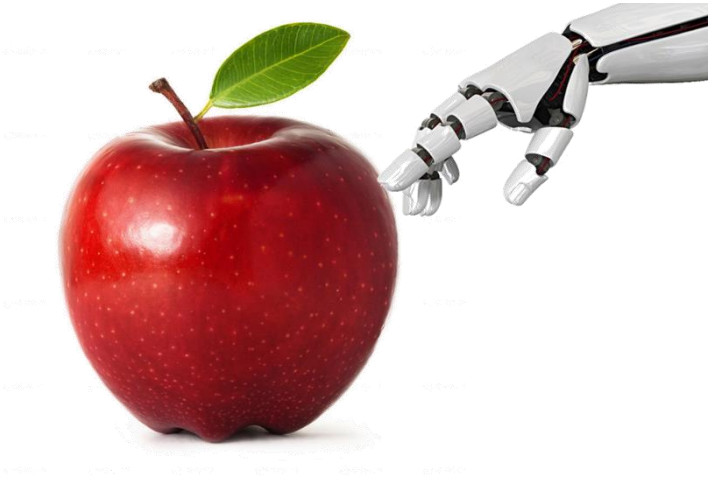
ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia Mecatrônica
Instrumentação
Profs. Larissa Driemeier/Marcilio Alves/Rafael T. Moura

Análise de Sinais usando MatLab

A aula de hoje será de estudo dirigido. Para isso você precisará do MATLAB®, ferramenta disponível na máquina virtual que você está logado.

Nas próximas páginas você encontrará dez exercícios, alguns resolvidos, outros parcialmente resolvidos e alguns você terá que encontrar a saída. Os exercícios têm o objetivo de fazer você entender um pouco mais sobre amostragem, leakage, windowing, filtros.



Não se preocupe em criar nenhum arquivo com explicações e respostas. **Somente as resoluções dos exercícios 8 e 9 devem ser enviadas por email (driemeie@usp.br) até as 00hs00 de amanhã. Coloque três arquivos ".m", zipados ou não. Quaisquer explicações necessárias, inclusive os nomes dos integrantes da dupla, devem estar nos comentários dos arquivos.** Todos os gráficos devem ter título, título dos eixos (com unidades) e legenda. A aula é também de estudo, faça suas anotações.

Ao analisar um exercício **com solução**, seja detalhista. Não deixe de entender o código. Isso será imprescindível para que você consiga fazer os exercícios **sem solução**.

A lista é longa e o tempo curto... Não se distraia com conversas paralelas.

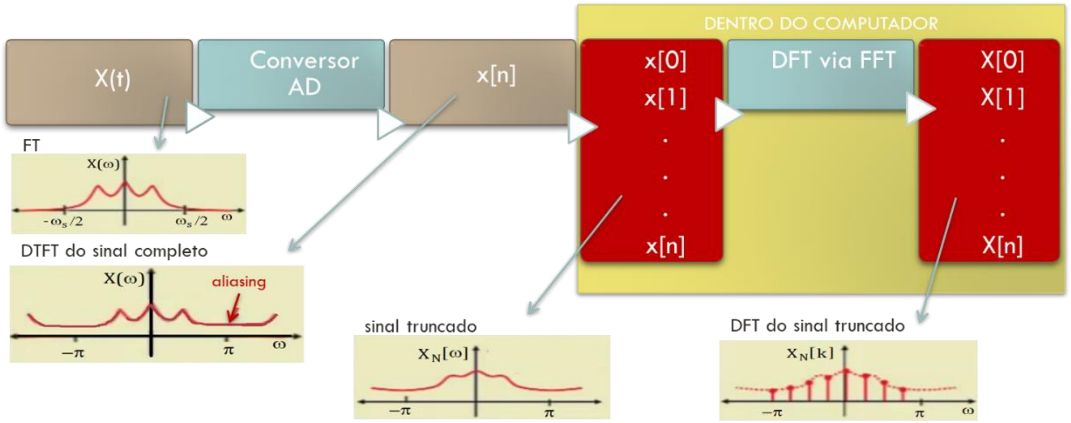
Boa aula!

Ferramentas matemáticas nos ensinam a fazer a FT (Transformada de Fourier) de um sinal contínuo e não periódico no tempo

A DTFT (Transformada de Fourier em Tempo Discreto) é a FT de um sinal amostrado em um intervalo de tempo infinito. Sua saída é periódica e contínua em frequência.

A DFT (Transformada de Fourier Discreta) pode ser vista como a versão de amostragem finita da DTFT. Ela é usada para calcular o espectro frequência de um sinal discreto no tempo usando o computador, já que os computadores só podem lidar com um número finito de valores. A DFT e a sua inversa estão implementadas no Matlab como fft and ifft.

Quais os erros que se comete entre a FT(o que queremos) e a DFT(o que medimos)? E como podemos evitar ou dimuniir esses erros?

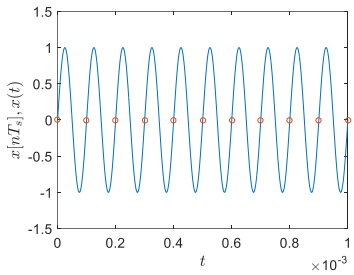


Exercício 01

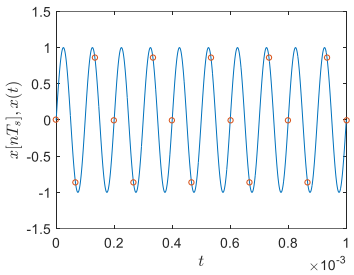
Vamos produzir uma senoide no MatLab e amostrá-la. Explore especialmente a influência da taxa de amostragem no resultado do sinal amostrado (vetor `sinal_sample`), comparando-o com o sinal contínuo (vetor `sinal`).

```
clear all; close all; clc
%% dados do sinal
f = 10000;%Freq entrada Hz
fs =15000;% Frequencia de amostragem Hz
%% gerar sinal
tempo = [0:1/(100*f):10/f];%Tempo amostral
sinal = sin(2*pi*f*tempo); % Geração onda senoidal
%% plotar sinal
plot(tempo,sinal)
hold;
%% sinal amostrado
Ts = 1/fs;
N=101;
n = [0:1:N-1];
t_sample = [0 : Ts : n(N)*Ts];
DigitalFrequency=2*pi*f/fs;
sinal_sample = sin (DigitalFrequency.*n);
plot(t_sample, sinal_sample,'o');
axis([0 10/f -1.5 1.5])
set(gca,'FontSize',16)
xlabel('$t$','Interpreter','LaTex','FontSize',18)
ylabel('$x[nT_s],x(t)$','Interpreter','LaTex','FontSize',18)
```

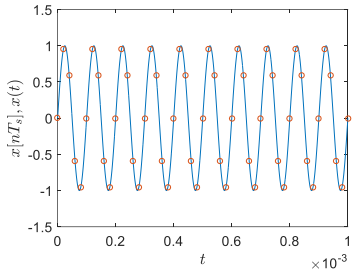
Nessas linhas você define o número de pontos amostrados `N`, período de amostragem `Ts` e, como consequência, o intervalo de tempo do sinal amostrado `(N-1)*Ts`. Mantendo `N` constante e aumentando `fs` você diminui o intervalo e vice versa.



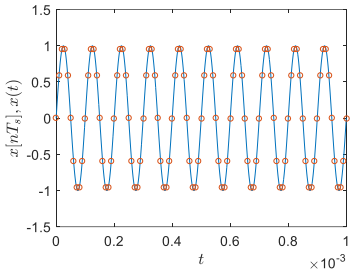
Sinal de 10 kHz com taxa de 10kHz



Sinal de 10 kHz com taxa de 15kHz



Sinal de 10 kHz com taxa de 50kHz



Sinal de 10 kHz com taxa de 100kHz

Um sinal analógico é definido por parâmetros de frequência e tempo. Para manter a analogia para sinais discretos, usa-se n (o número da amostra) como a unidade de tempo discreta. A frequência, portanto, terá unidade de *radianos/amostra*, já que a unidade de tempo passou a ser *número da amostra*. A frequência de um sinal discreto é diferente da frequência tradicional de um sinal contínuo. Chama-se **Frequência Digital** e usa-se, em geral, a letra grega Ω para representá-la,

$$\Omega = \frac{2\pi f}{f_s},$$

Onde f é a frequência do sinal e f_s a frequência de amostragem.

Exercício 02

FFT (FAST FOURIER TRANSFORM) é simplesmente uma forma mais rápida de calcular a DFT: A FFT utiliza alguns algoritmos que permitem reduzir o número de operações para $N\log_2 N$. Para utilizar a FFT, é

necessário que o número de amostras seja uma potência de 2 – a FFT é executada mais rapidamente com um vetor cujo comprimento é uma potência de 2.

Para $N = 1000$, $DFT = 1\,000\,000$, $FFT = 10\,000$ operações

A FFT no Matlab: Matlab permite o cálculo fácil da DFT via FFT. Se tivermos um vetor A, de n elementos,

$$>> FFTdeA = fft(A);$$

Agora vamos utilizar a FFT. No primeiro exemplo, aprenderemos a analisar a resposta da FFT. No segundo, apenas iremos ilustrar o papel da função `fftshift`. Por último, utilizaremos algumas propriedades bastante conhecidas e importantes da transformada. São elas:

- **Translação no domínio do tempo**: Transladar um sinal no domínio do tempo faz com que a transformada de Fourier seja multiplicada por uma exponencial complexa.

$$x(t - t_0) \underset{FT}{\rightarrow} e^{-j\omega t_0} X(\omega)$$

- **Convolução:** A transformada de Fourier da convolução de dois sinais é o produto das transformadas desses sinais.

$$y(t) = h(t) * x(t) \longrightarrow Y(\omega) = H(\omega)X(\omega)$$

- **Linearidade:** A propriedade de linearidade ou de superposição de efeitos estabelece que combinações lineares no domínio do tempo correspondem a combinações lineares no domínio da frequência.

