centro universitário FEI

PEL 304
Aplicações em Processamento de Sinais

https://fei.edu.br/~isanches/pel304/pel304-aulas_1_e_2.pdf

2020

Programação

- Objetivos da disciplina

  Apresentar aos alunos de Pós-Graduação aplicações na área de Processamento de Sinais e expor concretamente as diversas formas de representação, manipulação e transformação de sinais e imagens para o seu processamento eficiente e obtenção da informação e resultados finais desejados. A disciplina também visa dar uma visão generalista da área e contribuir para que o aluno se posicione e foque mais acertadamente sua pesquisa futura

- Metodologia

  Em aulas expositivas, aplicações em processamento de sinais serão apresentadas aos alunos, junto com os conceitos fundamentais necessários à compreensão

Programação

- Professores

Prof Aldo Artur Belardi
4353-2910 ramal 2184
belardi@fei.edu.br
sala K5-05

Prof Carlos Eduardo Thomaz
4353-2910 ramal 2183
cet@fei.edu.br
sala K5-01

Prof Ivandro Sanches
4353-2910 ramal 2208
isanches@fei.edu.br
sala K5-04

Profa Maria Claudia Ferrari de Castro
4353-2910 ramal 2224
mclaudia@fei.edu.br
sala K5-04

Prof Paulo Sergio Rodrigues
4353-2910 ramal 2181
psergio@fei.edu.br
sala K5-03

Programação

- Conteúdo (Professor – dia da aula)
  - Sinais e sistemas discretos (IS-03/mar)
  - Computação em imagens médicas, mapeamento cerebral e eye-tracking (CET-10/mar)
  - Análise de cenas naturais com modelos de engenharia (PSR-17/mar)
  - Reconhecimento de padrões em biopotenciais (MCFC-24/mar)
  - Métodos numéricos aplicados em processamento de sinais (AAB-31/mar)
  - Aplicação das wavelets e seus aspectos computacionais (AAB-07/abr)
  - Sinal de voz e reconhecimento automático de fala (IS-14/abr)
  - Percepção, modelagem e reconstrução de faces (CET-28/abr)
  - Reabilitação motora: restauração de movimento das mãos (MCFC-05/mai)
  - Análise de cenas naturais com modelos bio-inspirados (PSR-12/mai)
  - Discussão e definição de temas para apresentação(ao longo do período)
  - Semana para preparo/finalização das apresentações (sem aula - 19/mai)
  - Apresentação de trabalhos pelos alunos (última aula - 26/mai)

Programação

- Calendário, 1º período de pós-graduação, 2020

March

Su	Mo	Tu	We	Th	Fr	Sa
1	**2**	**3**	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

April

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

May

Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	**26**	27	28	**29**	30
31						

02/03 – início do 1° período
03/03 – primeira aula
21/04 – Tiradentes
26/05 – última aula
29/05 – término do 1° período

Programação

- Avaliação

  O aluno será avaliado por:

  - Exercícios solicitados na primeira aula de cada professor. A solução deverá ser entregue em aula futura do professor correspondente
  - Elaboração de trabalho escrito e sua apresentação a uma banca de professores e colegas da disciplina

- Cronograma previsto

PEL304 - Aplicações em Processamento de Sinais

Terças-feiras

2020													
Dia	03/mar	10/mar	17/mar	24/mar	31/mar	07/abr	14/abr	21/abr	28/abr	05/mai	12/mai	19/mai	26/mai
Aula	1	2	3	4	5	6	7		8	9	10	11	12
Professor	Ivandro	Carlos*	Paulo	Claudia*	Aldo	Aldo	Ivandro	*Tiradentes*	Carlos*	Claudia*	Paulo	Preparo	Apresentações

Nota = 50% + 5 * 10%

Apresentação

Exercícios

Entrega do exercício que foi solicitado em aula anterior desse professor

* início da aula dos professores Claudia e Carlos, excepcionalmente, às 19h00

Programação

- Avaliação (continuação)

O trabalho, de 1 ou mais páginas, se baseará em tema escolhido pelo aluno e relacionado à área de processamento de sinais, obedecendo ao formato do SICFEI*. Listam-se algumas possibilidades:

1. trabalho baseado em artigo científico publicado (http://www.periodicos.capes.gov.br, https://ieeexplore.ieee.org, etc.)
2. basear-se em ideia do próprio aluno e/ou em trabalho em processamento de sinais em andamento
3. o aluno pode procurar um dos professores da disciplina que indicará trabalho publicado ou um experimento que o aluno poderá reproduzir e apresentá-lo
4. o aluno é de outro programa de pós-graduação, por exemplo mecânica ou administração. Neste caso, o aluno é encorajado a buscar por trabalho científico na sua área de atuação em que alguma técnica de processamento de sinais seja empregada

Obs: Enviar a versão eletrônica do trabalho ao email de cada professor com pelo menos 1 dia de antecedência da apresentação. O arquivo não deverá ser maior do que 2 Mbytes. O correspondente arquivo da apresentação não é para ser enviado aos professores. Assunto (*subject*) da mensagem: **PEL304 – Trabalho Escrito**

* modelo: https://fei.edu.br/sites/sicfei/2019/modelo_resumo_2019.doc

Programação

- Bibliografia
  - L. R. Rabiner and R. W. Schafer. **Theory and Applications of Digital Speech Processing**. Prentice-Hall; 2010
  - B. Gold, N. Morgan, D. Ellis. **Speech and Audio Signal Processing**, 2nd edition. Wiley; 2011
  - K. Fukunaga. **Introduction to Statistical Pattern Recognition**, 2nd edition. Morgan Kaufmann; 1990
  - R. A. Johnson and D. W. Wichern. **Applied Multivariate Statistical Analysis**, 4th edition. Prentice Hall; 1998
  - P. A. Morettin. **Ondas e Ondaletas**. EDUSP; 1999
  - H. M. de Oliveira. **Análise de Sinais para Engenheiros, Uma abordagem via Wavelets**. Brasport; 2009
  - R. O. Duda, P. E. Hart and D. G. Stork. **Pattern Classification**, 2nd edition. Wiley; 2000
  - R. C. Gonzalez and R. E. Woods. **Digital Image Processing**, 3rd edition. Pearson; 2007
  - S. D. J. Barbosa e B. S. Silva. **Interação Humano-Computador**. Elsevier; 2010
  - K. J. Blinowska and J. Zygierewicz. **Practical Biomedical Signal Analysis Using MATLAB**. CRC Press; 2012

centro universitário FEI

PEL 304
Aplicações em Processamento de Sinais

Ivandro Sanches

Aula 1: Sinais e sistemas discretos

2020

Processamento de Sinais

- Referência clássica em processamento de sinais discretos no tempo, disponível na biblioteca:

  - A. V. Oppenheim and R. W. Schafer. **Processamento em tempo discreto de sinais**. Pearson, 3ª edição; 2013

Tópicos - Aula 1 e 2

- Aula 1
  - Ferramentas
  - Sinais Discretos
  - Sistemas Discretos
  - Análise Espectral com DFT
  - Espectrograma do Sinal de Voz
  - Exercício Proposto

  Geração de um tom e sua análise por transformada discreta de Fourier (DFT: discrete Fourier transform) via MATLAB e Python. Adição de um segundo tom. Adição de ruído. Projeto de filtro digital para recuperar o primeiro tom e análise espectral com DFT do do sinal filtrado. Apresentação de espectrograma do sinal de voz via diferentes programas. Enunciado de exercício a ser entregue na Aula 2

- Aula 2
  - O Sinal de Voz
  - Reconhecimento Automático de Fala
  - Outras Aplicações em Processamento de Voz
  - Entrega do Exercício Proposto

  Características do sinal de voz. Componentes de um sistema de reconhecimento automático de fala. Apresentação prática dos processos de treinamento e teste de um sistema criado com a ferramenta HTK (Hidden Markov Toolkit)

Ferramentas

- Aplicações em Processamento de Sinais: como o próprio nome da disciplina sugere, sinais reais devem ser processados de forma a se extrair a informação desejada e utilizá-la para um fim específico. Por exemplo, no reconhecimento automático de fala, um sinal de voz é processado, extraem-se as palavras contidas no sinal e conforme o sentido resultante (semântica) uma ação é executada

- Esse processo depende fortemente do uso de ferramentas para manipulação dos sinais. A representação básica de sinais reais em computador nada mais é do que uma sequência de números, seja em vetores, matrizes ou alguma outra estrutura conveniente. Uma ferramenta importante e muito utilizada no ambiente acadêmico é MATLAB (MATrix LABoratory). MATLAB foi desenvolvido em C, C++ e Java. Clones do MATLAB (Octave, Scilab, Euler) assim como Python e Julia Programming Language também são boas opções

Ferramentas - MATLAB

- Vetores e matrizes no MATLAB - Rudimentos

Sinais amostrados transformam-se em vetores. No MATLAB, o comando

```
x = [ 4 3 7 -9 1 ]
```

cria um vetor linha de 5 elementos. É importante saber que os índices dos vetores iniciam em 1. Assim, x(1) = 4, x(2) = 3. Note também que: x(end) = 1, x(2:4) = [3 7 -9], x(2:end-1) = [3 7 -9], etc.

O operador apóstrofe aplica o conjugado transposto. Assim

```
x = x'
```

Transforma x em vetor coluna.

```
x =

   4
   3
   7
  -9
   1
```

Ferramentas - MATLAB

- Vetores e matrizes no MATLAB - Rudimentos

A orientação de vetores em colula é preferível para sinais de 1 canal, pois ela se estende naturalmente para o caso multicanal. Dados multicanais têm cada canal representado em uma coluna. Nessa matriz de dados, tem-se em cada linha uma amostragem. Um sinal de 3 canais que consiste de x, 2x, e x/π é (obs.: 'π' no MATLAB é 'pi')

```
y = [ x 2*x x/pi ]
```

Que resulta em

```
y =

    4.0000    8.0000    1.2732
    3.0000    6.0000    0.9549
    7.0000   14.0000    2.2282
   -9.0000  -18.0000   -2.8648
    1.0000    2.0000    0.3183
```

Convém exercitar e explorar operações entre vetores e matrizes:
x'*x (igual a 156) , x*x' (matriz 5x5), x.*x, x./x, etc.

Ferramentas - MATLAB

- Vetores e matrizes no MATLAB - Rudimentos

Vetores podem ser complexos. Para isso, note que o MATLAB inicia as variáveis **i** e **j** com a raiz quadrada de -1 (unidade imaginária). Assim, por exemplo, sejam os vetores reais **a** e **b** abaixo

```
a = [ -1 0 2 ]'; b = [ 1 2 3 ]';
```

Pode-se construir o vetor complexo **c** da seguinte forma

```
c = a + 1i*b
```

```
c =

  -1.0000 + 1.0000i
       0 + 2.0000i
   2.0000 + 3.0000i
```

Note o resultado das seguintes operações com números, vetores ou matrizes complexos:

c' (transposto conjugado), c.' (apenas transposto), conj(c), real(c), imag(c), abs(c), angle(c), etc.

Note, por exemplo, que a notação 2**i**, ou 2**j**, também pode ser empregada

Ferramentas - MATLAB

- Vetores e matrizes no MATLAB - Rudimentos

Vetores também tem uma interpretação importante no MATLAB. Podem ser considerados como os coeficientes de polinômios. Por exemplo, sejam os seguintes polinômios em **x, p(x) = x^2+3x-1** e **q(x)=2x+1**. Os vetores **p** e **q** equivalentes a esses polinômios são:

```
p = [ 1 3 -1 ]; q = [ 2 1 ];
```

Note a aplicação das seguintes funções. Raízes de **p**

```
raizes=roots(p);
```

Obtenção dos coeficientes polinomiais a partir das raízes

```
poly(raizes)
```

Multiplicação de polinômios

```
pq = conv(p,q);
```

Divisão de polinômios

```
deconv(pq, q)
```

Ferramentas - Python (jupyter-qtconsole)

- Vetores e matrizes em Python - Rudimentos

Jupyter QtConsole

File Edit View Kernel Window Help

```
Jupyter QtConsole 4.3.1
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit
(AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import numpy as np
   ...:
   ...: # Sinais amostrados transformam-se em vetores. Em Python, o comando

In [2]: x = np.array([ 4, 3, 7, -9, 1, ])

In [3]: x
Out[3]: array([ 4,  3,  7, -9,  1])

In [4]: # cria um array de 5 elementos.
   ...: # É importante saber que os índices dos vetores iniciam em 0.
   ...: # Assim, x[0] = 4, x[1] = 3.
   ...: # Note também que: x[-1] = 1,
   ...: #                  x[1:4] = [3, 7, -9],
   ...: #                  x[1:-1] = [3 7 -9], etc.
   ...:
   ...: # criando-se uma array bidimensional:
   ...:

In [5]: y = np.array([ x, 2*x, x/np.pi])

In [6]: y
Out[6]:
array([[  4.        ,   3.        ,   7.        ,  -9.        ,
          1.        ],
       [  8.        ,   6.        ,  14.        , -18.        ,
          2.        ],
       [  1.27323954,   0.95492966,   2.2281692 ,  -2.86478898,
          0.31830989]])

In [7]: # y transposto:
   ...:
   ...: y.T # ou np.transpose(y)
Out[7]:
array([[  4.        ,   8.        ,   1.27323954],
       [  3.        ,   6.        ,   0.95492966],
       [  7.        ,  14.        ,   2.2281692 ],
       [ -9.        , -18.        ,  -2.86478898],
       [  1.        ,   2.        ,   0.31830989]])
```

Jupyter QtConsole

File Edit View Kernel Window Help

```
In [8]: # Equivalente a x'*x no MATLAB: (assume-se x vetor coluna e real)
   ...:
   ...: np.dot(x, x) # ou: x.dot(x) para x real
Out[8]: 156

In [9]: # Equivalente a x*x' no MATLAB: (assume-se x vetor coluna e real)
   ...:
   ...: np.outer(x, x)
Out[9]:
array([[ 16,  12,  28, -36,   4],
       [ 12,   9,  21, -27,   3],
       [ 28,  21,  49, -63,   7],
       [-36, -27, -63,  81,  -9],
       [  4,   3,   7,  -9,   1]])

In [10]: # Equivalente a x.*x no MATLAB:
    ...:
    ...: x*x
Out[10]: array([16,  9, 49, 81,  1])

In [11]: # Equivalente a x./x no MATLAB:
    ...:
    ...: x/x
Out[11]: array([1., 1., 1., 1., 1.])

In [12]: # Números/vetores complexos
    ...:
    ...: a = np.array([-1, 0, 2])
    ...: b = np.array([1, 2, 3])
    ...: c = a + 1j*b
    ...:

In [13]: c
Out[13]: array([-1.+1.j,  0.+2.j,  2.+3.j])

In [14]: np.conj(c)
Out[14]: array([-1.-1.j,  0.-2.j,  2.-3.j])

In [15]: np.real(c)
Out[15]: array([-1.,  0.,  2.])

In [16]: np.imag(c)
Out[16]: array([1., 2., 3.])

In [17]: # idem np.abs(c), np.angle(c), etc.
```

Ferramentas - Python (jupyter-qtconsole)

- Vetores e matrizes em Python - Rudimentos

Jupyter QtConsole

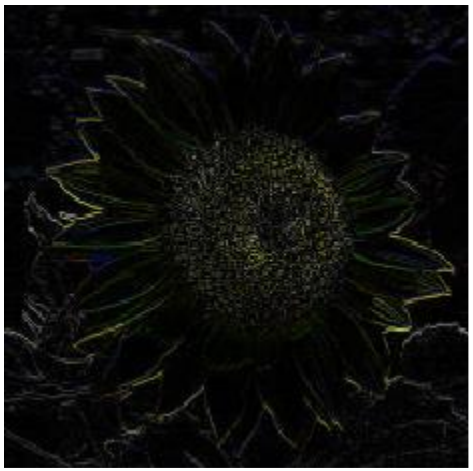
File Edit View Kernel Window Help

```
In [18]: # vetores como polinômios
    ...:
    ...: p = np.array([1, 3, -1])
    ...: q = np.array([2, 1])
    ...:

In [19]: # raízes do polinômio p (1*x² + 3*x - 1)
    ...: np.roots(p)
Out[19]: array([-3.30277564,  0.30277564])

In [20]: # coeficientes do polinômio dadas as raízes
    ...: np.poly(np.roots(p))
Out[20]: array([ 1.,  3., -1.])

In [21]: # multiplicação de polinômios
    ...: np.convolve(p, q)
Out[21]: array([ 2,  7,  1, -1])

In [22]: # divisão de polinômios (deconvolve() em scipy.signal)
    ...: from scipy import signal
    ...: signal.convolve(p, q)
    ...:
Out[22]: array([ 2,  7,  1, -1])

In [23]: signal.deconvolve(signal.convolve(p, q), q)
Out[23]: (array([ 1.,  3., -1.]), array([0., 0., 0., 0.]))
          ^^^^^^^^^^^^^^^^^^^^^^  ^^^^^^^^^^^^^^^^^^^^^^
          Quociente = p           Resto = 0

In [24]:
```

Sinais Discretos - MATLAB

- **Exemplo**: processamento rudimentar de uma imagem no MATLAB

```
% simples alteracao de imagem
img = imread('sunflower.jpg');
imgd = double(img);

maxi = 255; % max valor pixel
figure, subplot(211),
imshow(imgd/maxi);
imgd1 = abs(diff(imgd))/maxi;
subplot(212), imshow(imgd1);
```



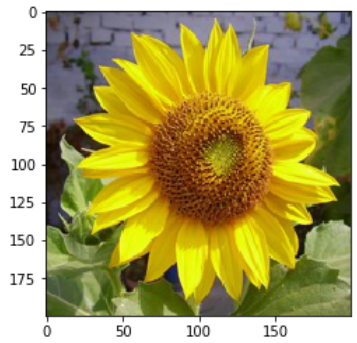
Sinais Discretos - Python

- **Exemplo**: processamnto rudimentar de imagem em Python[1]

Jupyter QtConsole

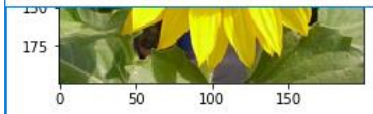
File Edit View Kernel Window Help

```
Jupyter QtConsole 4.3.1
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC
v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...: from matplotlib.pyplot import imread
   ...:

In [2]: img = imread('sunflower.jpg').astype(np.float)
   ...: maxi = 255; R = len(img); C = len(img[0])
   ...: plt.imshow(img/maxi)
   ...: plt.show()
   ...:
```

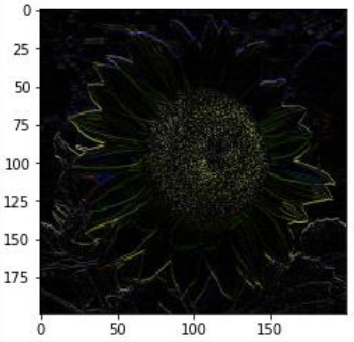


Jupyter QtConsole

File Edit View Kernel Window Help




```
In [3]: imgd = [[[abs((img[i+1,j,k] - img[i,j,k])/maxi)
   ...:                 for k in range(3)] for j in range(C)]
   ...:                 for i in range(R-1)]
   ...: # idem acima: imgd = abs(np.diff(img, axis=0)) / maxi
   ...: plt.imshow(imgd)
   ...: plt.show()
   ...:
```



