

HCS 7367  
Speech Perception Lab

Dr. Peter Assmann  
Fall 2011

## Assignment 2

### Assignment 1: Recording and analysis of American English vowels

- Download Hillenbrand vowel database
- <http://homepages.wmich.edu/~hillenbr/voweldata.html>
- Acoustic measurements (text files): *Coarse sampling*

## Assignment 2

- Save the file from your browser; store as **vowdata.txt** in your Matlab directory
- Use the Matlab M-file editor to remove the header information (first 30 lines).
- Load the acoustic measurements into Matlab from the text file:  
>> `help textread`

## Assignment 2

- >> `help textread`
- `TEXTREAD` Read formatted data from text file.
- `A = TEXTREAD('FILENAME')`
- `[A,B,C, ...] = TEXTREAD('FILENAME','FORMAT')`  
reads data from the file `FILENAME` into the variables `A,B,C,etc.`

## Assignment 2

- `TEXTREAD` works by matching and converting groups of characters from the file. An input field is defined as a string of non-whitespace characters extending to the next whitespace or delimiter character or until the field width is exhausted.

## Assignment 2

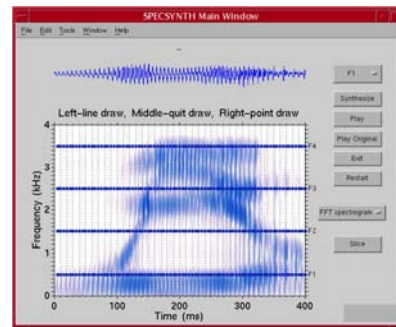
- The `FORMAT` string can contain whitespace characters (which are ignored), ordinary characters (which are expected to match the next non-whitespace character in the input), or conversion specifications.



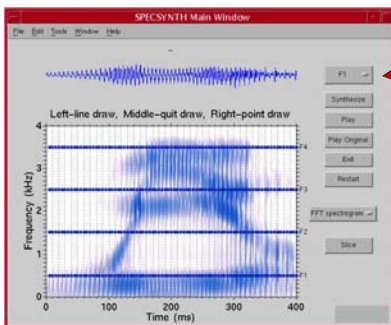
## Using TrackDraw

- » load wheel
- » `y=wheel;`
- `% --- OR ---`
- » `[y,rate]=wavread('wheel.wav');`
- `% --- launch TrackDraw ---`
- » `specsynth;`

