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

PROJETO 01 - PROJETO DE SINAIS E FILTROS EM MATLAB	
DISCIPLINA:	DEB0804 - Processamento Digital de Sinais
PROFESSOR:	Alan Cássio Queiroz Bezerra Leite
ALUNOS(A):	....
	....

PROJETO 01

Parte 01

Esta prática tem por propósito ilustrar a geração de sinais básicos e a trabalhar com eles com as principais funções do Matlab.

Ponto 1: geração de sinais básicos

Amostra unitária $\delta(n)$: esta função é gerada usando o código impseq(n0, inicio, fim). conforme exemplo:

```
[sinal1, n1] = impseq(0, -3, +3)
sinal1 =
     0     0     0     1     0     0     0
n1 =
    -3    -2    -1     0     1     2     3

[sinal2, n2] = impseq(3, 1, 5)
sinal2 =
     0     0     1     0     0
n2 =
     1     2     3     4     5
```

Degrau unitário $u(n)$: esta função é gerada usando o código stepseq(n0, inicio, fim). conforme exemplo:

```
[sinal1, n1] = stepseq(0, -3, +3)
sinal1 =
     0     0     0     1     1     1     1
n1 =
    -3    -2    -1     0     1     2     3

[sinal2, n2] = stepseq (3, 1, 5)
sinal2 =
     0     0     1     1     1
n2 =
     1     2     3     4     5
```

Sequência exponencial complexa: para entender melhor o comportamento de sinais, experimente gerar um e visualizá-lo conforme exemplo abaixo da função.

```
n=[-5:30];
a=exp((0.2+3j)*n);
subplot(2,2,1); stem(n, real(a));  title('Parte real');
subplot(2,2,2); stem(n, imag(a));  title('Parte imaginária');
subplot(2,2,3); stem(n, abs(a));   title('Magnitude');
subplot(2,2,4); stem(n, angle(a)); title('Fase');
b=exp(0.2*n);
c=exp(3j*n);
figure;
subplot(2,1,1); stem(n, b); title('Módulo');
subplot(2,1,2); stem(n, c); title('Fase');
```

Ponto 2: Amostrando Sinais

Para simular o processo de conversão de tempo contínuo para tempo discreto, utilize o código abaixo.

```
%% inicializacoes
F = 60;          %freq analogica
Fs = 400;        %freq aquisicao
ts = 1/Fs;       %tempo aquisicao
num_ciclos = 3;  %(equivale a 3 ciclos)
t = 0:1e-6:(num_ciclos*(1/F)); %tempo simulacao
n = 0:round(num_ciclos*(Fs/F));        %numero amostras

%% parte analogica
f_analogico = sin(2*pi*F*t);
plot(t,f_analogico);

%% parte digital
f_digital = sin(2*pi*(F/Fs)*n);
hold on;
stem(n.*ts,f_digital);
```

Ponto 3: Réplicas Espectrais

A ideia deste ponto é mostrar a réplica espectral. Para isto, considere um sistema cuja taxa de aquisição é $Fs = 6000$ amostras por segundo. Se considerarmos a frequência de $1kHz$, as frequências de $7k(1k + F_s)$,$13k(1k + 2F_s)$, $19k(1k + 3F_s)$ e assim por diante, passam pelos mesmos pontos que o sinal de $1kHz$ se confundindo com eles e gerando o indesejado fenômeno de Aliasing.

```
%% inicializacoes
F1 = 1000;          %freq analogica
Fs = 6000;        %freq aquisicao
ts = 1/Fs;       %tempo aquisicao
num_ciclos = 1;  %(equivale a 3 ciclos)
t = 0:1e-6:(num_ciclos*(1/F1)); %tempo simulacao
n = 0:round(num_ciclos*(Fs/F1));        %numero amostras

%% parte analogica
f_analogico1 = sin(2*pi*F1*t);
f_analogico2 = sin(2*pi*7000*t);
f_analogico3 = sin(2*pi*13000*t);

plot(t,f_analogico1,'r');
hold on;
plot(t,f_analogico2,'b');
plot(t,f_analogico3,'k');
legend('1k','7k','13k');

%% parte digital
f_digital = sin(2*pi*(F1/Fs)*n);
stem(n.*ts,f_digital);
```