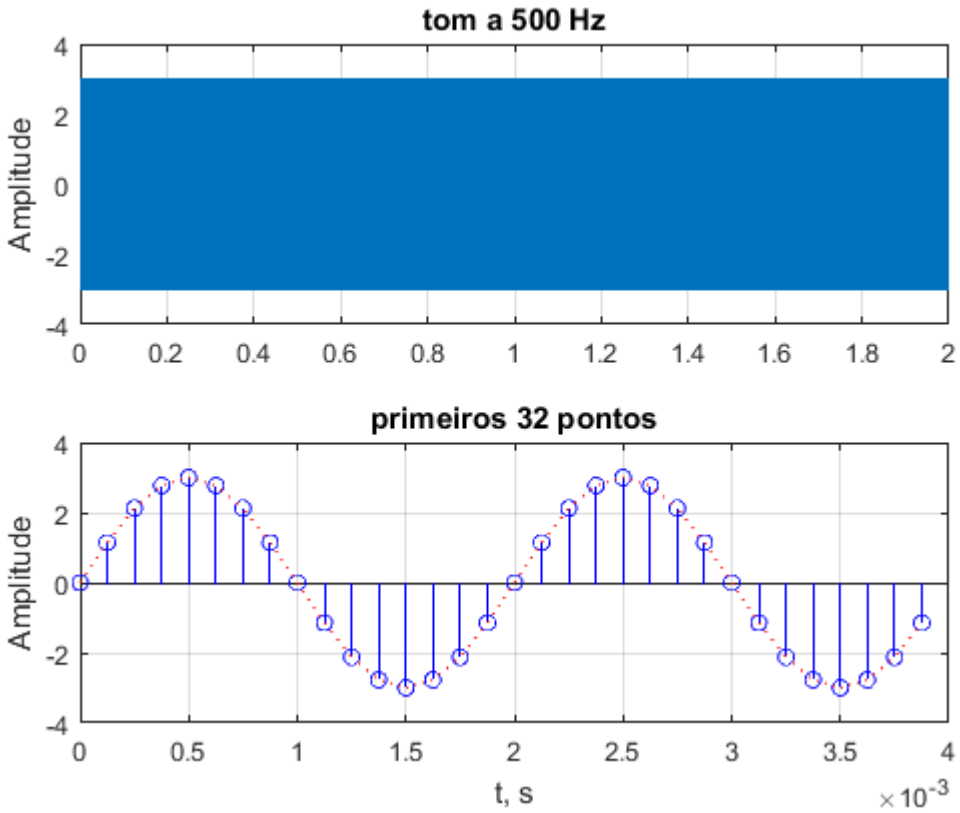

```
In [4]:
```

1 jupyter-qtconsole

Sinais Discretos - MATLAB

- **Exemplo**: geração de um tom de `f1` Hz na frequência de amostragem `fs` Hz

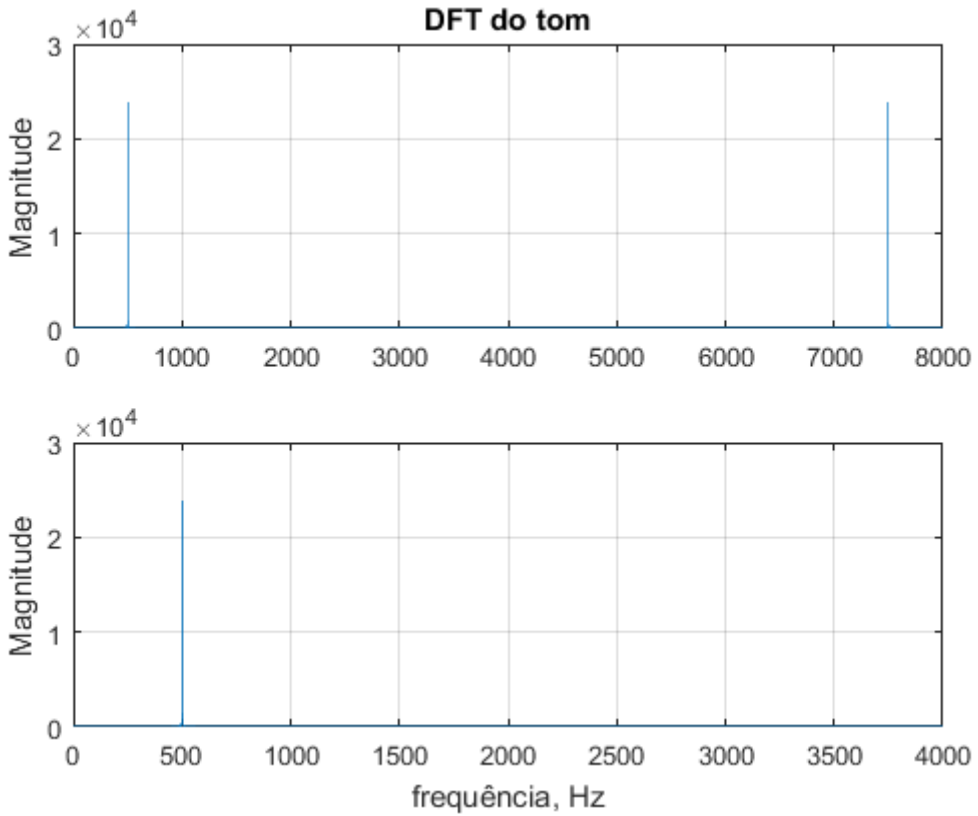
```
% geração de tom f1 a fs
f1 = 500; % freq. do sinal, Hz
fs = 8000; % freq. de amostragem, Hz
A1 = 3; % amplitude do tom
T = 2; % duracao em segundo
t = (0:1/fs:T)'; % indices de tempo
tom = A1 * sin(2*pi*f1*t);
subplot(211),
plot(t, tom);
title(sprintf('tom a %d Hz', f1));
grid; ylabel('Amplitude');
subplot(212),
N = 32; % numero de pontos para plot
stem(t(1:N), tom(1:N),'b');
hold on
plot(t(1:N), tom(1:N), 'r:');
title(sprintf('primeiros %d pontos', N));
grid; xlabel('t, s');
ylabel('Amplitude');
hold off
```



Sinais Discretos - MATLAB

- Transformada Discreta de Fourier (DFT*) do tom da página anterior

```
% DFT
tf = abs(fft(tom));
N = length(tf);
NN = floor(N/2);
f = (0:N-1)/N*fs;
subplot(211),
plot(f, tf);
grid,
ylabel('Magnitude');
title('DFT do tom');
subplot(212),
plot(f(1:NN), tf(1:NN));
grid,
xlabel('frequência, Hz');
ylabel('Magnitude');
```

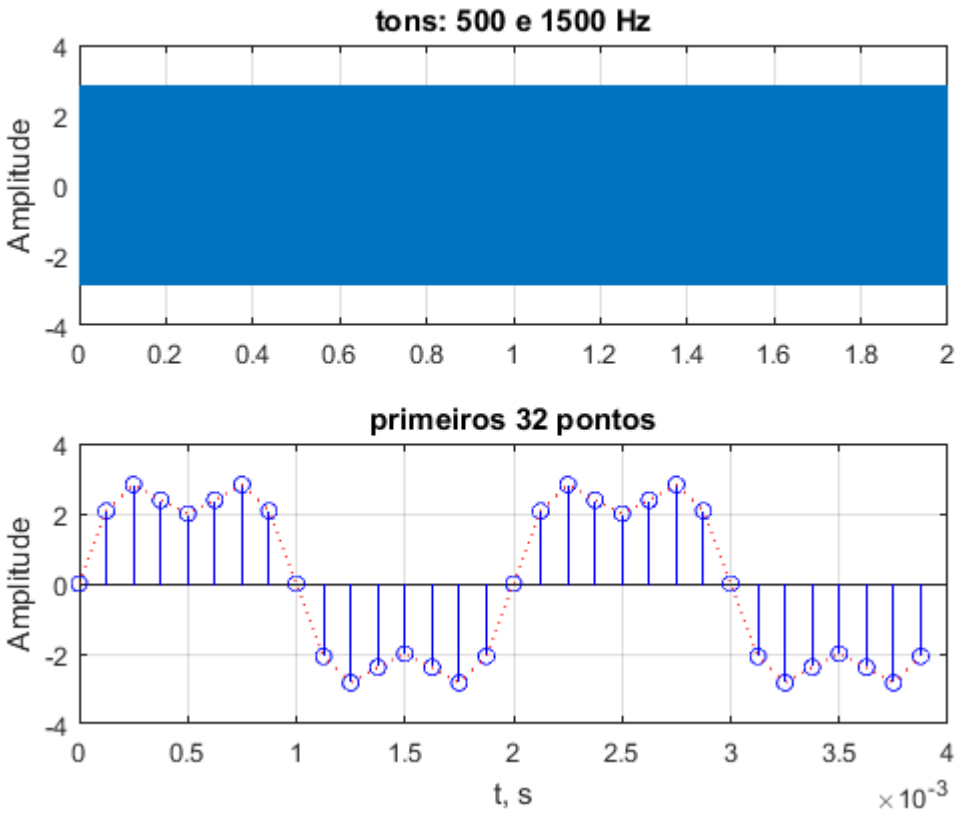


* DFT - Discrete Fourier Transform

Sinais Discretos - MATLAB

- Adição de tom em `f2` Hz ao anterior

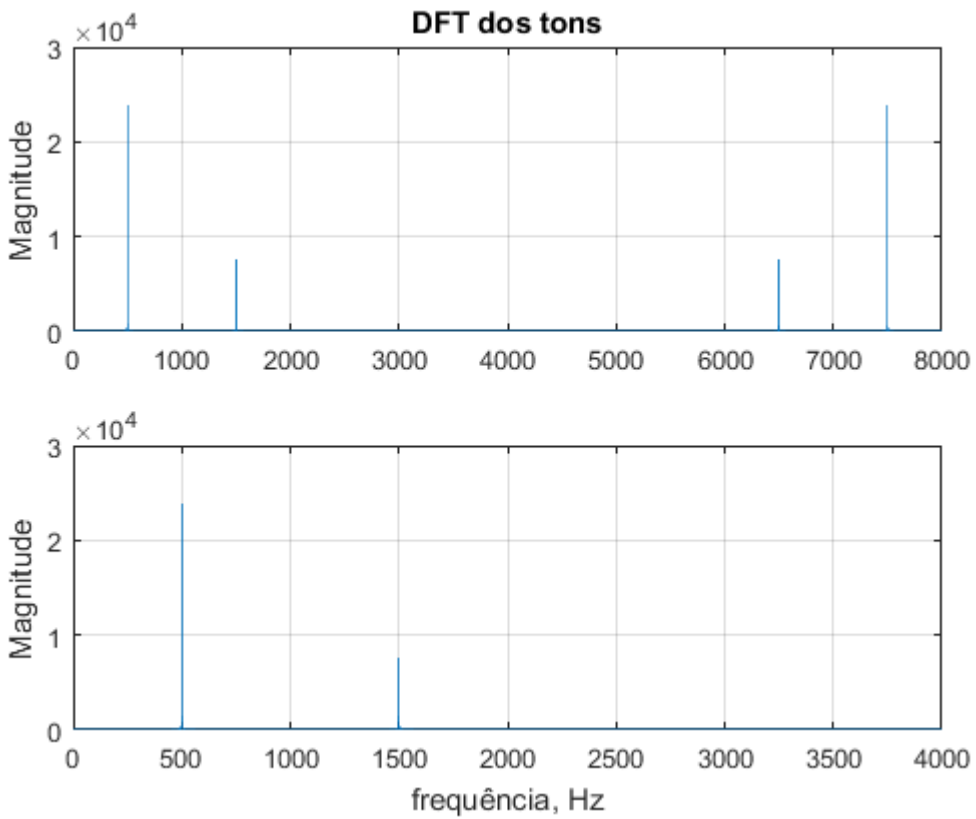
```
% adicionando tom em f2 Hz
f2 = 1500; % freq. do sinal, Hz
A2 = 1; % amplitude do novo tom
T = 2; % duracao em segundo
t = (0:1/fs:T)'; % indices de tempo
tons = tom + A2 * sin(2*pi*f2*t);
subplot(211),
plot(t, tons);
title(sprintf('tons: %d e %d Hz', f1, f2));
grid; ylabel('Amplitude');
subplot(212),
N = 32; % numero de pontos para plot
stem(t(1:N), tons(1:N),'b');
hold on
plot(t(1:N), tons(1:N), 'r:');
title(sprintf('primeiros %d pontos', N));
grid; xlabel('t, s');
ylabel('Amplitude');
hold off
```



Sinais Discretos - MATLAB

- Transformada Discreta de Fourier (DFT) dos tons da página anterior

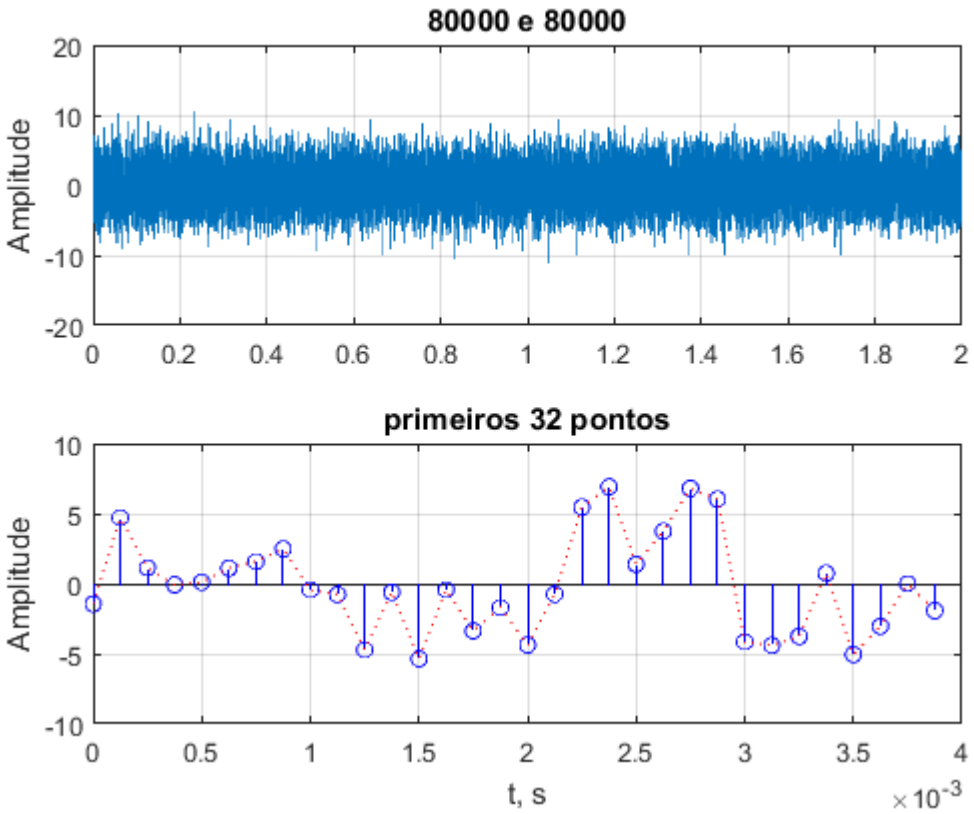
```
% DFT dos tons
tf2 = abs(fft(tons));
N = length(tf2);
NN = floor(N/2);
f = (0:N-1)/N*fs;
subplot(211),
plot(f, tf2);
grid,
ylabel('Magnitude');
title('DFT dos tons');
subplot(212),
plot(f(1:NN), tf2(1:NN));
grid,
xlabel('frequência, Hz');
ylabel('Magnitude');
```



Sinais Discretos - MATLAB

- Adição de ruído a 0 dB (SNR) aos tons

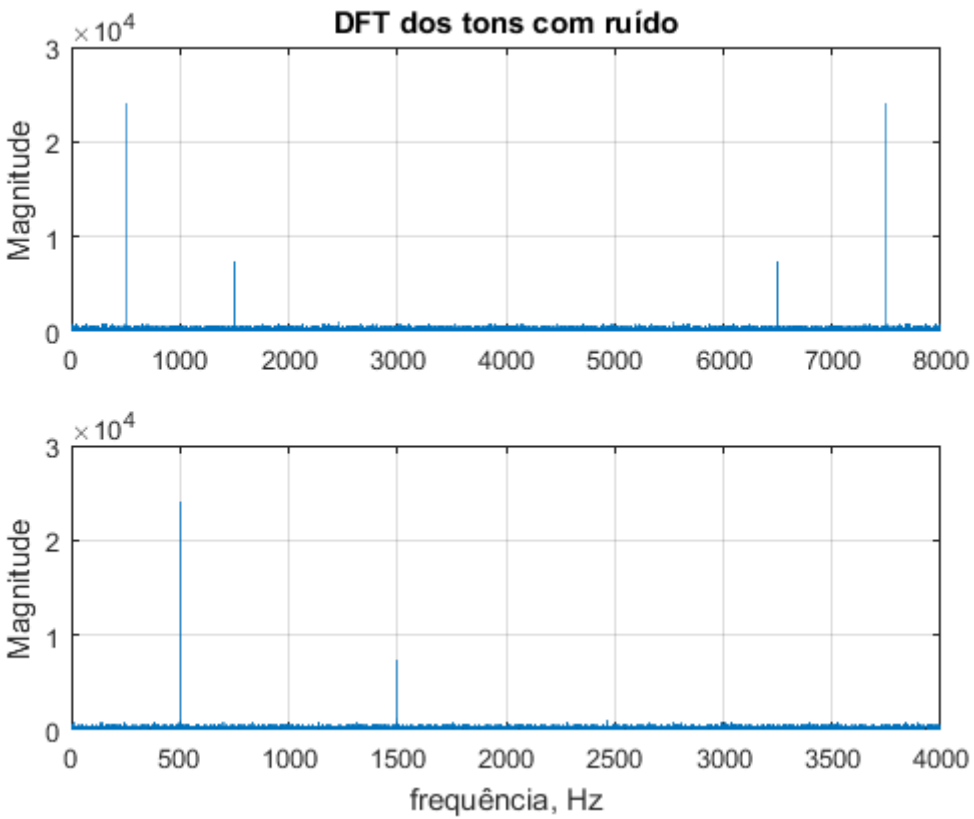
```
% Adicao de ruido a 0 dB (SNR)
rng(1); % semente
r = randn(size(tons));
Er = r'*r; % energia do ruido
Et = tons'*tons; % energia do sinal
r = r * (Et/Er)^0.5; % correcao
Er = r'*r; % nova energia do ruido
tonsR = tons + r; % sinal ruidoso
subplot(211),
plot(t, tonsR);
title(sprintf('%g e %g', Et, Er));
grid; ylabel('Amplitude');
subplot(212),
N = 32; % numero de pontos para plot
stem(t(1:N), tonsR(1:N),'b');
hold on
plot(t(1:N), tonsR(1:N), 'r:');
title(sprintf('primeiros %d pontos', N));
grid; xlabel('t, s');
ylabel('Amplitude');
hold off
```



Sinais Discretos - MATLAB

- Transformada Discreta de Fourier (DFT) dos tons com ruído

```
% DFT dos tons com ruido
tfR = abs(fft(tonsR));
subplot(211),
plot(f, tfR);
grid,
ylabel('Magnitude');
title('DFT dos tons com ruído');
subplot(212),
plot(f(1:NN), tfR(1:NN));
grid,
xlabel('frequência, Hz');
ylabel('Magnitude');
```



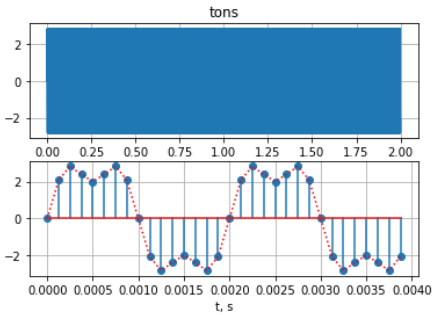
Sinais Discretos - Python

- Repetição em Python[1]

Jupyter QtConsole

File Edit View Kernel Window Help

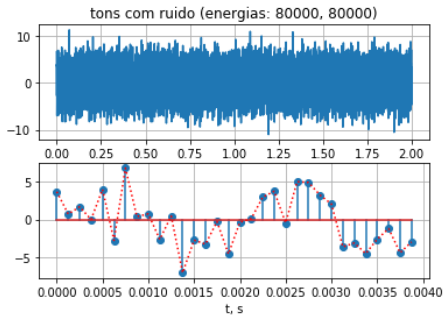
```
Jupyter QtConsole 4.3.1
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit
(AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import numpy as np
   ...: import matplotlib
   ...: import matplotlib.pyplot as plt
   ...: %matplotlib inline
   ...: # geracao de forma de onda
   ...: fs = 8000 # freq. amostragem, Hz
   ...: T = 2 # duracao em segundos
   ...: t = np.arange(fs*T)/fs # tempo
   ...: f1 = 500 # freq tom 1 em Hz
   ...: A1 = 3 # amplitude tom 1
   ...: f2 = 1500 # freq tom 2 em Hz
   ...: A2 = 1 # amplitude tom 2
   ...: tons = A1 * np.sin(2*np.pi*f1*t) + A2 * np.sin(2*np.pi*f2*t)
   ...: N = 32 # pontos para plot
   ...: plt.subplot(211); plt.plot(t,tons)
   ...: plt.title('tons'); plt.grid()
   ...: plt.subplot(212); plt.stem(t[:N],tons[:N])
   ...: plt.plot(t[:N],tons[:N], 'r:')
   ...: plt.grid(); plt.xlabel('t, s')
   ...: plt.show()
   ...:
```



Jupyter QtConsole

File Edit View Kernel Window Help

```
In [2]: # adicao de ruido a 0 dB
   ...: np.random.seed(1) # semente
   ...: r = np.random.randn(len(t)) # ruido
   ...: Er = np.sum(r*r); Et = np.sum(tons*tons)
   ...: r = r * (Et/Er)**.5
   ...: Er = np.sum(r*r) # energia corrigida
   ...: tonsR = tons + r
   ...: plt.subplot(211); plt.plot(t, tonsR)
   ...: m = 'tons com ruido (energias: '
   ...: m = m + "%d, %d)"%(Et, Er)
   ...: plt.grid(); plt.title(m)
   ...: plt.subplot(212)
   ...: plt.stem(t[:N],tonsR[:N])
   ...: plt.plot(t[:N],tonsR[:N], 'r:')
   ...: plt.grid(); plt.xlabel('t, s')
   ...: plt.show()
   ...:
```




```
In [3]:
```

1 jupyter-qtconsole

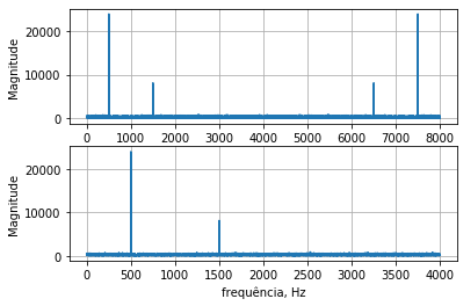
Sinais Discretos - Python

- Repetição em Python

Jupyter QtConsole

File Edit View Kernel Window Help

```
In [3]: # DFT (discrete Fourier Transform)
   ...: tfR = np.fft.fft(tonsR)
   ...: tfR = np.abs(tfR)
   ...: N = len(tfR)
   ...: f = np.arange(N)
   ...: f = f/N*fs
   ...: plt.subplot(211)
   ...: plt.plot(f, tfR)
   ...: plt.grid()
   ...: plt.ylabel('Magnitude');
   ...: plt.subplot(212)
   ...: NN = N//2
   ...: plt.plot(f[:NN], tfR[:NN])
   ...: plt.grid()
   ...: plt.xlabel('frequência, Hz');
   ...: plt.ylabel('Magnitude');
   ...: plt.show()
   ...:
```

