

Speech signal processing using MATLAB

Basics and applications

Stephan Kuberski

(`kuberski@uni-potsdam.de`)

April/May 2016

Slides and MATLAB scripts and data at <https://github.com/murtex/spl>

Digital signals

- Sampling

- Time domain

- Frequency domain

- Filters

Acoustic signals

- Short-time analysis

- Spectrograms

- Activity detection

- Landmarks detection

- Formants detection

Digital signals/Sampling

- ▶ **continuous signal** (normalized magnitude, length L in seconds)

$$x(t) \in [-1, 1] \quad \text{with} \quad t \in [0, L]$$

```
>> x = @( t ) sin( 2*pi*f * t ); % continuous sine with frequency f
```

- ▶ **sampling rate** f_S , quantization of time

$$t \rightarrow t_i = \frac{i-1}{f_S} \quad \text{with} \quad i \in \{1, \dots, N\} \quad \text{and} \quad N = \lfloor Lf_S \rfloor$$

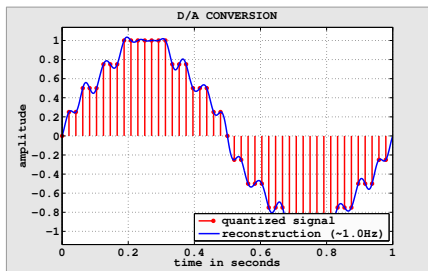
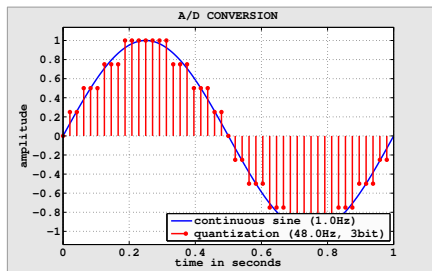
```
>> N = floor( L * fS ); % number of samples  
>> ti = (0:N-1) / fS; % quantized time values
```

- ▶ **bits per sample** n_S , quantization of amplitude

$$x(t) \rightarrow x_i = \frac{\lfloor 2^{n_S-1} x(t_i) \rfloor}{2^{n_S-1}}$$

```
>> xi = round( 2^(nS-1) * x( ti ) ) / 2^(nS-1); % quantized amplitudes
```

- ▶ example: matlab/sampling.m



- ▶ exercise:
 - ▶ verify from reconstruction that Nyquist frequency holds

$$f_{\text{Ny}} = \frac{f_s}{2}$$

- ▶ compare commonly used **sampling standards** (telephony, Audio-CD, professional audio equipment, ...)

Digital signals/Time domain

- ▶ total energy, average power and root mean square

$$E = \sum_{i=1}^N x_i^2, \quad P = \frac{1}{N} \sum_{i=1}^N x_i^2 \quad \text{and} \quad RMS = \sqrt{\frac{1}{N} \sum_{i=1}^N x_i^2}$$

```
>> E = sum( xi .* xi ); % total energy  
>> P = mean( xi .* xi ); % average power  
>> RMS = sqrt( mean( xi .* xi ) ); % root mean square
```

- ▶ decibel full scale, different for **power**- and **magnitude**-like quantities, e. g.

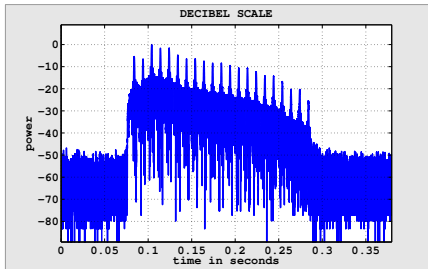
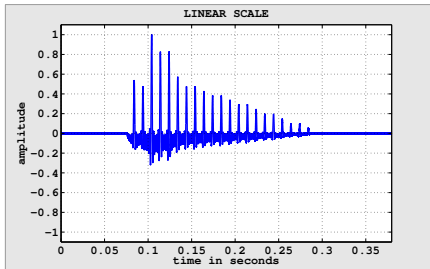
$$P_{\text{dB}} = 10 \log_{10}(P) \quad \text{and} \quad RMS_{\text{dB}} = 20 \log_{10}(RMS)$$

```
>> PdB = 10 * log10( P ); % power-like  
>> RMSdB = 20 * log10( RMS ); % magnitude-like
```