Ponto 4: Operação com Sinais

Soma: a soma de dois sinais em função de n pode ser feita usando a função sigadd($sinal1, n\_do\_sinal1, sinal2$ conforme exemplo abaixo:

```
x1 = [1 2 3 2 1];
n1 = [-2 -1 0 1 2];
x2 = [2 3 4];
n2 = [0 1 2];
[x3, n3] = sigadd(x1,n1,x2,n2);
stem(n3,x3);
```

Multiplicação: a multiplicação de dois sinais em função de n pode ser feita usando a função sigmult(sinal1, n_do_sinal1, sinal2, n_do_sinal2) conforme exemplo abaixo:

```
x1 = [1 2 3 2 1];
n1 = [-2 -1 0 1 2];
x2 = [2 -3 4];
n2 = [-1 0 1];
[x3, n3] = sigmult(x1,n1,x2,n2);
stem(n3,x3);
```

Deslocamento: o deslocamento (avanço ou atraso) de um sinal pode ser gerado através da função sigshift conforme mostra o exemplo:

```
x1 = [1 2 3 2 1];
n1 = [0 1 2 3 4];
[x2, n2] = sigshift(x1,n1,2); %mesmo que x1(n-2)
subplot(2,1,1); stem(n1,x1); title('Original');
subplot(2,1,2); stem(n2,x2); title('Atrasada 2 unid');
```

Convolução: embora o Matlab tenha uma função própria para convolução (*conv*), ela leva em conta que os sinais são sempre casuais (ou seja, $x[n] = 0$ para $n < 0$). Para evitar esta limitação, é aconselhável usar a função *conv_m* conforme ilustra exemplo abaixo.

```
x1 = [3, 11, 7, 0, -1, 4, 2];
n1 = [-3:3];
h = [2, 3, 0, -5, 2, 1];
nh = [-1:4];
[y, ny] = conv_m(x1,n1,h,nh);
stem(ny, y);
```

Parte 02

Nesta prática pretende-se demonstrar o uso da transformada discreta de Fourier (DFT) e a principal função do Matlab para cálculo da transformada discreta de Fourier: a FFT.

Ponto 1: Calcular Manualmente a Transformada

Para calcular a DFT de um sinal manualmente utilize o código abaixo. Vale destacar que neste caso consideramos o sinal como causal (em $n = 0$). Note que neste exemplo não se nota a preocupação em otimizar o código e se pode usar tanto a forma de *Euler* (linha 10) ou pela sua forma expandida (linha 11). Ambos casos conduzem a um mesmo resultado.

Código 3.1 - Cálculo manual da DFT

```
1   clear; clc;
2   sinal = [1 2 3 4 5]; %sinal
3   Fs = 1000; %taxa de aquisição
4   N = size(sinal,2); %num de amostras
5   y = zeros(1,N);
6   %% calcular a DFT
7   for m=0:(N-1)
8      acumulador = 0;
9      for n=0:(N-1)
10        acumulador = acumulador + sinal(n+1)*(cos(2*pi*n*m/N)-j*sin(2*pi*n*m/N));
11        %acumulador = acumulador + sinal(n+1)*exp(-j*2*pi*n*m/N);
12     end
13     y(m+1)=acumulador;
14  end
15  %% calcular eixo frequencias f:
16  m = 0:(N-1);
17  f = m*Fs/N;
18  %% plotar sinais
19  magX = abs(y);
20  angX = angle(y);
21  realX = real(y);
22  imagX = imag(y);
23  subplot(2,2,1); stem(f,magX); title('magnitude');
24  subplot(2,2,3); stem(f,angX); title('fase');
25  subplot(2,2,2); stem(f,realX); title('real');
26  subplot(2,2,4); stem(f,imagX); title('imaginario');
```

Ponto 2: fft

O código ilustrado anteriormente é capaz de calcular a *DFT* mas não tem eficiência computacional pois não faz uso das propriedades de simetria de cálculo da *DFT*. Uma função computacionalmente mais eficiente para cálculo da *DFT* é a *fft* (*fast Fourier transform*). O código 3.2 ilustra seu uso.