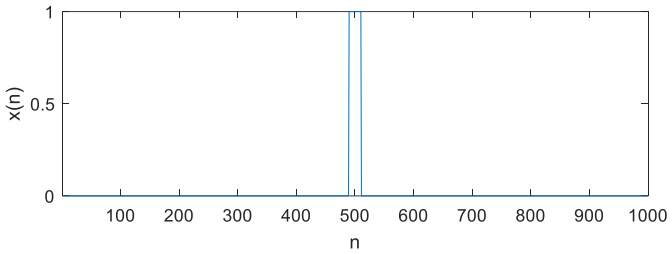


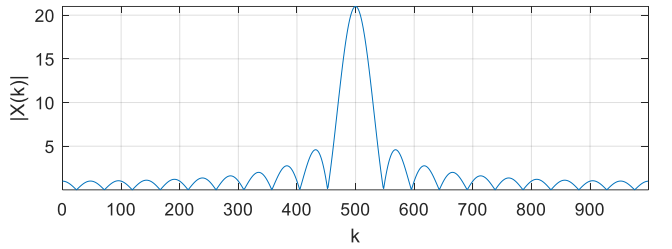

```
In [4]:
```

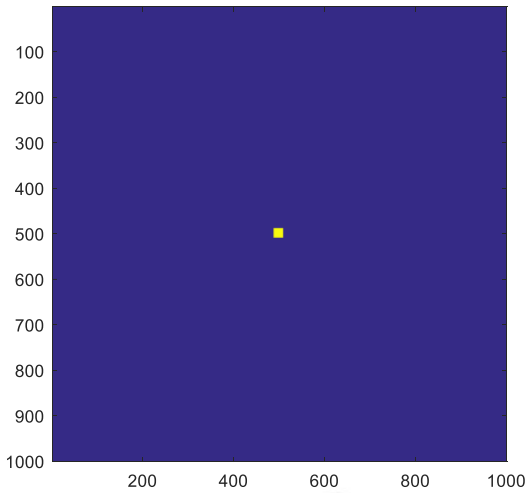
- Note a conveniência de se analisar o sinal no domínio da frequência. Os tons destacam-se do ruído de forma evidente

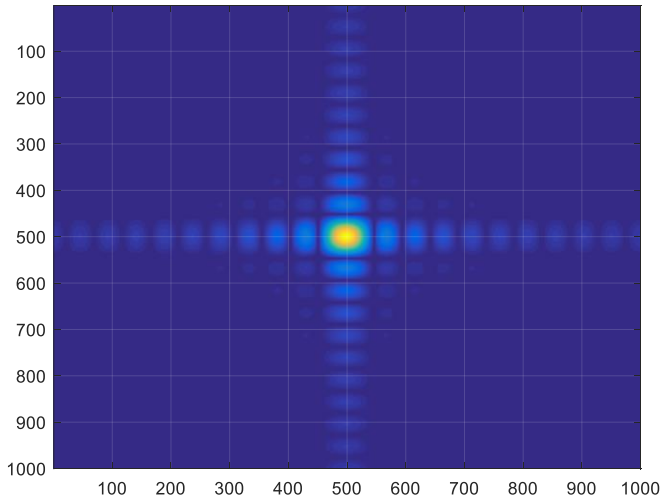
Sinais Discretos - MATLAB

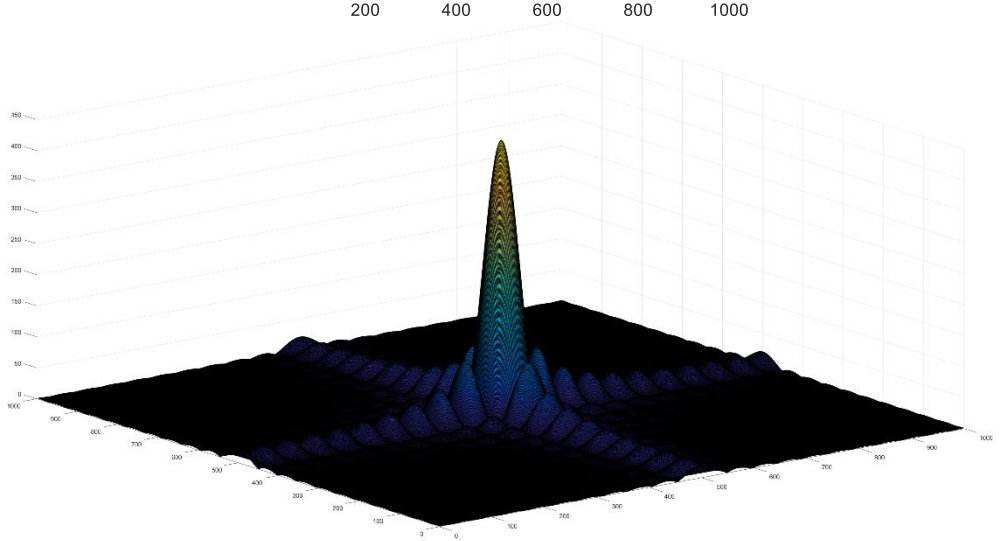
- Comparação DFT uni e bidimensional







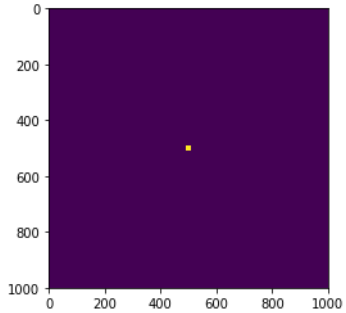




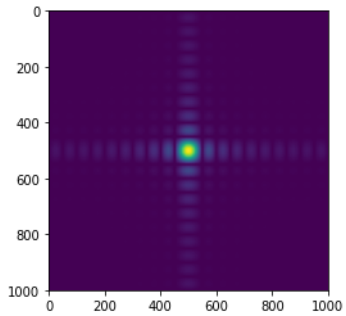
Sinais Discretos - Python

- DFT bidimensional

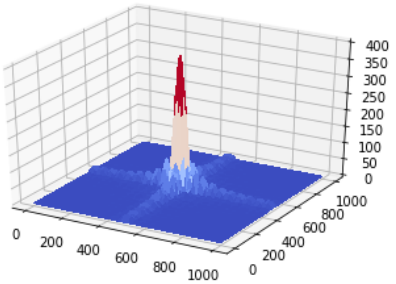
```
In [1]: # jupyter-qtconsole
   ...: %matplotlib inline
   ...: import matplotlib.pyplot as plt
   ...: import numpy as np
   ...: n = 1000
   ...: R = 10
   ...: M = np.zeros([n,n])
   ...: M[int(n/2-R):int(n/2+R), int(n/2-R):int(n/2+R)] = 1
   ...: plt.imshow(M)
   ...:
Out[1]: <matplotlib.image.AxesImage at 0x22df2c08e10>
```




```
In [2]: Mk = np.fft.fft2(M)
   ...: Ms = np.fft.fftshift(Mk)
   ...: Ma = np.abs(Ms)
   ...: plt.imshow(Ma)
   ...:
Out[2]: <matplotlib.image.AxesImage at 0x22df2ca2a90>
```




```
In [3]: from mpl_toolkits.mplot3d import Axes3D
   ...: from matplotlib import cm
   ...: fig = plt.figure()
   ...: ax = fig.gca(projection='3d')
   ...: X = np.arange(n)
   ...: Y = np.arange(n)
   ...: X, Y = np.meshgrid(X, Y)
   ...: # Plot the surface.
   ...: ax.plot_surface(X, Y, Ma, cmap=cm.coolwarm,
   ...:                 linewidth=0, antialiased=False)
   ...:
Out[3]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x22df2cc7940>
```



Sistemas Discretos

- Mapeamento (**processamento**) de uma sequência $x(n)$ de entrada em uma sequência $y(n)$ de saída

- Notação

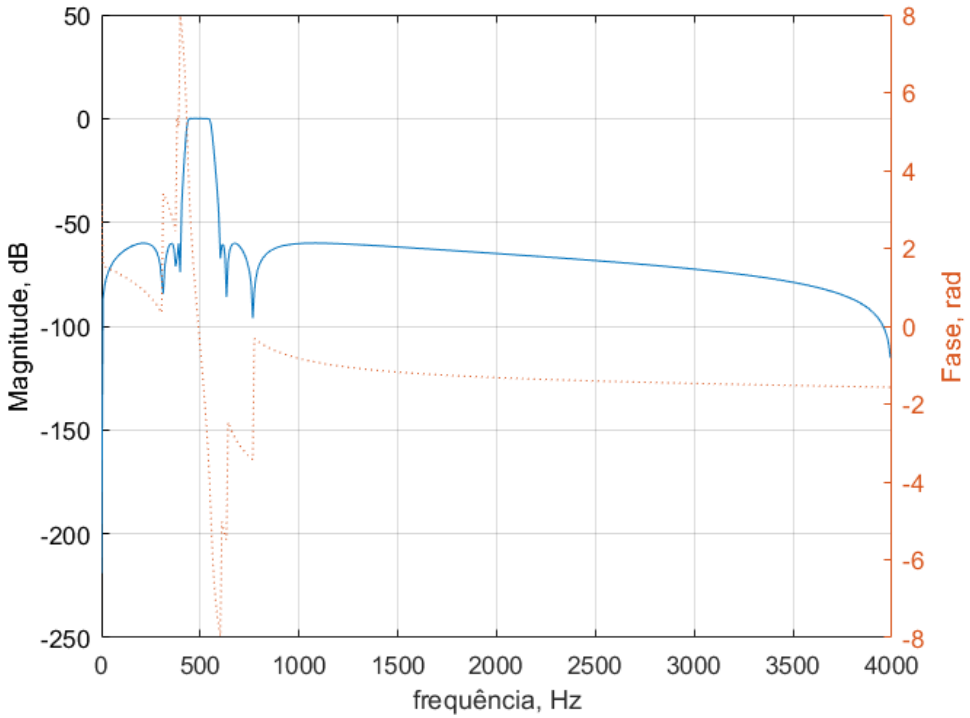
$$y(n) = H\left[x(n)\right]$$

- Representação

Sistemas Discretos - MATLAB

- Projeto de um filtro digital para preservar o tom em `f1` Hz. Chebyshev Tipo II, passa-faixa. **Faixa de passagem**: de 450 a 550 Hz, com atenuação máxima de 0.5 dB. **Faixa de rejeição**: de 0 a 400 Hz e de 600 Hz a fs, com atenuação mínima de pelo menos 60 dB

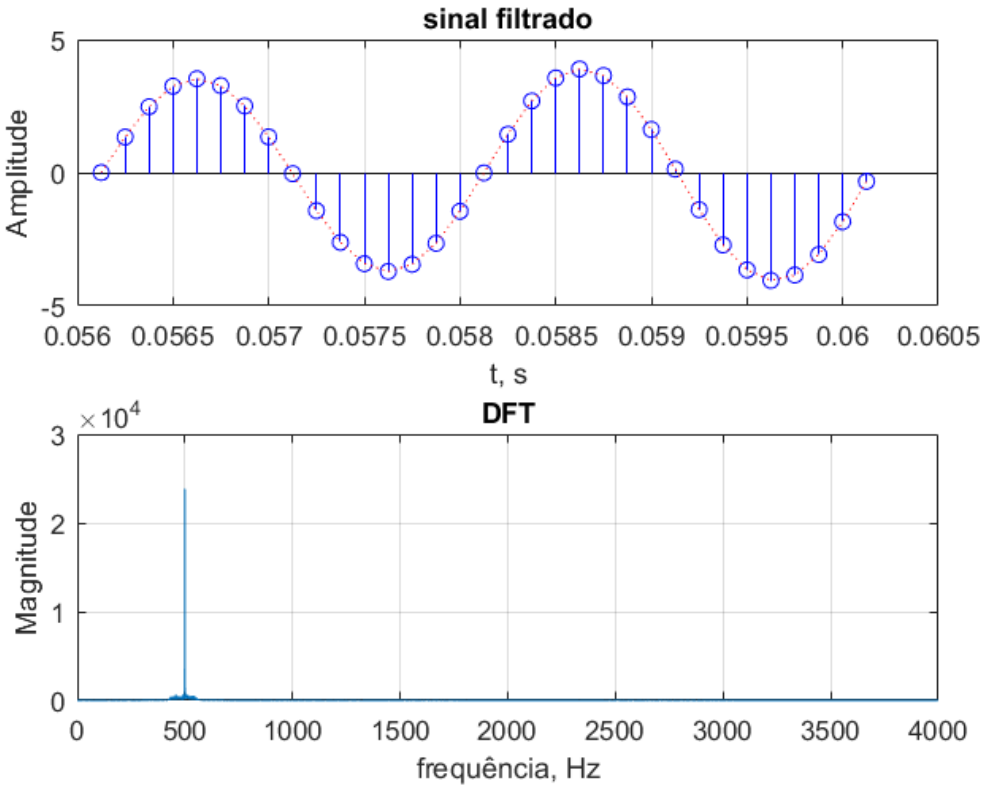
```
% Chebyshev Tipo II passa-faixa
fs = 8000; % freq. amostragem, Hz
fn = fs/2; % freq. de Nyquist
Wp = [450 550]/fn; % pass-band
Rp = 0.5; % ripple na faixa de passagem
Ws = [400 600]/fn; % stop-band
Rs = 60; % ripple na faixa de rejeição
[N, W] = cheb2ord(Wp, Ws, Rp, Rs);
[b, a] = cheby2(N, Rs, W); % coeficientes
[h, f] = freqz(b, a, 512, fs);
yyaxis left
plot(f, 20*log10(abs(h))),
ylabel('Magnitude, dB'),
yyaxis right
plot(f, unwrap(angle(h)), ':'),
ylabel('Fase, rad'),
grid, xlabel('frequência, Hz')
```



Análise Espectral com DFT - MATLAB

- Filtragem do sinal ruidoso (tons com ruído a 0 dB) pelo filtro recém projetado

```
% filtragem
y = filter(b, a, tonsR);
i = 450; % inicio arbitrário
N = 32; % numero de pontos para plot
subplot(211),
stem(t(i:i+N), y(i:i+N), 'b')
hold on
plot(t(i:i+N), y(i:i+N), 'r:')
title(sprintf('sinal filtrado'));
grid; xlabel('t, s');
ylabel('Amplitude');
hold off
% DFT
subplot(212),
tff = abs(fft(y));
N = length(tf); NN = floor(N/2);
f = (0:N-1)/N*fs;
plot(f(1:NN), tff(1:NN));
grid, title('DFT');
xlabel('frequência, Hz');
ylabel('Magnitude');
```



Sistemas Discretos - Python

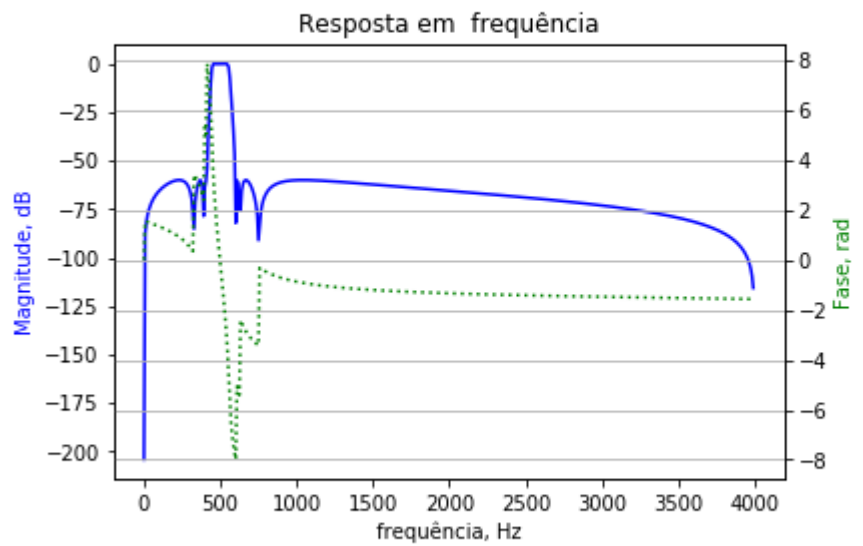
Jupyter QtConsole

File Edit View Kernel Window Help

```
Jupyter QtConsole 4.3.1
Python 3.6.3 |Anaconda, Inc.| (default, Oct 15 2017, 03:27:45) [MSC v.1900 64 bit
(AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.1.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: # projeto de filtro Chebyshev tipo II
   ...: # passa-faixa
   ...: # freqs passagem: 450, 550
   ...: # freqs rejeicao: 400, 600
   ...: # atenuacao passagem: 0.5 dB
   ...: # atenuacao rejeicao: 60 dB
   ...: import numpy as np
   ...: from scipy import signal
   ...: import matplotlib.pyplot as plt
   ...: fs = 8000 # freq. amostragem, Hz
   ...: fn = fs/2 # freq. de Nyquist
   ...: Wp = np.array([450, 550])/fn # pass-band
   ...: Rp = 0.5 # ripple na faixa de passagem
   ...: Ws = np.array([400, 600])/fn # stop-band
   ...: Rs = 60 # ripple na faixa de rejeição
   ...: N, W = signal.cheb2ord(Wp, Ws, Rp, Rs);
   ...: b, a = signal.cheby2(N, Rs, W, 'pass'); # coeficientes
   ...: f, h = signal.freqz(b, a, 512, whole=False)
   ...: f = f * fs / np.pi / 2
   ...: #fig = plt.figure(figsize=(12,10))
   ...: fig = plt.figure()
   ...: plt.title('Resposta em  frequência')
   ...: ax1 = fig.add_subplot(111)
   ...: plt.plot(f, 20 * np.log10(abs(h)), 'b')
   ...: plt.ylabel('Magnitude, dB', color='b')
   ...: plt.xlabel('frequência, Hz')
   ...: ax2 = ax1.twinx()
   ...: fase = np.unwrap(np.angle(h))
   ...: plt.plot(f, fase, 'g:')
   ...: plt.ylabel('Fase, rad', color='g')
   ...: plt.grid()
   ...: plt.axis('tight')
   ...: plt.show()
   ...:
```

Jupyter QtConsole

File Edit View Kernel Window Help




```
In [2]:
```

Análise Espectral com DFT - Python

- Filtragem do sinal ruidoso (tons com ruído a 0 dB) pelo filtro recém projetado

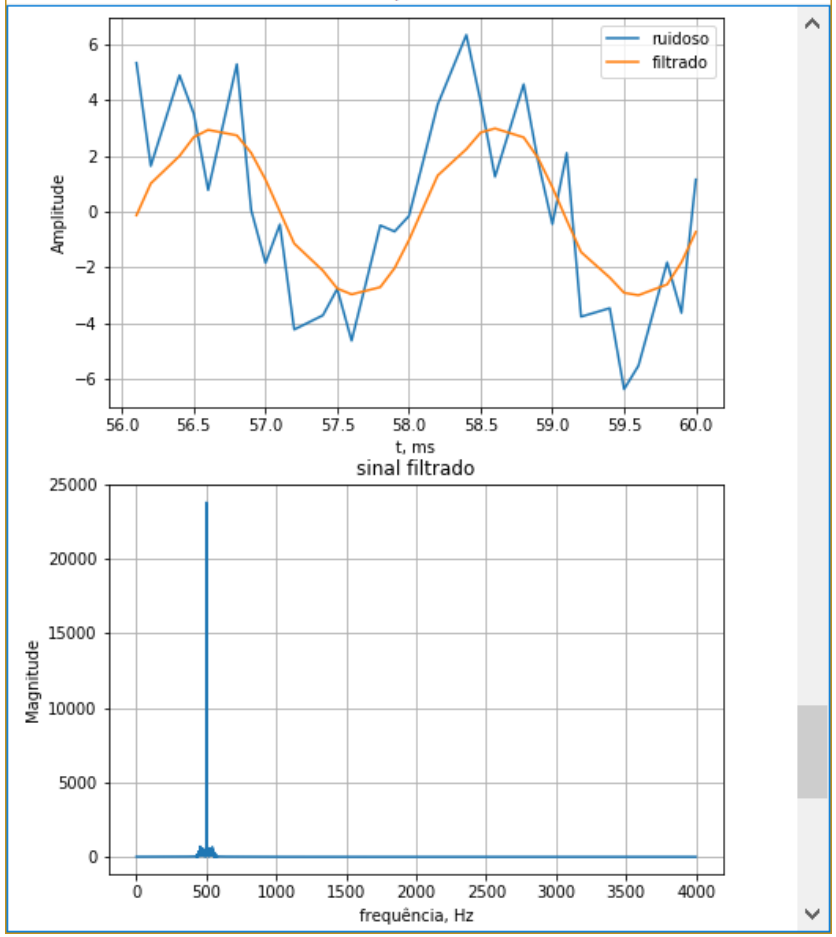
Jupyter QtConsole

File Edit View Kernel Window Help

```
In [6]: # filtragem
   ...: y = signal.lfilter(b, a, tonsR)
   ...: i = 449 # inicio arbitrário (Python index 0)
   ...: N = 32 # numero de pontos para plot
   ...: g, axarr = plt.subplots(2, 1, figsize=(7, 10))
   ...: ind1 = range(i, i+N)
   ...: tt = np.round(t*1000, 1) # agora, tt em ms
   ...: #axarr[0].set_title('tons com ruido')
   ...: axarr[0].plot(tt[ind1], tonsR[ind1], label='ruidoso')
   ...: axarr[0].set_xlabel('t, ms')
   ...: axarr[0].axis('tight')
   ...:
   ...: #axarr[0].set_title('sinal filtrado')
   ...: axarr[0].plot(tt[ind1], y[ind1], label='filtrado')
   ...: axarr[0].set_ylabel('Amplitude')
   ...: axarr[0].set_xlabel('t, ms')
   ...: axarr[0].axis('tight')
   ...: axarr[0].grid()
   ...: axarr[0].legend(loc='best')
   ...:
   ...: # DFT
   ...: tfY = np.fft.fft(y)
   ...: tfY = np.abs(tfY)
   ...: N = len(tfY)
   ...: f = np.arange(N)
   ...: f = f/N*fs
   ...: NN = N//2
   ...: axarr[1].plot(f[:NN], tfY[:NN])
   ...: axarr[1].set_title('sinal filtrado')
   ...: axarr[1].set_ylabel('Magnitude')
   ...: axarr[1].set_xlabel('frequência, Hz')
   ...: axarr[1].axis('tight')
   ...: axarr[1].grid()
   ...:
   ...: plt.show()
   ...:
```