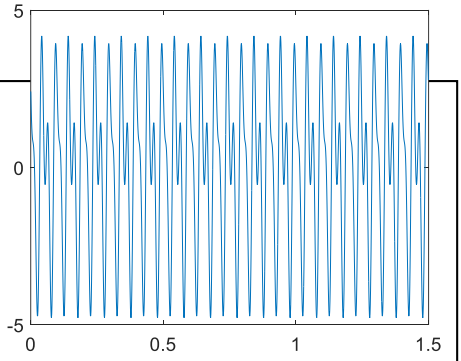
$$ax(t) + by(t) \longrightarrow aX(\omega) + bY(\omega)$$

i. Como exemplo, considera-se uma função cosseno,

$$x(t) = 3\cos(2\pi f_1 t + 0.2) + \cos(2\pi f_2 t - 0.3) + 2\cos(2\pi f_3 t + 2.4)$$

Considere a frequência de amostragem $f_s = 1000$ amostras/segundo, amostrando-se em um período total de 1.5 ms.

Com esse exemplo, vamos entender a FFT...




```
clear all; close all; clc

%% Senoides e aliasing
% Taxa de amostagem
fs=1000; Ts=1/fs;
% Frequencias, em Hz, do sinal
f1=20;  DigFreq1=2*pi*f1/fs;
f2=30;  DigFreq2=2*pi*f2/fs;
f3=40;  DigFreq3=2*pi*f3/fs;
%Plot
N=1500;
n = 0:1:N-1;
t_sample = [0 : Ts : (N-1)*Ts];
x=3.*cos(DigFreq1.*n+0.2)+cos(DigFreq2.*n-0.3)+2.*cos(DigFreq3.*n+2.4);
figure;
plot(t sample,x)
```

```
%% FFT do sinal
X=fft(x);
```

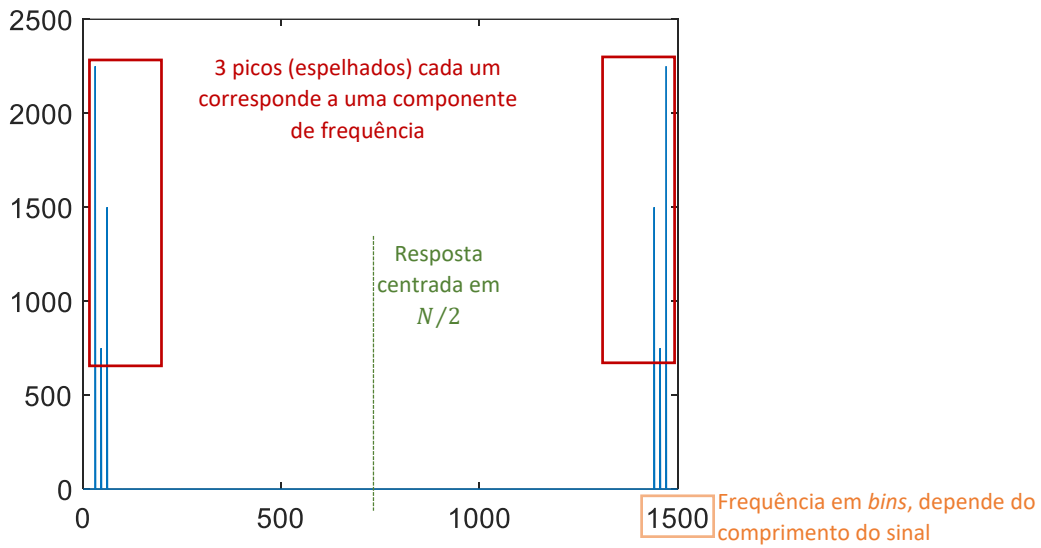
O comprimento das variáveis `x` (dominio do tempo) e `X` (dominio da frequência) são iguais. Verifique!

```
% os valores de x são valores reais, enquanto os valores de X são
% complexos. Veja alguns exemplos:
x(2:6)
X(30:34)
```

```
ans =

    2.0717    1.7535    1.4794    1.2572    1.0884


ans =

   1.0e+03 *

  -0.0000 - 0.0000i   2.2051 + 0.4470i   0.0000 - 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i
```

```
%X são valores complexos porque representam magnitude e fase!!!!
%Magnitude
figure;
X_mag=abs(X);
X_mag(30:34)
plot(X_mag)
set(gca,'FontSize',16)
```

```
ans =

   1.0e+03 *

    0.0000    2.2500    0.0000    0.0000    0.0000
```



Porque os valores espelhados? Sabe-se que a transformada de Fourier de um sinal discreto é um sinal periódico no domínio da frequência. O Matlab, obviamente, sempre trabalha em domínio discreto, de modo que, usando a função `fft`, você sempre obtém sinais de frequência periódicos.

Matlab considera apenas um período desse sinal (portanto, apenas as primeiras `N` amostras) e, para as convenções adotadas, como a primeira amostra vetorial $x(t)$ corresponde à amostra no tempo $t = 0$, então a primeira amostra de sinal $X(f)$ corresponde à amostra em $f = 0$. Porém, o MatLab não permite que uma *array* tenha índices negativos ou nulo. Por esta razão, após uma `fft`, o sinal obtido é aquele na Figura, onde as amostras de frequência negativa são rebatidas ao fundo, na segunda metade do vetor. Para ver o espectro de frequências matematicamente compatível com a entrada, deve-se utilizar a `ifft`, que será vista mais para frente.

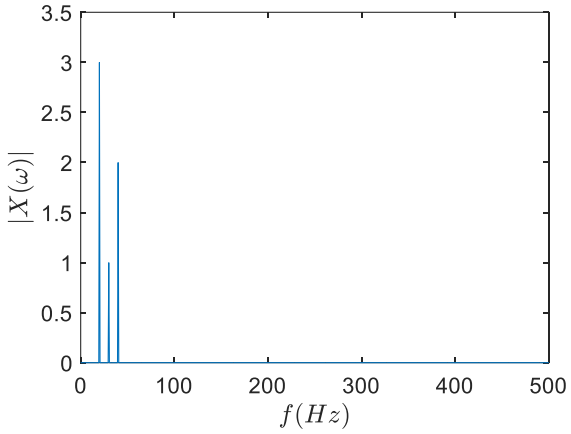
Além disso, faz sentido analisar somente metade do sinal (lembre-se da frequência de Nyquist!!!).

O eixo das abscissas corresponde a frequências, mas seu valor depende do comprimento do sinal. Os valores de X variam 0-1499, que é o tamanho do sinal criado por nós. Essas são chamadas de *bin frequencies*. Essa frequência em *bins* pode ser facilmente convertida em frequência em Hz,

$$f = F_{bins}\frac{f_s}{N}$$

No eixo das ordenadas, a magnitude está multiplicada também por $N/2$.

```
figure;
Fbin=0:1:N-1
plot((fs/N).*Fbin,X_mag/(N/2))
xlim([0 fs/2])
set(gca,'FontSize',16)
xlabel('$f(Hz)$','Interpreter','LaTex','FontSize',18)
ylabel('$|X(\omega)|$','Interpreter','LaTex','FontSize',18)
```



A fase do sinal também é uma informação importante, que pode ser tirada de X.

```
%Fase
X_fase=angle(X);
%se você verificar a fase em cada componente verá que coincide com a
% fase do sinal
%
X_fase(31)
X_fase(46)
X_fase(61)
```

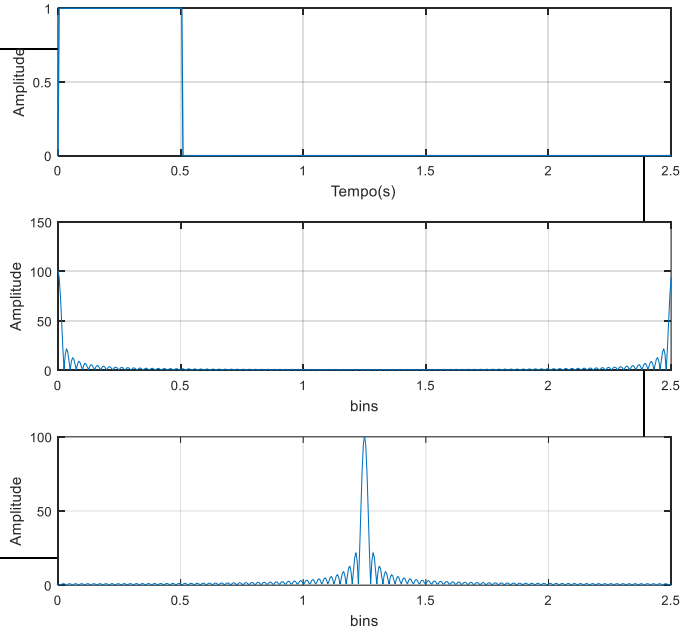
De onde saíram esses números, 31, 46 e 61? ☺

```
ans =

    0.2000


ans =

   -0.3000


ans =

    2.4000
```

ii. Apenas para ilustrar o comando `fftshift`, verifique as imagens abaixo,

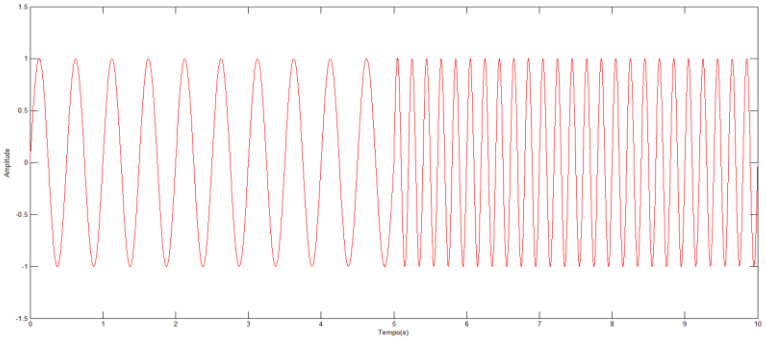
```
clear all; close all; clc
%%
Tmax=0.5;          % Intervalo de duração de cada onda
fs=200;                        % Frequência de amostragem
t=[0:(1/fs):Tmax+2];             % Amostragem no tempo

L=length(t);

% Pulso retangular
T0=0;                      % Instante de início do pulso retangular
T=Tmax;               % Duração do pulso retangular

% Definição do início e final da janela
L_ini=length([0:(1/fs):T0]);
L_pulse=length([0:(1/fs):T]);
L_fin=L-L_ini-L_pulse;

win = rectwin(L_pulse);
wRect1 = [zeros(L_ini,1); win;zeros(L_fin,1)]';
figure(1)
```

```
subplot(311)
plot(t,wRect1,'LineWidth',1)
grid on
xlabel('Tempo(s)');
ylabel('Amplitude');


X=fft(wRect1,length(t));
subplot(312)
plot(t,abs(X))
grid on
xlabel('bins');
ylabel('Amplitude');

Y=fftshift(X);
subplot(313)
plot(t,abs(Y))
ylim([0 100])
grid on
xlabel('bins');
ylabel('Amplitude');
```