Código 3.2 - Cálculo manual da DFT usando a função FFT do Matlab

```
1   clc; clear;
2   %% gera um sinal "sintetico" com 3 componentes de frequencia
3   Fs = 1000; %taxa de aquisicao do sinal
4   ts = 1/Fs;
5   N = 1000;  %numero amostras de amostras analisadas
6   t = (0:N-1)*ts;
7   sinal = 0.6*sin(2*pi*50*t) + 2*sin(2*pi*120*t) + 0.3*sin(2*pi*400*t);
8   ruido = 0.4*randn(size(t));
9   %% calcula a DFT usando a funcao fft
10  y = fft(sinal+ruido);
11  y = y/N; %se desejar pode-se dividir por N para "acomodar" os calculos
12  %% calcular o eixo das frequencias
13  m = 0:(N-1);
14  f = m*Fs/N;
15  y = y(1:N/2); %pegando so a primeira metade já que a outra é cópia
16  f = f(1:N/2); %pegando so a primeira metade já que a outra é cópia
17  %% calcular a potência do espectro em db
18  magnitude = abs(y);
19  fase = angle(y);
20  f_ref = max(magnitude); %escolhe o maior valor para ser a referencia para normalizacao
21  y_db = 20*log10(magnitude/f_ref);  %lembre que, por exemplo 0db = 1 unidade
22  %% plotar
23  subplot (3,1,1); plot(f ,magnitude); title('magnitude espectro'); xlabel('freq(Hz)');
24  ylabel('Amplitude');
25  subplot (3,1,2); plot(f ,y_db); title('potencia espectro');
26  subplot (3,1,3); plot(f ,fase); title('fase');
```

Note que ao sinal, na linha 10, foi inserido o ruído para cálculo da *fft*. Experimente executar o código usando a adição e sem a adição do ruído ao sinal para ver o efeito do ruído no espectro. Note ainda a amplitude do ruído que pode alcançar até 0.4 (linha 8) enquanto que a terceira componentes espectral de 400Hz (linha 7) tem amplitude máxima de 0.3. Mesmo assim ela é mais perceptível que o ruído uma vez que este tem pouca correlação com funções de seno/cosseno. Observe ainda o quanto a função logarítmica amplifica as pequenas componentes e atenua as grandes.

Ponto 3: usando janelas para calcular a FFT de sinais

Para diminuir os efeitos de diminuição do leakage do cálculo de uma DFT, usamos funções matemáticas (que chamamos de janelas) que são multiplicadas pelo sinal a ser analisado. O código abaixo mostra o uso de diferentes janelas. Ao lado do código pode ser visto o nome de diferentes janelas. Note, na linha 4, que algumas delas exigem parâmetros específicos.

Código 3.3 - Cálculo manual da DFT usando a função FFT do Matlab

```
1  N = 100;
2  janela1 = window(@blackmanharris,N);
3  janela2 = window(@hamming,N);
4  janela3 = window(@gausswin,N,2.5);
5  wvtool(janela1, janela2, janela3);
```

```
@barthannwin
@bartlett
@blackman
@blackmanharris
@bohmanwin
@chebwin
@flattopwin
@gausswin
@hamming
@hann
@kaiser
@nuttallwin
@parzenwin
@rectwin
@taylorwin
@triang
@tukeywin
```

Para ilustrar o uso de uma janela de Hanning, podemos proceder como no código 3.4 (que é uma versão adaptada do código 3.2). Veja a diferença da DFT de um sinal janelado e um ”puro”(não janelado).