Jupyter QtConsole

File Edit View Kernel Window Help



Espectrograma - MATLAB

frequência de amostragem

sinal

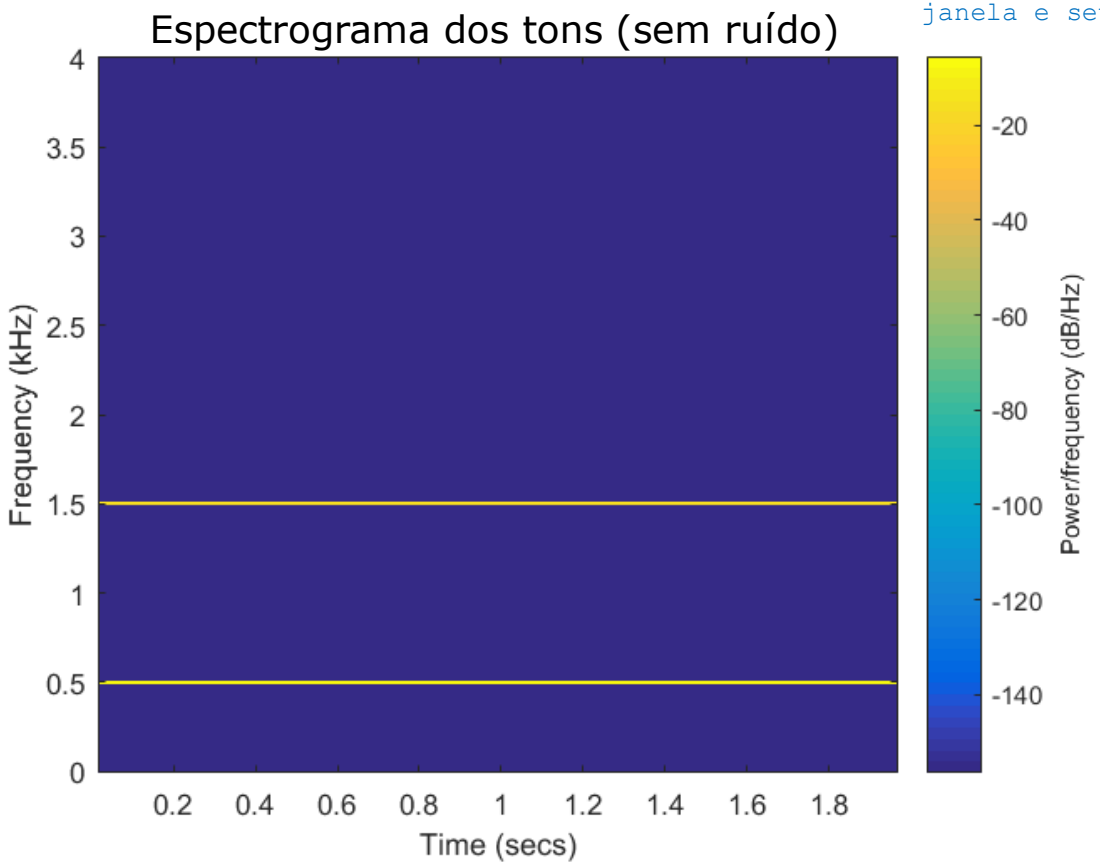
sobreposição

```
>>
>> N=256; spectrogram(tons, ones(2*N,1), N, 2*N, fs,'yaxis');
>>
```

janela e seu tamanho

número de pontos para DFT

frequência no eixo y

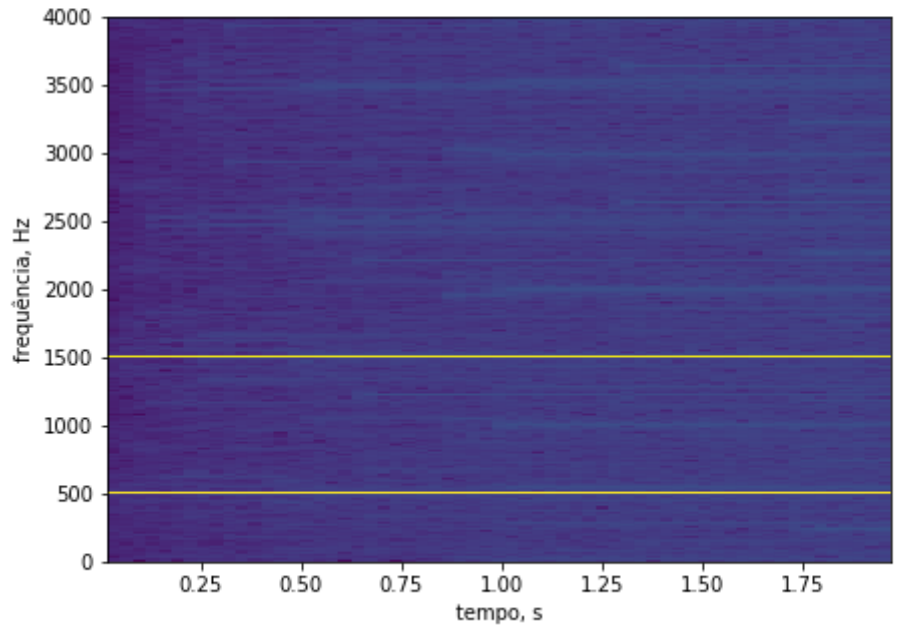


Espectrograma - Python

Jupyter QtConsole

File Edit View Kernel Window Help

```
In [3]: # espectrograma
   ...: nfft = 512
   ...: plt.figure(figsize=(7, 5))
   ...: plt.specgram(tons, nfft, fs, noverlap=nfft//2,
   ...:     window=np.ones(nfft)) # janela retangular
   ...: plt.xlabel('tempo, s')
   ...: plt.ylabel('frequência, Hz')
   ...: plt.show()
   ...:
```

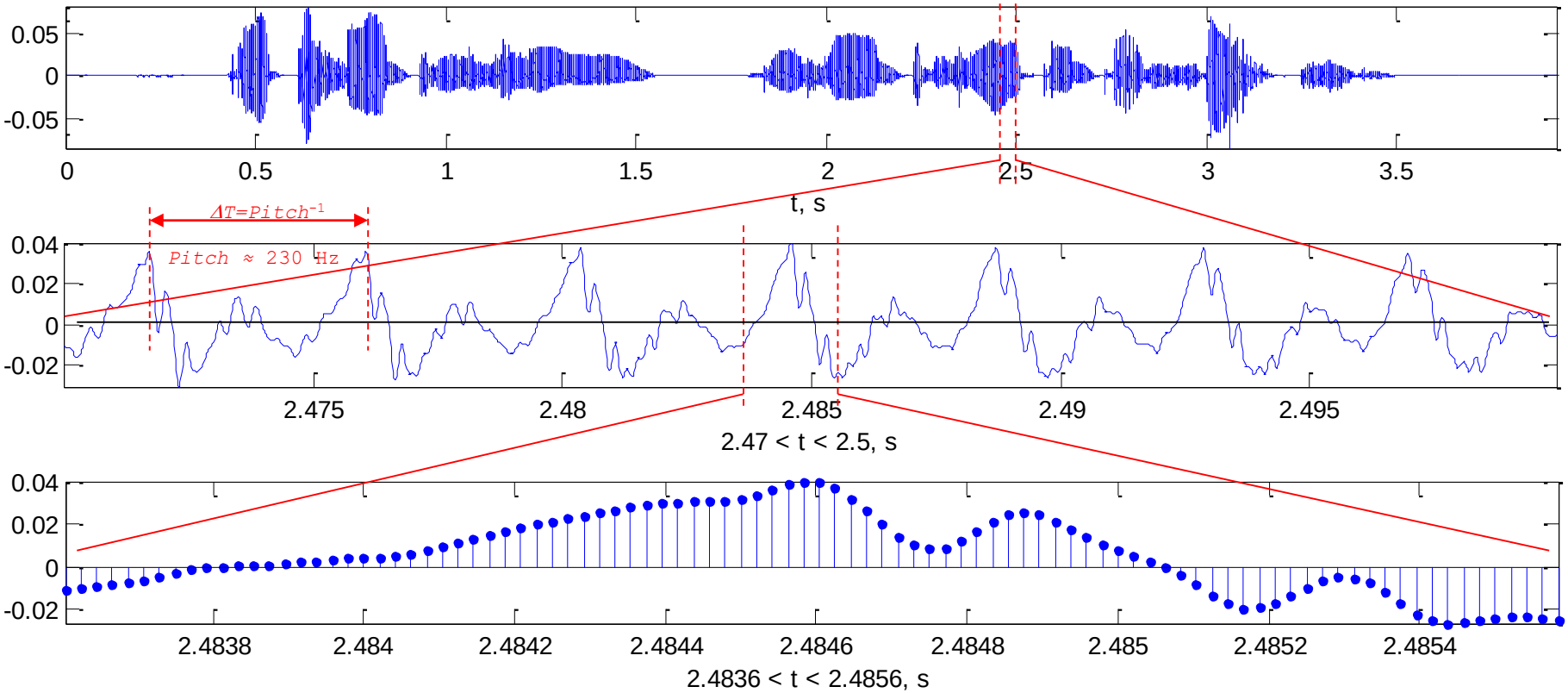


Espectrograma do Sinal de Voz

Texto: “oitocentos e um, duzentos e oitenta e sete” (801 287)

Áudio:

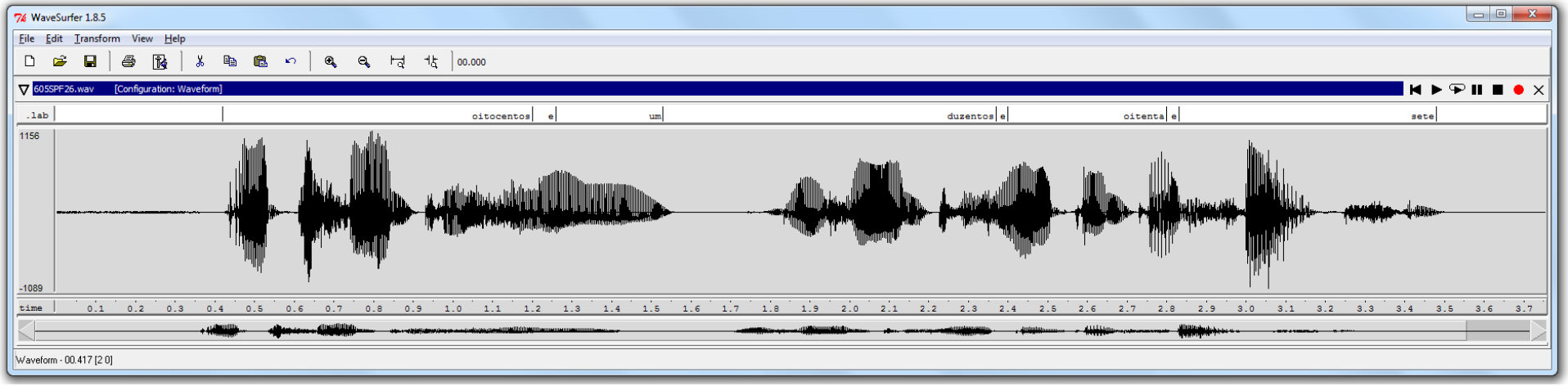
Sinal no tempo:

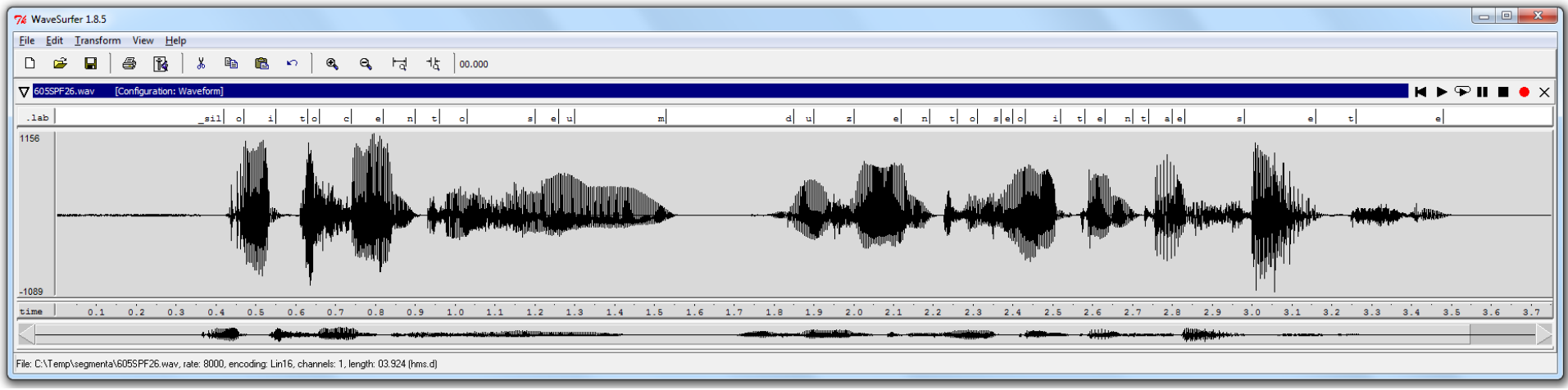


Espectrograma do Sinal de Voz

Texto: “oitocentos e um, duzentos e oitenta e sete” (801 287)

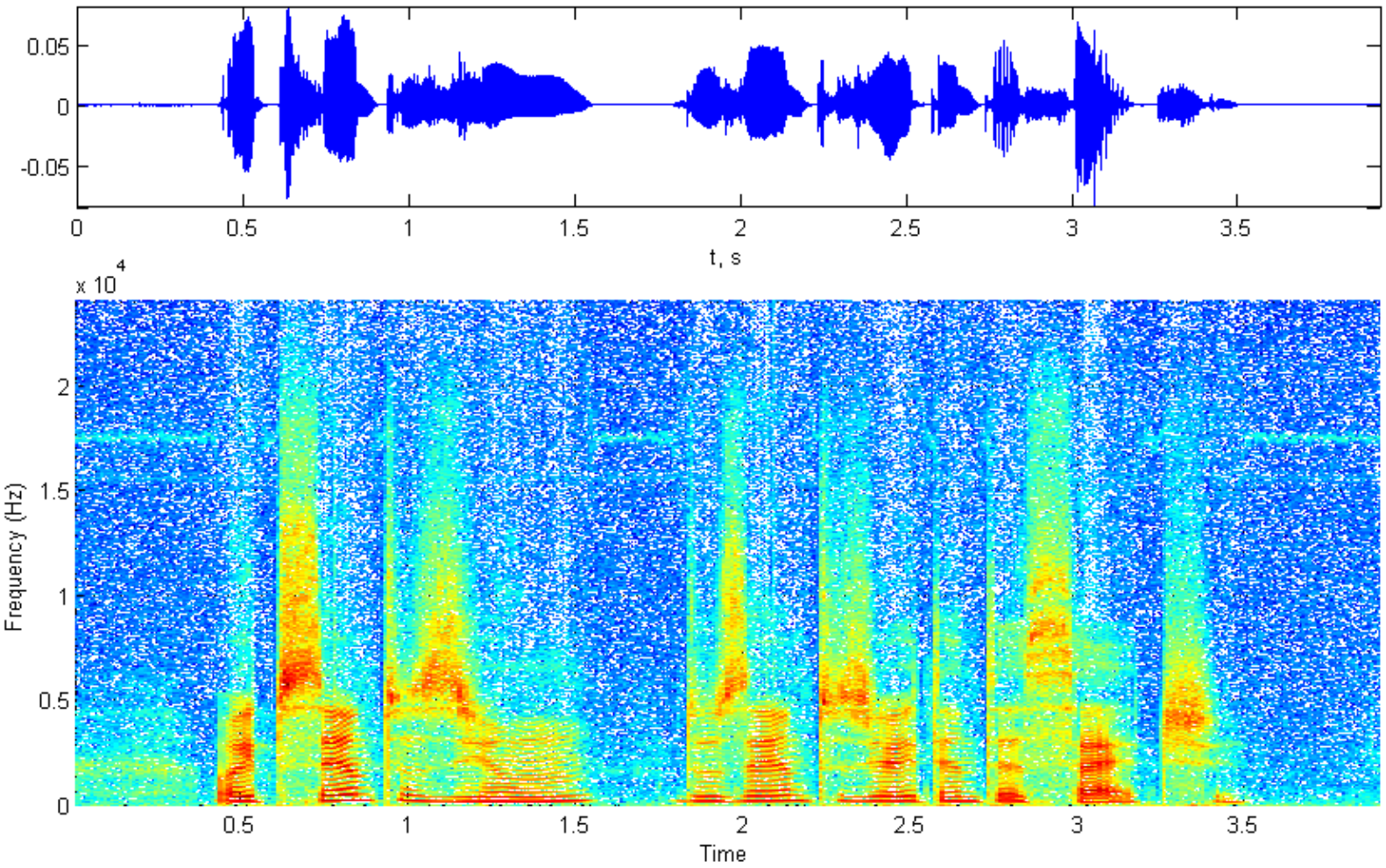
Segmentação automática em palavras e em modelos acústicos:





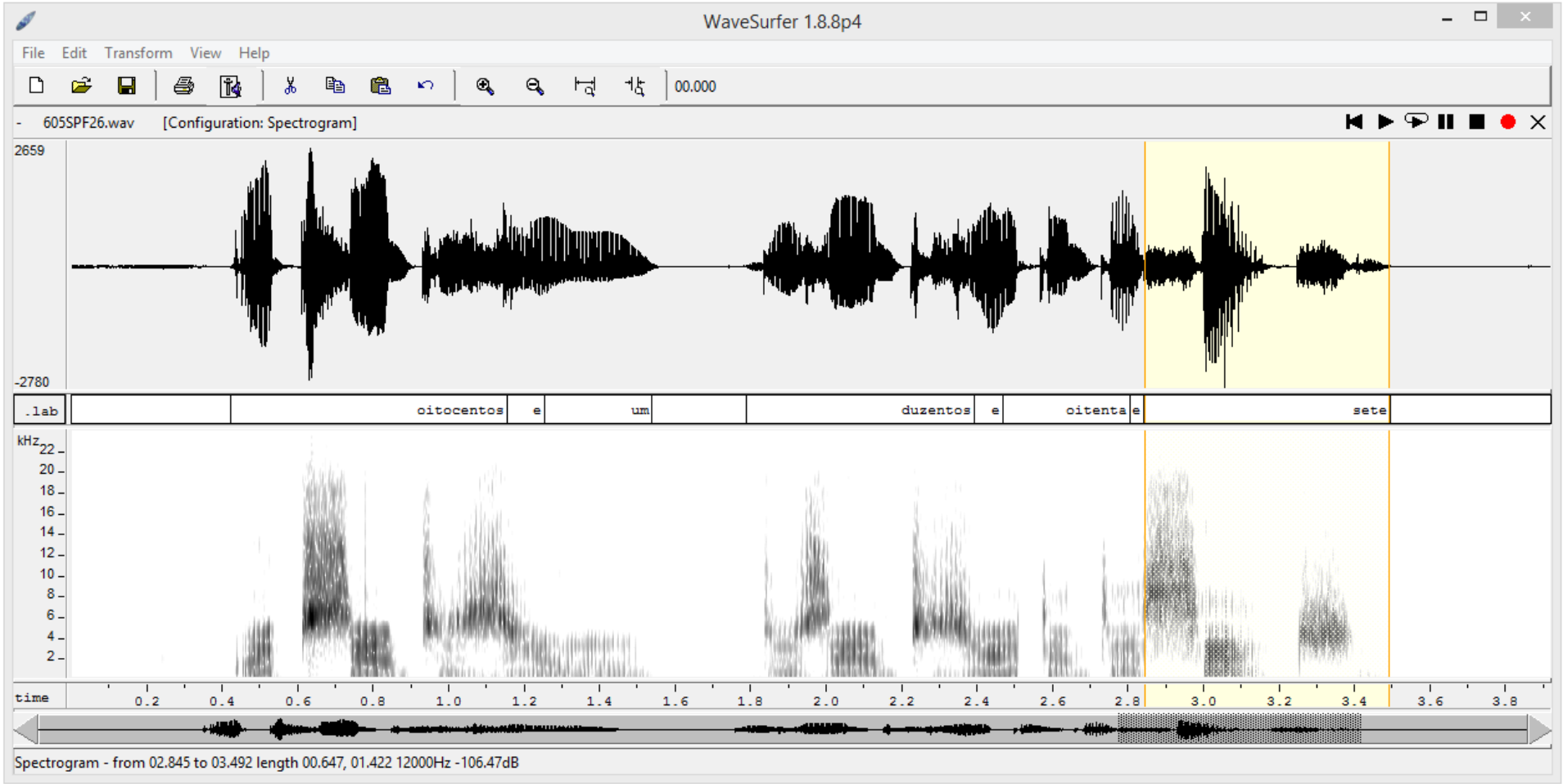
Espectrograma do Sinal de Voz

Espectrograma com MATLAB:



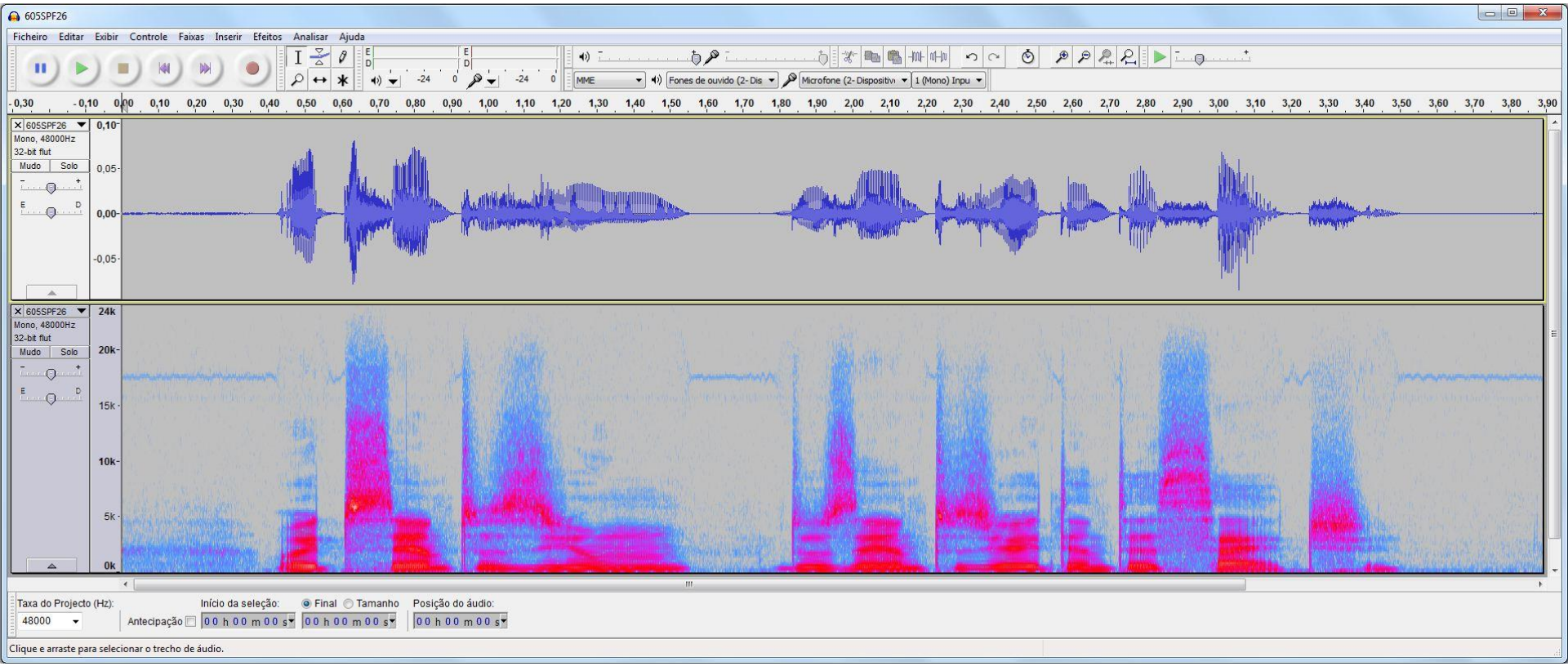
Espectrograma do Sinal de Voz

Espectrograma com WaveSurfer: http://www.speech.kth.se/wavesurfer/



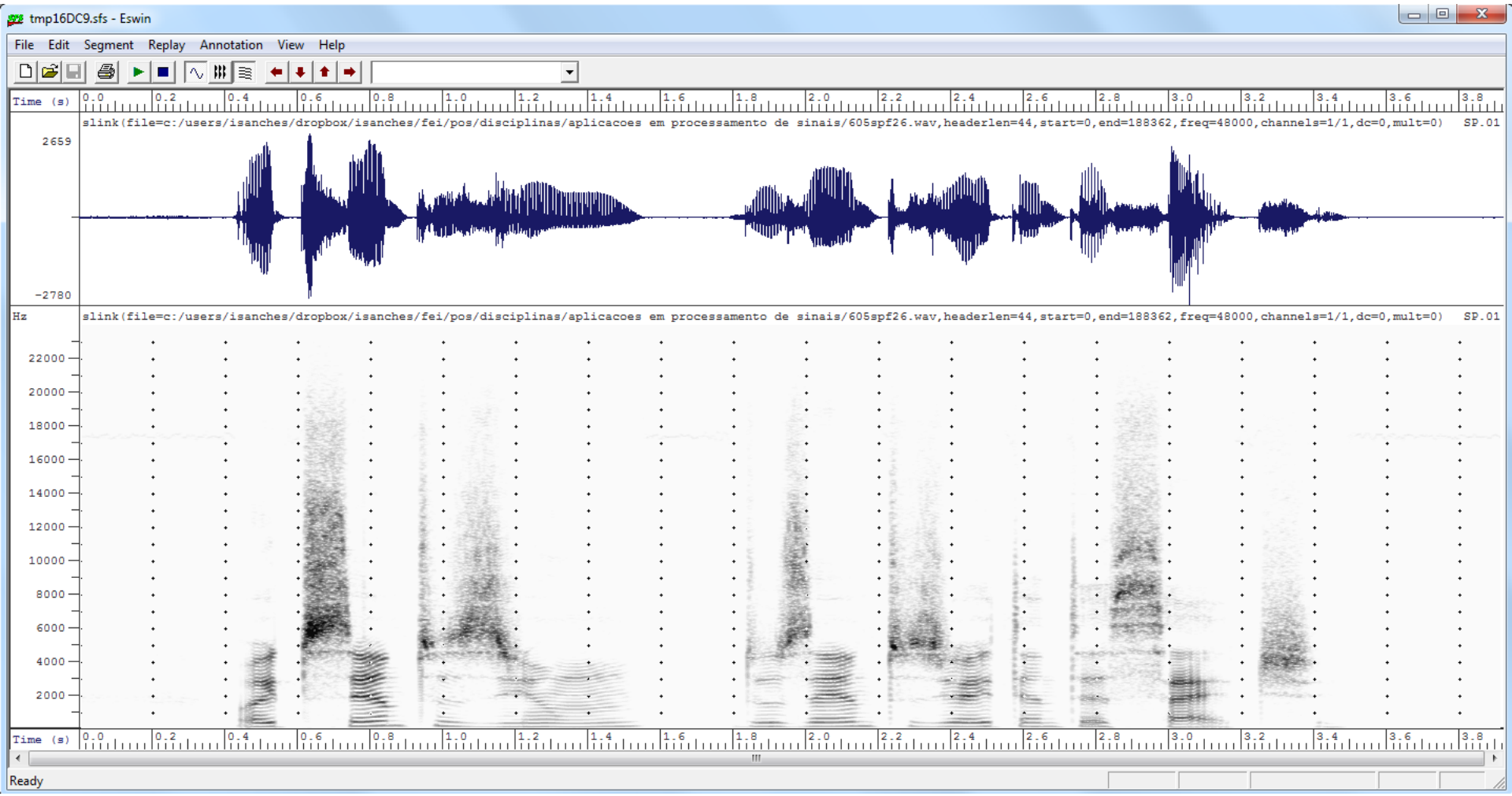
Espectrograma do Sinal de Voz

Espectrograma com Audacity: http://audacity.sourceforge.net/



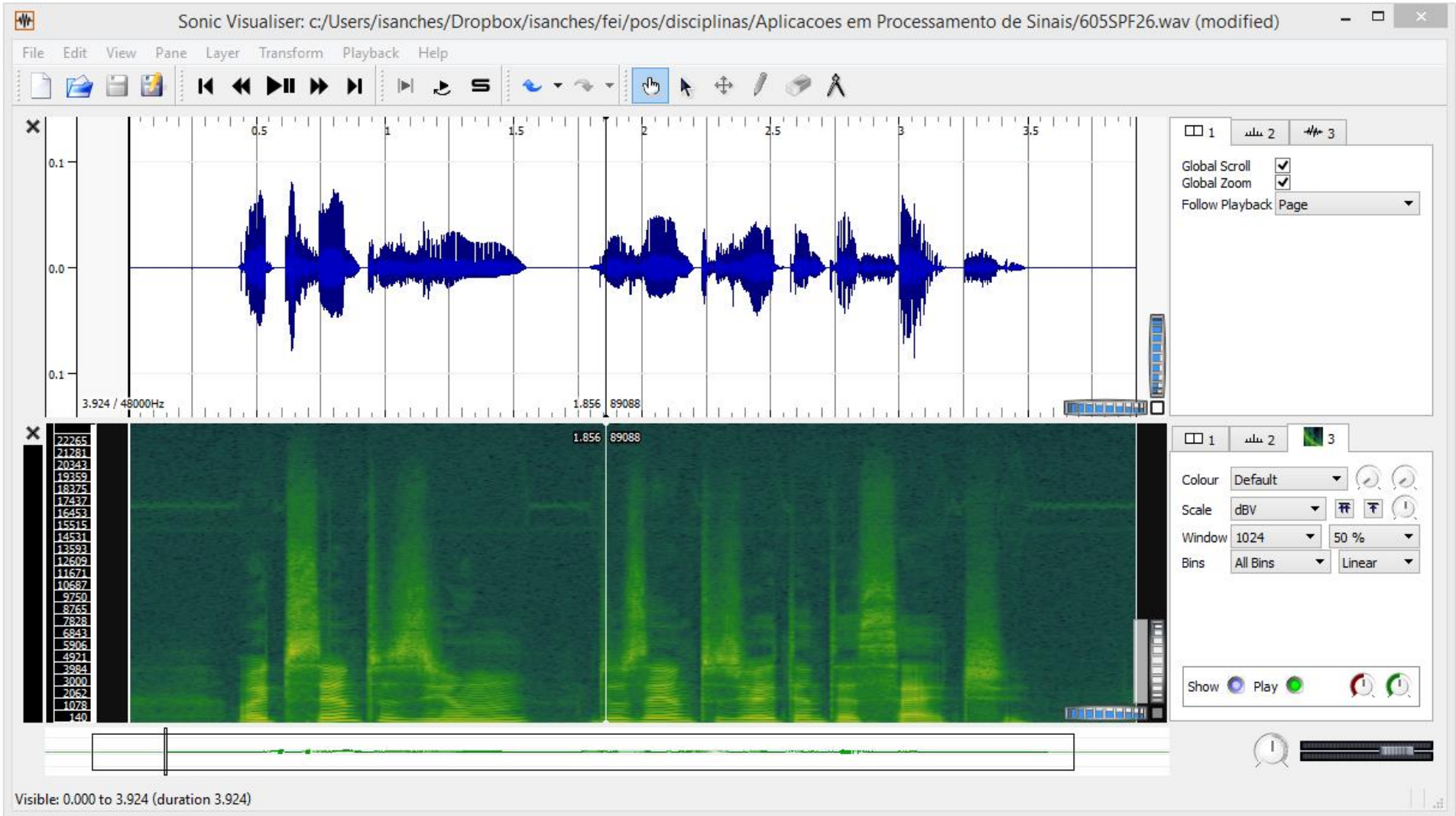
Espectrograma do Sinal de Voz

Espectrograma com SFS: http://www.phon.ucl.ac.uk/resource/sfs/



Espectrograma do Sinal de Voz

Espectrograma com SonicVisualizer: http://www.sonicvisualiser.org/



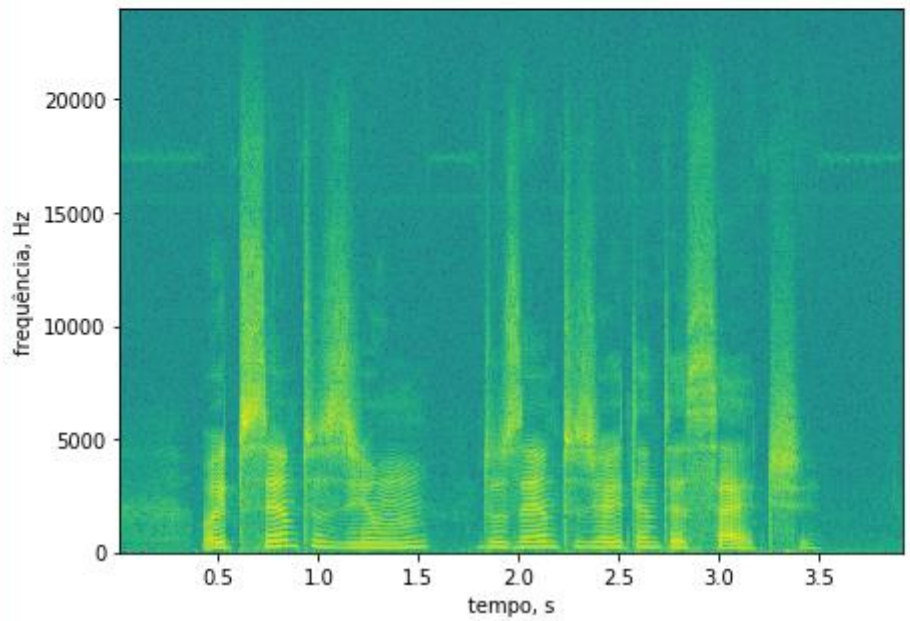
Espectrograma do Sinal de Voz

E, finalmente, espectrograma com Python:

Jupyter QtConsole

File Edit View Kernel Window Help

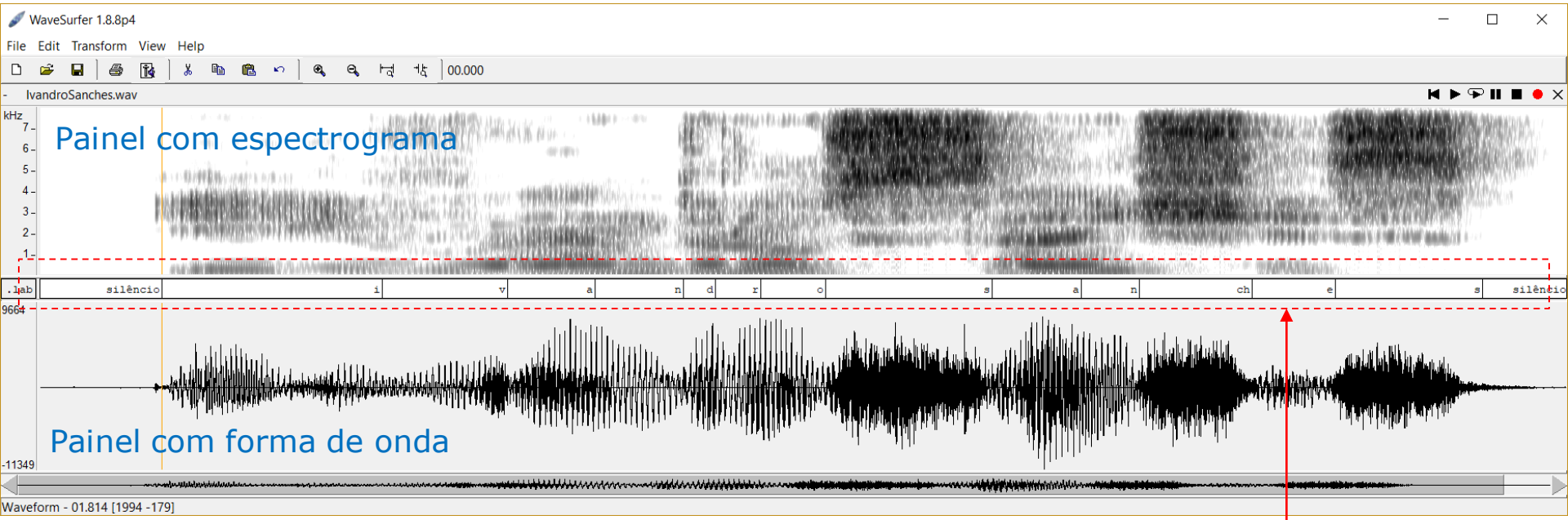
```
In [4]: # espectrograma do sinal de voz
   ...: from scipy.io import wavfile
   ...: wave = '../scripts e wavs/605SPF26.wav'
   ...: fs, data = wavfile.read(wave)
   ...: nfft = 1024
   ...: plt.figure(figsize=(7, 5))
   ...: plt.specgram(data, nfft, fs, noverlap=7*nfft//8)
   ...: plt.xlabel('tempo, s')
   ...: plt.ylabel('frequência, Hz')
   ...: plt.show()
   ...:
```



Exercício Proposto 1

1. Baixar o programa wavesurfer
2. Gravar o seu nome completo
3. Apresentar o painel com a forma de onda no tempo
4. Incluir o painel que apresenta o espectrograma (Spectrogram)
5. Incluir o painel que permite introduzir transcrição (Transcription) no qual você rotulará trechos do áudio que correspondam às vogais e consoantes do seu nome. Baseie-se no exemplo a seguir
6. Entregar em 1 página impressa (ou encaminhada por e-mail). O conteúdo será a imagem do sinal de voz com a transcrição correspondente. Isto é, não precisa do arquivo de áudio nem do arquivo de transcrição (labels). Ver exemplo na página seguinte
7. Para cada vogal de seu nome, estimar e apresentar o ***pitch*** (ver página 38) em Hz

Exercício Proposto 1



Vogal	pitch, Hz
i	133
a	115
o	136
a	121
e	108

Painel com transcrição

Dica1: para uma boa transcrição, baseie-se nas transições indicadas pelo espectrograma e na forma de onda, além da possibilidade de escutar os trechos demarcados.
Dica2: para confirmar a estimação de pitch, usar o painel "Pitch Contour" do wavesurfer

Exercício Proposto 2

1. Baixar o programa wavesurfer
2. Seguir os passos das **seções 2** e **3** da **Aula 2** para:
   1. Criar os arquivos `treinoI.wav`, `treinoI.lab`, `testeI.wav` e `testeI.lab`, onde `I` será um número atribuído a cada aluno: `I` = 1, 2, ... Para uniformizar as gravações entre os alunos, procurar realizar as aquisições com headset

   2. Os arquivos conterão as palavras do vocabulário do sistema de reconhecimento de fala a ser construído na Aula 2, mais os trechos de silêncio (_sil):

direita	esquerda	avance	retorne	_sil

Veja o exemplo na seção 3 para o aluno fictício `I` = 1. No exemplo há apenas 1 exemplo de cada palavra do vocabulário por arquivo, contudo, em seu caso, tente pronunciar pelo menos 5 repetições de cada palavra por arquivo. Exemplo: "*avance esquerda retorne direita esquerda avance direita retorne*..." Obs: usar apenas minúsculas nos labels

3. Trazer os 4 arquivos em um pendrive na minha próxima aula. Como mencionado, iremos construir um sistema de reconhecimento automático de fala com as vozes dos alunos.

centro universitário FEI

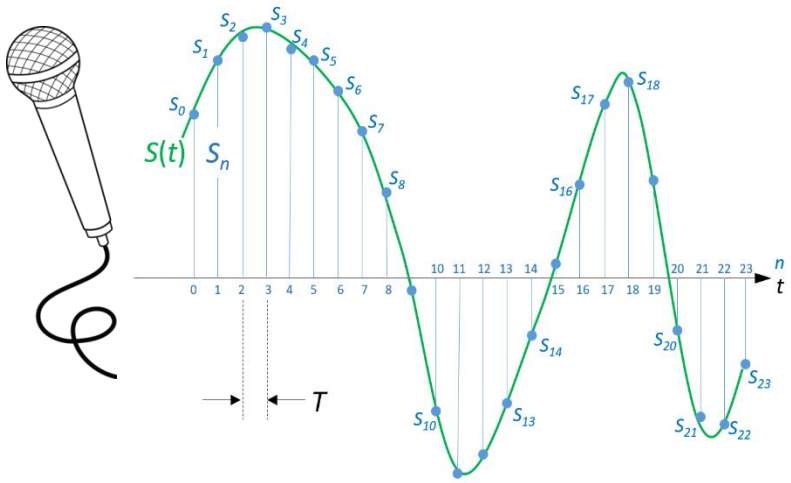
PEL 304
Aplicações em Processamento de Sinais

Ivandro Sanches

Aula 2: Sinal de voz e reconhecimento automático de fala

2020

O Sinal de Voz



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)

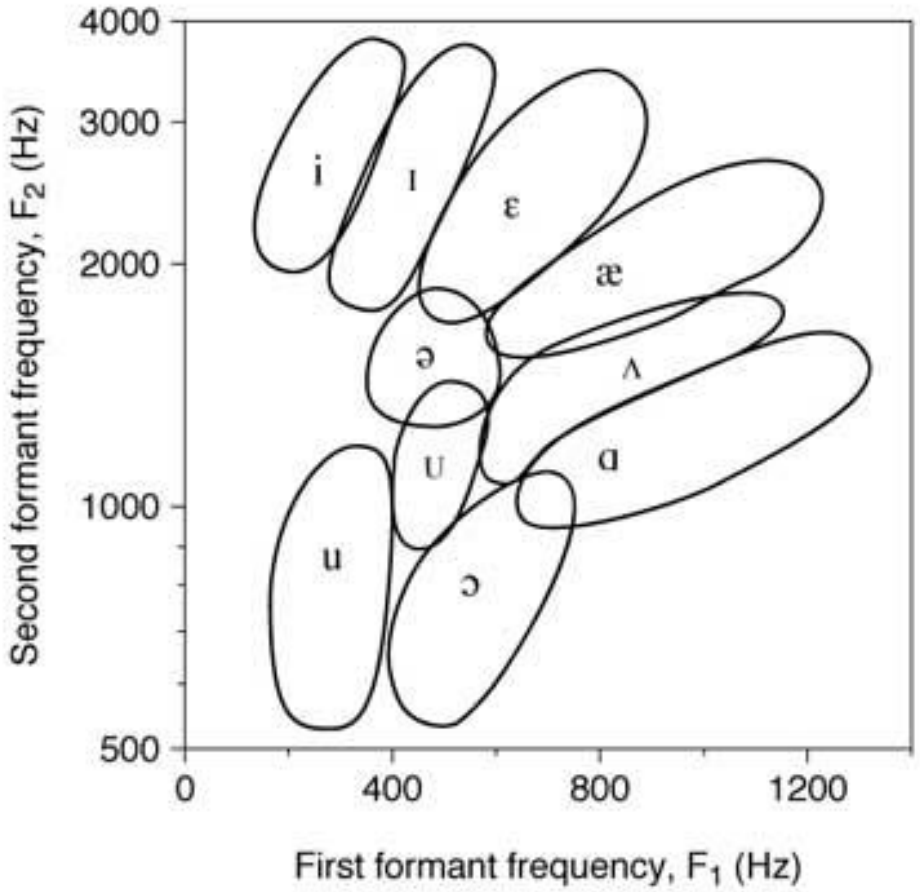
Telefonia: 8 kHz (T=125 μs)

