1= 3400/(Fs/2);
wp1= 4000/(Fs/2);
wp2= 8000/(Fs/2);
ws2= 8600/(Fs/2);
[n, wn]=buttord([wp1 wp2], [ws1 ws2], 1, 20);
[b, a]= butter(n,wn);
num_freq = 512;
freqz(b, a, num_freq, Fs);
```

Código 5.7 - Código de um filtro rejeita-faixas que exclui as faixas de 4500 a 8000Hz e largura de transição de 500Hz.

```
Fs = 40000;
wp1= 4000/(Fs/2);
ws1= 4500/(Fs/2);
wp2= 8500/(Fs/2);
ws2= 8000/(Fs/2);
[n, wn]=buttord([wp1 wp2], [ws1 ws2], 1, 20);
[b, a]= butter(n,wn,'stop');
num_freq = 512;
freqz(b, a, num_freq, Fs);
```

Ponto 4: Implementação do Filtro

Para executar uma filtragem, basta ter em mãos os coeficientes designados por b e a e usar a função *filter* conforme ilustra o código 5.8 na linha 15.

Código 5.8 - Execução de um filtragem IIR.

```
1  Fs = 40000;
2  wp1= 4000/(Fs/2);
3  ws1= 3400/(Fs/2);
4  wp2= 6000/(Fs/2);
5  ws2= 6600/(Fs/2);
6  [n, wn]=buttord([wp1 wp2], [ws1 ws2], 1, 20);
7  [b, a]= butter(n,wn);
8  %% cria sinal sintético
9  t = 0:(1/Fs):0.02;
10 N = max(t)/(1/Fs);
11 n = 0:N;
12 entrada_discretizada = sin(2*pi*900.*n/Fs) + sin(2*pi*14500.*n/Fs) + sin(2*pi*5000.*n/Fs) +
13 sin(2*pi*9000.*n/Fs);
14 %% aplica filtro
15 sinal_filtrado = filter(b, a, entrada_discretizada);
16 %% calcula espectros e plota sinais
17 fft_sinal_entrada = abs(fft(entrada_discretizada)/N);
18 fft_sinal_filtrado = abs(fft(sinal_filtrado)/N);
19 f_resol = Fs/N;
20 f = n.*f_resol;
21 subplot(1,2,1); plot( entrada_discretizada); hold on; plot(sinal_filtrado,'r');
22 subplot(1,2,2); plot(f(1:N/2), fft_sinal_entrada(1:N/2)); hold on;
23 plot(f(1:N/2),fft_sinal_filtrado(1:N/2),'r');
```