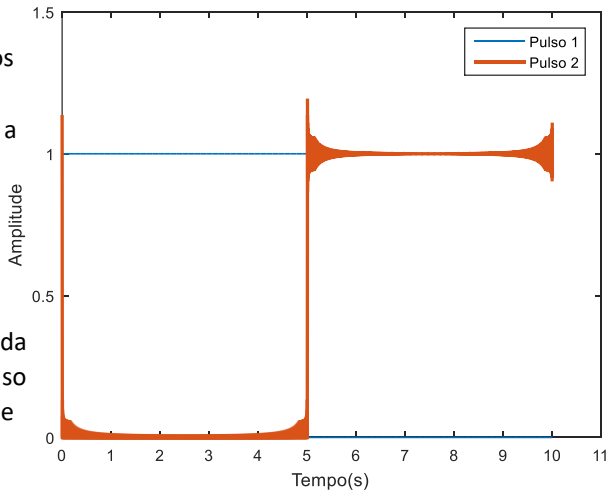
Portanto, a função `shiftfft` permite que você reordene as frequências e represente o sinal transformado centralizado no vetor de frequência, como esperado matematicamente.

iii. SEM USAR A FUNÇÃO LOOP, crie um sinal com frequência de aquisição de 400Hz de 0 a 10 segundos, no qual entre 0 e 5 segundos há um seno de 2Hz e entre 5 e 10 segundos um seno de 5Hz.



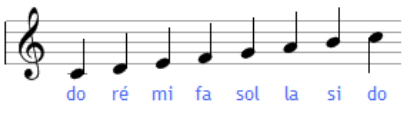
Nossa sequência de passos é:

- Cria-se os sinais senoidais e um pulso retangular;
- Utilizando a *propriedade de time shift* de um pulso retangular, desloca-se o sinal de 5s. Com a transformada inversa da FFT (`ifft`), cria-se um segundo pulso deslocado no tempo. Dessa forma, tem-se os pulsos definidos na figura ao lado;
- Crie a sequência de sinais utilizando a propriedade da linearidade;
- OPTATIVO: Convolução: Faça a multiplicação do sinal senoidal e o pulso retangular no tempo e faça a FFT. Compare com a FFT obtida da convolução das respostas da FFT da função seno com a FFT do pulso retangular. O script MatLab para esse caso está no anexo 1.



```
clear all;close all; clc

intervalo=5;          % Intervalo de duração de cada onda
numOndas=2;
Tmax=intervalo*numOndas;
Amplit=1;
fs=400;                         % Frequência de amostragem
t=[0:(1/fs):Tmax];              % Amostragem no tempo
f1=2;                           % Frequência do sinal senoidal 1
f2=5;                           % Frequência do sinal senoidal 2
seno1=Amplit*sin(2*pi*f1*t);    % Geração da onda senoidal
seno2=Amplit*sin(2*pi*f2*t);    % Geração da onda senoidal

L=length(t);

% Pulso retangular
T0=0;                       % Instante de início do pulso retangular
T=intervalo;                % Duração do pulso retangular

% Definição do início e final da janela
L_ini=length([0:(1/fs):T0]);
L_pulse=length([0:(1/fs):T]);
L_fin=L-L_ini-L_pulse;

win = rectwin(L_pulse);
wRect1 = [zeros(L_ini,1); win;zeros(L_fin,1)]';

freq=linspace(-1,1,L)*(fs/2);
F=fft(wRect1);
Fshifted=(fftshift(F));

theta=(2*pi*j*freq.*intervalo);
time_shifted=Fshifted.*exp(theta);

wRect2=ifft(ifftshift(time_shifted));

figure(1)
plot(t,wRect1,'LineWidth',1)
hold on
plot(t,wRect2,'LineWidth',2)
axis([0 11 0 1.5])

OndaFinal=seno1.*wRect1+seno2.*wRect2;

figure(3)
plot(t,OndaFinal,'r');
xlabel('Tempo(s)');
ylabel('Amplitude');
```

Exercício 3



Os arquivos '.wav' na pasta *NotasMusicais* deveriam estar numeradas em ordem crescente de frequência, que coincide com as notas musicais **Dó(261.63 Hz), Ré(293.66 Hz), Mi(329.63 Hz), Fá(349.23 Hz), Sol (392 Hz), Lá (440 Hz), Si (493.88 Hz)**. Alguém não fez o trabalho de forma adequada. Sua tarefa consiste em criar um programa Matlab® que reconheça a sequência de notas, de forma a completar a tabela:

Nota musical	Nome do arquivo
Dó	
Ré	
Mi	
Fá	
Sol	
Lá	
Si	

Use os comandos `audiowrite` e, se quiser ouvir, `audioread`, do MATLAB.

```
%Exercício resolvido em 2017 pelos alunos:
%Alexandre Hoppe Inoue e Luiz Motta da Silva
```

```
clear

identificaNota('sine1.wav',1);
identificaNota('sine2.wav',2);
identificaNota('sine3.wav',3);
identificaNota('sine4.wav',4);
identificaNota('sine5.wav',5);
identificaNota('sine6.wav',6);
identificaNota('sine7.wav',7);

function [ ] = identificaNota( tituloNota , numeroDaNota)

  %Amostragem sinal seno1
  [x1, Fs] = audioread(tituloNota);
  X1 = fft(x1);
  X1_mag = abs(X1);
  Y1 = fftshift(X1);
  Y1_mag = abs(Y1);
  N= length(Y1); %tamanho do vetor
  Fbin = 0:1:N-1;


  %titulo do sinal e mostrar dados obtidos
  formatSpec = 'sine%d.wav (f = %d)';
  subplot(3,3,numeroDaNota);
  plot((Fs/N).*Fbin,X1_mag/(N/2));
  xlim([0,Fs/2]);
  set(gca,'FontSize',10);
  indice = find(X1_mag==max(X1_mag));
  str = sprintf(formatSpec, numeroDaNota, indice(1));
  title(str);
  hold on;

end
```

Exercício 4

O sinal de frequência 1Hz

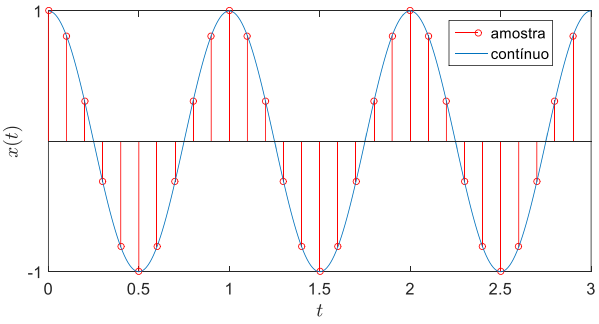
$$x = \cos(2\pi t)$$

é amostrado a uma taxa de 10Hz, pelo período total de 3s (isto é, foram amostrados 3 períodos do sinal).

Serão analisadas três situações:

a. **Truque do Zero padding**:

Depois de coletados os N pontos de amostragem, colocamos alguns zeros adicionais ao final da lista para *enganar* o processo (como são zeros não alteram os valores da Transformada de Fourier Discreta). Supondo que tenhamos N_z pontos, incluindo os zeros que adicionamos... O espaçamento entre os pontos da DFT será de $2\pi/N_z$, que é menor que $2\pi/N$.



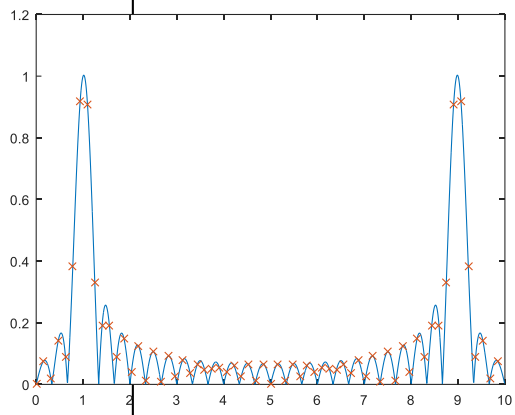
No MatLab, $X = fft(x, N_p)$

Isto é, a resposta no domínio da frequência é sempre a mesma curva, porém, o truque do zero padding permite que se observe mais pontos dessa curva. Vamos verificar no exemplo abaixo.

Entendendo diferentes valores de `Np` na fft: `fft(x,Np)` - use os valores $N_p = 30, 64, 128, 256$ **mantendo a frequência de amostragem**. Avalie o resultado comparando as respostas. Para comparar as respostas, considere a resposta com $N_p = 1024$ como contínua e faça quatro gráficos, comparando a resposta contínua com aquelas com diferentes valores de N_p . Abaixo, como ilustração, o gráfico de

comparação para $N_p = 64$. Responda: qual a influência de acrescentar zeros no final do sinal amostrado (*truque do zero-padding*)?

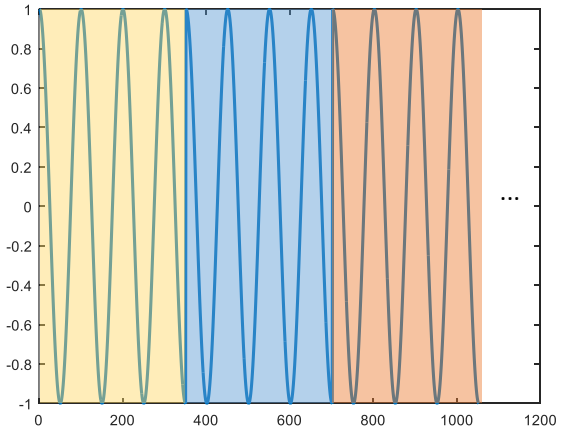
```
clear all; close all; clc

%% Senoide e fft - zero padding
N=30;
fs=10;
n = [0:N-1];
t=0:0.01:N/fs;
x_cont= cos(2*pi*t);
x = cos(2*pi*n/10);

N2 = 64;
n2=0:1:N2-1;
N5=2048;
n5=0:1:N5-1;
 %%%
 X2 = abs(fft(x,N2))./(length(x)/2);
 X5 = abs(fft(x,N5))./(length(x)/2);
figure;
plot((fs/2)/(N5/2).*n5,X5)
hold on
plot((fs/2)/(N2/2).*n2,X2,'x')
```



b. **Número de períodos de amostragem**: Mantendo o valor de $N_p = 1024$ constante, plote a fft para diferentes números de períodos amostrados. Você amostrou 3 períodos no item (a). Compare essa resposta com a resposta obtida amostrando 6, 9 e 12 períodos. Existe uma maneira rápida de aumentar o número de períodos no MatLab,

```
%% Aumentando o número de períodos
x1 = cos(2*pi*n/10); % 3 periodos
x2 = [x1 x1]; % 6 periodos
x3 = [x1 x1 x1]; % 9 periodos
x4 = [x1 x1 x1 x1]; % 12 periodos
```

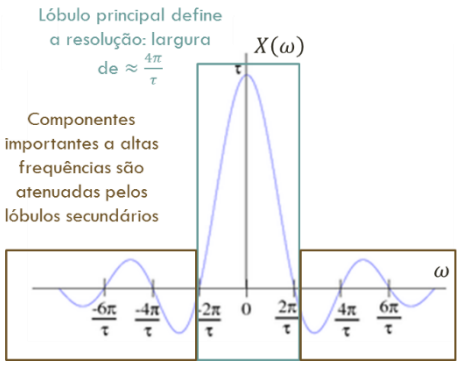
c. Se a amostragem do sinal não cobrir ciclos inteiros, o espectro calculado irá apresentar o fenômeno de vazamento espectral ("leakage"), que se traduz no seguinte erro básico: as amplitudes calculadas sofrem um achatamento e um espalhamento em torno das valores espectrais originais. Veja na figura abaixo a ilustração de como o sinal "é visto" pela transformada. Discuta o fenômeno de *leakage*, definindo períodos não completos de amostragem do exemplo.




```
%% Senoide e fft - leakage
N=35;
fs=10;
n = [0:N-1];
t=0:0.01:N/fs;
x_cont= cos(2*pi*t);
x = cos(2*pi*n/10);
%% Aumentando o número de períodos
x1 = cos(2*pi*n/10); % 3 periodos
x2 = [x1 x1]; % 6 periodos
x3 = [x1 x1 x1]; % 9 periodos
x4 = [x1 x1 x1 x1]; % 12 periodos
x_cont3=[x_cont x_cont x_cont];
figure;
plot(x_cont3,'Linewidth',2);
```

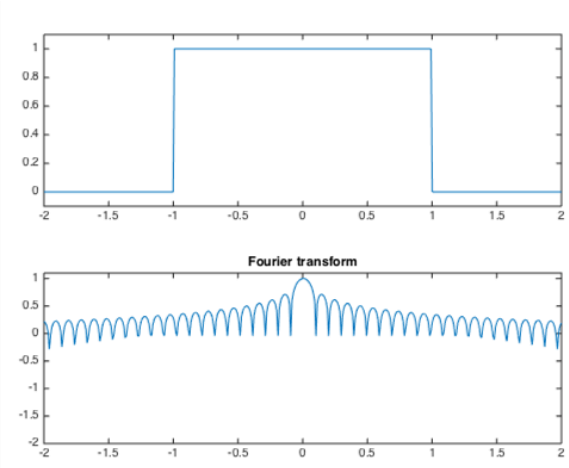
Exercício 5

Uma solução para o *leakage* é o janelamento (windowing). Diferentes tipos de janelas podem ser utilizados. A mais simples é a retangular, que é igual a 1 durante o intervalo de tempo que se pretende analisar, e igual a zero fora desse intervalo. Foi essa janela que usamos no exercício 2. Já aprendemos que a janela retangular no domínio do tempo, resulta em uma função *sinc* no domínio da frequência (figura ao lado).



Quanto mais estreito for o lóbulo principal, melhor a resolução frequencial. No entanto, quanto mais estreito o lóbulo principal, mais altos se tornam os lóbulos laterais, que aparecem como ruído de fundo no espectrograma.

A janela retangular fornece boa resolução frequencial, mas os lóbulos laterais são muito altos, resultando em muito ruído de fundo. Além disso, ocorre o fenômeno de leakage, discutido anteriormente, se os períodos de amostragem não forem completos.



Existem várias janelas, além da retangular, cujo objetivo é diminuir a influência dos extremos da amostragem. Veja, por exemplo, a janela de Hanning (também chamada de Hann), em homenagem ao vienense Julius Ferdinand von Hann (1839-1921). A janela Hanning é modelada como:

$$w(k) = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi k}{M-1}\right)\right] \qquad k = 0, \dots, M-1$$

No MatLab,

$w = hanning(M);$ ou $w = hann(M);$

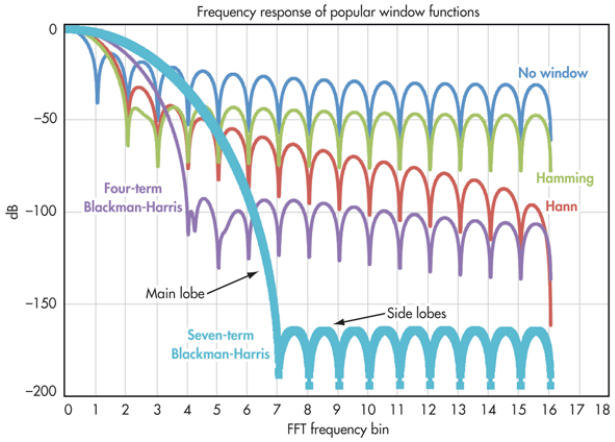
O janelamento Hamming começa em 0,08, sobe para 1 no meio do período, e depois cai novamente até 0,08 no final.

$$w(k) = 0{,}54 - 0{,}46\cos\left(\frac{2\pi k}{M-1}\right), \quad k = 0, \dots, M-1$$

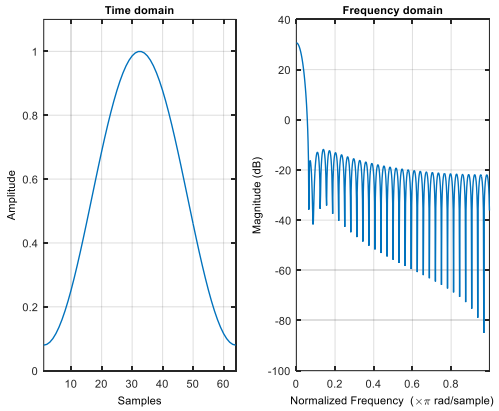
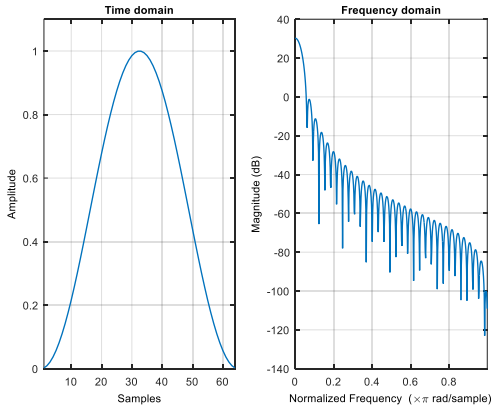
Ou seja, os valores iniciais e finais da amostragem são atenuados. No MatLab,

$$w = hamming(M);$$

A figura abaixo ilustra o comportamento de algumas janelas conhecidas.