- ▶ zero-crossings rate

```
>> fZ = sum( abs( diff( xi >= 0 ) ) ) / N * fS;
```

- ▶ **example:** `matlab/decibel.m` (`matlab/sound.wav`)



- ▶ **exercise:**
 - ▶ compare **linear** and **logarithmic** scales
 - ▶ explain **negative decibel values** (e. g. -3 dB power, -6 dB magnitude)
 - ▶ specify the power of silence in decibels

Digital signals/Frequency domain

- ▶ **discrete Fourier transform**, time domain \rightarrow frequency domain

$$X_k = \sum_{i=1}^N x_i e^{-2\pi i \frac{(i-1)(k-1)}{N}} \in \mathbb{C} \quad \text{with} \quad k \in \{1, \dots, N\}$$

```
>> Xk = fft( xi ) / N; % complex Fourier coefficients
```

- ▶ k is a **frequency index** (as i was a time index)

$$k \rightarrow f_k = \frac{k-1}{N} f_s$$

```
>> fk = (0:N-1) / N * fs; % frequency values
```

- ▶ frequencies beyond Nyquist frequency are **negative frequencies**

$$f_k \rightarrow \begin{cases} f_k - f_s & \text{if } f_k > f_{Ny} \\ f_k & \text{otherwise} \end{cases}$$

```
>> fk(fk > fNy) = fk(fk > fNy) - fs; % imply negative frequencies
```

- ▶ **power spectral density** (also known as **power spectrum**)

$$P_k = |X_k|^2 \in \mathbb{R} \quad \Leftarrow \quad \sum_{k=1}^N P_k = P$$

```
>> Pk = abs( Xk ) .^ 2; % power spectral density
```

- ▶ **real valued signals** ($x_i \in \mathbb{R}$) imply a special symmetry

$$X_{f_k} = X_{-f_k}^* \quad \Rightarrow \quad P_{f_k} = P_{-f_k}$$

- ▶ restrict to **one-sided spectrum**

```
>> Pk(fk < 0) = []; % remove negative frequency components
>> Xk(fk < 0) = [];
>> fk(fk < 0) = [];
>> Pk(2:end) = 2 * Pk(2:end); % rescale to match total power
>> Xk(2:end) = sqrt( 2 ) * Xk(2:end);
```

- ▶ P_1 is **DC offset**, $P_{k>1}$ are **contributions of sines** with frequencies $f_{k>1}$

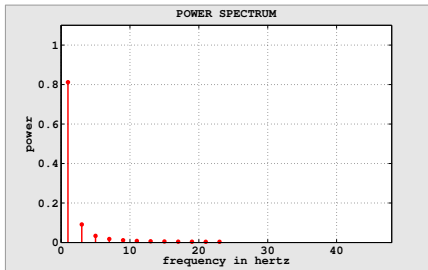
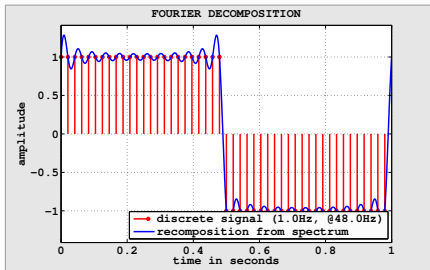
$$x(t) = \sqrt{P_1} + \sqrt{2} \sum_{k>1} \sqrt{P_k} \sin(2\pi f_k t)$$

Frequency domain

- ▶ complex valued but without loss of phase information

$$x(t) = X_1 + \sqrt{2} \sum_{k>1} X_k e^{2\pi i f_k t}$$

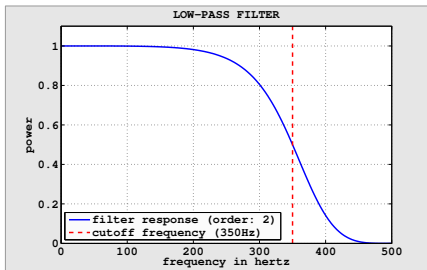
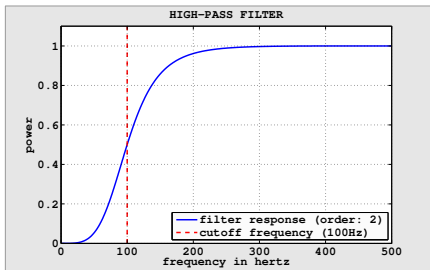
- ▶ example: matlab/fdomain.m