Um arquivo do tipo .WAV

cabeçalho
001010100111010 1
001010100110010 0
101010100001000 1
. . .
011000100101010 1

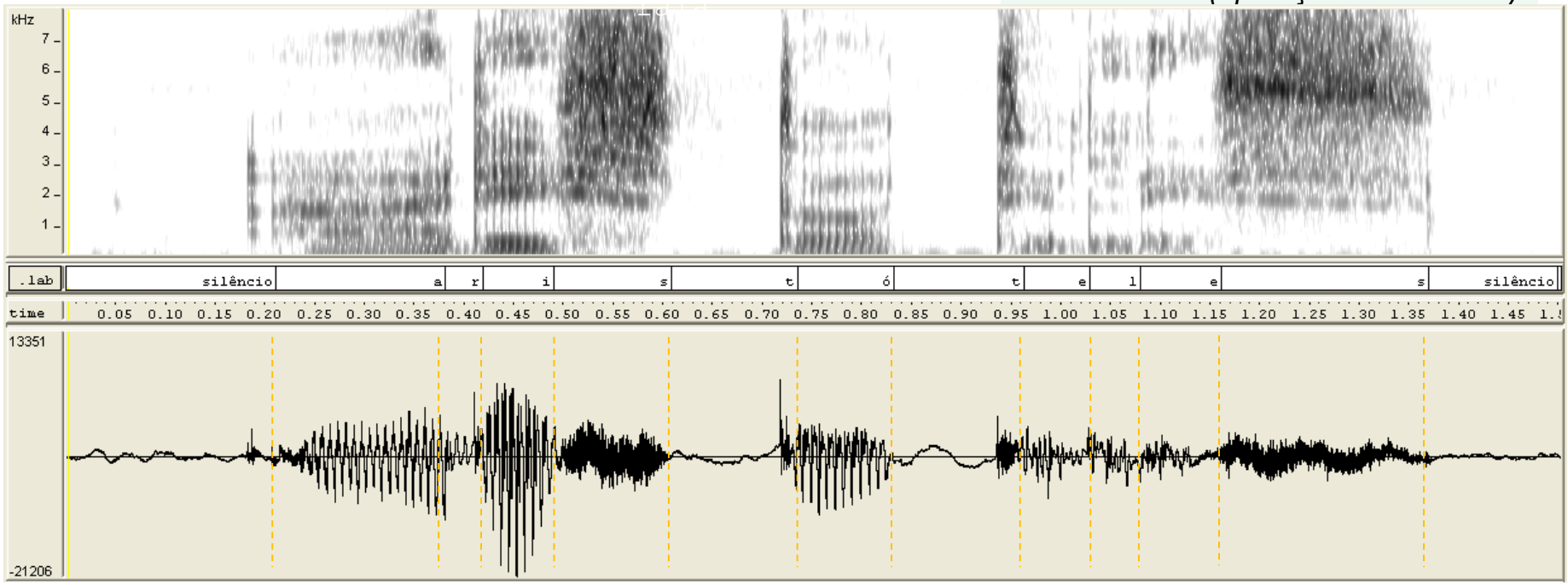
Reconhecimento Automático de Fala

F_1 versus F_2 para vogais da Língua Inglesa



Reconhecimento Automático de Fala

Parte Acústica (aplicação 10 cientistas)



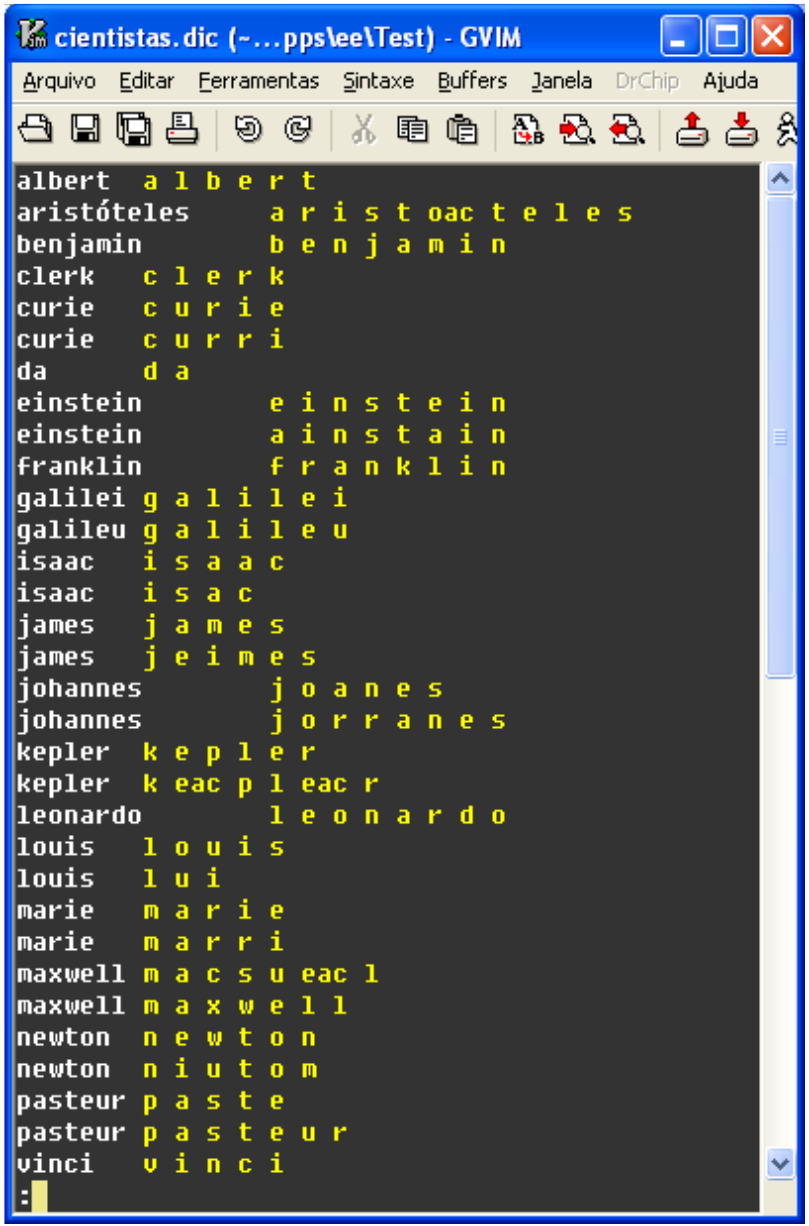
'sil' 'a' 'r' 'i' 's' 't' 'ó' 't' 'e' 'l' 'e' 's' 'sil'

Modelos Acústicos → fazem a correspondência entre a acústica e a linguística

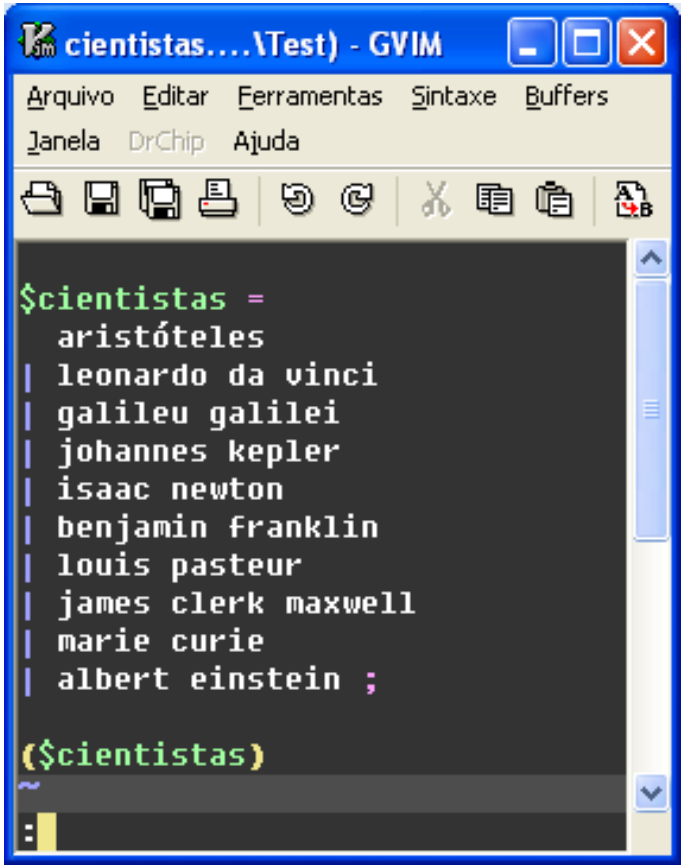
Reconhecimento Automático de Fala

Parte Linguística
(aplicação 10 cientistas)

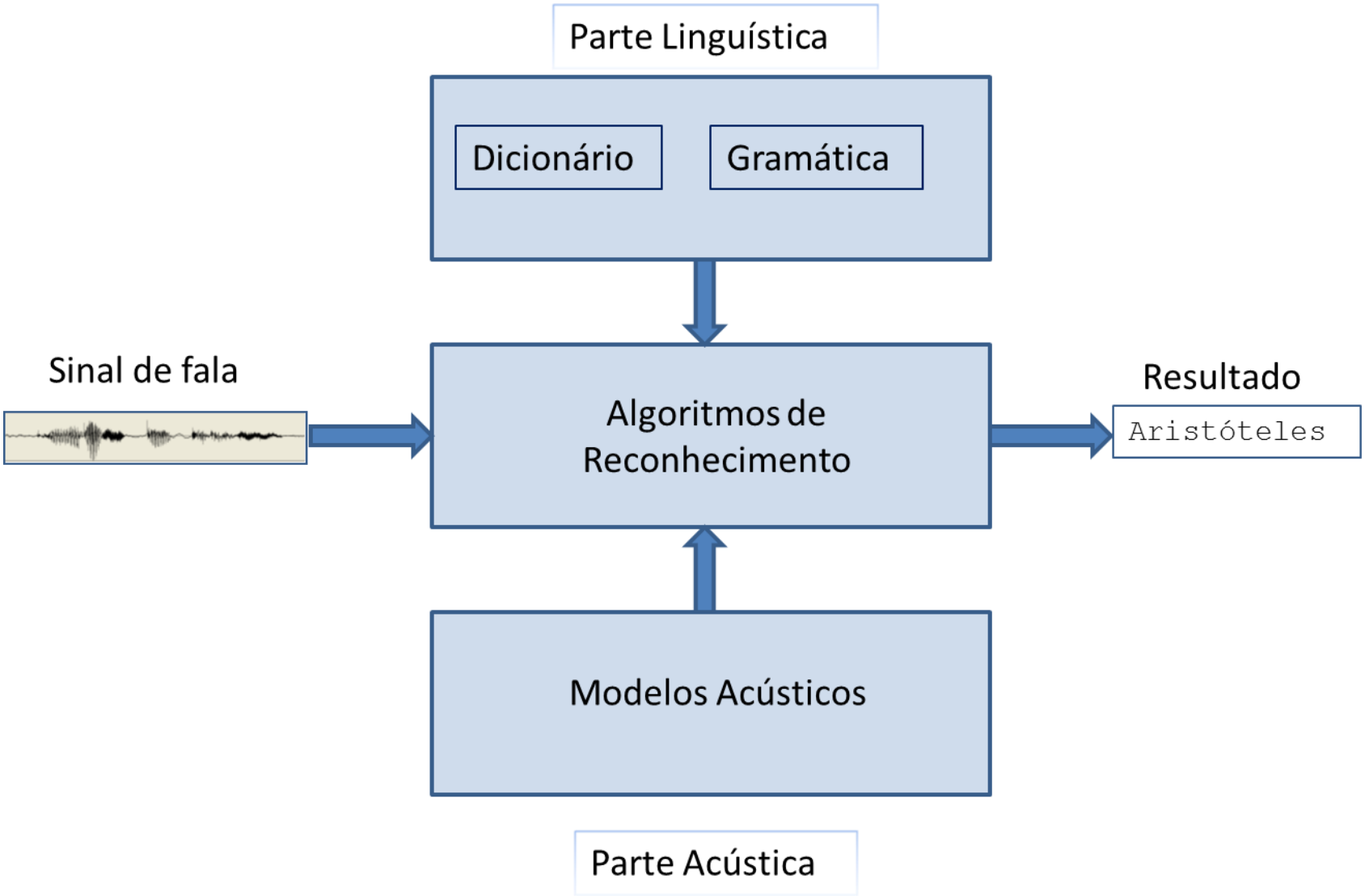
Dicionário



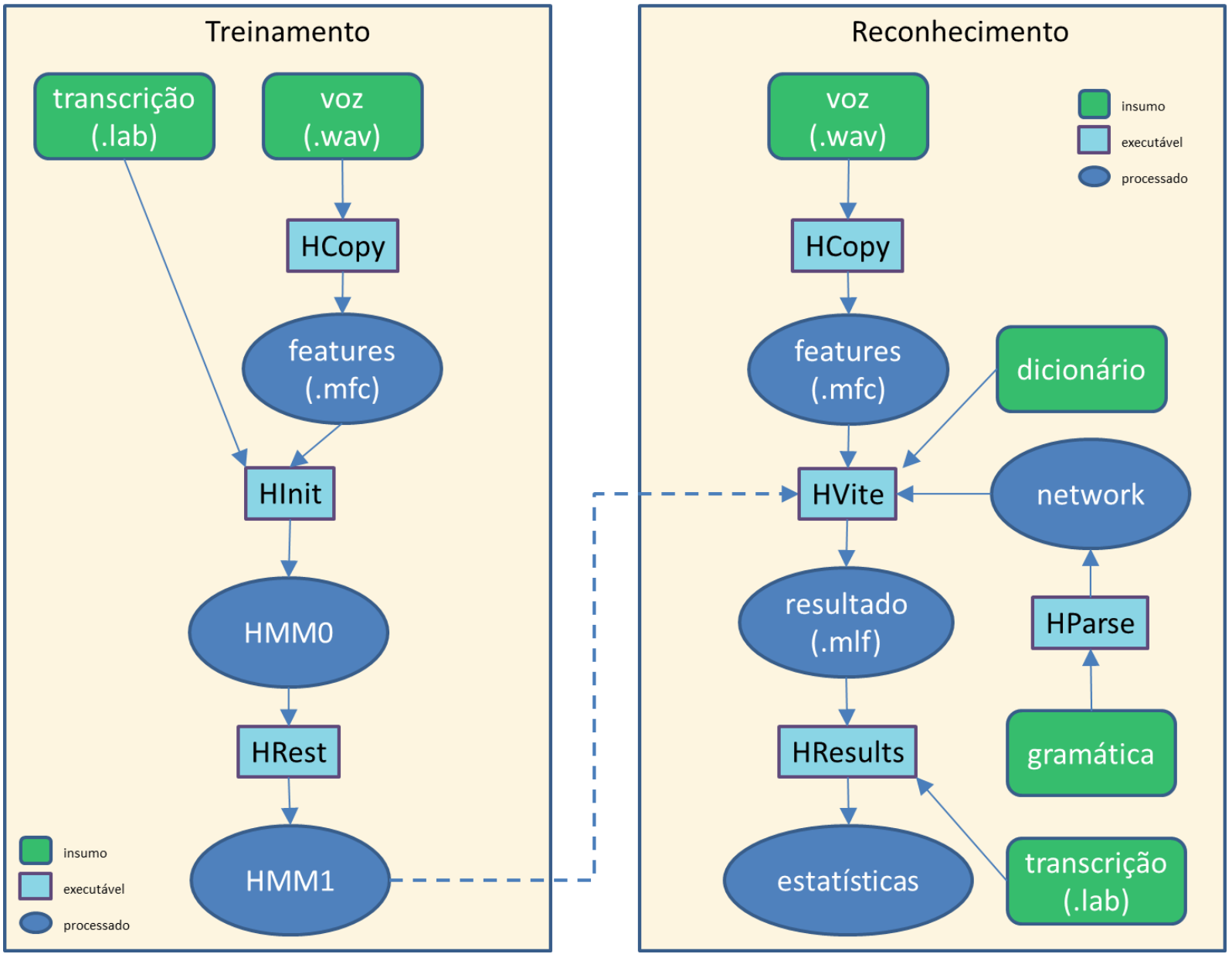
Gramática



Reconhecimento Automático de Fala



Reconhecimento Automático de Fala



Reconhecimento Automático de Fala

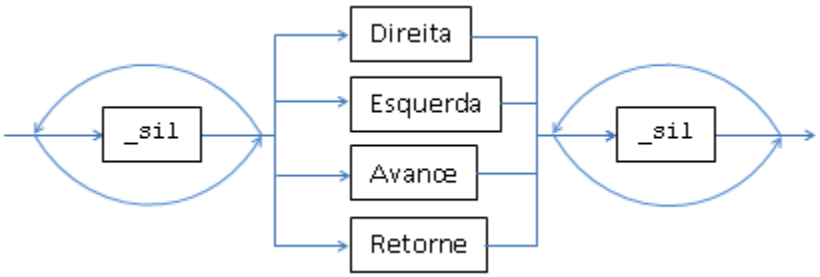
As etapas do slide anterior podem ser descritas de forma ampla pela seguinte sequência de atividades a ser realizadas a seguir:

1. definição do vocabulário e gramática
2. aquisição dos sinais de voz
3. transcrição dos sinais de voz
4. treinamento de modelos acústicos
5. avaliação da precisão de reconhecimento
6. testes 'live'

Reconhecimento Automático de Fala

1. Definição do vocabulário e gramática

Por exemplo:



Gramática:

```
({_sil} ( avance | direita | esquerda | retorne ) {_sil})
```

Dicionário: por simplicidade os modelos acústicos representarão palavras inteiras, e não unidades sonoras básicas. O silêncio corresponde ao símbolo `_sil`

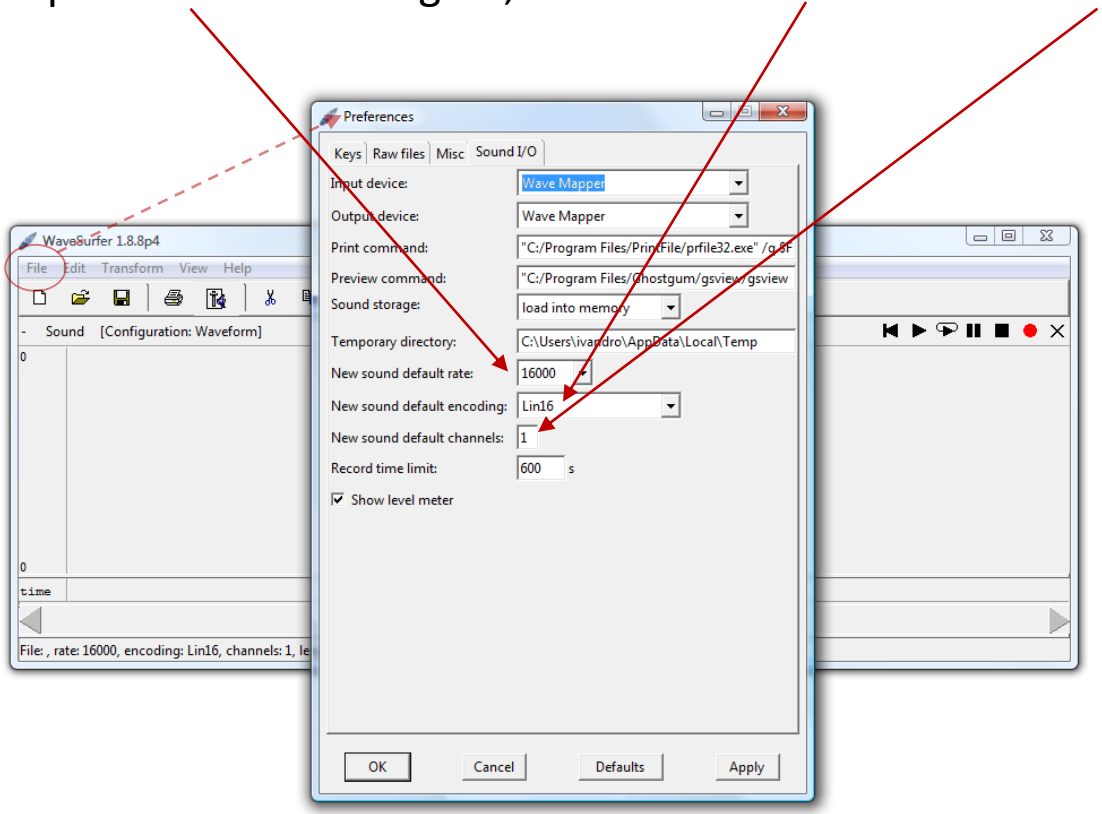
```
avance      [avance]    avance
direita     [direita]   direita
esquerda    [esquerda]  esquerda
retorne     [retorne]   retorne
_sil        []          _sil
```

Reconhecimento Automático de Fala

2. Aquisição dos sinais de voz

Usaremos o programa wavesurfer http://www.speech.kth.se/wavesurfer/

Ajuste de frequência de amostragem, tamanho da amostra e número de canais:



Reconhecimento Automático de Fala

2. Aquisição dos sinais de voz

Gravar o grupo de sinais para treino do sistema e o grupo de sinais para teste dos modelos acústicos treinados.

Por exemplo, aluno 1 gravará diversas palavras do vocabulário no arquivo

`treino1.wav`

e no arquivo

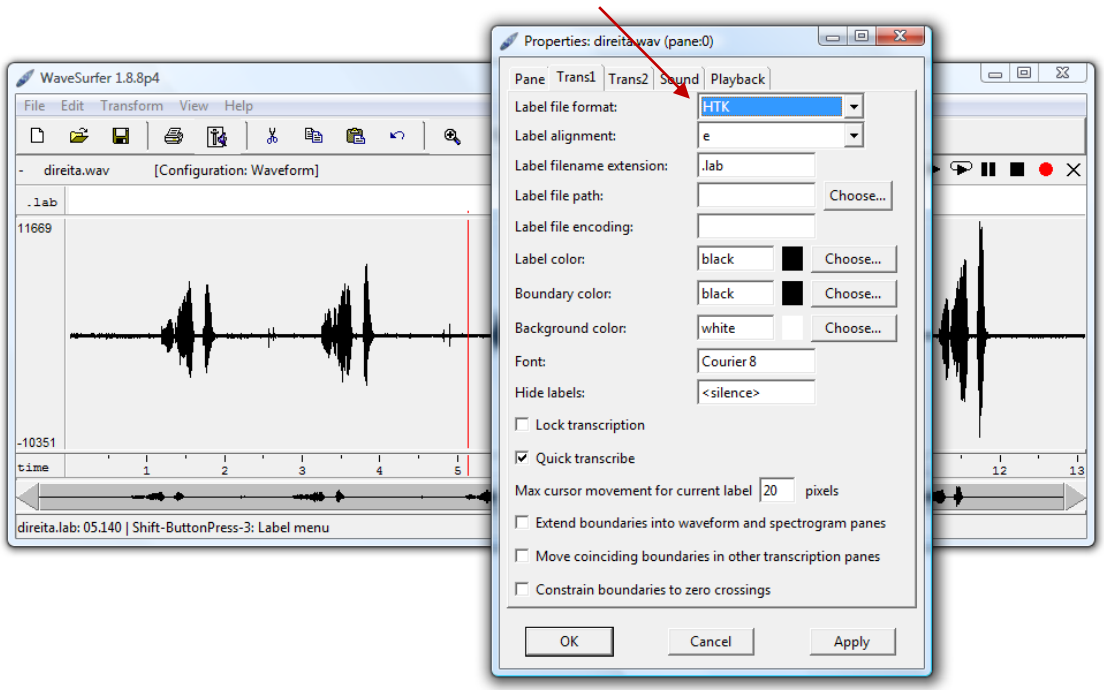
`teste1.wav`

O aluno 2, 3, etc. farão o mesmo

Reconhecimento Automático de Fala

3. Transcrição dos sinais de voz

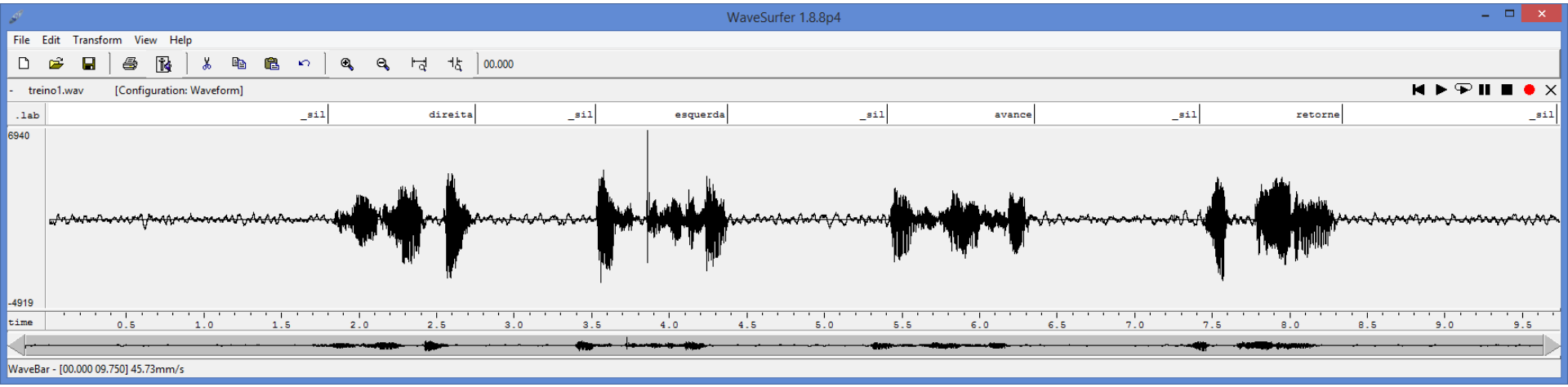
Adotar transcrição no formato HTK [1,2], que é o pacote a ser usado para o reconhecimento automático de fala: escolher “HTK transcription” na janela inicial antes da carga do arquivo ou abrir painel de transcrições e escolher formato ‘Label file format:’ HTK:



Reconhecimento Automático de Fala

3. Transcrição dos sinais de voz

Exemplo de sinal transcrito, `treino1.wav`, e correspondente arquivo de transcrição




```
> cat data/treino1.lab
0 17987968 _sil
17987968 27477273 direita
27477273 35245989 _sil
35245989 43744652 esquerda
43744652 54068182 _sil
54068182 63505348 avance
63505348 74193850 _sil
74193850 83370321 retorne
83370321 97239305 _sil
```

Obs: valores em unidades de 100ns, na transcrição no formato HTK

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos

Com os sinais de áudio gravados e respectivas transcrições, realiza-se agora o processo de treinamento dos modelos acústicos

Codificação dos arquivos de áudio

```
HCopy -A -T 1 -C cfg\hcopy.conf -S scripts\train_files.scp
```

onde

`hcopy.conf` contém parâmetros para codificação

`train_files.scp` contém origem dos sinais de áudio e destino dos sinais codificados, como por exemplo:

```
data/treino1.wav mfc/treino1.mfc
data/treino2.wav mfc/treino2.mfc
data/treino3.wav mfc/treino3.mfc
data/treino4.wav mfc/treino4.mfc
```

Reconhecimento Automático de Fala

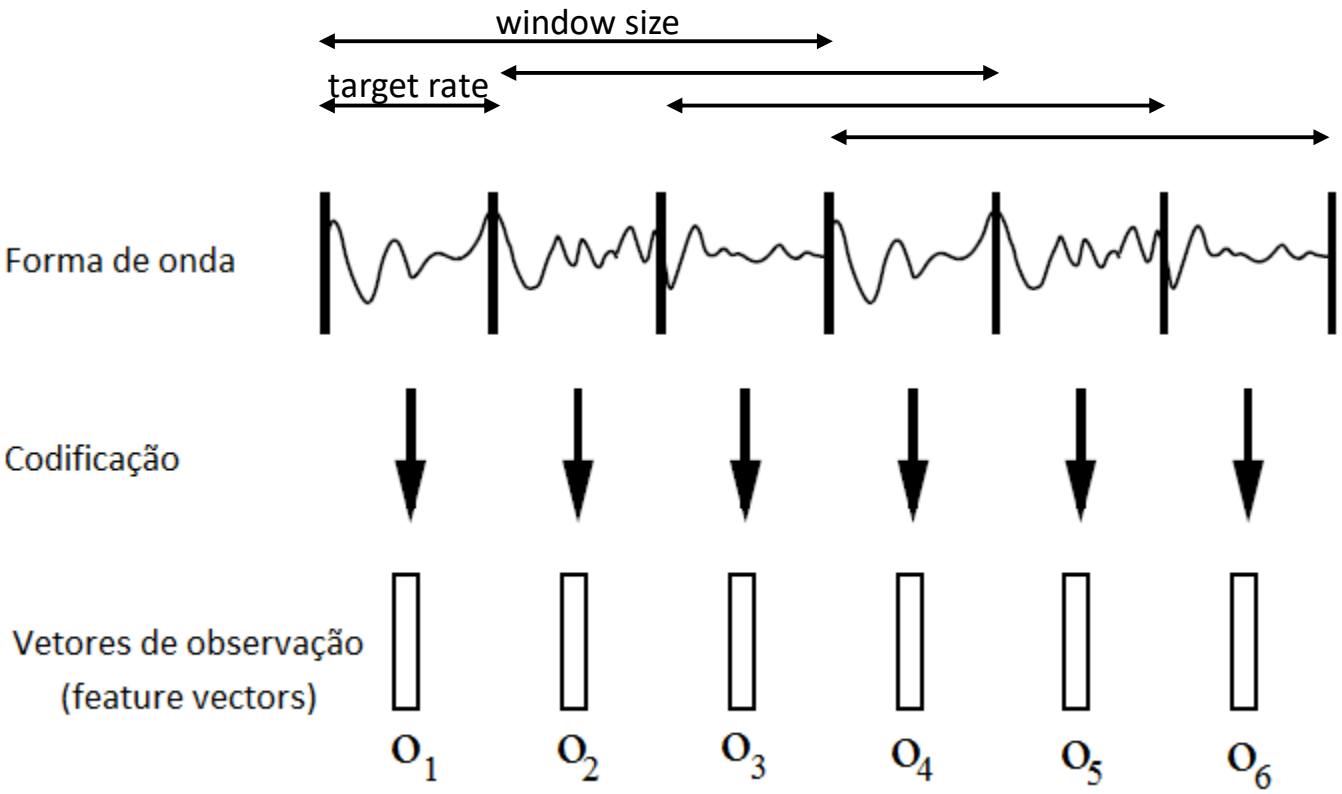
4. Treinamento de modelos acústicos (codificação)

Conteúdo de `hcopy.conf`

```
BYTEORDER       = VAX
SOURCEKIND      = WAVEFORM
SOURCEFORMAT    = WAV
SOURCERATE      = 625
ZMEANSOURCE     = TRUE

TARGETKIND      = MFCC_E_D_A
TARGETFORMAT    = HTK
TARGETRATE      = 100000
WINDOWSIZE      = 250000.0

NUMCHANS        = 24
ENORMALISE      = FALSE
```

Reconhecimento Automático de Fala

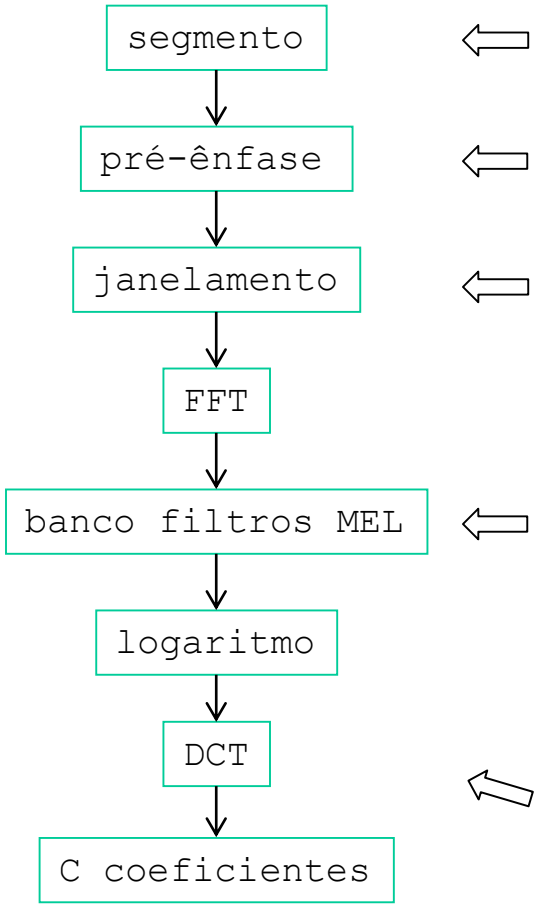
4. Treinamento de modelos acústicos (codificação)



Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos (codificação)

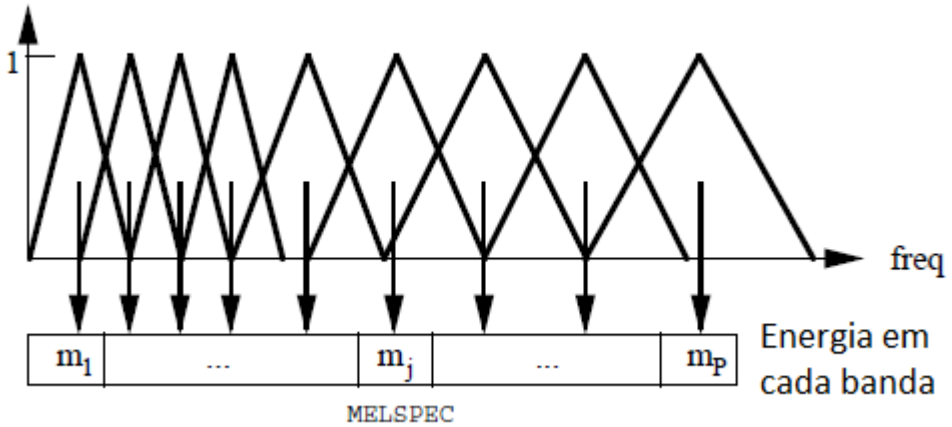
Cada vetor de observação (C coeficientes cepstrais) é obtido da seguinte forma



window size

$$y(n) = x(n) - \alpha\, x(n-1), \quad \alpha = 0.97$$ por exemplo

Hamming, Hanning, etc.



Discrete Cosine Transform

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos (codificação)

```
> HCopy -A -T 1 -C cfg\hcopy.conf -S scripts\train_files.scp
data/treino1.wav -> mfc/treino1.mfc
data/treino2.wav -> mfc/treino2.mfc
data/treino3.wav -> mfc/treino3.mfc
data/treino4.wav -> mfc/treino4.mfc

> HList -F WAV -h -e 18 data/treino1.wav
------------------------- Source: data/treino1.wav -------------------------
  Sample Bytes:  2        Sample Kind:   WAVEFORM
  Num Comps:     1        Sample Period: 62.5 us
  Num Samples:   156000   File Format:   WAV
------------------------------ Samples: 0->19 ------------------------------
    0:     155  -1373  -2492  -2176  -2316  -2269  -2220  -2432   -960    200
   10:     -76     42      3      6     26     28     23     32     38     31
---------------------------------- END -----------------------------------

> HList -h -e 0 mfc/treino1.mfc
------------------------------ Source: mfc/treino1.mfc -------------------------------
  Sample Bytes:  156      Sample Kind:   MFCC_E_D_A_K
  Num Comps:     39       Sample Period: 10000.0 us
  Num Samples:   973      File Format:   HTK
---------------------------------- Samples: 0->0 -----------------------------------
0:      -6.916  -3.314  -3.648  -6.929   7.129   2.228  -1.169  -1.504  12.015  -6.769
         5.253   1.210  17.330  -0.348   2.304   2.348   3.019  -0.810  -1.023   1.206
        -0.758  -2.775   2.295   0.094  -0.601  -0.904   0.027  -0.112   0.001   0.010
         0.154   0.040  -0.145   0.192   0.132  -0.052   0.000   0.128   0.261
---------------------------------------- END ----------------------------------------
```

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos (exemplo 'direita')

```
> HInit -A -T 3 -l direita -L data -M hmm0 proto\direita mfc\treino1.mfc mfc\treino2.mfc
mfc\treino3.mfc mfc\treino4.mfc
Initialising  HMM proto/direita . . .
 States   :   2  3  4  5  6 (width)
 Mixes  s1:   1  1  1  1  1 ( 39  )
 Num Using:   0  0  0  0  0
 Parm Kind:  MFCC_E_D_A
 Number of owners = 1
 SegLab   :  direita
 maxIter  :  20
 epsilon  :  0.000100
 minSeg   :  3
 Updating :  Means Variances MixWeights/DProbs TransProbs

 - system is PLAIN
 96 observations loaded from mfc/treino1.mfc
 113 observations loaded from mfc/treino2.mfc
 193 observations loaded from mfc/treino3.mfc
 100 observations loaded from mfc/treino4.mfc
5 Observation Sequences Loaded
Starting Estimation Process
Iteration 1: Average LogP = -6697.76074
Iteration 2: Average LogP = -6585.61572  Change =   112.14502
Iteration 3: Average LogP = -6581.09131  Change =     4.52441
Iteration 4: Average LogP = -6580.12500  Change =     0.96631
Iteration 5: Average LogP = -6579.86963  Change =     0.25537
Iteration 6: Average LogP = -6579.86963  Change =     0.00000
Estimation converged at iteration 7
Output written to directory hmm0
```

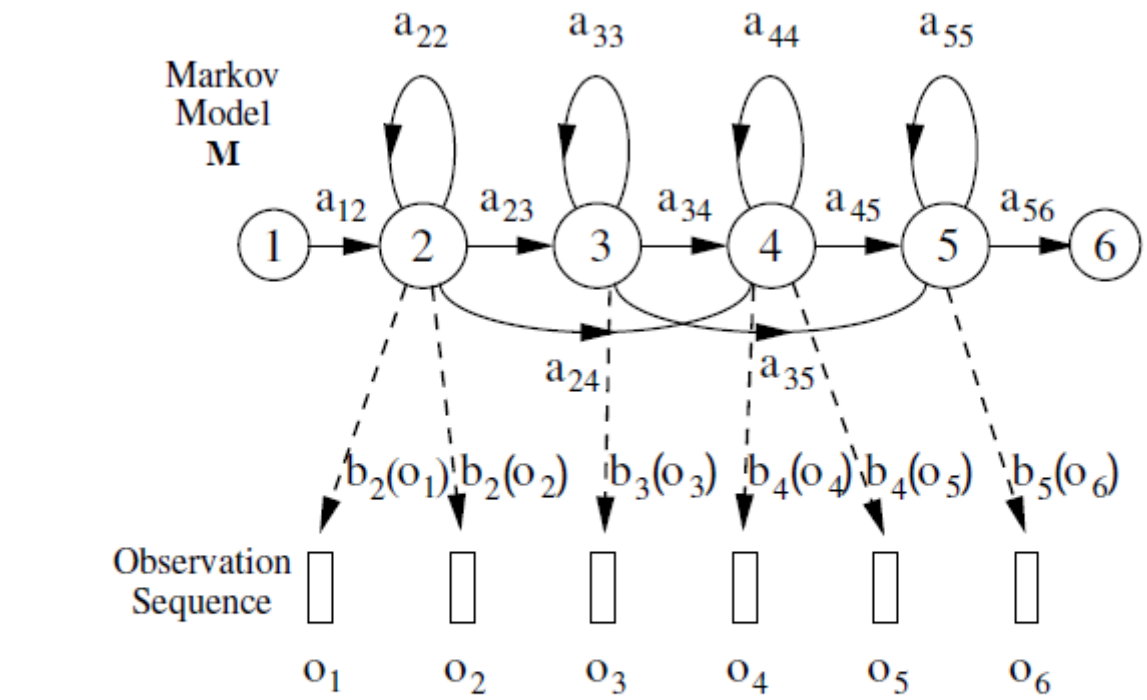
Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos (exemplo 'direita')

```
> HRest -A -T 3 -l direita -L data -M hmm1 -H hmm0\direita direita mfc\treino1.mfc mfc\treino2.mfc
mfc\treino3.mfc mfc\treino4.mfc
Reestimating HMM direita . . .
 States   :   2  3  4  5  6 (width)
 Mixes  s1:   1  1  1  1  1 ( 39  )
 Num Using:   0  0  0  0  0
 Parm Kind:  MFCC_E_D_A
 Number of owners = 1
 SegLab   :  direita
 MaxIter  :  20
 Epsilon  :  0.000100
 Updating :  Transitions Means Variances

 - system is PLAIN
 96 observations loaded from mfc/treino1.mfc
 113 observations loaded from mfc/treino2.mfc
 193 observations loaded from mfc/treino3.mfc
 100 observations loaded from mfc/treino4.mfc
5 Examples loaded, Max length = 113, Min length = 91
Ave LogProb at iter 1 = -6579.57178 using 5 examples
Ave LogProb at iter 2 = -6574.72021 using 5 examples  change =    4.85156
Ave LogProb at iter 3 = -6574.20459 using 5 examples  change =    0.51563
Ave LogProb at iter 4 = -6573.79053 using 5 examples  change =    0.41406
Ave LogProb at iter 5 = -6573.09082 using 5 examples  change =    0.69971
Ave LogProb at iter 6 = -6572.66406 using 5 examples  change =    0.42676
Ave LogProb at iter 7 = -6572.61084 using 5 examples  change =    0.05322
Ave LogProb at iter 8 = -6572.55225 using 5 examples  change =    0.05859
Ave LogProb at iter 9 = -6572.47656 using 5 examples  change =    0.07568
Ave LogProb at iter 10 = -6572.44141 using 5 examples  change =    0.03516
Ave LogProb at iter 11 = -6572.43750 using 5 examples  change =    0.00391
Ave LogProb at iter 12 = -6572.43604 using 5 examples  change =    0.00146
Ave LogProb at iter 13 = -6572.43604 using 5 examples  change =    0.00000
Estimation converged at iteration 13
```