Código 3.4 - Uso de janelas para cálculo da DFT visando diminuir o leakage

```
clc; clear;
%% gera um sinal "sintetico"
Fs = 200;
ts = 1/Fs;
N = 300;
t = (0:N-1)*ts;
sinal = 0.6*sin(2*pi*13*t);
%% cria um segundo sinal "janelado"
janela = window(@hann,N);
sinal_jan = sinal'.*janela;
%% calcula a DFT usando a funcao fft
y = fft(sinal);
y_jan = fft(sinal_jan);
%% calcular o eixo das frequencias
m = 0:(N-1);
f = m*Fs/N;
y = y(1:N/2);
y_jan = y_jan(1:N/2);
f = f(1:N/2);
%% calcular a potência do espectro em db
magnitude = abs(y);
magnitude_jan = abs(y_jan);
f_ref = max(magnitude);
f_ref_jan = max(magnitude_jan);
y_db = 20*log10(magnitude/f_ref);
y_db_jan = 20*log10(magnitude_jan/f_ref_jan);
%% plota
subplot (3,2,1); plot(t ,sinal); title('sinal nao janelado');
subplot (3,2,3); stem(f ,magnitude); title('magnitude espec sem jan'); ylim([0 max(magnitude)]);
subplot (3,2,5); plot(f ,y_db); title('potencia espectro sem janela');
subplot (3,2,2); plot(t ,sinal_jan); title('sinal janelado');
subplot (3,2,4); stem(f ,magnitude_jan); title('magnitude espec com jan'); ylim([0 max(magnitude)]);
subplot (3,2,6); plot(f ,y_db_jan); title('potencia espectro com janela');
```

Ponto 4: diferença entre resolução espectral e densidade espectral

Para ilustrar a diferença entre resolução espectral e densidade espectral utilizemos um exemplo. Considere o sinal $x[n] = cos(0.48\pi n) + cos(0.52\pi n)$. Calcula a DFT do sinal considerando

(a) um número de amostras $N = 10$ e preencha as outras 90 amostras com zero chegando a $N = 100$.

(b) um número de amostras N = 100 sem preenchimento com zero.

Código 3.5 - Código para solução dos dois itens anteriores. Observe que é usada a função DFT da biblioteca empregada neste apostila.

```
clc; clear;
n = [0:99];
Fs = 100;
x = cos(0.48*pi*n) + cos(0.52*pi*n);
%% calcula DFT com N=10 e plota
n1 = [0:9];
sinal1 = x(1:10);
Y1 = dft(sinal1, 10);
magY1 = abs(Y1(1:6));
N1 = 5;
m1= 0:5;
f1= m1*Fs/N1;
subplot(2,3,1); stem(sinal1); xlabel('n'); title('N=10')
subplot(2,3,4); stem(f1, magY1); xlabel('freq(Hz)');
%% calc DFT com N=10+90 zeros e plota (melhor resol espectro)
sinal2 = [x(1:10) zeros(1,90)]; %preencheu sinal que essencialmente eh o mesmo
Y2 = dft(sinal2,100);
magY2 = abs(Y2(1:50));
N2 = 50;
m2= 1:50;
f2= m2*Fs/N2;
subplot(2,3,2); stem(sinal2); xlabel('n'); title('N=10+90 zeros com boa resol. espec.')
subplot(2,3,5); stem(f2, magY2); xlabel('freq(Hz)');
%% calc DFT com N=100
sinal3 = x(1:100);
Y3 = dft(sinal3,100);
magY3 = abs(Y3(1:50));
N3 = 50;
m3= 1:50;
f3= m3*Fs/N3;
subplot(2,3,3); stem(sinal3); xlabel('n'); title('N=100 com boa densid. espec.')
subplot(2,3,6); stem(f3, magY3); xlabel('freq(Hz)');
```