Use os seguintes comandos para entender as janelas Hamming e Hanning,

`L = 64;` `wvtool(hamming(L))`	`L = 64;` `wvtool(hanning(L))`
	

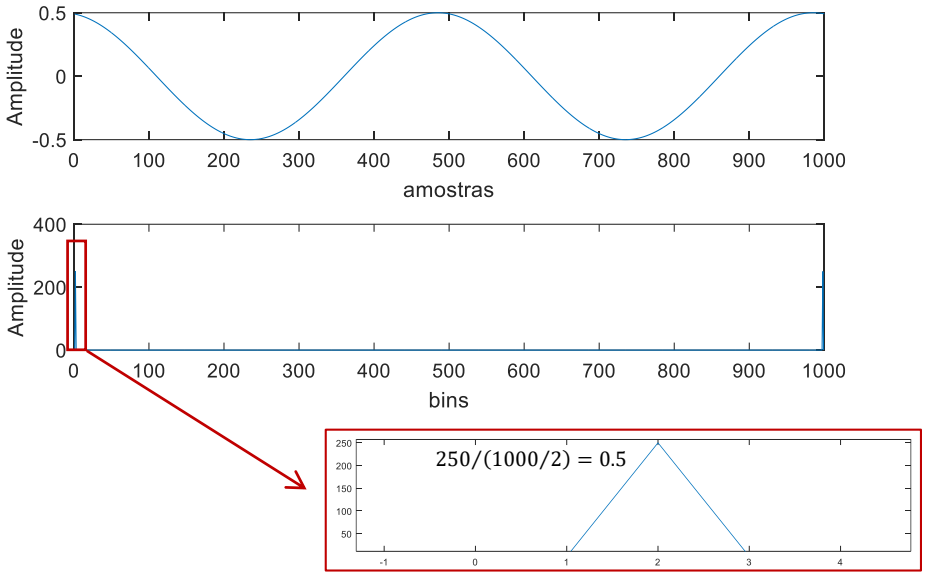
Exercício 6

Considere o sinal,

$$x = 0.5\cos(2\pi f_1 n + 0.2)$$

Segundo o código MATLAB abaixo,

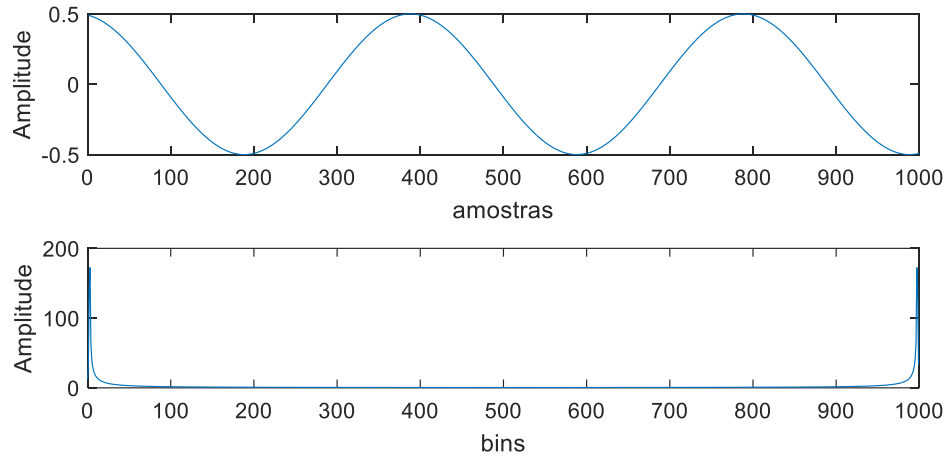
```
fs=1000;
t=0:1/fs:1-1/fs;
f1=2;
x1=0.5*cos(2*pi*f1*t+0.2);
subplot(2,1,1)
plot(x1)
set(gca,'FontSize',14)
xlabel('amostras','FontSize',16)
ylabel('Amplitude','FontSize',16)
%
X1=fft(x1);
subplot(2,1,2)
plot([0:length(X1)-1],abs(X1))
set(gca,'FontSize',14)
xlabel('bins','FontSize',16)
ylabel('Amplitude','FontSize',16)
```

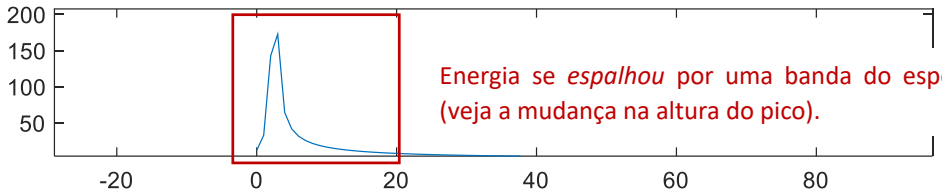


Pico em 2.0 bins (coincide com $2.0 * \frac{500}{500} = 2.0\ Hz$)

Se mudarmos a frequência de 2 para 2.5Hz, a amostragem não será de período inteiro e a resposta muda radicalmente.

```
fs=1000;
t=0:1/fs:1-1/fs;
f1=2.5;
x1=0.5*cos(2*pi*f1*t+0.2);
subplot(2,1,1)
plot(x1)
set(gca,'FontSize',14)
xlabel('Amostras','FontSize',16)
ylabel('Amplitude','FontSize',16)
%
X1=fft(x1);
subplot(2,1,2)
plot([0:length(X1)-1],abs(X1))
set(gca,'FontSize',14)
xlabel('bins','FontSize',16)
ylabel('Amplitude','FontSize',16)
```

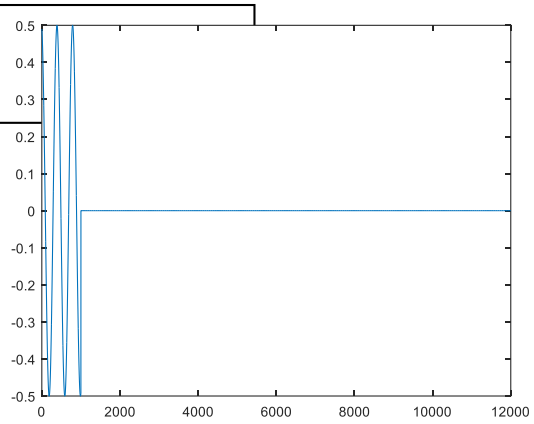


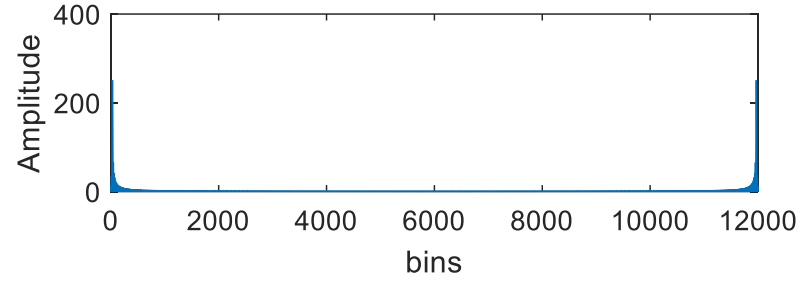


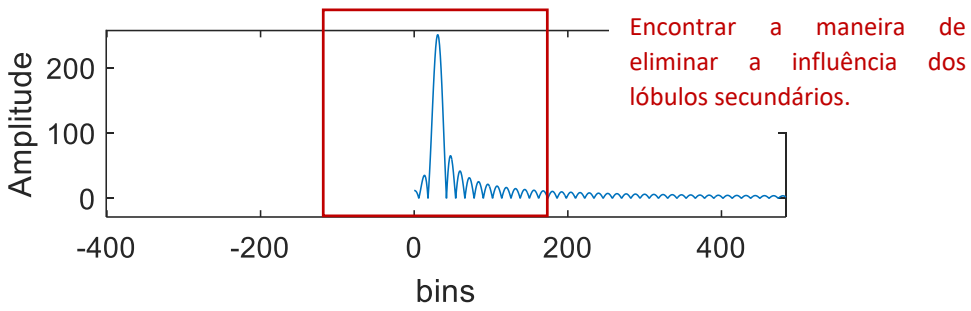
Energia se *espalhou* por uma banda do espectro (veja a mudança na altura do pico).

O uso do truque *zero padding* já entendemos que, nesse caso, não resolve. Veja que foi adicionado um número de zeros ao final do sinal (sem usar a variável N_p da fft) para termos controle do sinal.

```
x1=[x1 zeros(1,11000)];
X1=fft(x1);
subplot(2,1,2)
plot([0:length(X1)-1],abs(X1))
```



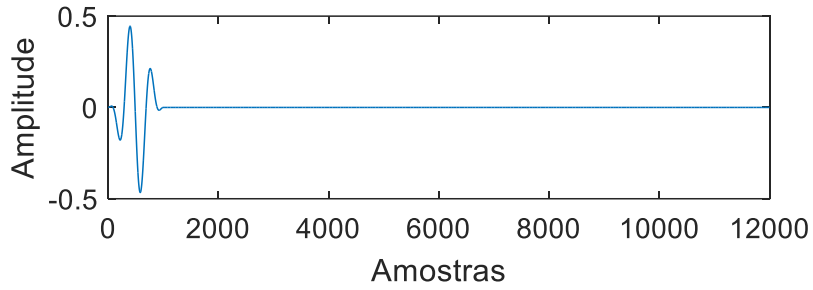


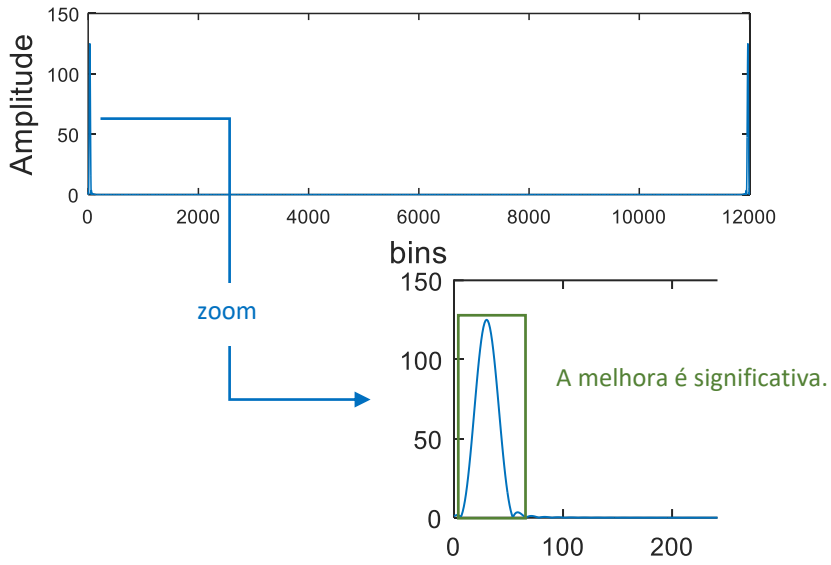


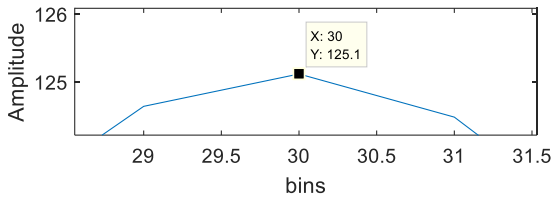
Encontrar a maneira de eliminar a influência dos lóbulos secundários.

Se usarmos o janelamento, antes do truque do zero-padding,

```
L=length(x1)
x1=x1.*hanning(L)';
x1=[x1 zeros(1,11000)];
X1=fft(x1);
subplot(2,1,2)
plot([0:length(X1)-1],abs(X1))
```







Para janela Hanning, a regra é que a magnitude é o valor dividido por $N/4$, e a fase é o valor encontrado. A frequência segue a regra já definida:

```
angle(X1(31))
abs(X1(31))/(N/4)
30*fs/N

ans =

    0.2000

ans =

    0.5005

ans =

    2.5000
```

Exercício 7

Analise o sinal sinusoidal composto de três frequências,

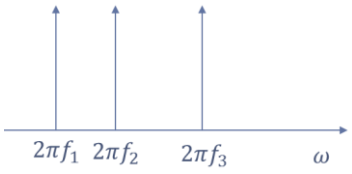
$$x = \cos(2\pi f_1 n) + \cos(2\pi f_2 n T_s) + \cos(2\pi f_3 n T_s)$$

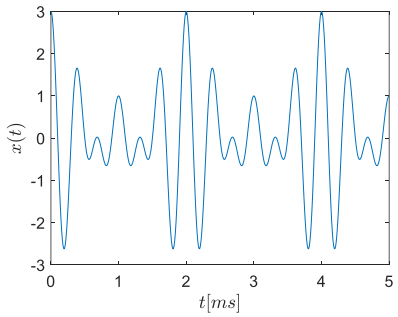
Onde,

$f_1 = 2000$ Hz

$f_2 = 2500$ Hz

$f_3 = 3000$ Hz





$f_s = 1000$ Hz, onde f_s é a taxa de amostragem, e $T_s = \frac{1}{f_s}$ é o período de amostragem.

Nesse exercício você deve definir uma frequência de amostragem constante, f_s, e utilizar um número diferente de amostras: $N = 10, 20, 40\ e\ 100$.

Utilize sempre as janelas retangular e hamming, conforme parte do código abaixo.

Analise a resposta das duas janelas a medida em que o período de amostragem (isto é, N) aumenta.

Discuta os resultados.

```
fs=10000; T=1/fs;  % Frequencia e período de amostragem
%Frequencias do sinal
f1=2e3;
f2=2.5e3;
f3=3e3;
% Frequencias para cálculo da DFT
w = 0:pi/1024:pi-pi/1024; %comprimento de w é usado em zero padding
figure(1)

L = 10;            % Define Numero de amostras
n=0:L-1;
x=cos(2*pi*f1*T.*n)+cos(2*pi*f2*T.*n)+cos(2*pi*f3*T.*n); % Calcular x(n)*w(n)
X=fft(x,length(w));
subplot(221)
plot(w./pi,abs(X))
axis([0 1 0 50])
title('Rectangular Window -- L = 10')

 h = hamming(L);       % Hamming window
 xh=x.*h';
 Xh=fft(xh,length(w));
 subplot(222)
 plot(w./pi,abs(Xh))
 axis([0 1 0 50])
 title('Hamming Window -- L = 10')
```

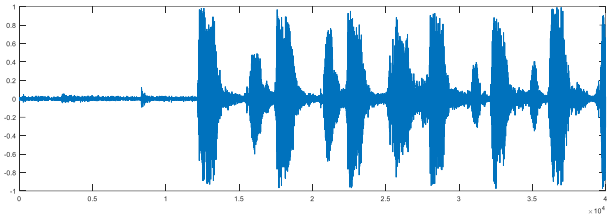
Exercício 8

Para detalhar outro assunto importante, vamos testar alguns filtros.

Inicialmente, vamos fazer um exemplo filtro de média móvel (*moving average filter).* É feita uma média de um número `M` de pontos do sinal da entrada `x[i]`, para produzir cada ponto do sinal de saída `y[i]`:

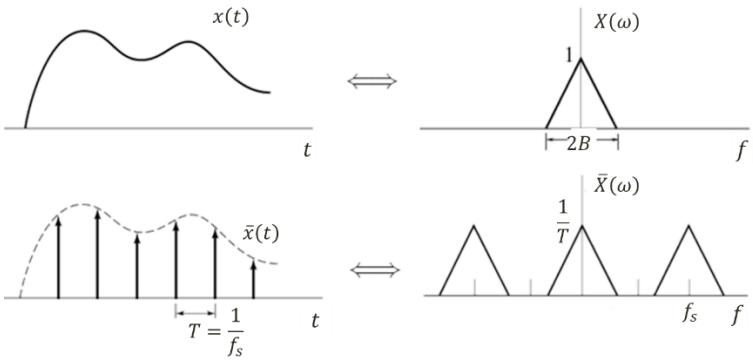
$$y[i] = \frac{1}{M}\sum_{j=1}^{m} x[i+j]$$

Grave sua voz por 5s e passe um filtro de média móvel para diferentes valores de `M` e reproduza a resposta.



```
clc; clear all; close all;
%
recObj = audiorecorder(44100, 16, 2);
get(recObj)
%Grave uma amostra de 5 segundos de sua voz com o microfone
recObj = audiorecorder;
disp('Start speaking.')
recordblocking(recObj, 5);
disp('End of Recording.');

% Escute sua gravacao
play(recObj);

% Armazene em um vetor em double-precision
myRecording = getaudiodata(recObj);

% Plote o sinal
plot(myRecording);

%% caso esteja sem microfone, basta usar uma função randômica
% % t=[0:0.001:100];
% % noise=randn(1,length(t));
% % figure;
% % plot(t,noise)


figure;
for i=2:length(myRecording)-1
    y(i)=(myRecording(i+1)+myRecording(i)+myRecording(i-1))/3;
end
figure;
plot(y)
soundsc(y)
```

Outro ponto importante para se tratar de filtro, é o uso de filtro anti aliasing. A figura abaixo ilustra a diferença entre a Transformada de Fourier de um sinal discreto e de um sinal contínuo. Isto é, quando o sinal é amostrado, seu espectro é replicado de acordo com a frequência de amostragem f_s.



$$\bar{x}(t) = \sum_k x(t)\,\delta(t - kT) \qquad \bar{X}(\omega) = \frac{1}{T} X(\omega - n\omega_s)$$

Por isso, o diz o teorema de Nyquist que:

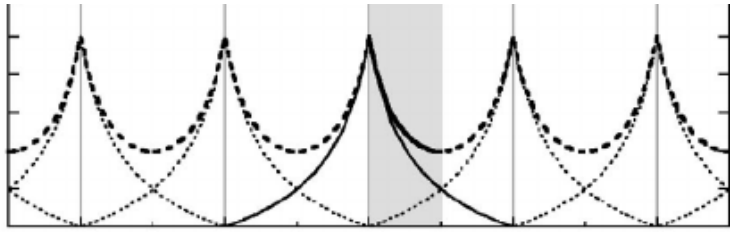
Se um sinal analógico $x(t)$ tem banda limitada, ou seja, se a frequência mais elevada do sinal é B ou seja,

$X(\omega) = 0$ para $|f| > B$,

então, é suficiente uma amostragem a qualquer taxa

$$f_s > 2B$$

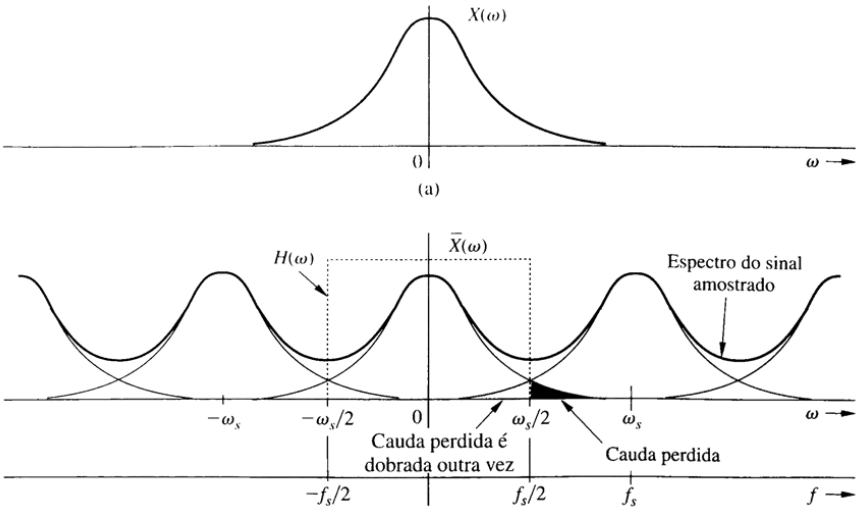
Porém, na maioria das vezes, não sabemos a priori, quais as frequências de nosso sinal. Além disso, muitos sinais não tem largura de banda finita ou não conhecemos. Dessa forma, existe uma grande chance de ocorrer sobreposição nas réplicas da frequência... o que impossibilitaria de reconstruir o sinal corretamente.



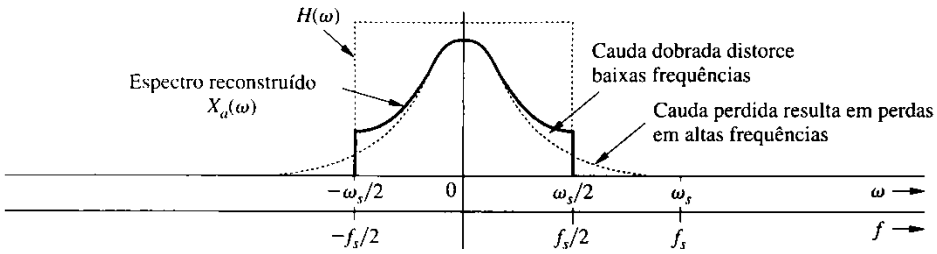
Quando há sobreposição de réplicas, diz-se que ocorreu *aliasing.*

Portanto, as frequências altas devem ser eliminadas **ANTES** da amostragem do sinal.

Como??? Emprega-se um filtro *passa-baixa* com frequência de corte $f_s/2$. Esse filtro é chamado de filtro *anti-aliasing.* O espectro das componentes de baixa frequência permanece intacto.

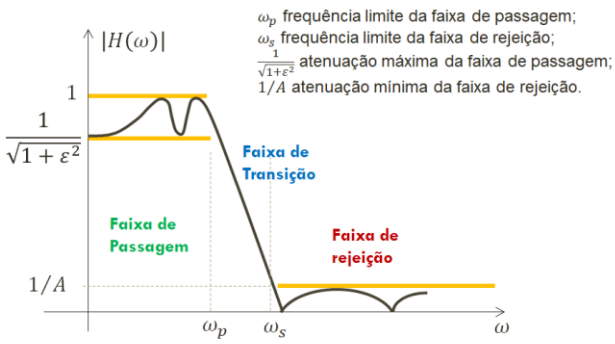


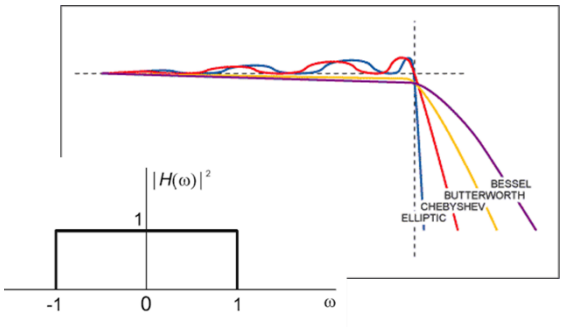
Como perdemos as componentes de alta frequência, determina-se a frequência de corte com base nas frequências de interesse do sinal e na frequência de amostragem. Para extrair corretamente a informação fundamental do sinal analisado é necessário selecionar as frequências de interesse que compõe esse sinal.



Como fazer isso?

Com um **FILTRO**. Filtros são SLIT capazes de modificar as características dos sinais de entrada de tal modo que apenas uma parcela específica dos seus componentes de frequência chega à saída do filtro. A resposta em frequência do filtro é caracterizada por **uma faixa de passagem** e **uma faixa de rejeição**, separadas por uma **faixa de transição ou faixa de guarda.**





Filtro passa baixa Butterworth

Para o filtro Butterworth, define-se

$$H(\omega) = \frac{1}{\sqrt{1+\varepsilon\left(\frac{\omega}{\omega_c}\right)^{2N}}}$$

Função `Buttord`

```
[N,Wc]=buttord(Wp,Ws,Rp,Rs)
```

`Wp` frequência limite da banda de passagem (em rad/s);
`Ws` frequência limite da banda de rejeição (em rad/s);
`Rp` máxima atenuação da banda de passagem (em dB, em `Wp`);
`Rs` mínima atenuação na banda de rejeição (em dB, em `Ws`)
OBS: O comando `buttord` define `Wc` usando a especificação de banda de rejeição e, possivelmente, deixando uma margem de segurança para banda de passagem.

ou função `butter`

```
[a,b]=butter(n,Wn)
```

retorna os coeficientes da função de transferência de um filtro passa-baixa de ordem `n` do tipo Butterworth com freq de corte normalizada `Wn`.

Baixe do STOA o som com nome de *Bach44100_10s.wav* (10s de Bach para você). O código abaixo ensina a adquirir o sinal a frequências mais baixas. Verifique a presença de aliasing no som quando a taxa de aquisição diminui.

```
clear all
% [Dados, FDados, Filtrados, Num, Frequencia]
% Filtro: Butterworth de segunda ordem a um quarto da frequencia
% original (ou metade da frequencia de reamostragem)

% audioread('MyFile') - lê o arquivo MyFile e retorna os dados amostrados 'x'
% e a frequencia de amostragem 'Fs'(=44100Hz)
[x,Fs] = audioread('Bach44100_10s.wav');
% audiowrite('MyFile',x,Fs) - escreve no arquivo 'MyFile' os dados 'x'
% à frequencia de amostragem 'Fs'(=44100Hz)
audiowrite('orig.wav',x,Fs);
```

```
% armazenando os dados amostrados x na frequencia original na primeira
% coluna da matriz 'Dados'
Dados(:,1)=x;

% O primeiro valor do vetor 'Frequencia' é a frequência padrão de aquisição
% de um sinal de áudio Fs=44100 Hz (Os sons audíveis pelo ouvido humano têm
% uma frequência entre 20Hz e 20kHz.)
Frequencia(1)=uint32(Fs);

 for n=2:5
  Frequencia(n) = Frequencia(n - 1)/2;

  y= downsample(x,2^(n-1));
  Y=ifft(fft(y));
  Dados(1:length(Y),n)=Y;
 end

 for n=2:5
    %filename = sprintf('%s_%d','filename',n)
    myFile = ['path' int2str(n) '.wav'];
    audiowrite(myFile,Dados(:,n),Frequencia(n));
 end
```

Utilize um filtro tipo Butter e verifique se o problema de aliasing diminui. Perceba que `Wn` é a frequência de corte normalizada pela frequência de Nyquist, $2\pi f_s/2$. Isso quer dizer que `Wn=1`, a frequência de corte é a frequência de Nyquist. Atenção: verifique o uso do comando `filter`.

```
% butter(n,Wn) - retorna os coeficientes da função de transferência de um
% filtro passa-baixa de ordem n do tipo Butterworth com freq de corte
% normalizada Wn
wnb=[4000 8000]/(Fs/2);   %bandpass
[b,a]=butter(6,wnb);
audio_filtrado=filter(b,a,audio);
fvtool(b,a);
%
wn=0.25;
[b,a] = butter(2,wn);  %lowpass
fvtool(b,a,'Fs',44100)
```

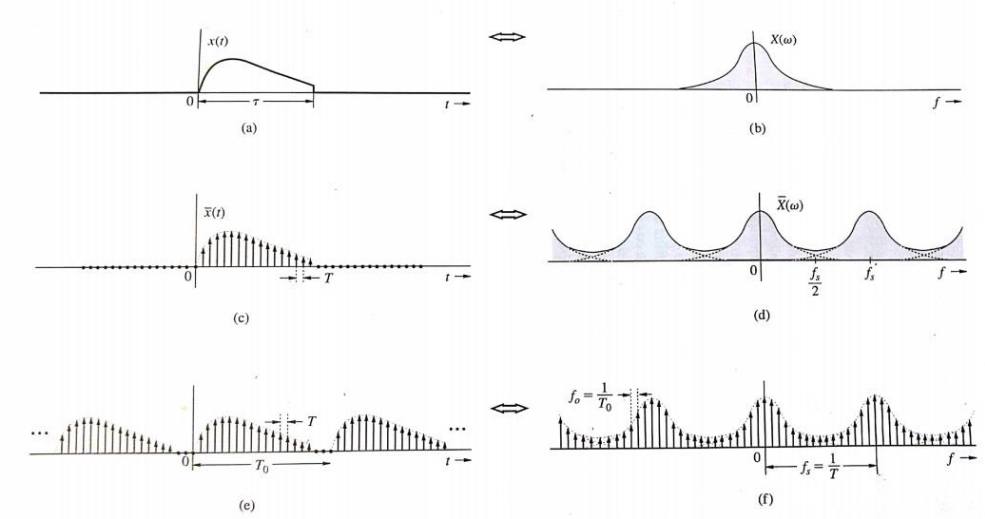
Verifique a utilização do filtro tipo Butterworth na eliminação de ruídos, completando o *script* abaixo.

```
t = 0:0.01:2;
y1= sin(2*pi.*t); % 1 Hz signal
y10 = sin(20*pi.*t); % 10 Hz signal
% soma-se y1 e y10 para gerar um final com ruído em alta frequencia
yt = y1 + 0.3*y10;

%utilizaremos um filtro de sexta ordem, e analisaremos ele até uma
%frequencia de corte menor ou igual a fc = 10Hz. Vemos que o ultimo
%teste elimina totalmente o ruido de alta frequencia (10 Hz)

[b,a] = butter(6,0.4); %distante da frequencia de corte ideal, fc = 10Hz ainda passa
pelo filtro
Filtrado1 = filter(b,a,yt);

[b,a] = butter(6,0.2);
Filtrado2 = filter(b,a,yt);

[b,a] = butter(6,0.15); %proximo a frequencia de corte ideal
Filtrado3 = filter(b,a,yt);

[b,a] = butter(6,0.1); %(fc/(fs/2)) idealmente, cortando frequencia de valor fc = 10Hz
Filtrado4 = filter(b,a,yt);

subplot(6,1,1), plot (t,y1);
subplot(6,1,2), plot (t,y10);
subplot(6,1,3), plot (t,Filtrado1);
subplot(6,1,4), plot (t,Filtrado2);
subplot(6,1,5), plot (t,Filtrado3);
subplot(6,1,6), plot (t,Filtrado4); %melhor caso, como ilustrado nas contas acima
```

Exercício 9

a. Extraído da referência [1] : Um sinal analógico de faixa limitada é amostrado a 7500 Hz (suficiente para assegurar que não haja aliasing), e N amostras são coletadas.
   i. Qual é a resolução em freqüência da DFT em Hz, se N = 1250?
   ii. Para atingir uma resolução em frequência de 4.5 Hz, qual deve ser N?

b. Extraído da referência [2] : Um sinal analógico de faixa limitada é amostrado (N=980, sem aliasing) a 500 Hz. A DFT destas 980 amostras é calculada. Queremos calcular o valor do espectro do sinal amostrado a 120 Hz.
   i. Qual índice *k* da DFT está mais próximo de 120 Hz, e qual é a sua frequência em hertz?
   ii. Qual é o número mínimo de zeros que devemos preencher além das 980 amostras para obter um valor da DFT exatamente a 120 Hz? Qual é o índice *k* da DFT correspondente a 120 Hz?



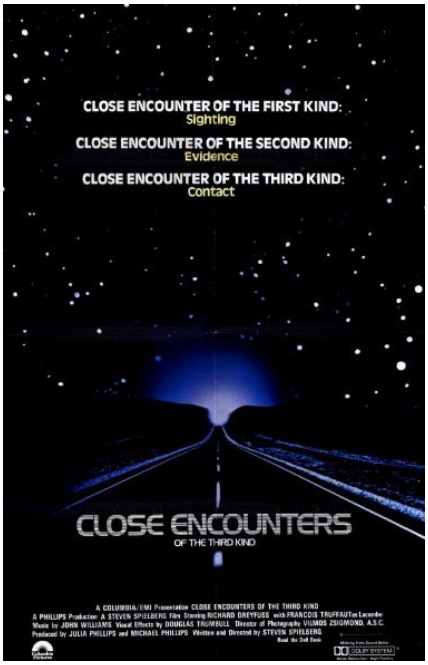
Relações entre as amostras de $x(t)$ e $X(\omega)$. Figura e discussão abaixo extraídas de [3].

Cálculos numéricos da transformada de Fourier (a DFT) de $x(t)$ necessitam dos valores amostrados de $x(t)$, pois um computador digital pode trabalhar somente com dados discretos. Além disso, um computador pode calcular $X(\omega)$ apenas para valores discretos de ω. Portanto, é umportante relacionar as amostras de $X(\omega)$ com as amostras de $x(t)$. A Figura (a) mostra um sinal limitado no tempo $x(t)$ e, por consequencia, $X(\omega)$ não é limitado em faixa – Figura (b). De aordo com o *teorema da amostragem,* o espectro $\bar{X}(\omega)$ do sinal amostrado $\bar{x}(t)$ é constituído de $X(\omega)$ repetindo a cada f_s Hz, sendo $f_s = 1/T$ como indicado nas Figuras (c,d). No passo seguinte, o sinal amostrado da Figura (c) é repetido periodicamente a cada T_0 segundos, como ilustrado na Figura (e). De acordo com o *teorema de amostragem espectral*, tal operação resulta na amostragem do espectro a uma taxa de T_0 amostras/Hz. Essa taxa de amostragem significa que as amostras são separadas por $f_0 = 1/T_0$ Hz – Figura (f).

A discussão mostra que quando um sinal $x(t)$ é amostrado e, então, periodicamente repetido, o espectro correspondente também é amostrado e periodicamente repetido. Quando amostramos $x(t)$ dentro de um espaço de tempo limitado, $T_0 < \infty$, então, matematicamente, estamos repetindo o sinal periodicamente, pois a transformada de Fourier (para sinal não periódico) é a Série de Fourier (sinal periódico), com período infinito...

Exercício 10 - Desafio

Adaptado de [3]



O filme **Contatos Imediatos do 3º grau** (em inglês Close Encounters of the Third Kind, algumas vezes abreviado como CE3K ou simplesmente Close Encounters), de 1977, foi escrito e dirigido por Steven Spielberg. O título é tirado da *classificação de contatos imediatos com alienígenas* criada pelo ufologista J. Allen Hynek, em que o terceiro grau indica observações humanas de verdadeiros alienígenas ou *seres animados.*

A comunicação entre os humanos e a raça alienígena era feita através de uma sequência de tons que os cientistas acreditavam ser reconhecida pelos alienígenas. Esta sequência era composta por 5 tons nas frequências **493,9Hz, 554,4Hz, 440Hz, 220Hz e 329,6Hz**.

Veja o vídeo do filme disponível no STOA ou no site,

https://www.youtube.com/watch?v=m2JL0xABlrQ

Sua tarefa consiste em criar um programa Matlab® que gere esta sequência de tons **no domínio da frequência**, considerando todos com a mesma duração. Mais precisamente você deve criar uma função `contatos(T)` onde `T` é a duração de cada tom da sequência. Por exemplo, ao digitar:

```
>> contatos(5)
```

deverá ser gerada a sequência de tons nos alto-falantes do PC com duração total de 25s. Você pode usar quase todas as ferramenta do Matlab **exceto qualquer forma de loop**. **Além disso, o comando sound pode ser utilizado uma única vez**.

```
%Exercício resolvido em 2017 pelos alunos:
%Alex Majima e Pietro Domingues
function contatos(duracao)
    intervalo=duracao; % Intervalo de duração de cada onda
    numOndas=5;
    Tmax=intervalo*numOndas;
    Amplit=1;
    fs=5000; % Frequência de amostragem
    t=[0:(1/fs):Tmax]; % Amostragem no tempo
    f1=493.9; % Frequência do sinal senoidal 1
    f2=554.4; % Frequência do sinal senoidal 2
    f3=440; % Frequência do sinal senoidal 3
    f4=220; % Frequência do sinal senoidal 4
    f5=329.6; % Frequência do sinal senoidal 5
    seno1=sin(2*pi*f1*t); % Geração da onda senoidal
    seno2=sin(2*pi*f2*t); % Geração da onda senoidal
    seno3=sin(2*pi*f3*t); % Geração da onda senoidal
    seno4=sin(2*pi*f4*t); % Geração da onda senoidal
    seno5=sin(2*pi*f5*t); % Geração da onda senoidal
    L=length(t);

    % Janela retangular unitaria com duracao de parametro da funcao e t0=0
    win = rectwin(duracao*fs)';
    wRect = [win zeros(1,(Tmax-duracao)*fs+1)];
    freq=linspace(-1,1,L)*(fs/2);
    FR=fftshift(fft(wRect));

    %delay do pulso unitário
    theta=(2*pi*j*freq.*intervalo);

    %utiliza-se a janela unitaria e modifica-se o delay no dominio da frequencia
    window1=FR;
    window2=FR.*exp(4*theta);
    window3=FR.*exp(3*theta);
    window4=FR.*exp(2*theta);
```

```
    window5=FR.*exp(theta);

    %Calcula-se FFT dos senos a serem utilizados
    S1=fftshift(fft(seno1));
    S2=fftshift(fft(seno2));
    S3=fftshift(fft(seno3));
    S4=fftshift(fft(seno4));
    S5=fftshift(fft(seno5));

    %Faz-se a convolução de cada janela com cada seno. Lembrando que a convolução
    % no domínio da frequência é igual à multiplicação dos sinais no domínio do tempo
    xw1=conv(S1,window1);
    xw2=conv(S2,window2);
    xw3=conv(S3,window3);
    xw4=conv(S4,window4);
    xw5=conv(S5,window5);

    %Calcula-se a transformada inversa da soma dos espectros calculados acima.
    OndaFinal=ifft(ifftshift(xw1+xw2+xw3+xw4+xw5))/(length(t)/2);
    t2=[0:(1/(2*fs)):Tmax];

    %Plot
    figure(3)
    plot(t2,real(OndaFinal),'r');
    xlabel('Tempo(s)');
    ylabel('Amplitude');
    %Som
    sound(real(OndaFinal),2*fs);
end
```

Referências

Os exercícios aqui apresentados foram extraídos e adaptados das seguintes fontes:

[1] Bombois, X. *Signal analyses* http://www.dcsc.tudelft.nl/~xbombois/SR3exercises.pdf

[2] Cuff, P. *Signal analyses* https://www.princeton.edu/~cuff/ele301/files/lecture8_2.pdf

[3] Lathi, B.P. *Sinais e Sistemas Lineares*, 2ª edição, Bookman, 2007.

[4] Oppenheim, A.V. *Signals and Systems*, http://ocw.mit.edu

[5] http://paloalto.unileon.es/ts/first/archives/html/p4_43_0.htm

[6] http://eeweb.poly.edu/iselesni/EL6113/DSP_Exercises.pdf

Anexo 01

```
close all; clear all; clc

%% Dados de amostragem
intervalo=5;               % Intervalo de duração de cada onda
numOndas=2;
Tmax=intervalo*numOndas;
%
Tmax=10;
fs=400;                    % Frequência de amostragem
t=[0:(1/fs):Tmax];            % Amostragem no tempo
L=length(t);

%% Funcao Seno
Amplit=1; f1=20;             % Amplitude e Frequência do sinal senoidal 1
seno1=Amplit.*sin(2*pi*f1.*t);   % Geração da onda senoidal
FFTy1=fft(seno1);

figure(10)
plot(abs(FFTy1))
hold on
%% Pulso retangular
T0=0;                      % Instante de início do pulso retangular
T=intervalo;               % Duração do pulso retangular

% Definição do início e final da janela
L_ini=length([0:(1/fs):T0]);
L_pulse=length([0:(1/fs):T]);
L_fin=L-L_ini-L_pulse;

win = rectwin(L_pulse);
wRect1 = [zeros(L_ini,1); win;zeros(L_fin,1)]';
FFTy2=fft(wRect1);
plot(abs(FFTy2))

figure(1);plot(t,seno1,t,wRect1);

%% Convolução
z=conv(FFTy1,FFTy2);Zshifted=(fftshift(z));
Z_mag=abs(z);
Z_mag=Z_mag(1:length(t));
N=length(seno1);
Fbin=0:1:N-1;

figure(2);
subplot(2,1,1)
plot((fs/N).*Fbin,Z_mag/(N/2))
xlim([0 fs/2])
xlabel('Hz'); ylabel('Magnitude');
grid on;


%% Multiplicação no tempo
SenoRect=seno1.*wRect1;
FFTy3=fft(SenoRect);Yshifted=(fftshift(FFTy
Y_mag=abs(FFTy3)
subplot(2,1,2)
plot((fs/N).*Fbin,Y_mag/(N/2))
xlim([0 fs/2])
xlabel('Hz'); ylabel('Magnitude');
grid on;
```