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos - HMM



The Markov Generation Model

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos (exemplo 'direita')

```
> cat hmm1/direita
~o
<STREAMINFO> 1 39
<VECSIZE> 39<NULLD><MFCC_E_D_A><DIAGC>
~h "direita"
<BEGINHMM>
<NUMSTATES> 7
<STATE> 2
<MEAN> 39
 -5.741704e+000 7.520256e+000 5.533943e+000 3.059285e+000 3.685120e+000 2.324062e-001 3.423630e+000
-8.268481e-001 1.245726e+000 -1.451842e+000 6.718845e-001 -7.393590e-001 1.618561e+001 1.863393e-001
1.609667e-001 2.302677e-001 7.814348e-002 -5.339318e-002 -1.629242e-001 -1.482808e-001 -2.381710e-
001 -1.974270e-001 -2.023672e-001 2.898869e-002 -2.000098e-001 9.396002e-002 -6.538748e-002 -
8.399925e-002 6.066366e-003 -4.244983e-002 -1.209197e-001 -7.545752e-002 -2.073816e-002 -2.596173e-
002 -3.734947e-002 -1.885379e-002 8.843935e-002 -3.115564e-002 1.360707e-002
<VARIANCE> 39
 1.319786e+001 1.513757e+001 1.402997e+001 6.723221e+000 7.245210e+000 1.141768e+001 1.339027e+001
1.366583e+001 1.383768e+001 1.331502e+001 9.294785e+000 1.090619e+001 2.160706e+000 9.349041e-001
1.286147e+000 9.383922e-001 6.469566e-001 1.312899e+000 1.210882e+000 1.516942e+000 1.555333e+000
1.867312e+000 1.230428e+000 1.206075e+000 8.861809e-001 1.338089e-001 1.613421e-001 1.951574e-001
1.785131e-001 1.181565e-001 3.747819e-001 2.885528e-001 2.762757e-001 3.261987e-001 4.471501e-001
2.305780e-001 2.407505e-001 1.848699e-001 2.326322e-002
<GCONST> 8.055302e+001
<STATE> 3
<MEAN> 39
etc. (continuação no próximo slide)
```

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos (exemplo 'direita')

```
(trecho cortado/continuação)
<STATE> 6
<MEAN> 39
 -9.628569e-001 -5.021389e+000 -1.095100e+000 -1.800025e+000 -5.136259e+000 -3.792818e+000 -
3.592028e+000 1.500133e+000 5.542352e+000 -1.517571e+000 -2.893304e+000 -3.495200e+000 1.713465e+001
-3.864798e-001
6.491910e-001 4.059693e-002 2.848185e-001 6.958485e-001 7.041485e-002 5.139159e-001 -1.139621e-001
1.201493e-001 8.230814e-002 1.044833e-001 3.818709e-001 -2.223516e-001 -6.555285e-002 1.027952e-001
1.658931e-002 -4.739230e-003 -3.125330e-003 -5.558917e-002 1.223769e-002 -3.395791e-002 3.459720e-
002 3.526522e-002 4.626533e-002 4.657534e-002 -4.616988e-003
<VARIANCE> 39
 1.797602e+001 4.114077e+001 6.571036e+000 1.141293e+001 3.571069e+001 1.410005e+001 3.154197e+001
2.517715e+001 1.766061e+001 2.902270e+001 2.130890e+001 2.305840e+001 4.458227e+000 3.039902e-001
8.849986e-001 5.684738e-001 5.882849e-001 1.297806e+000 1.523089e+000 1.486329e+000 1.601807e+000
1.667073e+000 2.697398e+000 1.675963e+000 1.190604e+000 9.361739e-002 5.875409e-002 2.372969e-001
1.158895e-001 1.213620e-001 3.096076e-001 2.683584e-001 3.590322e-001 3.346377e-001 3.304111e-001
4.890463e-001 3.183975e-001 2.798463e-001 1.584858e-002
<GCONST> 8.686790e+001
<TRANSP> 7
 0.000000e+000 1.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
 0.000000e+000 9.473104e-001 5.268962e-002 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
 0.000000e+000 0.000000e+000 9.543110e-001 4.568904e-002 0.000000e+000 0.000000e+000 0.000000e+000
 0.000000e+000 0.000000e+000 0.000000e+000 9.387555e-001 6.124453e-002 0.000000e+000 0.000000e+000
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 9.505826e-001 4.941746e-002 0.000000e+000
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 9.564669e-001 4.353317e-002
 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000 0.000000e+000
<ENDHMM>
```

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos (teste de sanidade)

```
> HVite -A -T 1 -d hmm1 -S scripts\test_trainfiles.scp -i mlf\test_trainfiles.mlf -w
net\network_train dics\dictionary lists\models
Read 5 physical / 5 logical HMMs
Read lattice with 9 nodes / 14 arcs
Created network with 17 nodes / 22 links
File: mfc/treino1.mfc
_sil _sil _sil _sil direita _sil _sil esquerda _sil _sil avance _sil _sil retorne _sil _sil  ==
[973 frames] -56.5951 [Ac=-55067.0 LM=0.0] (Act=14.9)
File: mfc/treino2.mfc
_sil _sil _sil _sil _sil direita _sil _sil esquerda _sil _sil avance _sil _sil _sil retorne _sil  ==
[1048 frames] -56.4438 [Ac=-59153.1 LM=0.0] (Act=14.9)
File: mfc/treino3.mfc
_sil _sil _sil direita _sil _sil _sil esquerda _sil _sil _sil avance _sil _sil _sil _sil retorne
_sil _sil direita _sil _sil esquerda _sil _sil avance _sil _sil retorne _sil _sil _sil _sil _sil  ==
[1711 frames] -56.6933 [Ac=-97002.2 LM=0.0] (Act=14.9)
File: mfc/treino4.mfc
_sil _sil direita _sil _sil esquerda _sil _sil avance _sil _sil retorne _sil _sil  ==  [861 frames]
-56.4611 [Ac=-48613.0 LM=0.0] (Act=14.9)
```

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos (resultados de sanidade)

```
> HResults -A -h -z _sil -f -t -p -L data lists\models mlf\test_trainfiles.mlf
    ,-----------------------------------------------------------.
    | HTK Results Analysis at Fri Jan 23 22:21:52 2015          |
    | Ref: data                                                 |
    | Rec: mlf\test_trainfiles.mlf                              |
    |-----------------------------------------------------------|
    |      FILE       |  Corr    Sub    Del    Ins    Err       |
    |-----------------------------------------------------------|
    |     treino1.rec | 100.00   0.00   0.00   0.00   0.00      |
    |-----------------------------------------------------------|
    |     treino2.rec | 100.00   0.00   0.00   0.00   0.00      |
    |-----------------------------------------------------------|
    |     treino3.rec | 100.00   0.00   0.00   0.00   0.00      |
    |-----------------------------------------------------------|
    |     treino4.rec | 100.00   0.00   0.00   0.00   0.00      |
    |===========================================================|
    |           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
    |-----------------------------------------------------------|
    | Sum/Avg |    4  | 100.00   0.00   0.00   0.00   0.00   0.00 |
    `-----------------------------------------------------------'
----------------------- Confusion Matrix -------------------------
       a   d   e   r
       v   i   s   e
       a   r   q   t
       n   e   u   o
       c   i   e   r  Del [ %c / %e]
avan   5   0   0   0    0
dire   0   5   0   0    0
esqu   0   0   5   0    0
reto   0   0   0   5    0
Ins    0   0   0   0
```

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos (referências [1] e [2])

Let each spoken word be represented by a sequence of speech vectors or *observations* $\boldsymbol{O}$, defined as

$$\boldsymbol{O} = o_1, o_2, \ldots, o_T \tag{1.1}$$

where o_t is the speech vector observed at time t. The isolated word recognition problem can then be regarded as that of computing

$$\arg\max_i \{P(w_i|\boldsymbol{O})\} \tag{1.2}$$

where w_i is the i'th vocabulary word. This probability is not computable directly but using Bayes' Rule gives

$$P(w_i|\boldsymbol{O}) = \frac{P(\boldsymbol{O}|w_i)P(w_i)}{P(\boldsymbol{O})} \tag{1.3}$$

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos

$$P(\boldsymbol{O}, X|M) = a_{12}b_2(\boldsymbol{o}_1)a_{22}b_2(\boldsymbol{o}_2)a_{23}b_3(\boldsymbol{o}_3)\ldots \tag{1.4}$$

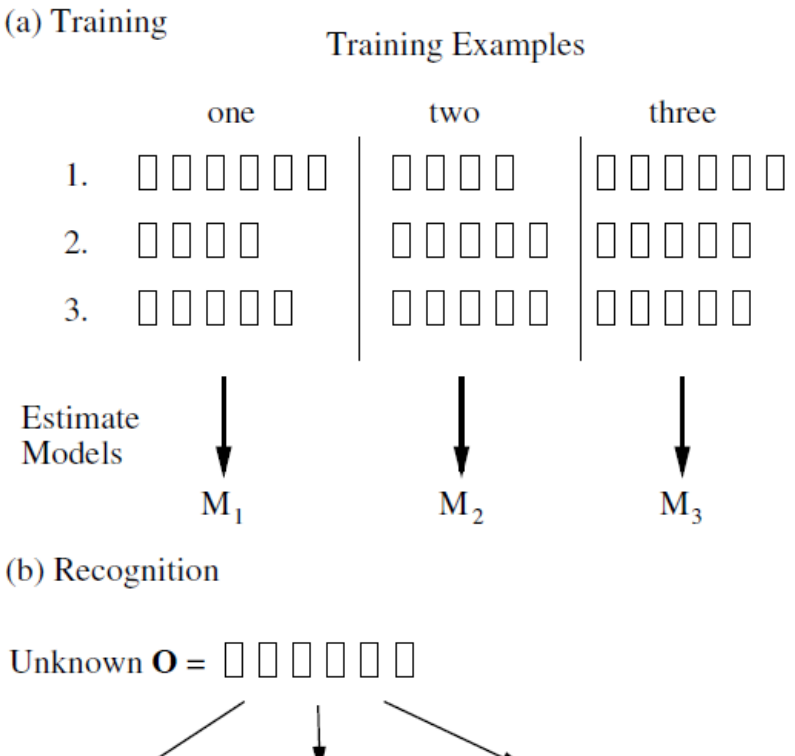
$$P(\boldsymbol{O}|M) = \sum_X a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(\boldsymbol{o}_t)a_{x(t)x(t+1)} \tag{1.5}$$

$$\hat{P}(\boldsymbol{O}|M) = \max_X \left\{ a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(\boldsymbol{o}_t)a_{x(t)x(t+1)} \right\} \tag{1.6}$$

$$P(\boldsymbol{O}|w_i) = P(\boldsymbol{O}|M_i). \tag{1.7}$$

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos



Using HMMs for Isolated Word Recognition

Reconhecimento Automático de Fala

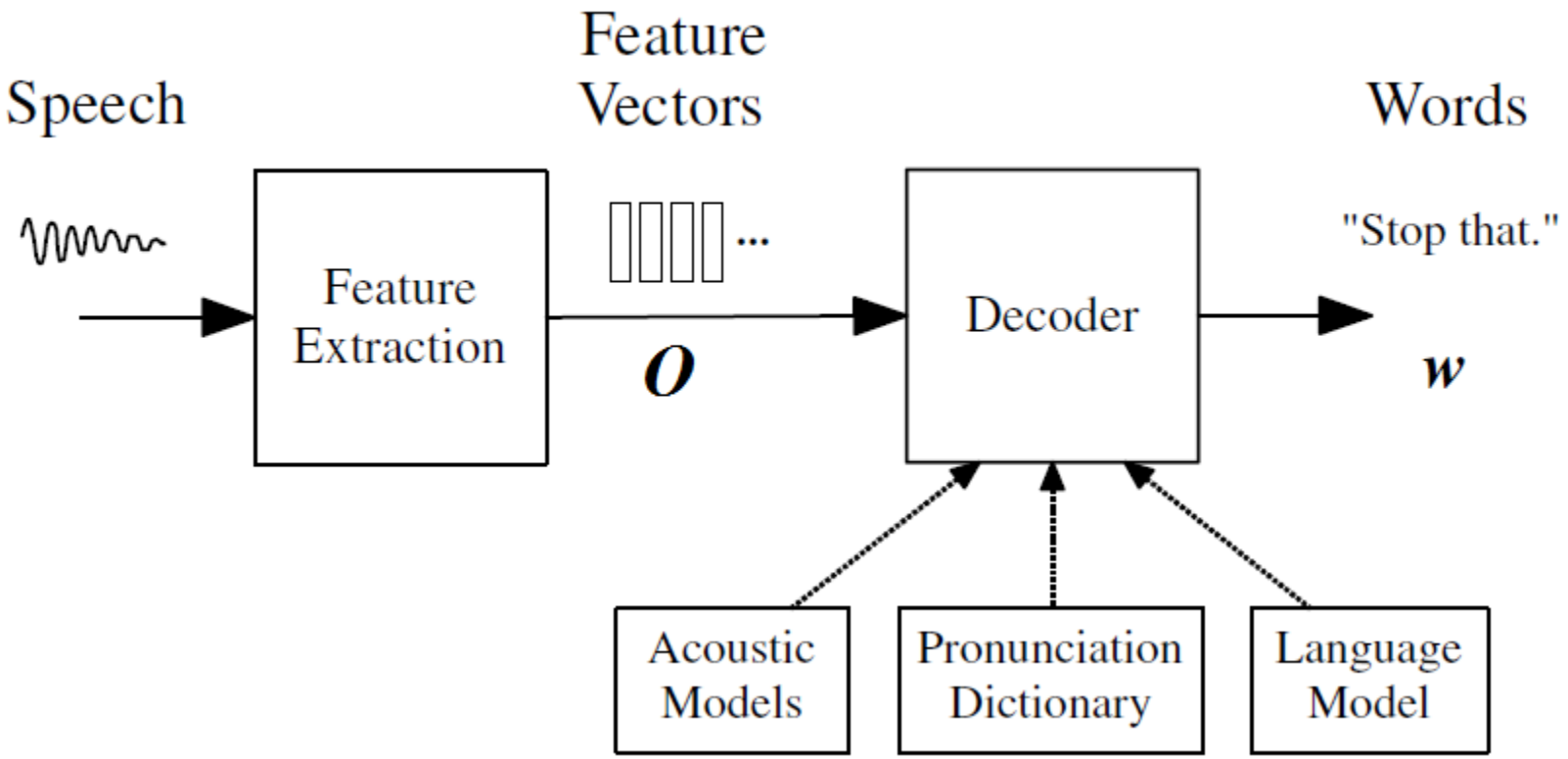
4. Treinamento de modelos acústicos

$$b_j(o_t) = \prod_{s=1}^{S} \left[\sum_{m=1}^{M_s} c_{jsm} \mathcal{N}(o_{st}; \mu_{jsm}, \Sigma_{jsm}) \right]^{\gamma_s} \tag{1.8}$$

$$\mathcal{N}(o; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o-\mu)^\top \Sigma^{-1} (o-\mu)} \tag{1.9}$$

Reconhecimento Automático de Fala

4. Treinamento de modelos acústicos



Reconhecimento Automático de Fala

5. Avaliação da precisão de reconhecimento

Para a avaliação dos modelos acústicos deve-se utilizar os arquivos de voz rotulados como `teste*.wav`, os quais não tiveram participação no processo de treinamento. Isto é, os arquivos de teste devem ser independentes dos arquivos utilizados no treinamento

Codificação dos arquivos de teste:

```
> HCopy -A -T 1 -C cfg\hcopy.conf -S scripts\test_files.scp
data/teste1.wav -> mfc/teste1.mfc
data/teste2.wav -> mfc/teste2.mfc
```

Reconhecimento Automático de Fala

5. Avaliação da precisão de reconhecimento

A gramática deve ser elaborada de acordo com os dados de teste. Por exemplo, se os arquivos de teste contêm apenas uma pronúncia de uma palavra, a gramática deve ser da forma:

```
> cat net/grammar
/*
 * algumas regras da gramatica
 * |  alternatives
 * [] options
 * {} zero or more repetitions
 * <> one or more repetitions
 *
 */

( {_sil}  (avance | direita | esquerda | retorne)  {_sil} )
```

Caso haja mais de uma pronúncia por arquivo de teste, a gramática fica:

```
(<(<_sil> (avance | direita | esquerda | retorne) <_sil >) >)
```

Reconhecimento Automático de Fala

5. Avaliação da precisão de reconhecimento

O comando `HParse` processa o arquivo de gramática e produz o arquivo de network, o qual é a representação textual do grafo definido pela gramática

```
> HParse -A -T 1 net\grammar net\network
Creating HParse net from file net\grammar
Generating Lattice with 10 nodes and 14 links
Writing Word Lattice to net\network
```

Reconhecimento Automático de Fala

5. Avaliação da precisão de reconhecimento

O comando `HVite` realiza o processo de reconhecimento (como visto no teste de sanidade para os dados de treino)

```
> HVite -A -T 1 -d hmm1 -S scripts\test_testfiles.scp -i mlf\test_testfiles.mlf -w net\network_train
dics\dictionary lists\models
Read 5 physical / 5 logical HMMs
Read lattice with 9 nodes / 14 arcs
Created network with 17 nodes / 22 links
File: mfc/teste1.mfc
_sil _sil _sil _sil direita _sil _sil esquerda _sil _sil avance _sil _sil _sil retorne _sil _sil
direita _sil _sil esquerda _sil _sil avance _sil _sil _sil retorne _sil  ==  [1842 frames] -55.5355
[Ac=-102296.4 LM=0.0] (Act=14.9)
File: mfc/teste2.mfc
_sil _sil _sil _sil direita _sil _sil _sil esquerda _sil _sil avance _sil _sil _sil esquerda _sil
_sil retorne _sil  ==  [929 frames] -56.4599 [Ac=-52451.2 LM=0.0] (Act=14.9)
```

Reconhecimento Automático de Fala

5. Avaliação da precisão de reconhecimento

O comando `HResults` apresenta os resultados com formatação conveniente

```
> HResults -A -h -z _sil -f -p -t -L data lists\models mlf\test_testfiles.mlf
    ,-----------------------------------------------------------.
    | HTK Results Analysis at Fri Jan 23 22:21:53 2015          |
    | Ref: data                                                 |
    | Rec: mlf\test_testfiles.mlf                               |
    |-----------------------------------------------------------|
    |      FILE       |  Corr    Sub    Del    Ins    Err       |
    |-----------------------------------------------------------|
    |      teste1.rec | 100.00   0.00   0.00   0.00   0.00      |
    |-----------------------------------------------------------|
    |      teste2.rec | 100.00   0.00   0.00  25.00  25.00      |
Aligned transcription: data/teste2.lab vs mfc/teste2.rec
 LAB: direita esquerda avance          retorne
 REC: direita esquerda avance esquerda retorne
    |===========================================================|
    |           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
    |-----------------------------------------------------------|
    | Sum/Avg |    2  | 100.00   0.00   0.00   8.33   8.33  50.00 |
    `-----------------------------------------------------------'
----------------------- Confusion Matrix -------------------------
       a   d   e   r
       v   i   s   e
       a   r   q   t
       n   e   u   o
       c   i   e   r  Del [ %c / %e]
avan   3   0   0   0    0
dire   0   3   0   0    0
esqu   0   0   3   0    0
reto   0   0   0   3    0
Ins    0   0   1   0
```

Reconhecimento Automático de Fala

5. Avaliação da precisão de reconhecimento

Ao analisar os resultados do slide anterior vê-se que, das $N = 12$ pronúncias existentes nos dados de teste, todas foram reconhecidas corretamente. Contudo, no arquivo `teste2.wav` houve o reconhecimento de uma pronúncia inexistente. Diz-se que ocorreu uma *inserção* (I). Caso uma das palavras tivesse sido reconhecida erradamente, diz-se que ocorreu uma *substituição* (S). Finalmente, caso uma pronúncia existente não tiver resultado correspondente, diz-se que ocorreu uma *deleção* (D).
A taxa de erro final por palavra é dada pela seguinte expressão:

$$Err = \frac{S + D + I}{N} \times 100\%$$

Assim, a precisão (*accuracy*) do sistema é dada por $Acc = 100 - Err$

Do slide anterior, vê-se que $N = 12$, $S = 0$, $D = 0$, $I = 1$, resultando $Err = 8.33\%$, e consequentemente, $Acc = 91{,}67\%$

Obs: desconsiderar neste caso o erro por sentença, $S.\ Err$

Reconhecimento Automático de Fala

6. Testes 'live'

```
> HVite -C cfg\hvite_live.conf -g -e -d hmm1 -i mlf\testliveout.mlf -w net\network dics\dictionary lists\models
```

Onde

```
> cat cfg/hvite_live.conf
BYTEORDER      = VAX
SOURCEKIND     = HAUDIO
SOURCEFORMAT   = HTK
SOURCERATE     = 625
ZMEANSOURCE    = TRUE

TARGETKIND     = MFCC_E_D_A
TARGETRATE     = 100000
WINDOWSIZE     = 250000.0

NUMCHANS       = 24

# Waveform capture
ENORMALISE     = FALSE
USESILDET      = TRUE
MEASURESIL     = FALSE
OUTSILWARN     = TRUE
```

```
> cat net/grammar

( {_sil}  (avance   |
           direita  |
           esquerda |
           retorne)
  {_sil} )
```

```
> HParse -A -T 1 net\grammar net\network
```

Referências/Bibliografia adicional

[1] The HTK Book (HTK version 3.5)
Steve Young et al
Cambridge University, 2015
http://htk.eng.cam.ac.uk/
acesso em 18/02/2020

[2] The Application of Hidden Markov Models in Speech Recognition
Mark Gales and Steve Young
Foundations and Trends in Signal Processing,
Vol. 1, No. 3 (2007), 195-304
https://mi.eng.cam.ac.uk/~mjfg/mjfg_NOW.pdf,
acesso em 18/02/2020

Algumas outras aplicações

Microphone array - filtragem espacial e localização de fontes sonoras

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.

Análise eficiente - estudo de algoritmos

Speech enhancement - redução de ruído

Aplicações em música – análise/transformação de música/canto