- ▶ exercise:
 - ▶ examine spectra of different wave forms (sines, square, sawtooth, ...)
 - ▶ examine spectral **frequency range**
 - ▶ verify loss of **phase information** in (real valued) power spectra

Digital signals/Filters

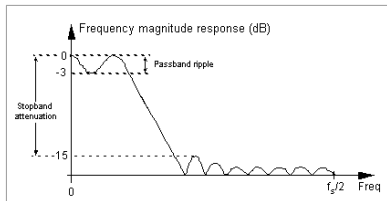
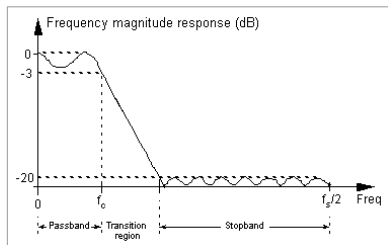
- ▶ **general filter types:**
 - ▶ **low-pass:** passes low frequencies (cuts high ones)
 - ▶ **high-pass:** passes high frequencies (cuts low ones)
 - ▶ **band-pass:** passes a range of frequencies (combination of low- and high-pass)
 - ▶ **band-stop (notch):** cuts a range of frequencies (opposite of band-pass)
- ▶ **cutoff frequency** at which output power is (generally) reduced by -3 dB
- ▶ **example:** `matlab/filters.m`



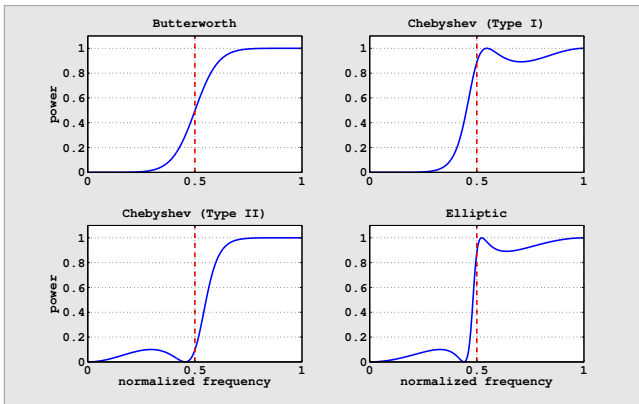
- ▶ filters are represented by **filter coefficients** b_j (feedforward) and a_j (feedback)
- ▶ high **filter order** m increases computational complexity but thereby quality

$$y_i = \frac{1}{a_1} \left(\overbrace{\sum_{j=0}^m b_{j+1} x_{i-j}}^{\text{IIR}} - \underbrace{\sum_{j=1}^m a_{j+1} y_{i-j}}_{\text{FIR}} \right) \quad \text{with } i \in \{1, \dots, N\}$$

- ▶ **FIR filters** (finite impulse response) are slow to compute but stable
- ▶ **IIR filters** (infinite impulse response) are fast to compute but might be unstable
- ▶ some often used **additional terms** (images from <http://dspguru.com>)



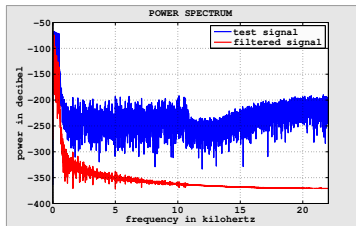
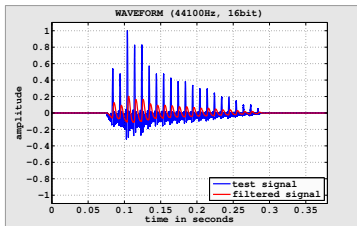
- ▶ example: `matlab/filters2.m`



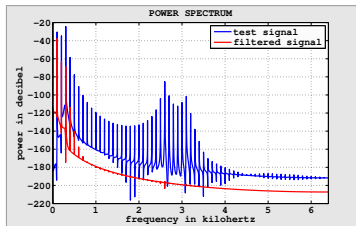
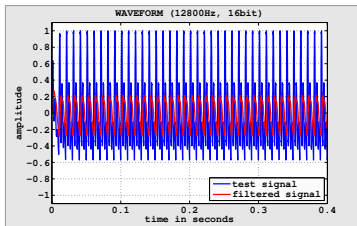
- ▶ many filter families with different characteristics
- ▶ normalized frequency

$$\tilde{f}_k = \frac{f_k}{f_{Ny}} = \frac{2f_k}{f_s} \in [0, 1] \quad \text{with } k \in \{1, \dots, N\}$$

- ▶ example: `matlab/spectrum.m` (`matlab/sound.wav`)



- ▶ example: `matlab/spectrum.m` (`matlab/ivowel.wav`)



- ▶ exercise:

- ▶ observe the occurrence of filter delay

- ▶ **Butterworth filter** (high-pass, second-order, 100 Hz cutoff)

```
>> m = 2; % filter order
>> cutoff = 100; % cutoff frequency
>> [b, a] = butter( m, cutoff / (fs/2), 'high' );
```

- ▶ **Chebyshev filter** (high-pass, 1 dB ripple, 40 dB attenuation, 100 Hz cutoff)

```
>> cutoff = 100; % cutoff frequency
>> stopband = 90; % stopband frequency
>> ripple = 1; % passband ripple
>> attenuation = 40; % stopband attenuation
>> m = cheb2ord( cutoff / (fs/2), stopband / (fs/2), ripple, attenuation );
>> [b, a] = cheby2( m, attenuation, stopband / (fs/2) );
```

- ▶ **apply any filter**

```
>> y = filter( b, a, x ); % filter signal x using coefficients a, b
```

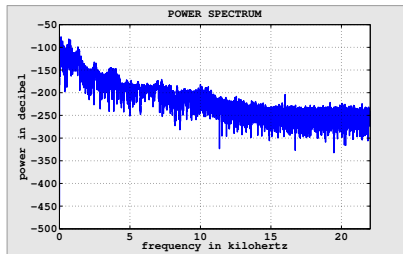
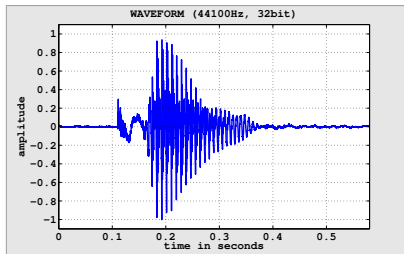
- ▶ **or in zero-phase version (without filter delay)**

```
>> y = filtfilt( b, a, x ); % zero-phase filtering
```

Acoustic signals/Short-time analysis

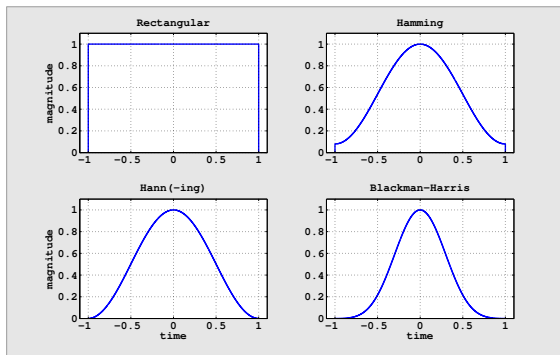
Short-time analysis

- ▶ **spectral and temporal analysis** is essential for speech acoustics
- ▶ problem:
 - ▶ **power spectrum** has no temporal information anymore (matlab/tam.wav)



- ▶ solution:
 - ▶ choose **short overlapping segments** (windows) at different time points
 - ▶ length of the segments (**window size**) is crucial
 - ▶ **overlap** and **window function** control spectral leakage
 - ▶ aligning Fourier transforms of these (altered) segments leads to **spectrograms**

- ▶ example: `matlab/windows.m`



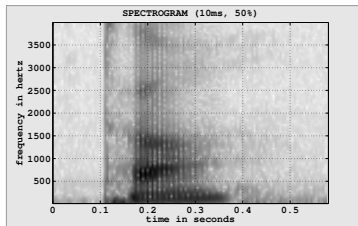
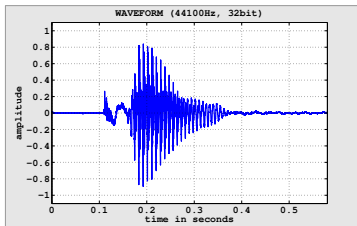
- ▶ optimal overlapping for minimal spectral leakage

Rectangular:	any value
Hamming:	50%
Hann(-ing):	50%
Blackman-Harris:	66.1%

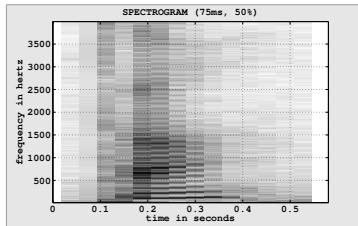
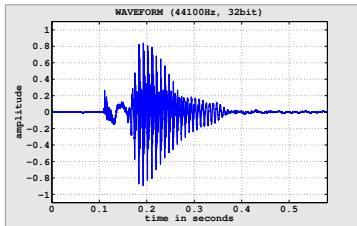
- ▶ other commonly used window functions: **Welch, Kaiser, Gaussian, ...**

Acoustic signals/Spectrograms

- ▶ example: `matlab/spectrogram.m` (`matlab/tam.wav`)



- ▶ example: `matlab/spectrogram.m` (`matlab/tam.wav`)



- ▶ exercise:

- ▶ impact of window size → broad-band vs. narrow-band spectrogram

- ▶ **broad-band spectrograms** have good temporal but poor spectral resolution
- ▶ **narrow-band spectrograms** have poor temporal but good spectral resolution

spectrogram:	broad-band	narrow-band
window size:	< 20 ms	> 20 ms
structures:	formants	harmonics

- ▶ set up windowing

```
>> wsize = 10; % window size in milliseconds
>> woverlap = 66; % window overlap in percent
>> wfunc = @blackmanharris; % window function
```

- ▶ compute the spectrogram

```
>> [Xk, fk, ti] = spectrogram( xi, ... % signal
    wfunc( ceil( wsize/1000 * fS ) ), ... % window function values
    ceil( woverlap/100 * wsize/1000 * fS ), ... % window overlap samples
    4096, fS ); % fourier transform
```

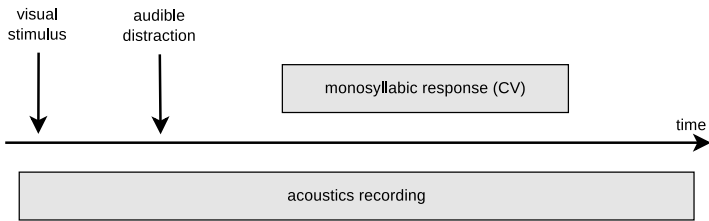
- ▶ plot the spectrogram

```
>> colormap( flipud( colormap( 'gray' ) ) ); % set color coding
>> imagesc( ti, fk, Pk .^ 0.1 ); % plot spectral powers
```


Acoustic signals/Activity detection

Activity detection

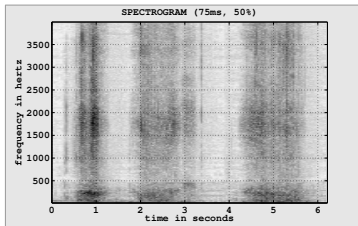
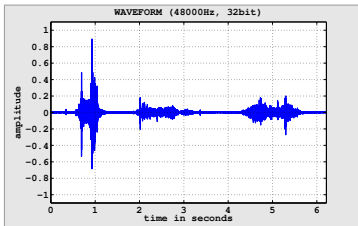
- ▶ **experimental data** often contain a lot of noise and little of information
- ▶ for **automatic processing** restriction to important parts is essential
- ▶ consider the following experiment:



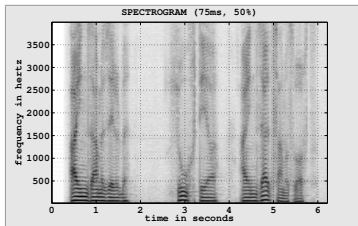
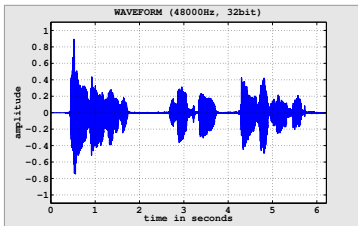
- ▶ with features of interest:
 - ▶ responded syllable (out of a specific set → classification task)
 - ▶ voice onset time (→ **landmarks detection**)
 - ▶ formants onsets (frequency and time → **formants tracking/detection**)
- ▶ all of these require (human) **activity detection** as an initial processing pass

Activity detection

- ▶ in literature usually called **voice activity detection** (VAD)
- ▶ exploiting **spectral differences** in human speech and ambient sound/noise
- ▶ example: `matlab/spectgram.m` (`matlab/chair.wav`)