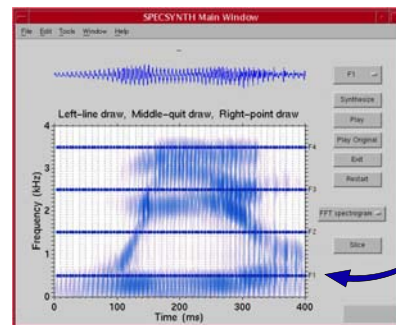
## TrackDraw – main window



## TrackDraw – select F1 track

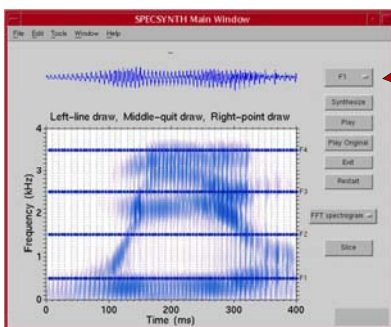


## TrackDraw – re-draw F1 track

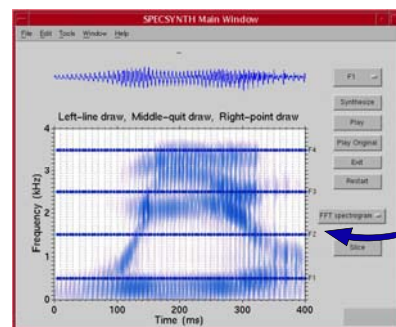


left mouse button: click-and-draw to draw short line segments

## TrackDraw – select F2 track

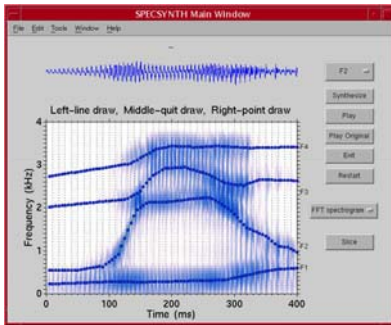


## TrackDraw – modify F2 track

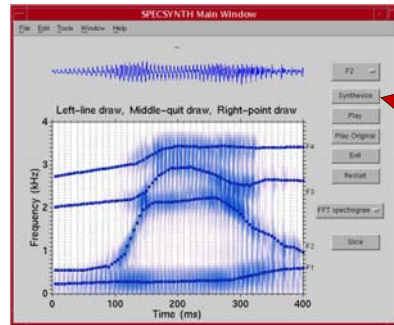


left mouse button: click-and-draw to draw short line segments

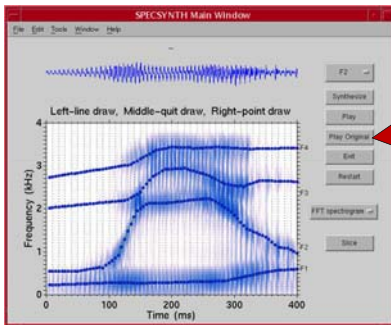
### TrackDraw: finished tracks



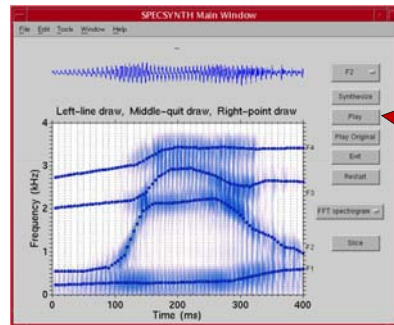
### TrackDraw – synthesize tracks



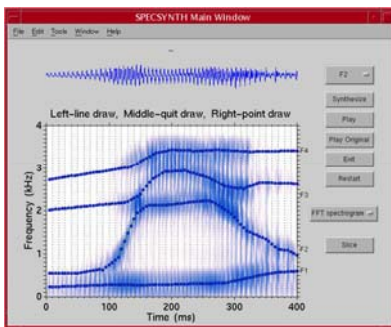
### TrackDraw – Play original



### TrackDraw – Play synthesized

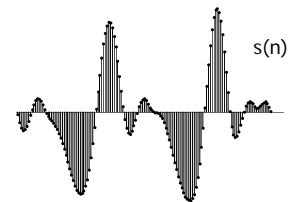


### TrackDraw: repeat steps



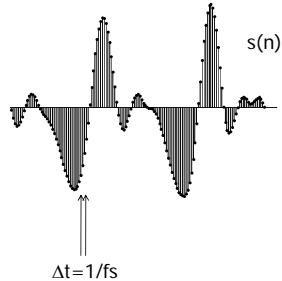
### Discrete-time representation

- D/A conversion: continuous signal is transformed to discrete form by **sampling** (time) and **quantizing** (amplitude)



## Discrete-time representation

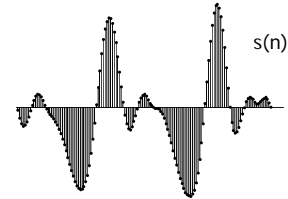
- Only the discrete amplitudes  $s(n)$  are preserved, along with the sampling interval ( $1/f_s$ ).



## Discrete-time representation

- The maximum signal frequency preserved in the discrete-time representation is  $f_s/2$ .

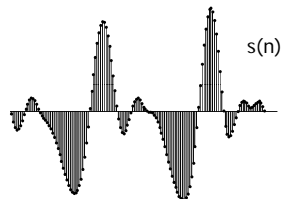
- $f_s=8000$  (telephone)
- $f_s=44100$  (CD quality)



## Discrete-time representation

- Quantization leads to an integer representation with  $2^n$  amplitude levels where  $n$  is the number of bits:

- 8 bits (-128 to 127)
- 16 bits (-32768 to 32767)



## Sliding window analysis

- Speech is a time-varying signal with complex spectral changes
- A sliding time window is applied to capture fluctuations in signal properties
- Different time windows may be needed depending on the aims of the study (e.g. measuring formants or measuring F0)
  - Frame size:** 10-20 ms
  - Frame overlap:** 50% of the frame length

## Window functions

- Rectangular window:

$$w[n] = \begin{cases} 1 & 0 \leq n \leq L-1 \\ 0 & \text{otherwise} \end{cases}$$

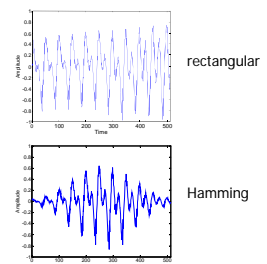
- Hamming window

$$w[n] = \begin{cases} 0.54 - 0.46 \cos\left(\frac{2\pi n}{L-1}\right) & 0 \leq n \leq L-1 \\ 0 & \text{otherwise} \end{cases}$$

## Hamming window

% Waveforms: rectangular & Hamming windowed signals  
% Extract 512 samples from vowel midpoint

```
>> start = length(y) ./ 2;
>> stop = start + 511;
>> y = y ( start:stop );
>> plot ( y );
>> h = hamming(512);
>> plot ( y .* h );
```



## Hamming window

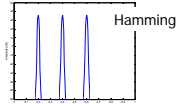
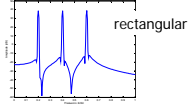
% Spectra: rectangular & Hamming-windowed tones

```
>> y=fgen([200 400 600],[50 50 50],[0 0 0],2048,10000);
```

```
>> fp(y,10000);
```

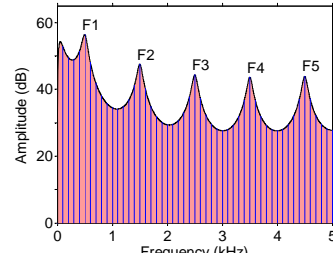
```
>> axis([0 1 -60 50]);
```

```
>> fp(y.*hanning(length(y)),10000);
```



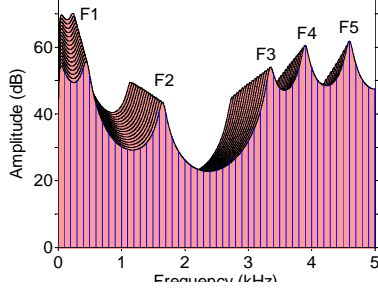
## Formant pattern

- **Formant pattern:** Frequencies of the formants (F1, F2, F3, ...)

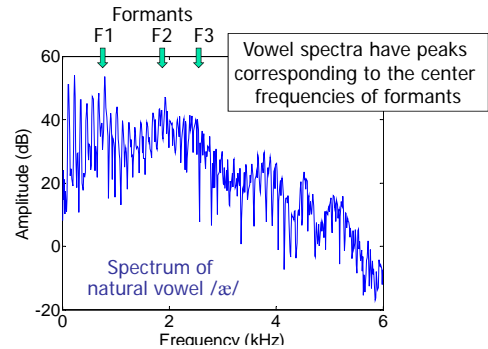


## Formant Dynamics

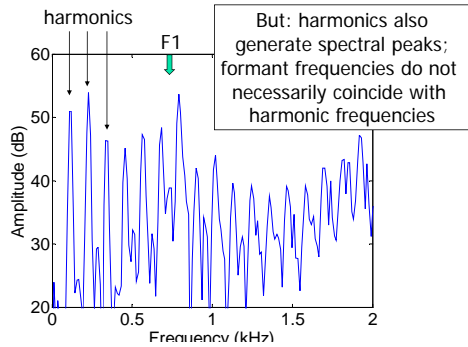
- **Formant tracks:** Formant frequency changes over time:



## Formant Estimation



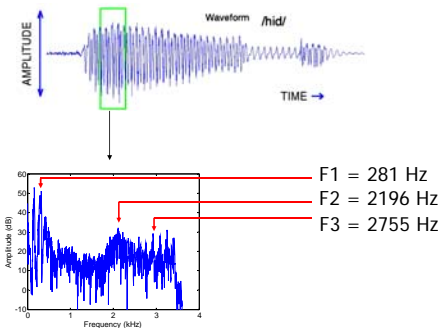
## Formant Estimation



## Spectral representations for estimating formants

- Short-term amplitude spectrum
- Spectrogram
- LPC spectrum
- Cepstral smoothing

## Short-term amplitude spectrum



## Short-term amplitude spectrum

### Analysis window properties

- Length of each time frame (e.g., 20 ms)
- Number of frequency bins (e.g., 512)
- Analysis window type (e.g., Hamming)
- Number of samples per analysis frame (optionally, append zeros)

## Short-term amplitude spectrum

### Analysis strategy

- Choose the analysis window properties so that formant peaks are clearly resolved, but harmonic peaks are "smeared."
- Search for peaks and enter their frequencies in a table.

## Short-term amplitude spectrum

### Problems

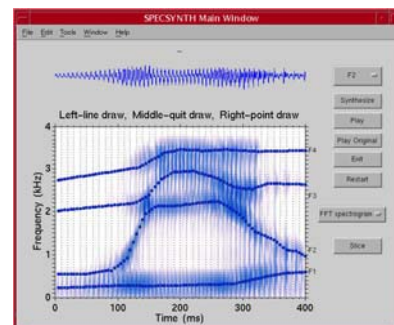
- Time-consuming method, prone to errors
- Difficulty estimating formant frequencies when  $F_0$  is high (as in children's voices)
- Formant frequencies can be very close together (e.g. F2 and F3 in the vowel /a/)
  - "missing formants"
  - "merged formants"

## Short-term amplitude spectrum

### Problems

- Time-frequency tradeoff: Frequency resolution is controlled primarily by the length of the analysis window (improved frequency resolution can be obtained by increasing the length of the analysis window; but a longer window causes rapidly-changing events to be blurred in time).

## Formant tracking from spectrograms



## Formant tracking from spectrograms

- Advantages:
  - Visual continuity helps track formants through noisy regions, and in regions where adjacent formants appear to merge.
  - Track-based synthesis (monitoring by ear as well as by eye) provides independent checks on formant estimates.
  - Iterative process allows for progressive tuning of measurements to the desired level of precision.

## Formant tracking from spectrograms

- Limitations:
  - Mismatches identified by ear are sometimes hard to repair by parameter track manipulations.
  - TrackDraw uses cascade formant synthesis, and its application is currently limited to voiced speech.
  - Formant analysis/synthesis does not capture all of the perceptually relevant information in voiced speech.
  - TrackDraw is a manual procedure rather than automatic, hence its accuracy is user-dependent.

## Linear predictive coding

- **LPC:** Linear Predictive Coding
  - Each sample of the speech signal is modeled as the weighted sum of past samples. The weights (or coefficients) are used to derive a linear prediction filter.

## Linear predictive coding

- **LPC:** Linear Predictive Coding
  - Peaks in the spectrum of the LP filter provide better estimates of formant peaks than the amplitude spectrum of the original signal.

## LPC analysis: preliminary steps

- Preliminary steps
  - (1) The speech signal is low-pass filtered and sampled at  $F_s$  samples/second.
  - (2) The signal is divided into successive overlapping frames (e.g. 20 ms).
  - (3) In each frame, samples are multiplied by a Hamming window, pre-emphasis is applied, and an LPC analysis of order  $p$  is applied.

## LPC analysis

- LPC analysis
  - Input: digital speech samples,  $s(n)$
  - Output: set of  $p$  coefficients  $[a_1, a_2, a_3, \dots, a_p]$
  - The number  $p$  is the *order* of the linear prediction filter, and determines the accuracy of the predictions.

## LPC analysis

- The coefficients  $[a_1, a_2, a_3, \dots, a_p]$  are chosen to best predict the current data sample,  $s(n)$ , from the set of  $p$  previous samples,  $s(n-1), s(n-2), \dots, s(n-p)$ :

$$\tilde{s}(n) = \sum_{k=1}^p a_k s(n-k)$$

## LPC analysis

- The prediction error,  $e(n)$ , is defined as the difference between the actual and predicted signal:

$$e(n) = s(n) - \tilde{s}(n)$$
$$e(n) = s(n) - \sum_{k=1}^p a_k s(n-k)$$

## LPC analysis

- This equation specifies the coefficients of a digital filter whose output is the prediction error:

$$e(n) = s(n) - \sum_{k=1}^p a_k s(n-k)$$

## LPC analysis using Matlab

```
>> help lpc
```

LPC Linear Predictor Coefficients.

A = LPC(X,N) finds the coefficients, A=[ 1 A(2) ... A(N+1) ], of an Nth order forward linear predictor

$X_p(n) = -A(2)*X(n-1) - A(3)*X(n-2) - \dots - A(N+1)*X(n-N)$

such that the sum of the squares of the errors is minimized.

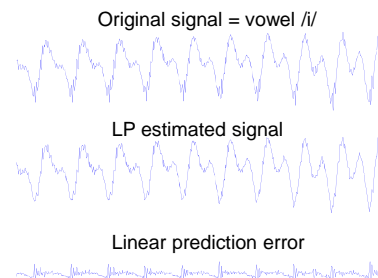
$err(n) = X(n) - X_p(n)$

NOTE: The function *lpc.m* is part of the Signal Processing Toolbox, and is not included with the Student Edition of Matlab.

## LPC analysis using Matlab

```
>> load wheel
>> x = wheel ( 1501:2001 );
>> x = x .* hamming ( 500 );
>> p = 12;
>> a = real ( lpc ( x, p ) );
>> xp = filter( [ 0 - a ( 2 : end ) ], 1, x );
>> err = x - xp;
```

## LPC analysis using Matlab



## LPC analysis

- The **system function** for the LP filter is a complex polynomial of order  $p$ :

$$A(z) = 1 - \sum_{k=1}^p a_k z^{-k}$$

where  $z = r \exp(i 2 \pi f / F_s)$

and  $i = \sqrt{-1}$  (a complex number)

## LPC analysis

- The **frequency response** of a digital filter,  $A(f)$ , or  $A \exp(i 2 \pi f / F_s)$  specifies a set of modifications in amplitude and phase as a function of frequency.

## LPC analysis

- From the source-filter theory of speech production, the frequency response of the LP filter provides an approximation to the **spectrum envelope** of speech:

$$|H(f)| \equiv |1 / A(f)|$$

## LPC analysis

$$|H(f)| \equiv |1 / A(f)|$$

- The peaks (maxima) of  $|H(f)|$  specify the resonant frequencies of the vocal tract, i.e. the center frequencies of the formants.

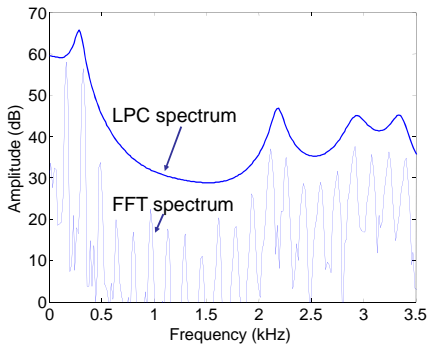
## LPC spectrum using Matlab

- Compute vector of linear prediction coefficients (a)
- Append zeros to the vector:  $a(i) = 0$ ;  
[where  $i = p+1, \dots, \text{nfft}$ ]
- Compute the magnitude of the Fourier Transform
- $|H(f)|$  `>> abs(fft(a, nfft))`
- Convert linear magnitudes to log scale for plotting:  
`>> logmag = 20 * log10(abs(fft(a, nfft)))`;

## LPC spectrum using Matlab

```
>> load wheel
>> x = wheel(1501:2001);
>> x = x .* hamming(500);
>> p = 12;
>> a = lpc(x, p);
>> nfft = 512;
>> logmag = -20 * log10(abs(fft(a, nfft)));
```

## LPC spectrum using Matlab

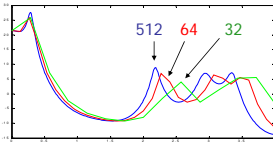


## LPC formant estimation

- The frequencies (and bandwidths) of the formants can be estimated from  $|H(f)|$  in several ways.
  1. Estimate the frequencies of the peaks in  $|H(f)|$ .
  2. Solve for the roots of the predictor polynomial  $H(f)$ .

## LPC peak-picking

- Choose the FFT length (e.g. `nfft=512`) to achieve the desired frequency resolution; append zeros to the LP coefficients as needed.



```
>> logmag = -20 * log10 ( abs( fft ( a, 512 ) ) );
```

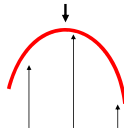
## LPC peak-picking

- Scan for peaks in the LPC spectrum
  - A peak is defined as a spectral sample,  $s_k$  that exceeds the level of both its neighbors on either side by some specified amount:

$$s_{k-1} < [ s_k - 3 \text{ dB} ] > s_{k+1}$$

## LPC peak-picking

- Scan for peaks in the LPC spectrum
  - To save time, use a shorter transform length and interpolate to find the **exact** peak location, e.g. by using a 3-point parabolic fit to the spectral amplitudes that straddle the peak.



## LPC root-solving

- Solving for the roots of the predictor polynomial
  - The roots of  $H(f)$  are also called **poles**. They always occur in pairs; one is the **complex conjugate** of the other (mirror image in the complex plane).

## LPC root-solving

- Solving for the roots of the predictor polynomial
  - A polynomial of order  $p$  has  $p$  roots. Each root is a complex number,  $r_k \exp ( i 2 \pi f_k / F_s )$  which represents a sinusoid of frequency  $f_k$  and damping  $r_k$ .

## LPC root-solving

- Solving for the roots of the predictor polynomial
  - The frequency  $f_k$  and bandwidth  $b_k$  of a root,  $\alpha_k$ , can be computed as:

$$f_k = ( F_s / 2\pi ) \tan^{-1} [ \text{Im}(\alpha_k) / \text{Re}(\alpha_k) ]$$

$$b_k = ( F_s / 2\pi ) \ln ( 1 / r_k )$$

## LPC root-solving

- Solving for the roots of the predictor polynomial
- Not all roots correspond to formants. If  $p$  (the order of the LPC analysis) is set too high, there will be more roots than formants. The bandwidths of roots that do not correspond to formants often have large bandwidths ( $> 300$  Hz).
- If  $p$  is too low, some formants will be missed.

## Choosing LPC order

Rules of thumb: Define the LPC order  $p$  as follows:

$$p = ( 2 * nft ) + 2$$

where  $nft$  is the number of expected formants in the frequency range of  $0 - f_s/2$ .

## Choosing LPC order

- For an adult male with a vocal tract length of 17.5 cm, the formant frequencies of the neutral vowel schwa [ə] are approximately 1000 Hz apart: i.e., 500, 1500, 2500, 3500, 4500, ... Hz
- If the sample rate  $f_s$  is 10 kHz then the number of expected formants in the range  $[ 0 f_s/2 ]$  is 5, so we choose an LPC filter order  $p = 12$ .
- On average, adult females have 12-15% higher formants than adult males, and children's formants are about 20% higher ( $p \approx 10$ ).

## LPC root-solving using Matlab

`>> help roots`

**ROOTS** Find polynomial roots.

ROOTS(C) computes the roots of the polynomial whose coefficients are the elements of the vector C. If C has N+1 components, the polynomial is  $C(1)*X^N + \dots + C(N)*X + C(N+1)$ .

## LPC root-solving using Matlab

```
% compute linear prediction coefficients
>> load wheel
>> x = wheel(1501:2001);
>> x = x .* hamming ( 500 );
>> p = 12;
>> a = lpc ( x, p );
```

## LPC root-solving using Matlab

```
% estimate formant frequencies, fk and
% formant bandwidths, bk, from the roots
% of the predictor polynomial
>> r = roots ( a );
>> fk = rate / ( 2*pi ) * atan2( imag(r), real(r) );
>> bk = rate / ( 2*pi ) * log( 1 ./ abs(r) );
```

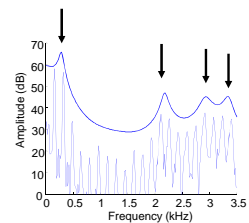
## LPC root-solving using Matlab

```
% eliminate roots with frequencies < 100 Hz
% eliminate roots with bandwidths > 500 Hz
>> indices=find(fk<100);
>> fk(indices)=[];
>> bk(indices)=[];
>> indices=find(bk>500);
>> fk(indices)=[];
>> bk(indices)=[];
```

## LPC root-solving using Matlab

```
% list the frequencies and bandwidths
>> disp( round( [ fk bk ] ) );
```

3327	74
2905	100
2165	54
289	37



## Summary of LPC methods

- Four main sources of error in LPC analysis:
  1. Incorrect choice of filter order,  $p$ 
    - Too high: "spurious" formants
    - Too low: "missed" formants
  2. Errors due to  $F_0$  – formant interactions
    - Formant estimates vary as a function of  $F_0$
  3. Errors due to root solving
    - Poor frequency estimates of low-frequency formants
  4. Errors due to peak picking
    - Formant estimates "skewed" toward adjacent harmonics

## LPC formant estimation

- Reference:  
Vallabha, G.K. & Tuller, B. (2002). Systematic errors in the formant analysis of steady-state vowels. *Speech Communication* 38: 141–160.

## Cepstral analysis

## Cepstral analysis

- Cepstral analysis provides a method for separating the properties of source and filter in speech
- Source – periodic glottal pulses or noise
- Filter – vocal tract resonators

## Cepstral analysis

- The real cepstrum of a signal  $x$  is computed (1) by taking the natural log of the magnitude spectrum of  $x$ ; then (2) taking the inverse Fourier transform of the result.

$$c_x = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log |X(e^{j\omega n})| e^{j\omega n} d\omega$$

## Cepstral analysis

- The real cepstrum is computed in Matlab as follows:

$$y = \text{real}(\text{ifft}(\log(\text{abs}(\text{fft}(x))))));$$

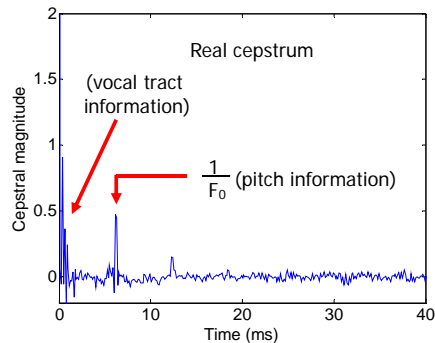
## Cepstral analysis

- The double Fourier transform produces a pseudo-time scale (originally referred to as “quefrency”).
- Low-order cepstral components (<25) represent the slowly-varying components in the spectrum, corresponding to vocal tract information.

## Cepstral analysis

- At longer time intervals, the cepstrum shows a peak at the period of the fundamental,  $F_0$ .
- Strategy: Apply a cepstral window to isolate the vocal tract from the glottal source components.

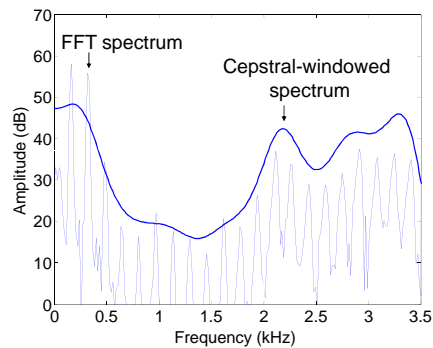
## Cepstral analysis



## Cepstral smoothing

- Cepstral smoothing of the spectrum: apply a rectangular window ( $n = 24$  samples) to the cepstrum, then compute the log magnitude spectrum of the windowed cepstrum.

## Cepstral smoothing



## Cepstral analysis

### Reference:

Oppenheim, A.V., and R.W. Schaffer. 1999. *Discrete-Time Signal Processing*. Upper Saddle River, NJ: Prentice-Hall, pp. 788-789.

## Sine wave synthesis

- Generate a 500 Hz cosine wave  
% 100 ms duration; 60 dB; 10 kHz sample rate)

```
>> dur=100;
>> db=60;
>> amp=10.^(db/20);
>> freq=500;
>> Fs=10000;
>> T=1/Fs;
>> n=dur*Fs/1000;
>> phase = 0;
>> y=amp .* cos(2*pi*T*(1:n)*freq+phase);
```

## Sine wave synthesis

- Generate 5-component complex with equal amplitudes and frequencies of 200, 400, 600, 800 and 1000 Hz; 100 ms duration; 60 dB; 10 kHz sample rate)

```
dur=100;
db=[60 60 60 60 60];
amp=10.^(db/20);
freq=[200 400 600 800 1000];
Fs=10000;
T=1/Fs;
n=dur*Fs/1000;
phase = [0 0 0 0 0];

% introduce a for-loop:
y=zeros(1,n);
nharm=length(freq);
for i=1:nharm
    harmi=amp(i). *cos(2*pi*T*(1:n)*freq(i)+phase(i));
    y=y+harmi;
end;
```

## Group project

- