centro universitário FEI

Curso de Verão
Teoria e Prática de Processamento de Sinais e Imagens

Processamento do Sinal de Voz

Ivandro Sanches

2017

Programação

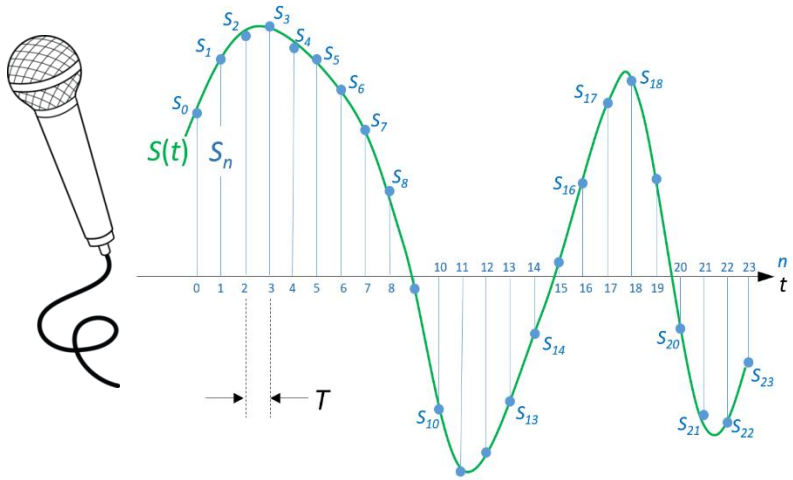
- Amostragem e quantização
- Áudio no MATLAB
  - geração
  - gravação
  - importação/exportação
- Processamento do áudio
  - segmentação
  - energia/taxa de cruzamentos por zero
  - convolução/filtragem
  - transformada discreta de Fourier
- Visualização
  - sinal no tempo e em frequência
  - espectrograma

Programação

- Produção da voz
  - pitch
  - formantes
  - vogais/consoantes
- Percepção da voz
  - curvas de loudness
  - bandas críticas
  - mascaramento
- Algumas aplicações e referências bibliográficas
- Laboratório
  - introdução de eco, inversão no tempo, áudio estéreo
  - filtragem de ruído
  - energia e taxa de cruzamentos por zero
  - função de autocorrelação

Amostragem e quantização

- Amostragem e quantização



cabeçalho
0010101001110101
0010101001100100
1010101000010001
. . .
0110001001010101

T : período de amostragem, s

$f_s = 1/T$: frequência de amostragem, Hz

Exemplos de f_s:

Áudio de CD: 44.1 kHz (T = 22.7 µs)

Telefonia: 8 kHz (T = 125 µs)

Amostragem e quantização

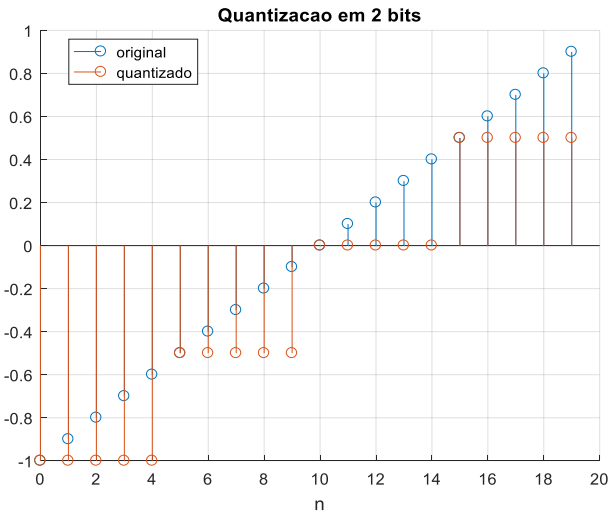
- Amostragem e quantização

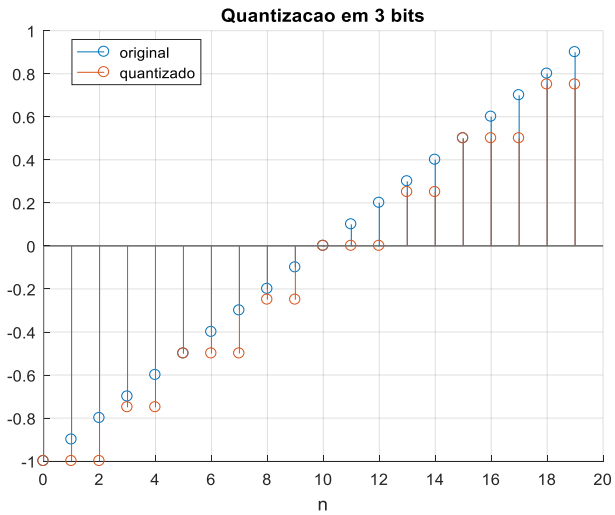
Application	Sample rate, resolution	Used how
telephony	8 kHz, 8–12 bits	64 kbps A-law or μ-law
voice conferencing	16 kHz, 14–16 bits	64 kbps SB-ADPCB
mobile phone	8 kHz, 14–16 bits	13 kbps GSM
private mobile radio	8 kHz, 12–16 bits	<5 kbps, e.g. TETRA
long-play audio	32 kHz, 14–16 bits	minidisc, DAT, MP3
CD audio	44.1 kHz, 16–24 bits	stored on CDs
studio audio	48 kHz, 16–24 bits	CD mastering
very high end	96 kHz, 20–24 bits	for *golden ears* listening

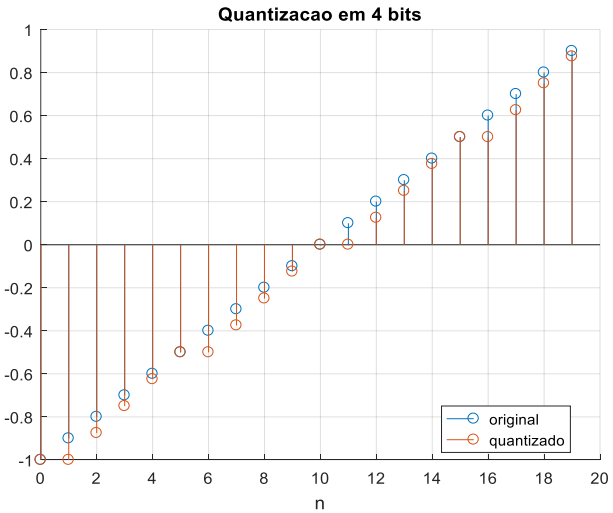
Extraído de Ian McLoughlin. Applied Speech and Audio Processing. Cambridge University Press. 2009

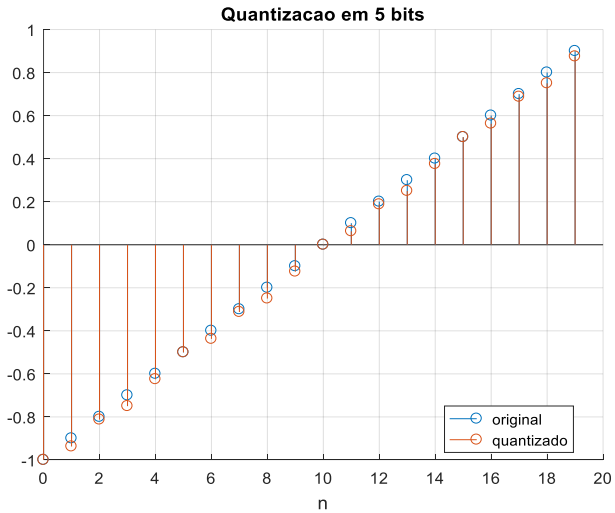
Amostragem e quantização

- Amostragem e quantização (resolução em nº de bits)



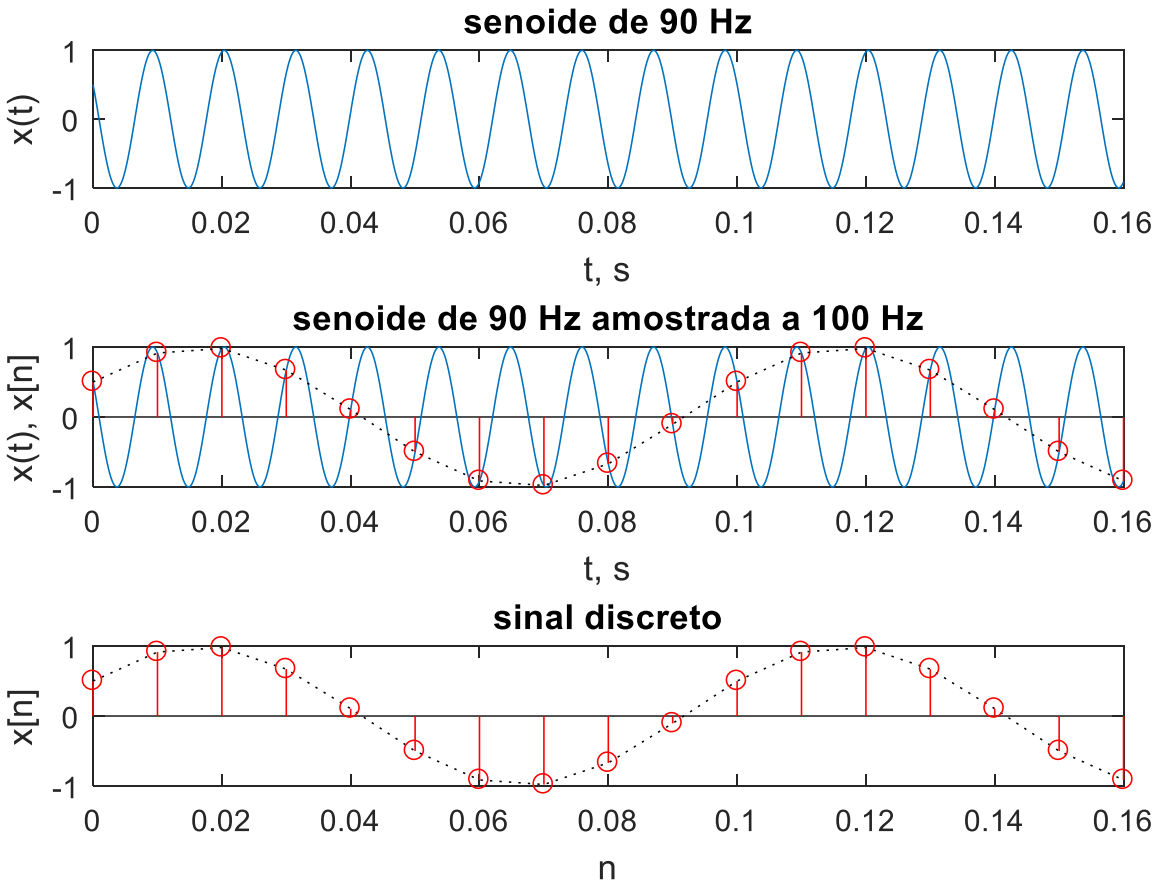






Amostragem e quantização

- Taxa de Nyquist (mínima freq. de amostragem)



Áudio no MATLAB

- Geração

A geração (simulação) de áudio no MATLAB pode ser feita de diversas maneiras. Abaixo, ilustram-se alguns exemplos.

1. Geração de senóide de f_1 = 5 kHz
2. Geração de ruído aleatório
3. Geração de senóide com ruído aleatório a 5 dB de relação sinal-ruído (*signal-to-noise ratio*, SNR)

```
% geracao de senoide na frequência f1
fs = 44100; % frequencia de amostragem, Hz
f1 = 5000;  % frequencia da senoide, Hz
T = 3;      % duracao em segundos
N = fs * T; % numero total de amostras
A = 0.8;    % amplitude as senoide
n = 0:N-1;  % indices das amostras
t = n/fs;   % indices de tempo, em segundo

s = A * sin (2*pi*f1*t); % amostras da senoide
sound(s, fs);            % reproducao no alto-falante
```

Áudio no MATLAB

- Geração

```
% Geracao de ruido aleatorio (gaussiano, média 0, variância 1)
r = randn(1, N);
% Adicao da senoide ao ruído ajustado para atender SNR
% energia do ruido
er = sum(r.^2);
% energia do sinal
es = sum(s.^2);
% SNR desejada, dB
snr = 5;
k = 10^(snr/10); % deseja-se que es/er seja k, logo:
r = r * (es / (er*k))^0.5;
% energia ajustada do ruido
er = sum(r.^2);
% SNR
10*log10(es/er)
% sinal resultante
x = s + r;
```

Áudio no MATLAB

- Geração

```
% Geracao de tom de frequencia variavel linearmente
% function sinal = genlin(td, fi, ff, fs)
%   td: duracao do tom em segundos, ex. 10 s
%   fi: freq. inicial (fi < fs/2),  ex. 100 Hz
%   ff: freq. final (ff < fs/2),    ex. 12000 Hz
%   fs: freq. de amostragem,        ex. 44100 Hz
%   sinal: sinal gerado
function  sinal = genlin(td, fi, ff, fs)
    t = 0:1/(td*fs):1;
    if fi <= ff,
        freqlin = fi + (ff-fi)*t;
    else
        freqlin = fi - (fi-ff)*t;
    end
    sinal = freqgen(freqlin, fs); % funcao definida no proximo slide
end
```

Áudio no MATLAB

- Geração

```
% Geracao de tom de frequencia variavel dada por fr
function [sinal] = freqgen(fr, fs)
    fr = fr*2*pi/fs; % normalizacao para rad/s
    phi = cumsum(fr); % soma acumulada
    sinal = sin(phi);
end
```

- Definição

Se uma forma de onda é definida como:

$$x(t) = \sin(\phi(t))$$

então, sua frequência instantânea é definida pela taxa da fase:

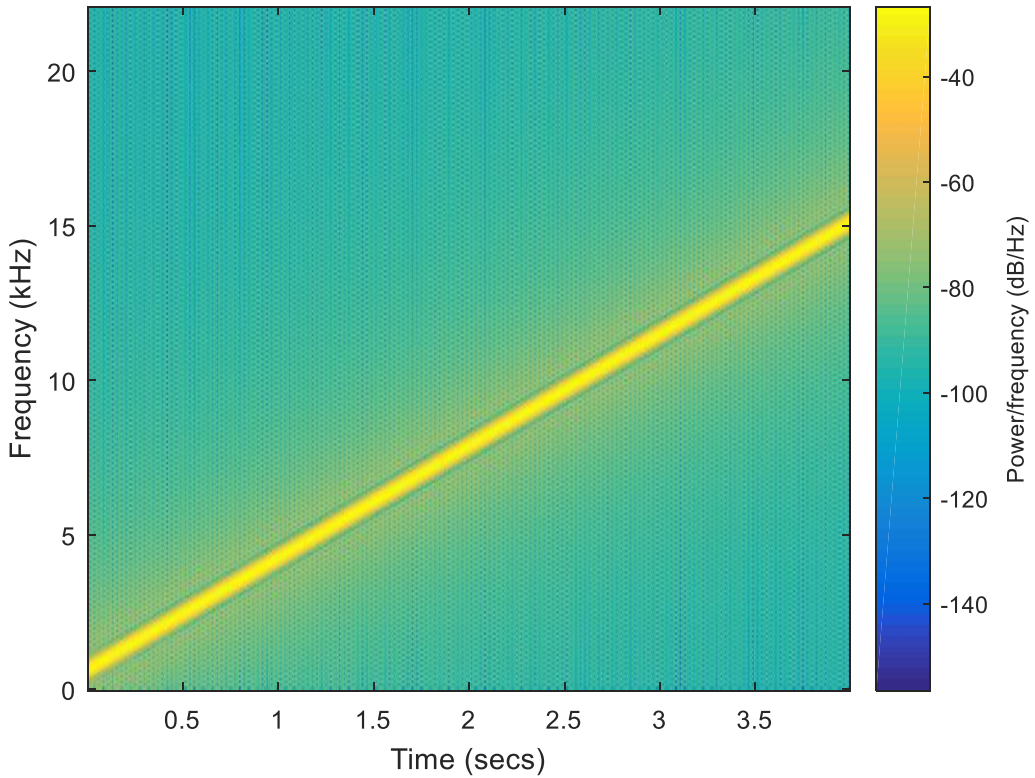
$$\omega(t) = \frac{d\phi(t)}{dt} \quad \rightarrow \quad f(t) = \frac{1}{2\pi}\frac{d\phi(t)}{dt}$$

Daí o uso de `cumsum()` em `freqgen()` acima.

Áudio no MATLAB

- Geração

```
fs = 44100;
sinal1 = genlin(4, 500, 15000, fs);
soundsc(sinal1, fs);
spectrogram(sinal1, hamming(256), 128, 512, fs, 'yaxis');
```



Áudio no MATLAB

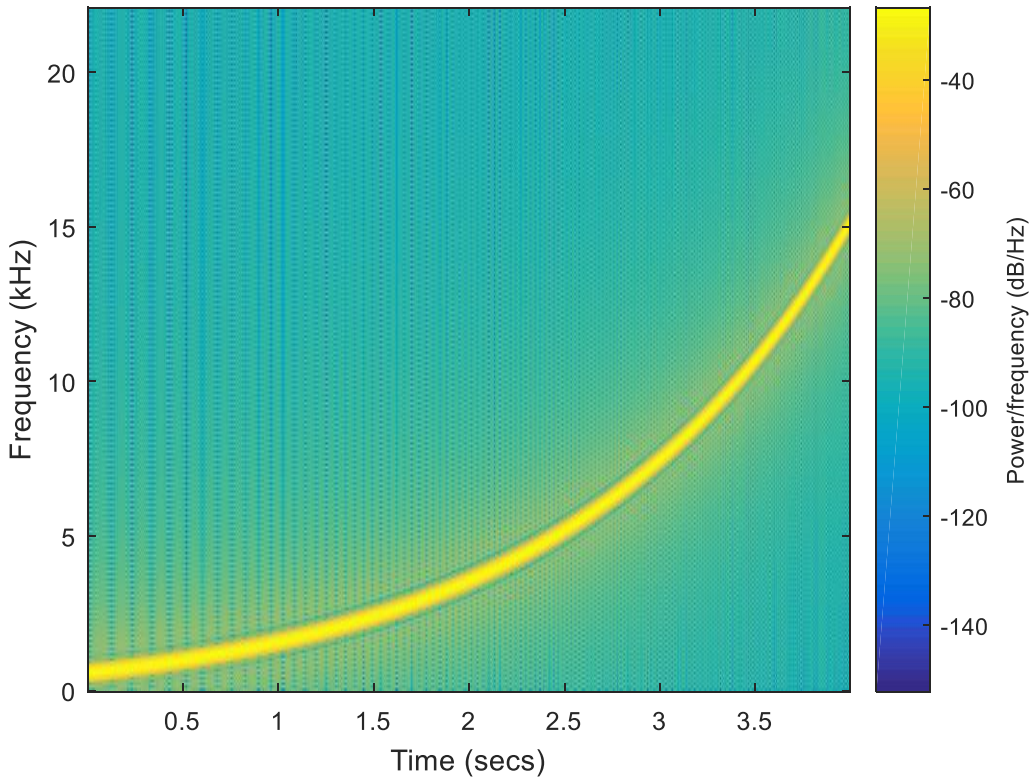
- Geração

```
% Geracao de tom de frequencia variavel exponencialmente
% function sinal = genexp(td, c, fi, ff, fs)
%   td: duracao do tom em segundos, ex. 10 s
%   c: grau de curvatura, (c >= 0), ex. 1
%   fi: freq. inicial (fi < fs/2),  ex. 100 Hz
%   ff: freq. final (ff < fs/2),    ex. 12000 Hz
%   fs: freq. de amostragem,        ex. 44100 Hz
%   sinal: sinal gerado
function  sinal = genexp(td, c, fi, ff, fs)
    x = 0:1/(td*fs):1; m = exp(c); % m constante de warping
    M = exp(m);
    if fi <= ff,
        freqexp = fi + (exp(m*x)-1)/(M-1)*(ff-fi);
    else
        freqexp = ff + (exp(m*x)-1)/(M-1)*(fi-ff);
        freqexp = freqexp(end:-1:1);
    end
    sinal = freqgen(freqexp, fs);
end
```

Áudio no MATLAB

- Geração

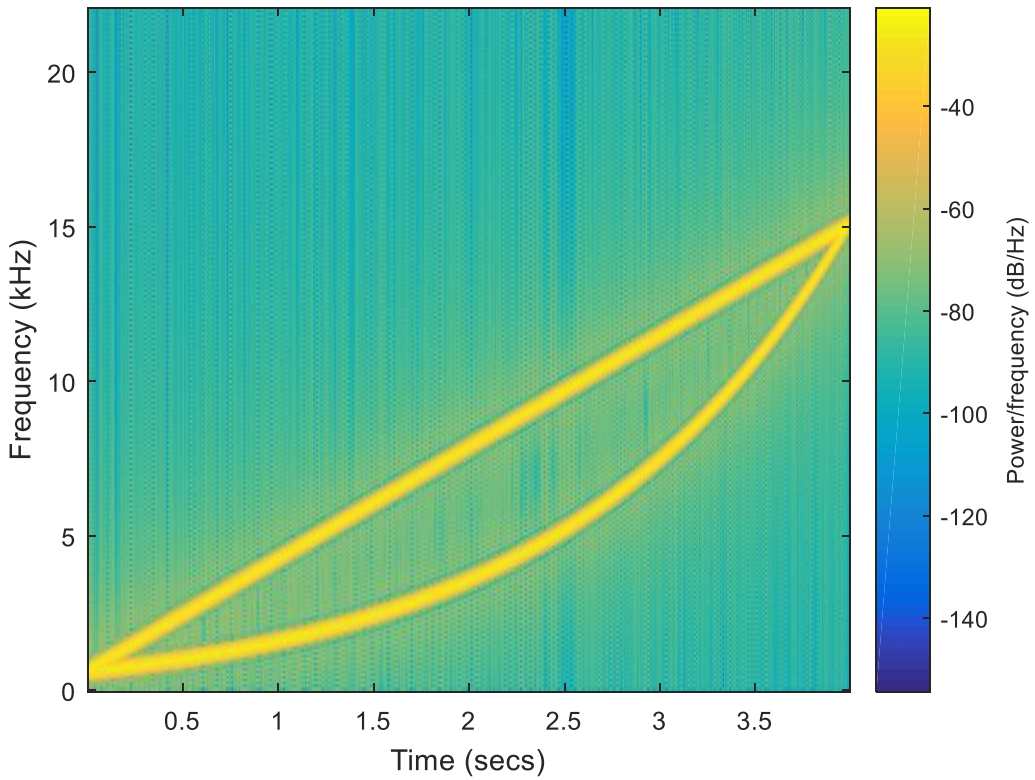
```
fs = 44100;
sinal2 = genexp(4, 1, 500, 15000, fs);
soundsc(sinal2, fs);
spectrogram(sinal2, hamming(256), 128, 512, fs, 'yaxis');
```



Áudio no MATLAB

- Geração

```
fs = 44100;
sinal3 = 0.5*sinal1 + 0.5*sinal2;
soundsc(sinal3, fs);
spectrogram(sinal3, hamming(256), 128, 512, fs, 'yaxis');
```



Áudio no MATLAB

- Reprodução

Um vetor contendo amostras de áudio pode ser reproduzido (tocado) via

1. amostras do vetor `sinal1` serão enviadas ao alto-falante/fone e tocadas à frequência de amostragem `fs`

```
sound(sinal1, fs);
```

2. idem acima, porém o áudio é amplificado (scaled) pelo maior fator sem que ocorra saturação na reprodução (clippling)

```
soundsc(sinal1, fs);
```

Áudio no MATLAB

- Gravação (recording)

A gravação de áudio pode ser efetuada de diversas maneiras. Por exemplo,

1. direto do MATLAB

```
nbits = 16; % numero de bits/amostra
nchans = 1; % numero de canais
r = audiorecorder(fs, nbits, nchans)
disp('Comece a falar...')
record(r);
pause(5); % 5 segundos de gravacao
stop(r);
falad = getaudiodata(r, 'double');
soundsc(falad, fs); % ou p = play(r);
```

2. criação de arquivo .wav via algum programa externo e sua posterior importação para o MATLAB

   Exemplos de programas para manipulação de áudio

   a. Wavesurfer http://www.speech.kth.se/wavesurfer
   b. Audacity http://www.audacityteam.org
   c. SFS http://www.phon.ucl.ac.uk/resource/sfs
   d. SonicVisualizer http://www.sonicvisualiser.org

Áudio no MATLAB

- Exportação de dados (cópia em arquivo)

A exportação das variáveis pode ser efetuada de diversas maneiras. Por exemplo,

1. Arquivo de áudio (.wav)

```
audiowrite('sinal1.wav', sinal1, fs); % fs=44100 Hz
audiowrite('estereo.wav', [sinal1' sinal2'], fs);
```

Obs: lembra-se que `sinal1` *e* `sinal2` *são vetores linha*

atenção: ações diferentes

2. Arquivo MATLAB binário (.mat)

```
save('tudo.mat'); % salva todas variaveis
save('algumas.mat', 'sinal1', 'sinal2', 'fs'); % escolhidas
```

3. Arquivo texto (.txt)

```
save('sinal1l.txt', 'sinal1', '-ascii'); % salva em modo texto
sinal1v = sinal1'; % sinal1v agora vetor coluna
save('sinal1v.txt', 'sinal1v', '-ascii'); % 1 valor por linha
```

Áudio no MATLAB

- Importação de dados (leitura de arquivo)

A leitura/restauração das variáveis copiadas em arquivos pode ser feita de diversas formas. Por exemplo,

1. Arquivo .wav

```
[sinal1, fs] = audioread('sinal1.wav');
[estereo, fs] = audioread('estereo.wav');
```

Lembrando que `estereo` acima é uma matriz, pois o arquivo `estereo.wav` contém 2 canais, os mesmos `sinal1` e `sinal2`, que foram gerados anteriormente

2. Arquivo .mat

```
load('tudo.mat'); % importa espaço salvo anteriormente
load('algumas.mat', 'sinal1', 'sinal2', 'fs'); % escolhidas
load('algumas.mat', 'fs'); % importa apenas fs
load('algumas.mat'); % importa todas variáveis do arquivo
```

3. Arquivo texto

```
load('sinal1v.txt', 'sinal1v', '-ascii');
```

Processamento do áudio

- Segmentação

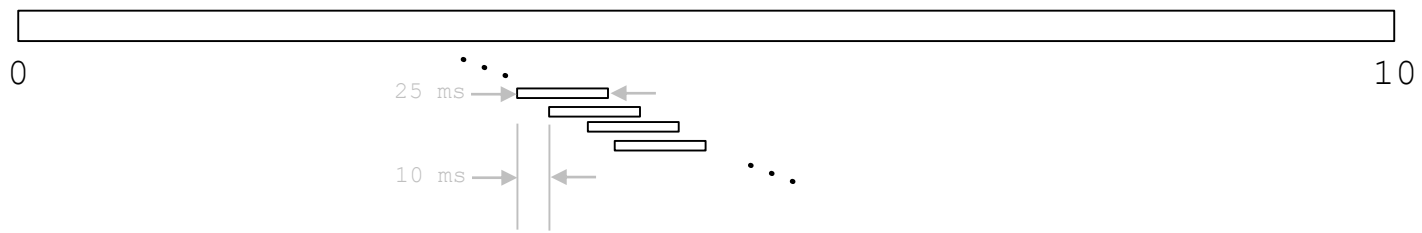
Como o sinal de voz apresenta trechos com características bastante diversas, seu processamento precisa ser feito por segmentos. O ideal é que cada trecho com características semelhantes pudesse ser tratado individualmente. Como isso não é possível, o sinal de voz (e de áudio) é processado por segmentação.

O tamanho de cada segmento não pode ser muito pequeno, pois a informação contida não é suficiente, e também não pode ser muito grande, quando trechos com características distintas seriam tratados em conjunto.

Dessa forma há um compromisso e o tipo de processamento e sua finalidade acabam por sugerir tamanhos adequados para segmentos.

Por exemplo, em reconhecimento de fala é usual que segmentos tenham tamanhos de 20 ms. Adicionalmente, ocorre sobreposição entre segmentos sendo analidados.

Concretamente, suponha um sinal com duração de 10 s. Caso esse sinal seja analisado por segmentos de 25 ms aplicados a cada 10 ms, qual é o número total de segmentos a ser processado?



Processamento do áudio

- Segmentação

Do exemplo anterior, sejam

T = 10 s, a duração total do sinal
S = 0.025 s, o tamanho de um segmento
dS = 0.010 s, o tempo entre deslocamentos dos segmentos

Chega-se que o número de segmentos, M, resulta em

$$M = (T - S) / dS + 1$$

ou

$$M = (T - S + dS) / dS$$

Substituindo os valores

$$M = (10 - 0.025 + 0.010) / 0.010 = 998.5$$

A parte decimal deve ser descartada pois não se processam trechos com duração menor do que um segmento

Portanto, resulta um total de 998 segmentos

Processamento do áudio

- Energia

Uma informação importante sobre um sinal é o comportamento da energia ao longo do tempo

Dado um segmento de duração S de um sinal que foi amostrado à frequência de amostragem f_s, é fácil calcularmos o tamanho em amostras desse segmento.

Seja N o número de amostras em um segmento de duração S, em segundos, extraído de um sinal que foi amostrado à taxa f_s, em Hz. Tem-se que

$$N = S\ f_s$$

Sejam então as N amostras contidas no segmento de duração S

$$s_0, s_1, s_2, \cdots s_{N-1}$$

A energia desse segmento é dada por:

$$E = s_0^2 + s_1^2 \cdots + s_{N-1}^2$$

ou, ainda, de forma mais compacta

$$E = \sum_{i=0}^{N-1} s_i^2$$

Processamento do áudio

- Taxa de cruzamentos por zero

Uma informação relativamente simples de ser computada e que dá uma ideia do conteúdo em frequência do sinal sendo analidado é a taxa de cruzamentos por zero (TCZ)

Dado um segmento com N amostras, a taxa de cruzamentos por zero pode ser computada pela expressão

$$Z = \frac{1}{2\,(N-1)} \sum_{i=1}^{N-1} \left| \operatorname{sign}(s_i) - \operatorname{sign}(s_{i-1}) \right|$$

Onde

$$\operatorname{sign}(s_i) = \begin{cases} 1, & s_i \geq 0 \\ -1, & s_i < 0 \end{cases}$$

Vê-se que

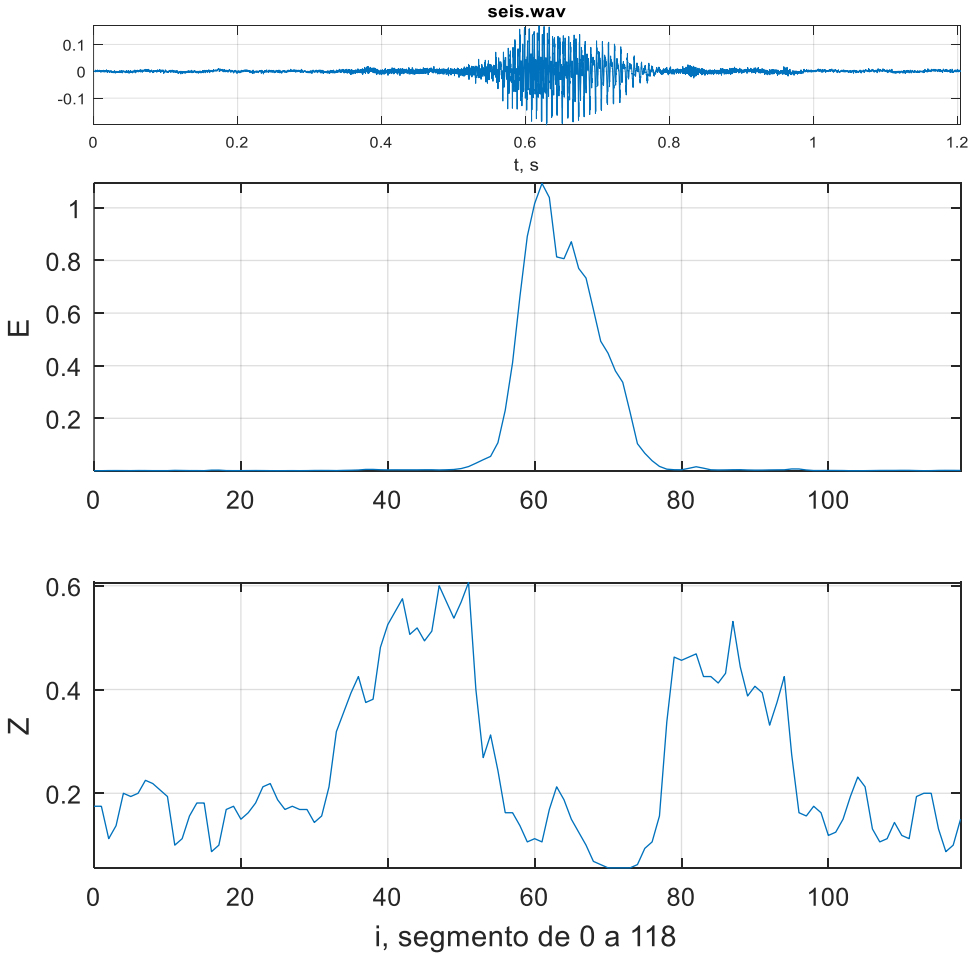
$$0 \leq Z \leq 1$$

Quanto mais próximo Z for de 1, maior o conteúdo em frequência do segmento. Note que quando Z = 0, o sinal não cruzou o eixo. Vê-se que essa grandeza é muito sensível a eventuais niveis DC presentes no sinal. Recomenda-se que se retire o nível médio do sinal ou que seja filtrado adequadamente.

Processamento do áudio

- Energia e TCZ

Seja o sinal de áudio **seis.wav**. A figura ilustra o comportamento da energia, E, e da TCZ, Z, desse sinal em segmentos de 20 ms a cada 10 ms. O aspecto geral dessas curvas é o esperado?



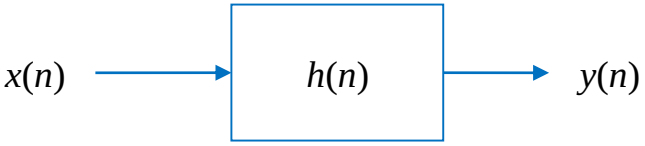
Processamento do áudio

- Convolução

Seja uma sequência discreta $x(n)$ ou $x[n]$ ou x_n. Seja um sistema linear invariante no tempo cuja resposta ao impulso é $h(n)$. A saída desse sistema, $y(n)$, dá-se pela convolução entre $x(n)$ e $h(n)$:

$$y(n) = h(n) * x(n)$$

convolução



A operação é comutativa:

$$y(n) = x(n) * h(n)$$

Explicitamente:

$$y(n) = \sum_{i=-\infty}^{\infty} h(i)\, x(n-i)$$

No MATLAB:

```
y = conv(h, x)
```

No domínio da frequência (propriedade da transformada de Fourier discreta no tempo):

$$Y(e^{j\omega}) = H(e^{j\omega})\, X(e^{j\omega})$$

Processamento do áudio

- Filtragem

Quando $h(n)$ corresponde à resposta ao impulso de um filtro digital, a convolução corresponde a um processo de filtragem e a saída desse processo pode ser obtida no MATLAB por:

```
y = conv(h, x)
```

ou:

```
y = filter(h, 1, x)
```

Vamos a seguir ilustar o projeto de um filtro digital. Nesse projeto simplificado, o resultado será a resposta ao impulso finita `h`, ou os coeficientes `a` e `b` que caracterizam o filtro, e que nos permitirá o processo de filtragem para uma entrada `x` qualquer.

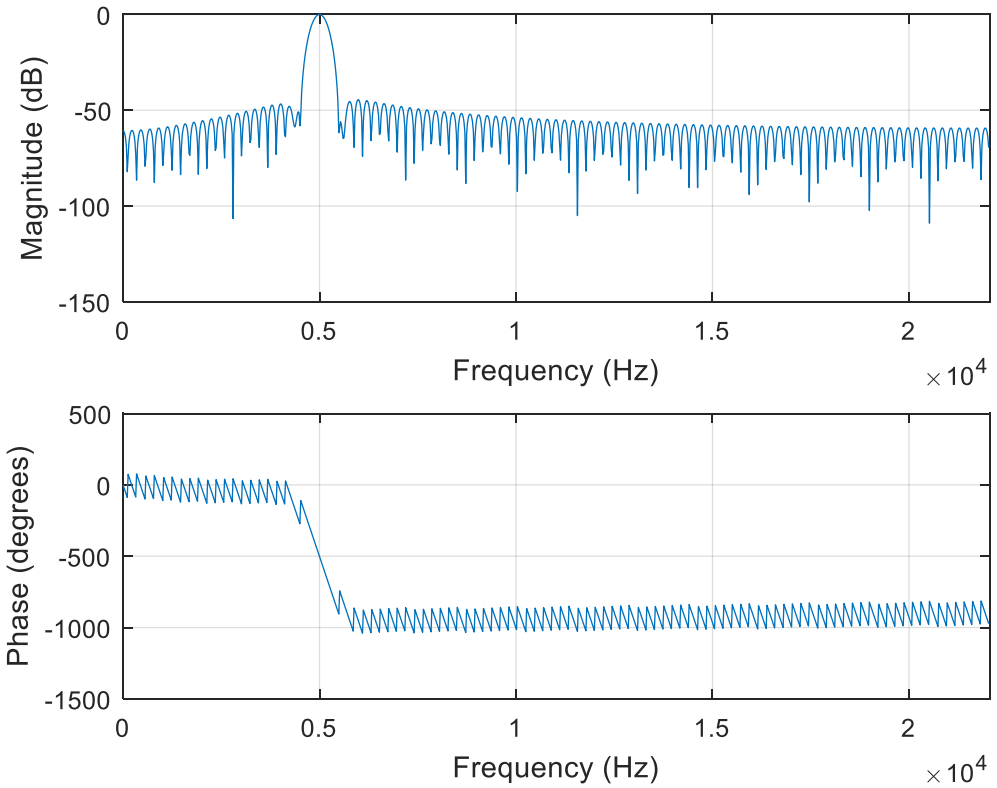
Posteriormente, vamos ilustrar o projeto de um filtro digital que possui resposta ao impulso infinita. Neste caso, o projeto retornará 2 vetores, `a` e `b`, que utilizaremos no processo de filtragem da seguinte forma:

```
y = filter(b, a, x)
```

Processamento do áudio

- Filtragem

Projeto de um filtro digital de resposta finita ao impulso. Dados do projeto:
frequência de amostragem: 44100 Hz
frequência de corte inferior: 4900 Hz
frequência de corte superior: 5100 Hz
ordem do filtro: 200

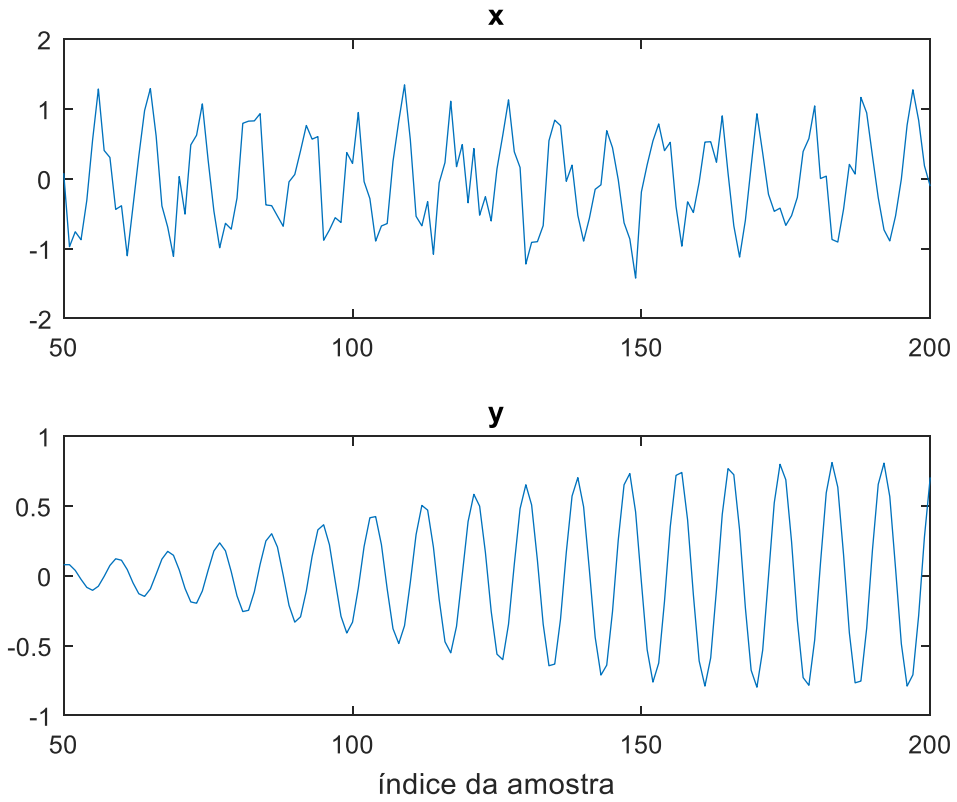



```
% projeto do filtro, h
fs = 44100; % freq. amostragem
fb = 4900; % freq. baixa/inferior
fa = 5100; % freq. alta/superior
N = 200; % ordem
h = fir1(N+1, [fb fa]/(fs/2));
freqz(h, 1, 1024, fs);

% x: sinal de 5kHz com ruido a 5dB
y = filter(h,1,x); %ou conv(h, z);
ind = 50:200; subplot(211),
plot(ind, x(ind)), title('x'),
subplot(212),
plot(ind, y(ind)), title('y')
xlabel('índice da amostra');
```

Processamento do áudio

- Filtragem

Lembrando que o sinal x é a senoide de 5 kHz mais ruído gaussiano a 5 db de SNR. O sinal y é o sinal x filtrado pelo filtro projetado no slide anterior. Note que após o transitório inicial, que é da ordem do filtro, praticamente se obtém s, a senoide original de 5 kHz (quase sem ruído)



Processamento do áudio

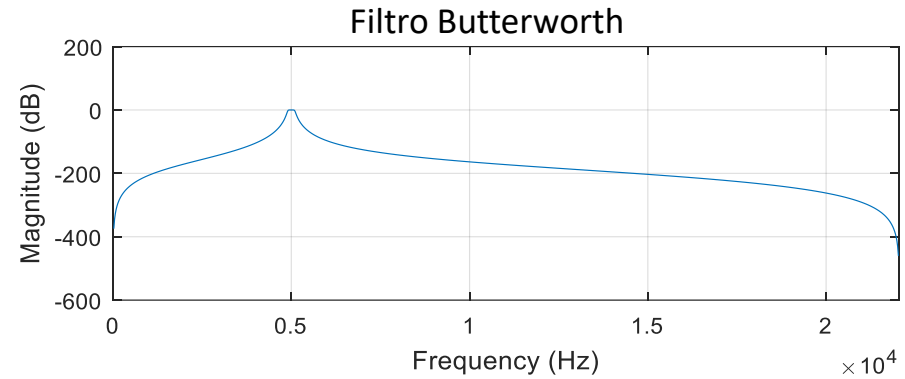
- Filtragem

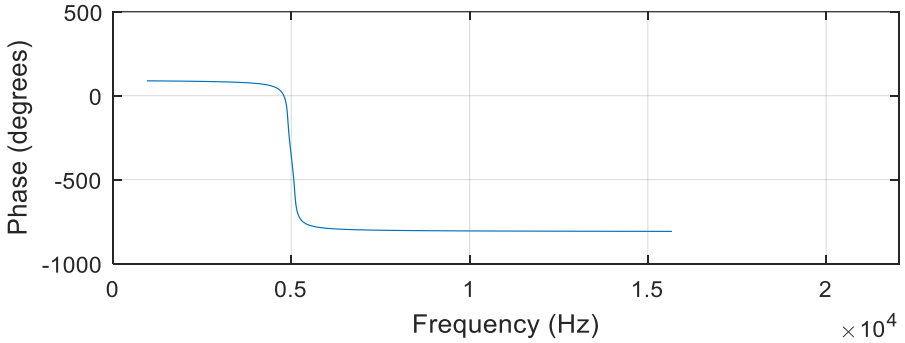
Projeto de um filtro digital de resposta infinita ao impulso. Dados do projeto:
frequência de amostragem: 44100 Hz
frequência de corte inferior: 4900 Hz
frequência de corte superior: 5100 Hz
ordem do filtro Butterworth: 10

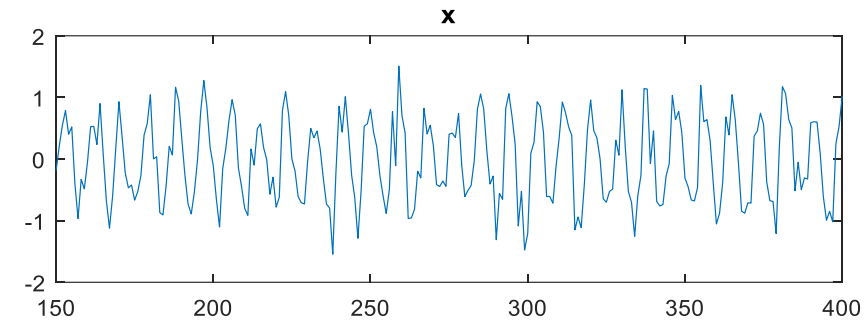
```
% projeto do filtro, [b, a]
fs = 44100; % freq. amostragem
fb = 4900; % freq. baixa/inferior
fa = 5100; % freq. alta/superior
N = 5; % ordem/2
[b, a] = butter(N, [fb fa]/(fs/2));
freqz(b, a, 1024, fs);

% x: sinal de 5kHz com ruido a 5dB
z = filter(b, a, x); ind = 150:400;
figure, subplot(211),plot(ind, x(ind)), title('x'),
subplot(212),plot(ind, z(ind)), title('z'),
xlabel('índice da amostra');
```

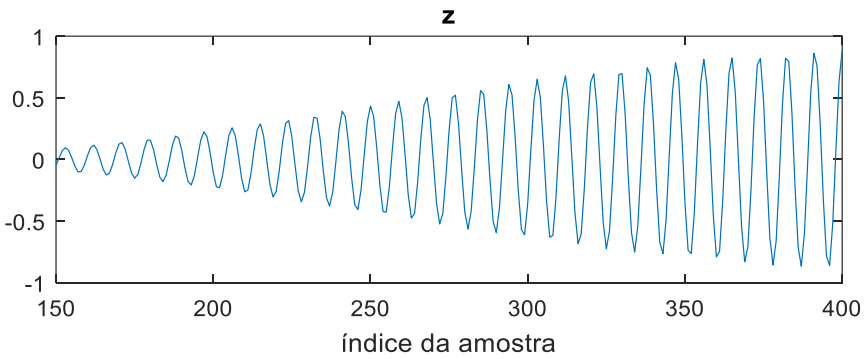
Processamento do áudio

- Filtragem









Processamento do áudio

- ## Transformada Discreta de Fourier

A transformada discreta de Fourier (Discrete Fourier Transform, DFT) converte uma sequência finita no tempo em sua correspondente versão em frequência. Isto é, a DFT é a representação no domínio da frequência da sequência original de entrada.

As expressões de análise e síntese, para a sequência original $x(n)$, $n = 0, 1, ..., N\text{-}1$, são:

Análise:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\frac{2\pi}{N}kn} \quad , k = 0, 1, \ldots, N-1$$

Síntese:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, e^{j\frac{2\pi}{N}kn} \quad , n = 0, 1, \ldots, N-1$$

Indica-se o par de transformadas na forma:

$$x(n) \leftrightarrow X(k)$$

No MATLAB:

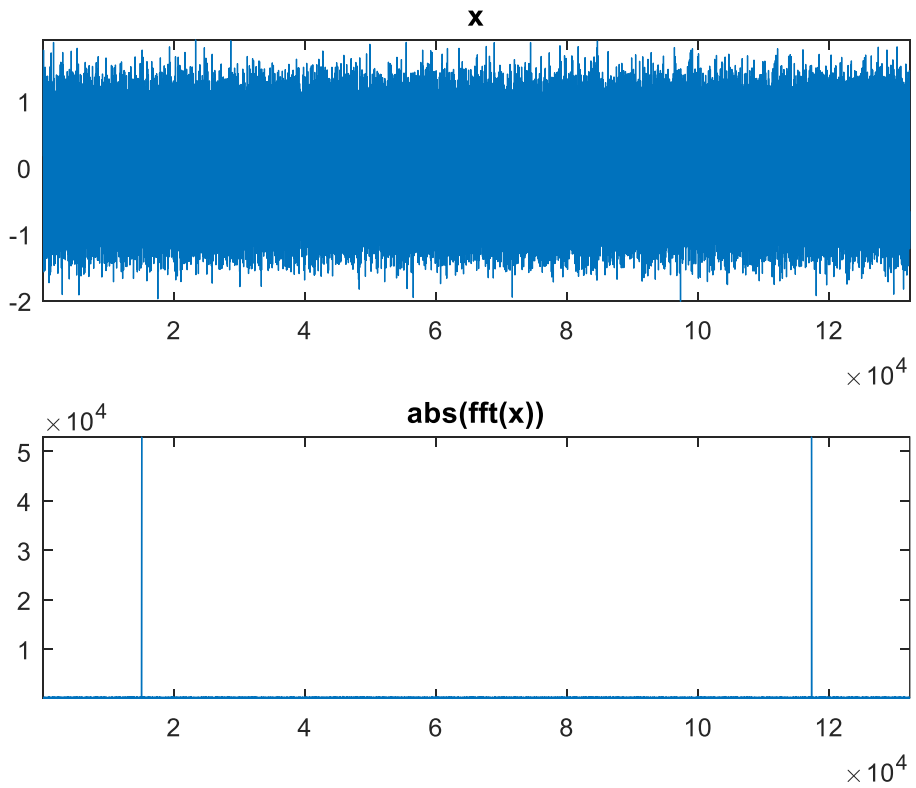
```
X = fft(x);  % transformada direta de x
x = ifft(X); % transformada inversa
```

Processamento do áudio

- Transformada Discreta de Fourier

Geralmente trabalha-se com o sinal no tempo como uma sequência real. Normalmente a sequência transformada é complexa. Portanto, note abaixo o uso da função `abs()` nas transformadas de sequências vistas anteriormente.

```
% x é a senoide com ruído
% adicionado a 5 dB de SNR
X = fft(x);
Xa = abs(X);
subplot(211), plot(x),
axis tight, title('x'),
subplot(212), plot(Xa),
axis tight,
title('abs(fft(x))'),
```



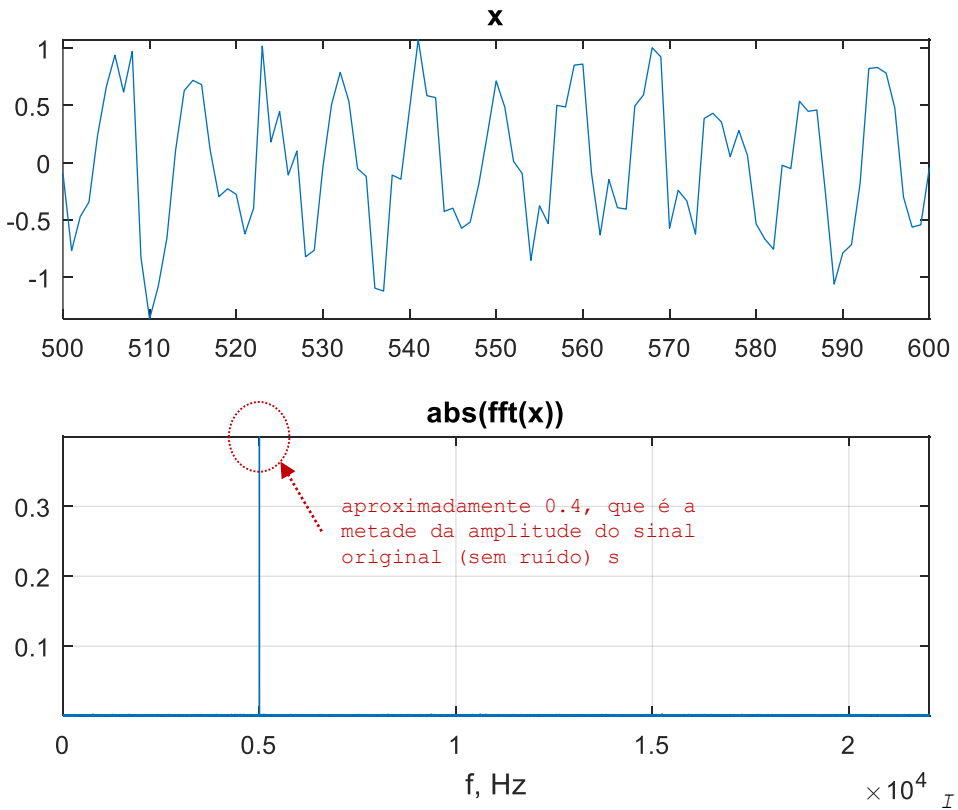
Visualização

- Sinal no tempo e em frequência

Vamos enxergar melhor o que está ocorrendo nas figuras da página anterior ao melhorar a visualização de seu conteúdo. Assim, vamos:

1. olhar um pequeno trecho em detalhe do sinal **x**
2. converter o eixo x da DFT para Hz e mostrar apenas a primeira metade, já que a segunda metade é a imagem Hermitiana da primeira (isto é, as partes reais são idênticas e as partes imaginárias têm o sinal trocado)

```
% trecho de x
N = length(x);
ind = 500:600;
subplot(211),
plot(ind, x(ind)),
axis tight, title('x'),
% eixo da DFT em Hz
f = (0:N-1)/N*fs;
ind = 1:floor(N/2);
subplot(212),
plot(f(ind), Xa(ind)/N),
axis tight, grid
title('abs(fft(x))'),
xlabel('f, Hz'),
```

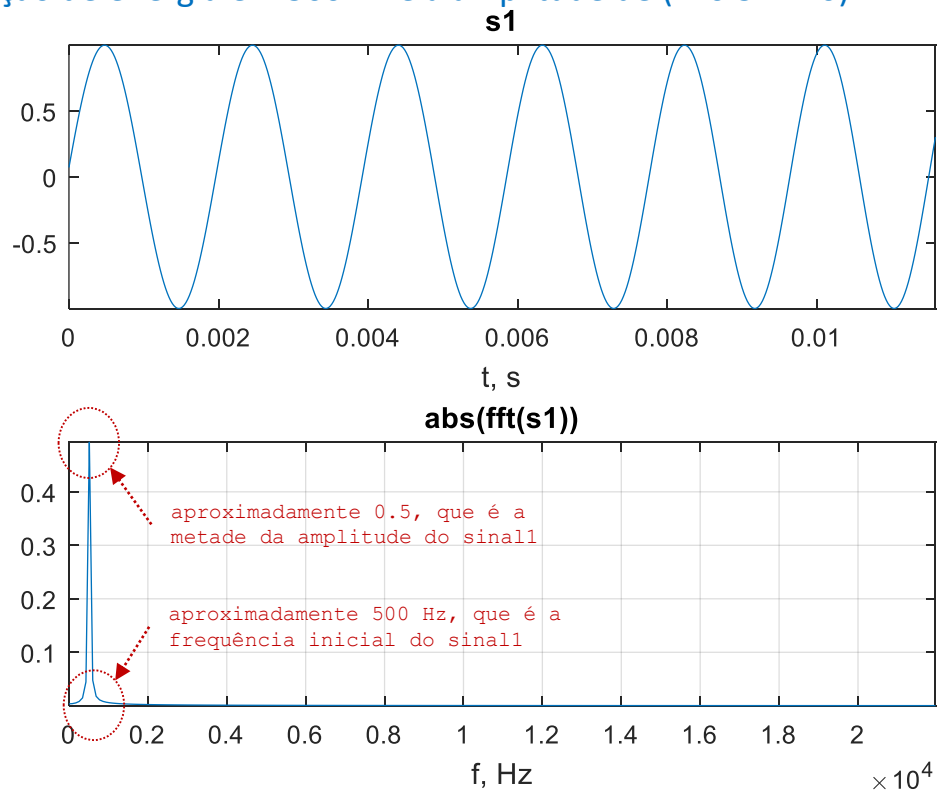


Visualização

- Sinal no tempo e em frequência

sinal1, visto anteriormente, tem sua frequência saindo de 500 Hz e chegando a 15 kHz de forma linear. Vamos analisar 2 trechos desse sinal, no início e no fim. Cada trecho terá o comprimento de N = 512 amostras. Note que a DFT revela a concentração de energia em 500 Hz e a amplitude de (2x0.5 = 1.0)

```
% trecho 1 de sinal1
N = 512;
ind = 1:N;
t = (ind-1)/fs;
s1 = sinal1(ind);
S1 = abs(fft(s1));
subplot(211), plot(t, s1),
axis tight, title('s1'),
xlabel('t, s');
% eixo da DFT em Hz
f = (0:N-1)/N*fs;
ind = 1:floor(N/2);
subplot(212),
plot(f(ind), S1(ind)/N),
axis tight, grid
title('abs(fft(s1))'),
xlabel('f, Hz'),
```

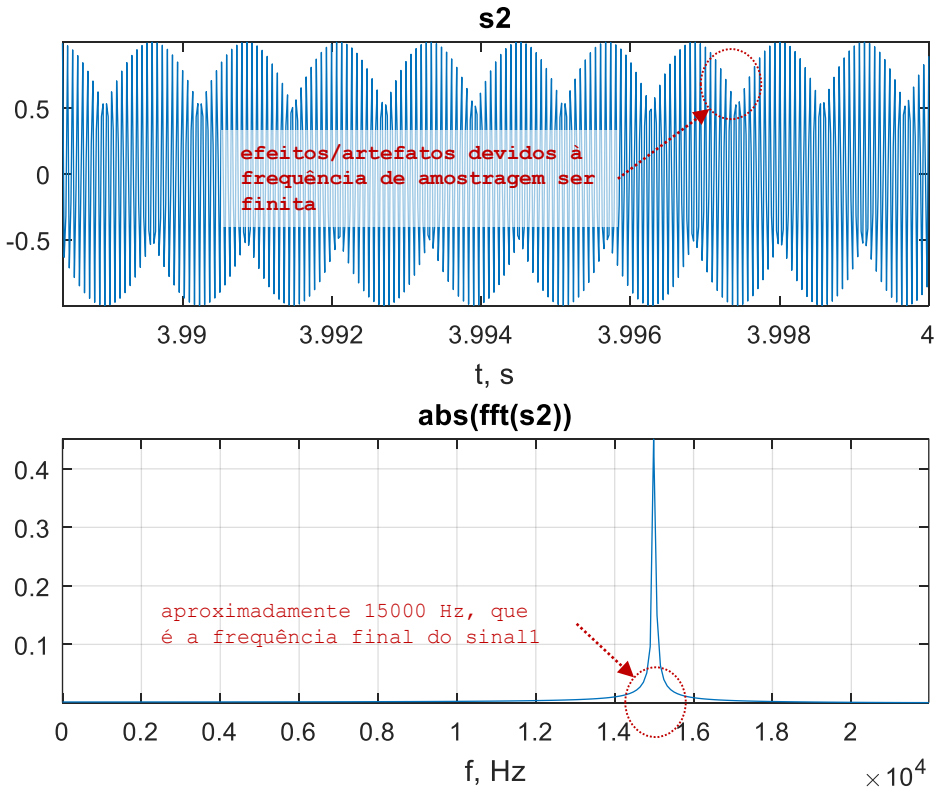


Visualização

- Sinal no tempo e em frequência

Análise do trecho final de N = 512 amostras do sinal1, quando sua frequência atinge 15 kHz.

```
% trecho 2 de sinal1
N = 512; M=length(sinal1);
ind = M-N:M;
t = (ind-1)/fs;
s2 = sinal1(ind);
S2 = abs(fft(s2));
subplot(211), plot(t, s2),
axis tight, title('s2'),
xlabel('t, s');
% eixo da DFT em Hz
f = (0:N-1)/N*fs;
ind = 1:floor(N/2);
subplot(212),
plot(f(ind), S2(ind)/N),
axis tight, grid
title('abs(fft(s2))'),
xlabel('f, Hz'),
```

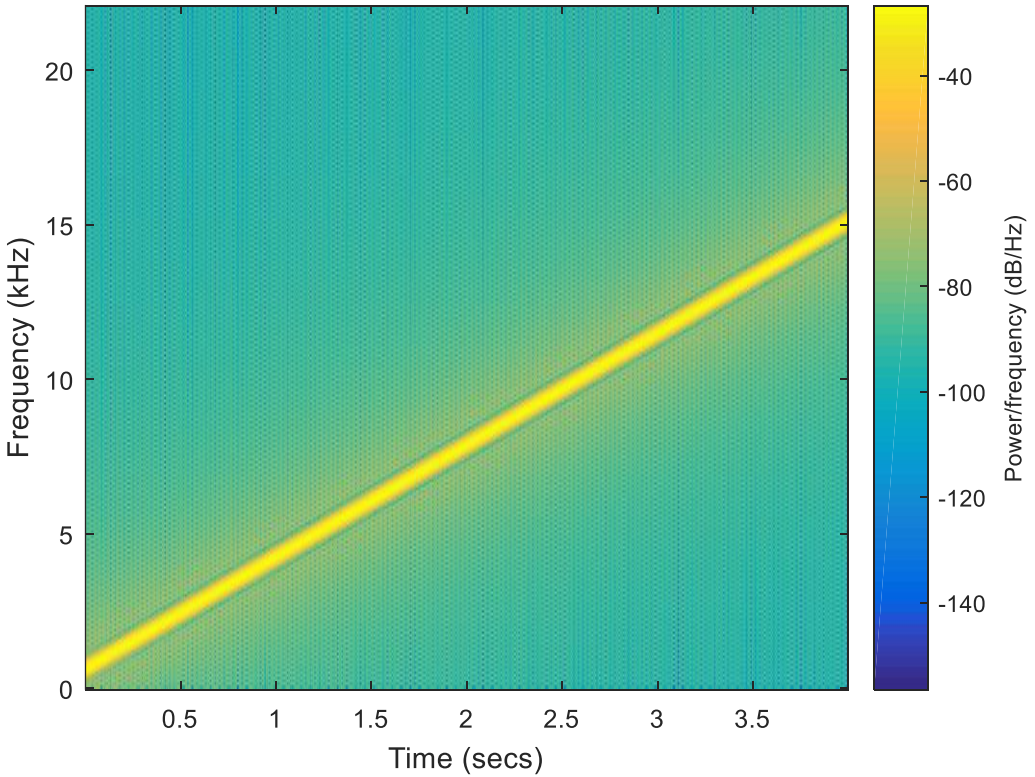


Visualização

- Espectrograma

O espectrograma combina os dois tipos de gráficos vistos nas páginas anteriores em um único gráfico. Usualmente representa-se o tempo ao longo do eixo x, a frequência ao longo do eixo y e a amplitude/intensidade como a cor do ponto (x,y). Como visto anteriormente, abaixo é ilustrado o espctrograma do sinal1

```
sinal1=genlin(4, 500, 15000, fs);
spectrogram(
   sinal1,       % sinal
   hamming(256),% janela e tamanho
   128,          % sobreposicao
   512,    % no. de pontos para FFT
   fs,         % freq. de amostragem
   'yaxis'); % freq. no eixo y
```

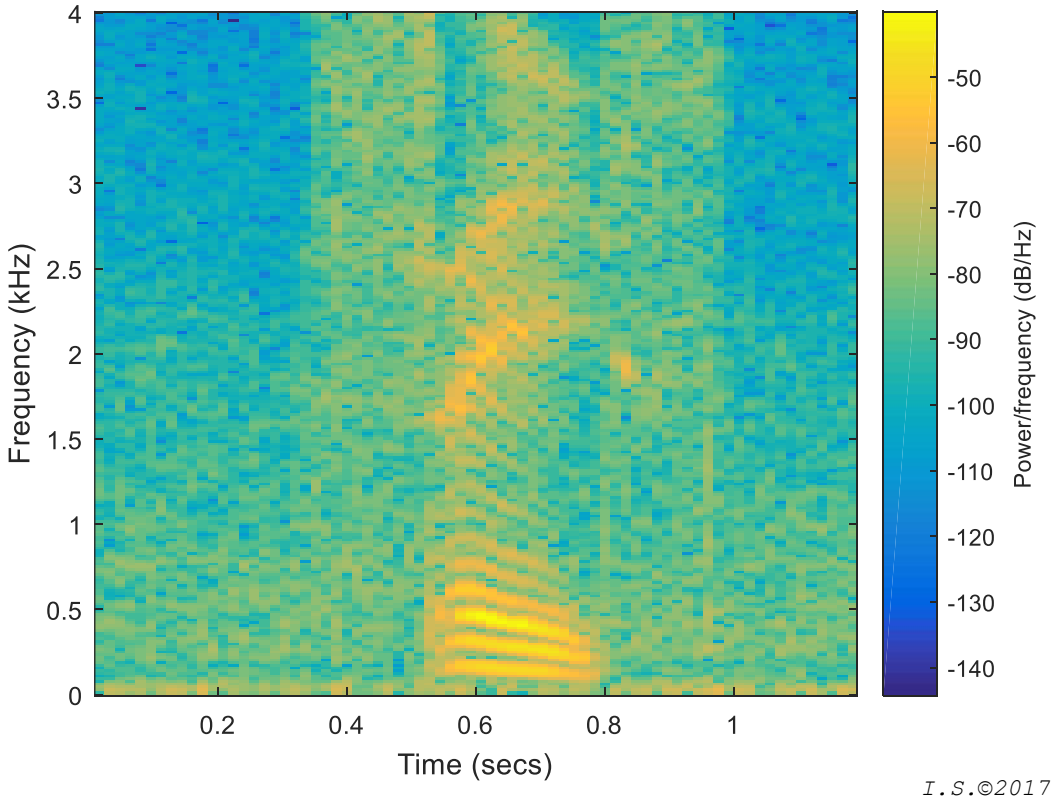


Visualização

- Espectrograma

Espectrograma do sinal de voz no arquivo **seis.wav**
Note o limite no valor da escala de frequência: 4 kHz. Esse valor corresponde à metade da frequência de amostragem do sinal sendo analisado. Neste caso, f_s = 8 kHz.

```
[y, fs] = audioread('seis.wav');
spectrogram(
    y,              % sinal
    hamming(256),% janela e tamanho
    128,            % sobreposicao
    512,    % no. de pontos para FFT
    fs,         % freq. de amostragem
    'yaxis'); % freq. no eixo y
```

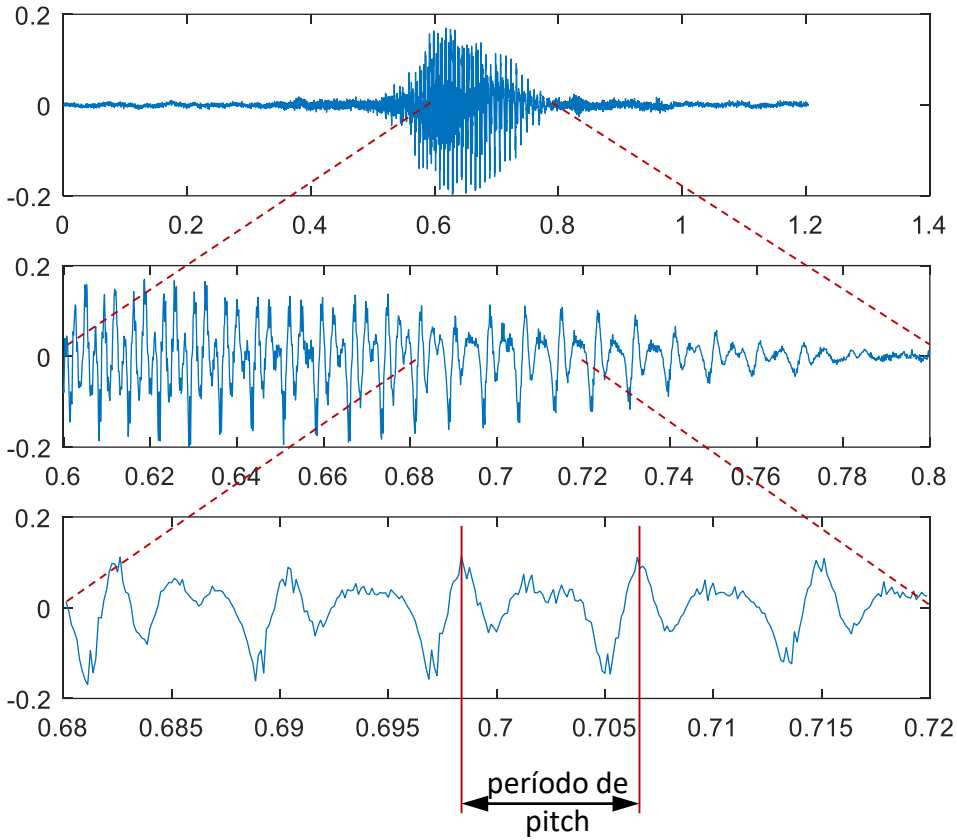


Produção da voz

- Pitch

Pitch, F_0, ou frequência fundamental, é a frequência de vibração das cordas vocais. Valores baixos, entre 85 e 180 Hz, são frequentes no sexo masculino (adulto em fala normal). Valores femininos podem variar de 165 a 255 Hz. Crianças e, principalmente bebês, podem atingir valores acima de 300 Hz.

```
[y, fs] = audioread('seis.wav');
M = length(y);
t = (0:M-1)/fs;
subplot(311), plot(t, y),
ind = 4800:6400;
subplot(312),
plot(t(ind), y(ind)),
ind = find(t>0.68 & t<0.72);
subplot(313),
plot(t(ind), y(ind)),
```



Na figura, estima-se uma duração de cerca de 0.008 s, equivalendo a um pitch F_0 = 1/0.008 = 125 Hz.
Esse valor sugere um locutor do sexo masculino.

Produção da voz

- Pitch

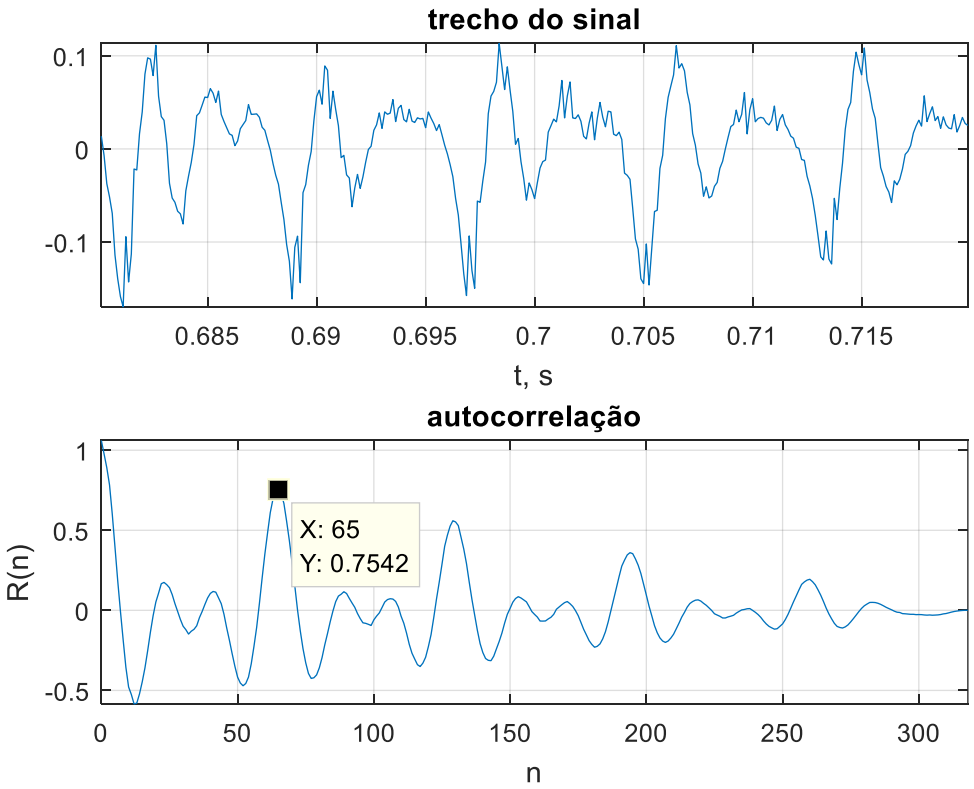
Um método simples de se estimar a frequência de pitch, F_0, e por isso nem sempre muito eficiente, é pela função de autocorrelação. Abaixo apresenta-se a função de autocorrelação e sua aplicação a um trecho selecionado do sinal da página anterior. Dado um trecho de N amostras do sinal $x(n)$, sua autocorrelação, $R(n)$, pode ser dada por:

$$R(k) = \sum_{n=0}^{N-1-k} x(n+k)\, x(n) \quad , k = 0, 1, \ldots, N-1$$

Calculada a autocorrelação, basta agora determinar o valor de n onde $R(n)$ é máxima após os valores iniciais. Esse processo pode ser feito automaticamente, contudo, da figura, nota-se que esse valor de n vale 65. Lembrando que a frequência de amostragem do sinal **seis.wav** é 8 kHz, chega-se a

$$F_0 = \frac{8000}{65} \cong 123\,\text{Hz}$$

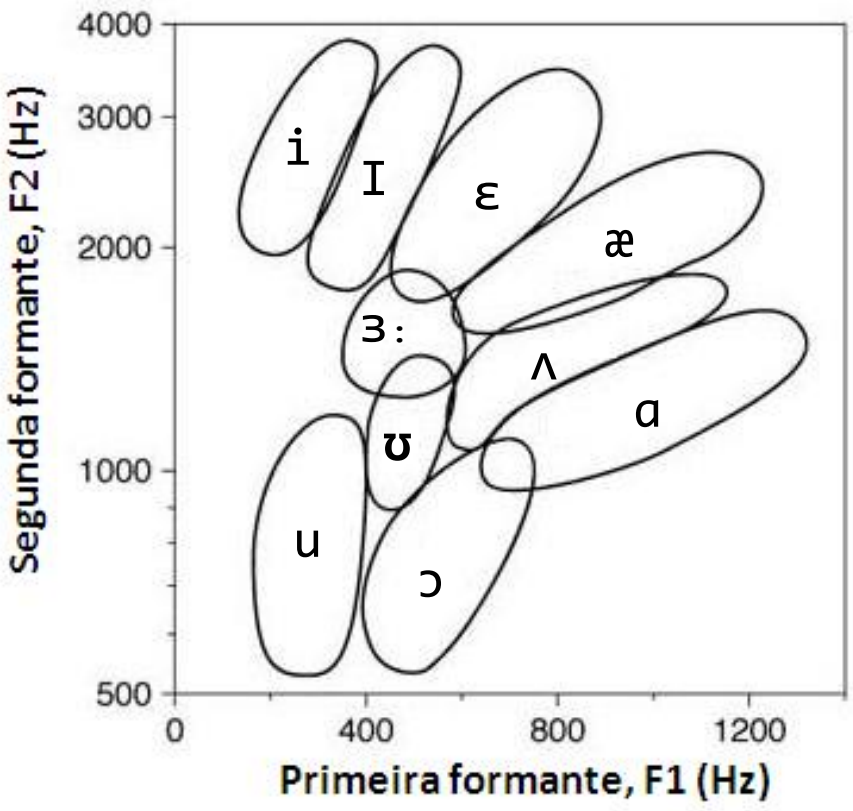
que corresponde ao valor estimado na página anterior para o mesmo trecho de sinal



Produção da voz

- Formantes

As formantes são as frequências de ressonância do trato vocal. Por exemplo, para cada vogal o trato vocal assume uma configuração relativamente estável, determinando frequências de ressonância específicas para cada vogal. A figura ilustra a relação entre a primeira, F_1, e a segunda, F_2, formantes para vogais da língua inglesa.

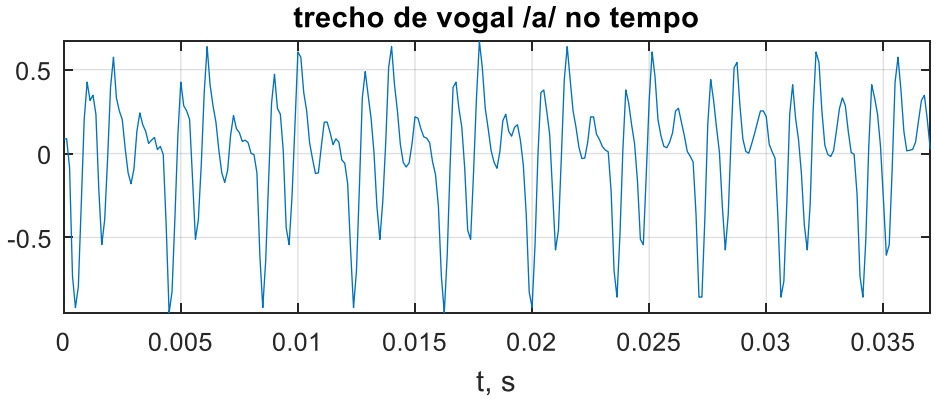


vogal	exemplo
[i]	beet
[I]	bit
[ɛ]	bet
[æ]	bat
[ʌ]	but
[ɑ]	hot
[ɔ]	bought
[ʊ]	foot
[u]	boot
[ɜ:]	bird

Produção da voz

- Vogais e consoantes

Abaixo é apresentado o exemplo de uma vogal. Note a quase periodicidade do sinal devido à vibração das cordas vocais. Processou-se esse trecho pela técnica de predição linear e compararam-se os espectros obtidos pela FFT e pelos coeficientes de predição linear (LPC). Note que esta última técnica ressalta as ressonâncias do trato vocal, permitindo uma melhor análise das frequências formantes.



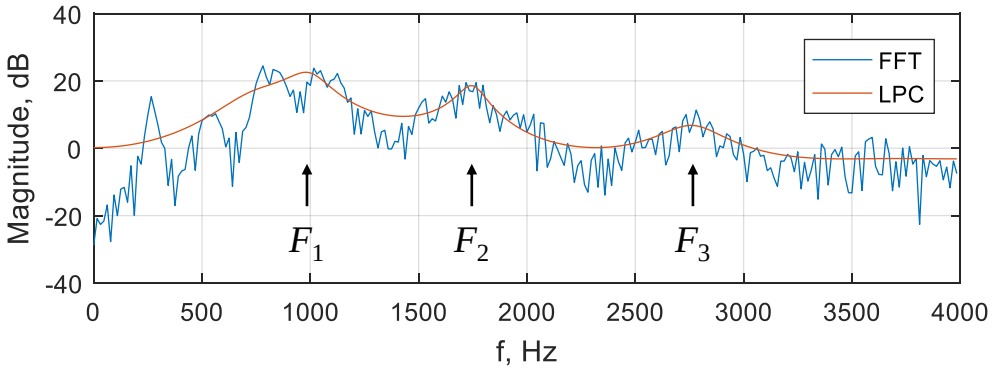
Da figura:

$F_1 \cong 1\,\text{kHz}$

$F_2 \cong 1.75\,\text{kHz}$

$F_3 \cong 2.75\,\text{kHz}$

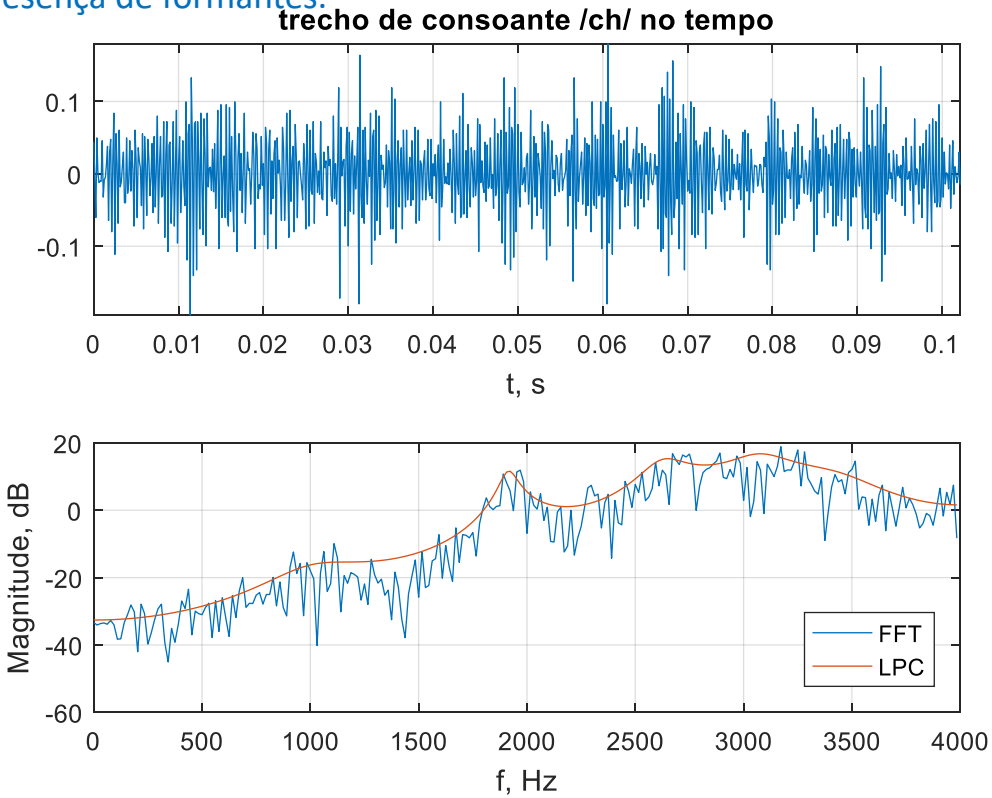
compare com o gráfico F_1xF_2 da página anterior para essa vogal.



Produção da voz

- Vogais e consoantes

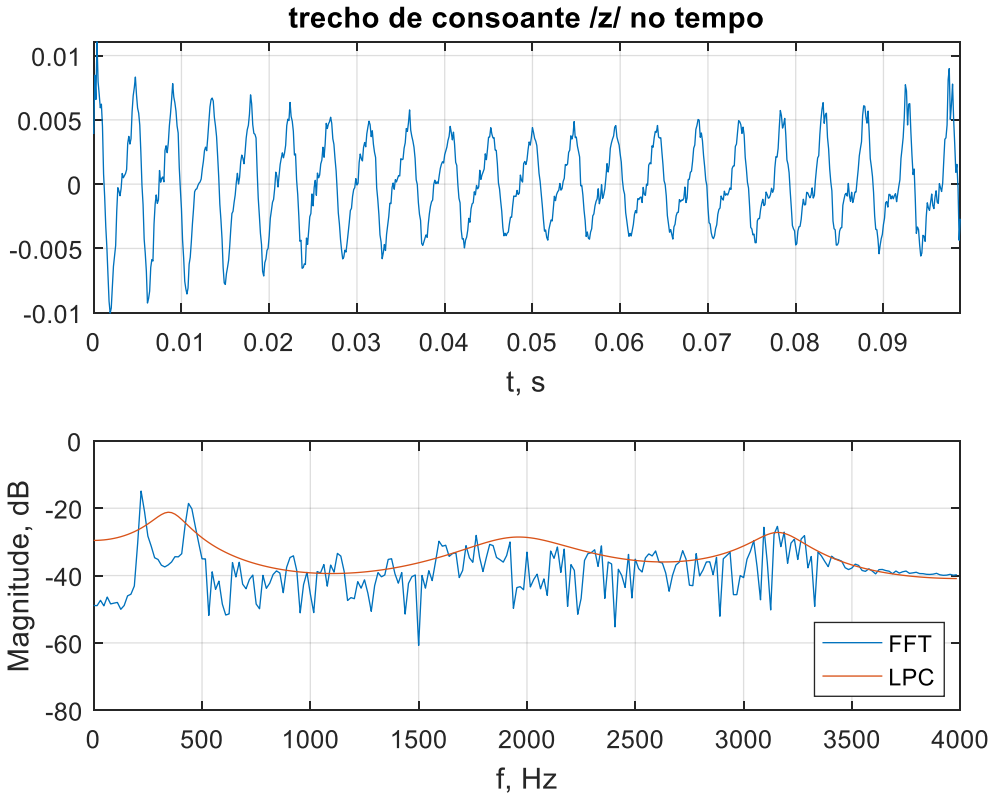
Abaixo é apresentado um exemplo de uma consoante. Consoantes podem ser **sonoras**, quando há vibração das cordas vocais, e **surdas**, quando não há vibração das cordas vocais. Abaixo é apresentada uma consoante surda, como o 'ch' de 'chá'. Note o aspecto de ruidoso desse sinal no tempo e que a concentração de energia do sinal se encontra em frequências relativamente mais altas, sem existir um claro padrão de presença de formantes.



Produção da voz

- Vogais e consoantes

Abaixo é apresentada uma consoante sonora, como o 'z' de 'zê'. Note que existe uma periodicidade no sinal, devido à vibração das cordas vocais. O espectro tem um comportamento razoavelmente plano quando comparado aos dois exemplos anteriores.

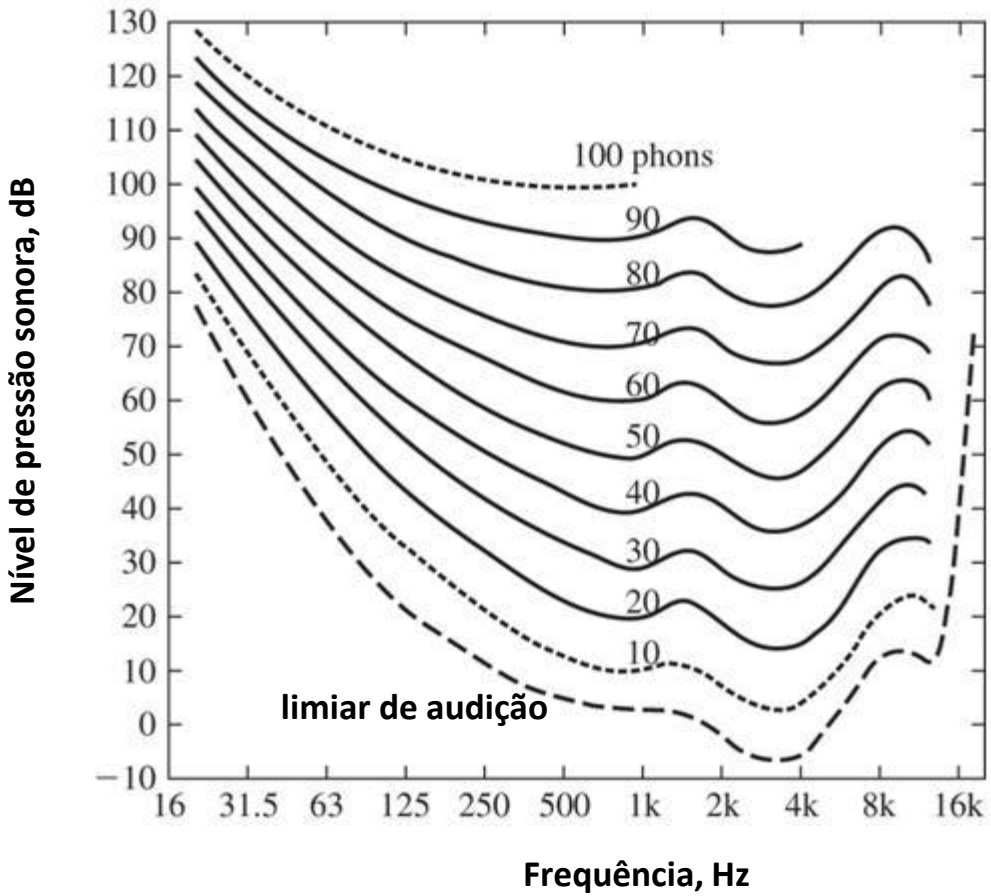


Percepção da voz

- Curvas de loudness (Fletcher-Munson, Equal Loudness Curves for Pure Tones, 1933)

A sensibilidade auditiva varia com a frequência do tom. Note a maior sensibilidade para tons em 3.5 kHz

Contornos de mesma altura subjetiva



phon: unidade de nível de intensidade percebida

Use a função `genlin()` , definida anteriormente, para testar sua percepção com alguns tons:

```
T = 5; % duraçãoo em s
fs = 44100;
f1 = 3500; % freq. tom 1
s1=genlin(T, f1, f1, fs);
f2 = 250; % freq. tom 2
s2=genlin(T, f2, f2, fs);
% etc.
```

Percepção da voz

- Bandas críticas

O termo banda crítica é usado de diversas formas na literatura. Originalmente foi usado por Fletcher (1940) com respeito ao mascaramento de um tom por ruído branco. Investigava-se quanto do ruído branco era responsável para o mascarameto do tom. Em outras palavras, o objetivo era determinar se toda a banda espectral do ruído contribuía para mascarar o tom ou se há uma certa banda espectral limitada, ou "banda crítica", do ruído que já bastava para o mascaramento completo. Foi descoberto que apenas uma banda crítica, centralizada na frequência do tom já era suficiente para o mascaramento.

- Baixas frequências têm-se bandas mais estreitas (filtros mais seletivos)
- Altas frequências têm-se bandas menos estreitas (filtros menos seletivos)

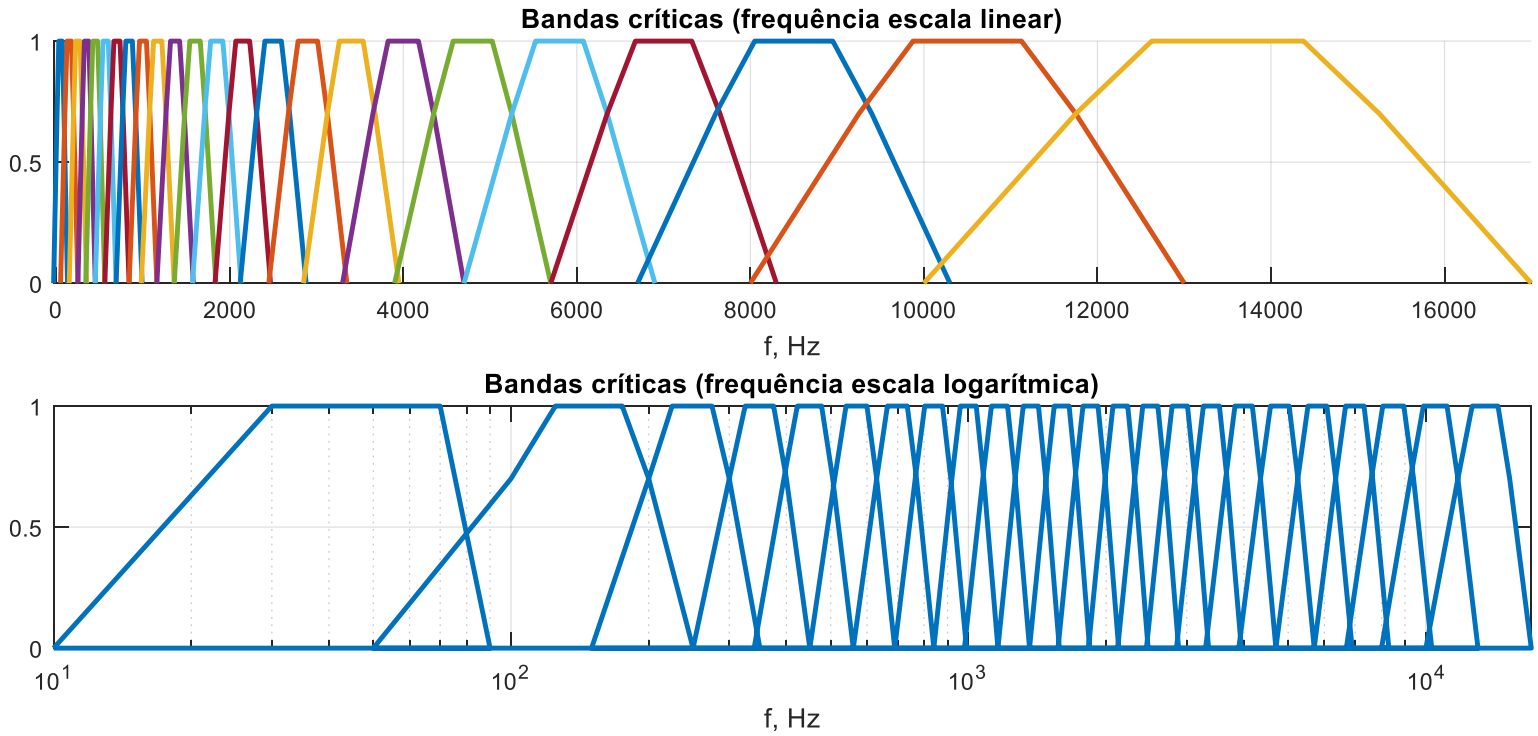
Número	Frequência Central, Hz	Largura de Banda, Hz
1	50	80
2	150	100
3	250	100
4	350	100
5	450	110
6	570	120
7	700	140
8	840	150
9	1000	160
10	1170	190
11	1370	210
12	1600	240

Número	Frequência Central, Hz	Largura de Banda, Hz
13	1850	280
14	2150	320
15	2500	380
16	2900	450
17	3400	550
18	4000	700
19	4800	900
20	5800	1100
21	7000	1300
22	8500	1800
23	10500	2500
24	13500	3500

Percepção da voz

- Bandas críticas

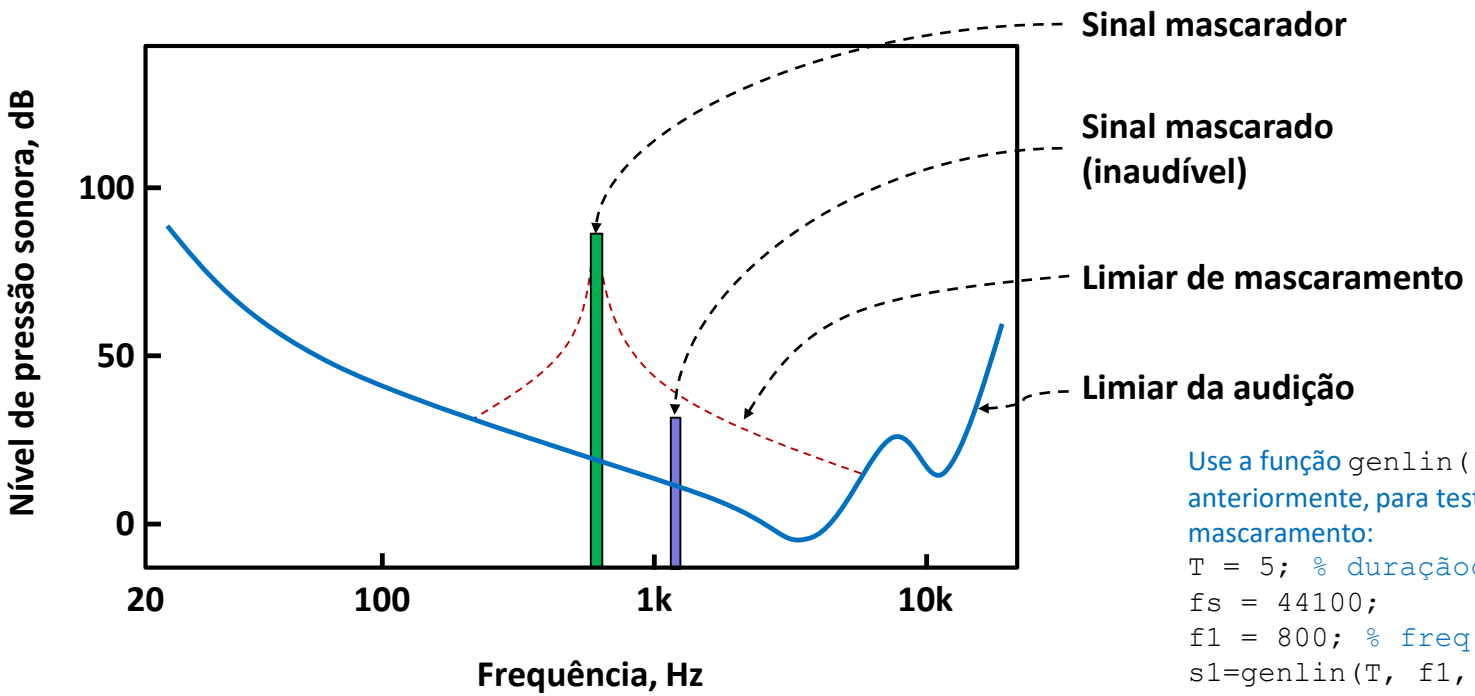
Notar o comportamento exponencial/logarítmico das bandas críticas em função da frequência



Percepção da voz

- Mascaramento

Um sinal mais fraco é completamente mascarado por um sinal mais intenso com conteúdo similar em frequência (próximo em termos de banda crítica). Este fenômeno é convenientemente empregado na codificação **mp3** de forma a comprimir a informação pela eliminação de tons perceptuais desnecessários.



Um tom de 800 Hz com 80 dB mascara um tom em 1.1 kHz com 30 dB

Use a função `genlin()` , definida anteriormente, para testar o mascaramento:

```
T = 5; % duraçãoo em s
fs = 44100;
f1 = 800; % freq. tom 1
s1=genlin(T, f1, f1, fs);
f2 = 1100; % freq. tom 2
s2=genlin(T, f2, f2, fs);
s3 = (s1+5e-2*s2)*.9;
sound(s3, fs)
```

Algumas aplicações

- Microphone array - filtragem espacial e localização de fontes sonoras
- Reconhecimento automático de fala
- Autenticação de locutor – biometria
- TTS (text-to-speech) - síntese de voz em leitura de textos
- Voice morphing - modificação de voz
- Codificação - representação mais eficiente para transmissão/armazenam.
- Speech enhancement - redução de ruído
- Aplicações em música – análise/transformação de música/canto
- Estimação de faixa etária/sentimento/stress

Algumas referências bibliográficas

- L.R. Rabiner and R.W. Schafer. Theory and Applications of Digital Speech Processing. Prentice-Hall; 2010
- A.V. Oppenheim, R.W. Schafer. Discrete-Time Signal Processing. 3rd edition. Prentice Hall. 2011
- L.R. Rabiner and B.H. Juang. Fundamentals of Speech Recognition. Prentice Hall; 1993
- X. Huang, A. Acero and H.W. Hon. Spoken Language Processing: A Guide to Theory, Algorithm and System Development. Prentice Hall; 2001
- P. Taylor. Text-to-Speech Synthesis. Cambridge University Press; 2009
- I.J. Tashev. Sound Capture and Processing. John Wiley; 2009

Laboratório

- Geração de tom (cossenoide)

Transformar os passos descritos anteriormente na geração de tom na frequência `f1` em uma função do MATLAB, como descrita abaixo:

```
function tom = geracos(A, f1, ph, fs, T)
% Entrada:
%          A  : amplitude
%          f1 : frequência desejada para o tom, Hz
%          ph : fase, rad
%          fs : frequência de amostragem, Hz
%          T  : duração, s
% Saída:
%          tom: sinal com as características dos parâmetros acima
%               A * cos(2*pi*f1*n/fs + ph), n = 0, 1, ..., T*fs
%
```

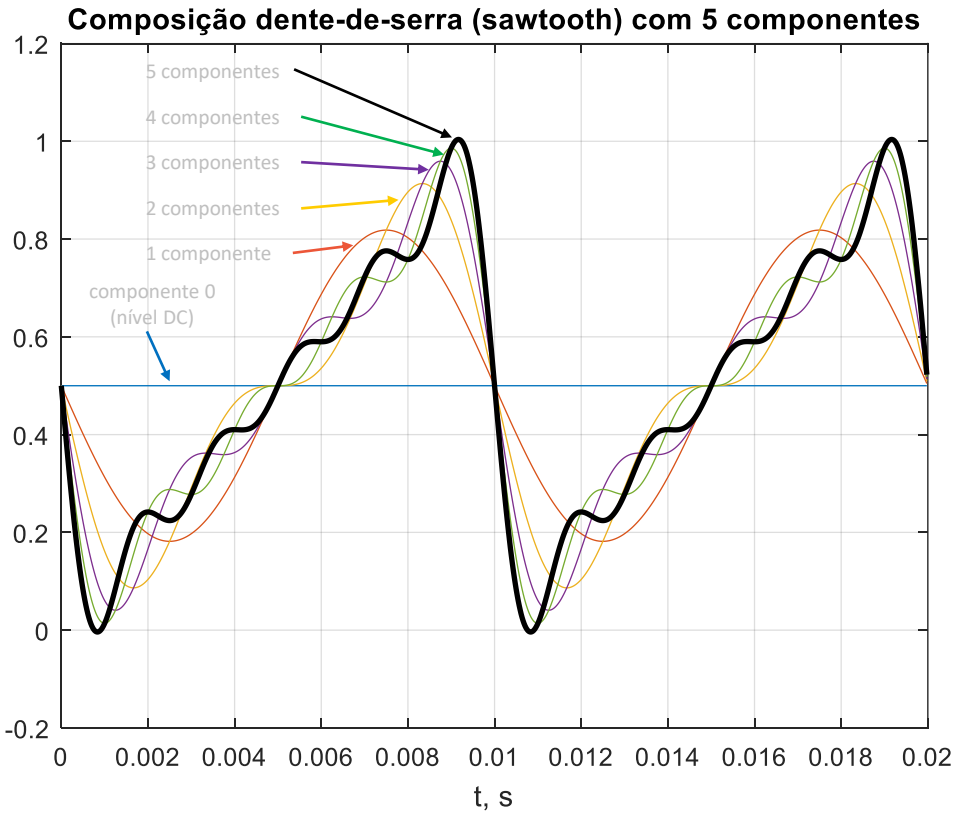
Nota: para padronizar, usar função cosseno, `cos()`

Laboratório

- Geração de tom (cossenoide)

Se a sua implementação do gerador de tom cossenoidal, como descrito na página anterior, estiver correta, o seguinte código gerará a composição via série de Fourier da sequência dente-de-serra (sawtooth)

```
fs = 44100;          % freq amostragem, Hz
f1 = 100;            % freq fundamental, Hz
T = 2;               % duracao, s
N = floor(fs/f1);    % pontos por periodo
n = 0:floor(T*fs);   % indice das amostras
t = n/fs;            % escala de tempo, s
C = 5;      % no. componentes para serie
P = 2;      % no. de periodos para visualizacao
ind = 1:P*N; % indice eixo x para visualizacao
figure(1),
s = geracos(1/2, 0, 0, fs, T); % nivel DC
plot(t(ind), s(ind));
hold on,
for i = 1:C, % soma C componentes a s
   s = s + geracos(1/pi/i, i*f1, pi/2, fs, 2);
   plot(t(ind), s(ind));
end
plot(t(ind), s(ind), 'k', 'LineWidth', 2);
msg = 'Composição dente-de-serra (sawtooth)';
msg = sprintf('%s com %d componentes', msg, C);
title(msg); xlabel('t, s');
grid, hold off,
```



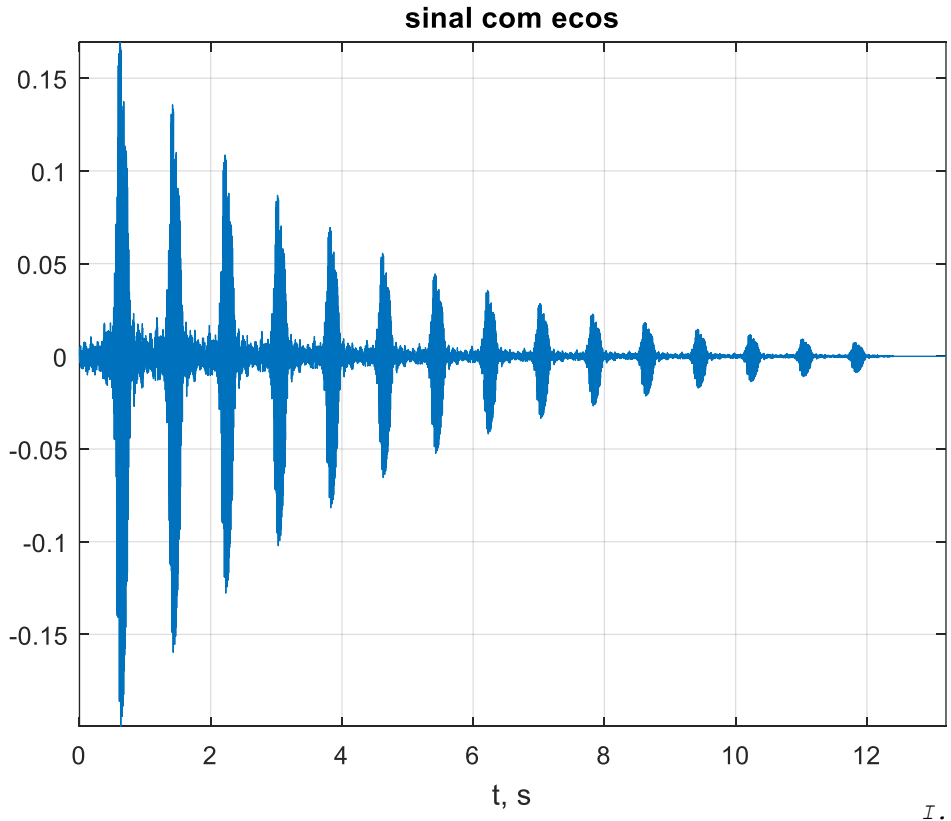
Posteriormente, varie o número de componentes, `C`, e experimente a geração de outras formas de onda

Laboratório

- Introdução de eco

Iremos a seguir criar um novo sinal composto por ecos do sinal original, no caso o sinal de voz do arquivo **seis.wav** . A ideia é simplesmente somar repetições do mesmo sinal atrasadas pelos intervalos de eco desejado. Isso é facilmente feito pela convolução do sinal original por um trem de impulsos, como descrito a seguir. Reproduza os passos abaixo e escute o sinal resultante. Entenda e varie a receita abaixo para poder controlar os intervalos de eco, assim como as amplitudes dos sinais ecoados.

```
[y, fs] = audioread('seis.wav');
T = 800; % echo em ms
N = T * fs/1000;
A = 1;    % amplitude do echo
pulse = [A zeros(1, N)];
trem = pulse;
for i=1:14,
    pulse = .8*pulse;
    trem = [trem pulse];
end
s = conv(trem, y);
sound(s, fs);
```



Laboratório

- Inversão no tempo

Importe as amostras do arquivo **seis.wav** e escute-as de trás para frente. Crie um novo arquivo, **sies.wav**, com as amostras invertidas no tempo.
Dica: entenda o código abaixo

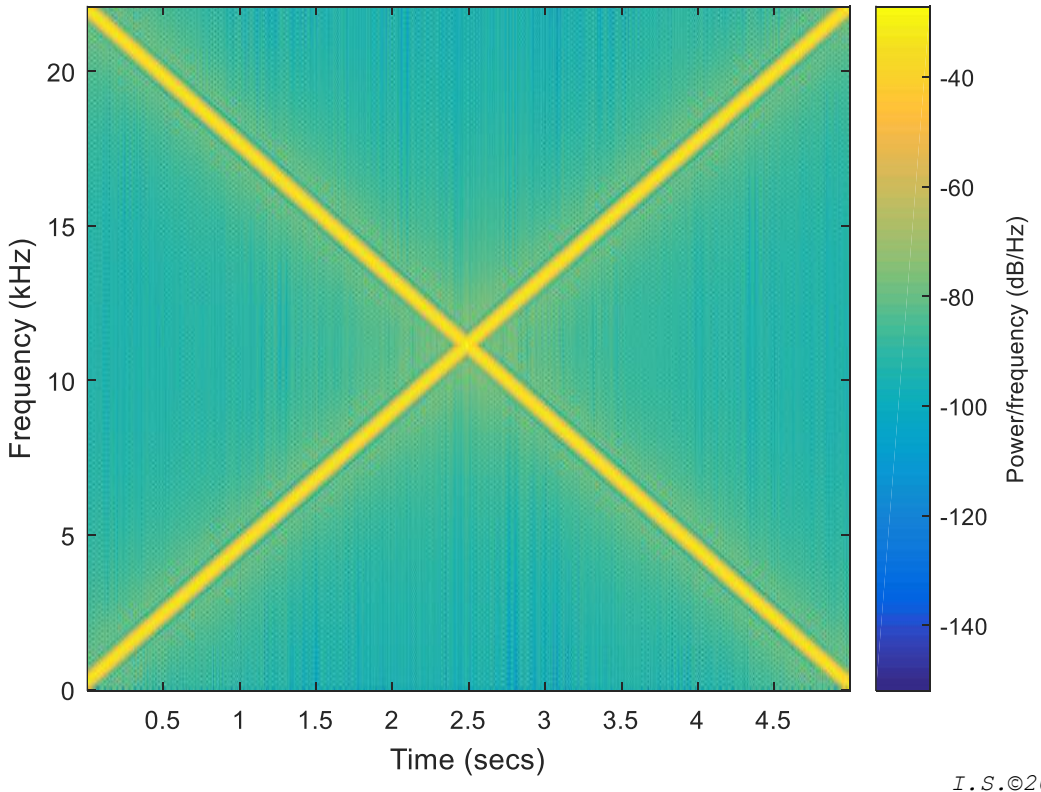
```
[y, fs] = audioread('seis.wav');
w = y(end:-1:1);
```

Laboratório

- Áudio estéreo

Crie um arquivo **.wav** de 2 canais (estéreo) no qual em um canal se tem um tom que varia sua frequência linearmente de 100 Hz a 22 kHz em 5 segundos e no outro canal um tom que varia linearmente de 22 kHz a 100 Hz em 5 segundos. Use procedimentos análogos aos apresentados anteriormente. Adote frequência de amostragem de 44100 Hz.

Espectrograma da soma dos sinais:



Laboratório

- Redução do ruído no sinal senoidal com SNR = 5 dB

  Reproduza o processo de filtragem realizado anteriormente para outras especificações de filtro: teste ordens e frequências diferentes.

Processamento do áudio

- Energia e taxa de cruzamentos por zero (TCZ)

Dado o arquivo de áudio **seis.wav**, compute a energia, E, e a TCZ, Z, desse sinal em segmentos de 20 ms a cada 10 ms.

Dicas:

1. De acordo com a notação usada anteriormente, determine
   T, S, dS, M e N
2. Retire o nível médio do sinal (apenas recomendação geral, pois o sinal em questão já teve seu nível médio removido)
3. Entre em um loop, que sabemos terá M passadas, e para cada passada i compute a energia do segmento i, $E(i)$, e a correspondente taxa de cruzamentos por zero, $Z(i)$.
4. Apresente os gráficos de E e Z. Deve resultar nos gráficos vistos quando foram abordados os conceitos de segmentação, energia e TCZ.

Laboratório

- Função de autocorrelação

Implemente a função de autocorrelação e chegue à mesma figura de $R(n)$ apresentada anteriormente.
Para se chegar ao mesmo gráfico, obviamente, use o mesmo trecho de sinal usado anteriormente.
Dica: faça `x = y(ind)`, como no código correspondente, e trabalhe com `x` para a reprodução do gráfico de $R(n)$, após sua implementação.