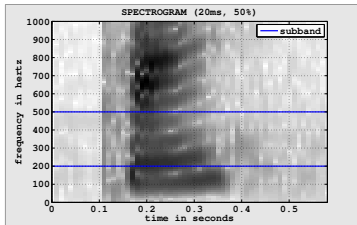
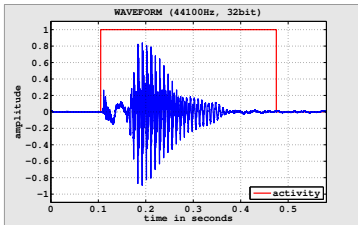


- ▶ example: `matlab/spectgram.m` (`matlab/haiku.wav`)

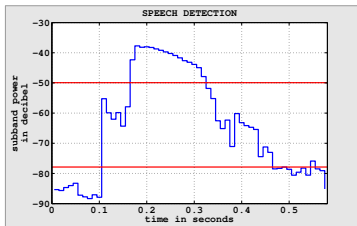
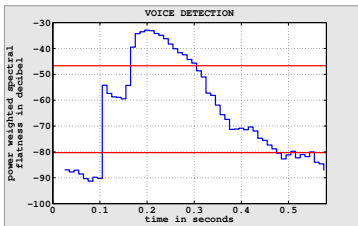


Activity detection

- ▶ **example:** matlab/activity.m (matlab/tam.wav)
- ▶ applying an **equal loudness filter** and limiting to **frequency band 0 ... 1000 Hz**



- ▶ **adaptive thresholds** for power-weighted **spectral flatness** and **subband power**



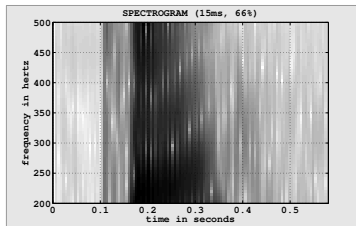
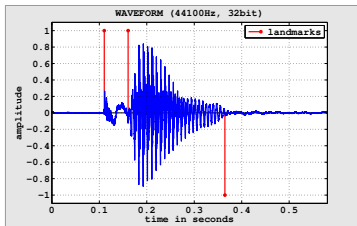
- ▶ combining **activity states** based on thresholds gives overall **voice activity**

- ▶ D. Robinson. *Equal loudness filter*. 2001.
- ▶ D. Burileanu, L. Pascalín, C. Burileanu, M. Puchiu. *An adaptive and fast speech detection algorithm*. Springer, 2000.
- ▶ M. Prcin, L. Müller. *Heuristic and statistical methods for speech/non-speech detector design*. Springer, 2002.
- ▶ Y. Ma, A. Nishihara. *Efficient voice activity detection algorithm using long-term spectral flatness measure*. Springer, 2013.

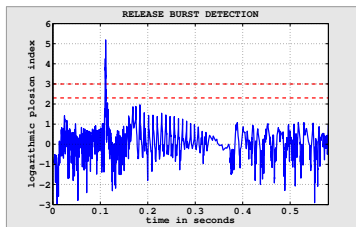
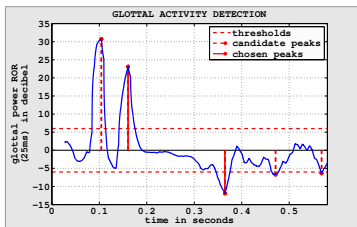
Acoustic signals/Landmarks detection

Landmarks detection

- ▶ **example:** matlab/landmarks.m (matlab/tam.wav)
- ▶ applying an **equal loudness filter** and limiting to **frequency band 200 ... 500 Hz**



- ▶ **power rate-of-rise and sophisticated peak detection, thresholding and pairing**



- ▶ computation of **plosion indices** to find release bursts

- ▶ D. Robinson. *Equal loudness filter*. 2001.
- ▶ P. Mermelstein. *Automatic segmentation of speech into syllabic units*. ASA, 1975.
- ▶ S. Liu. *Landmark detection for distinctive feature-based speech recognition*. MIT, 1995.
- ▶ T. Ananthapadmanabha, A. Prathosh, A. Ramakrishnan. *Detection of the closure-burst transitions of stops and affricates in continuous speech using the plosion index*. ASA, 2014

Acoustic signals/Formants detection

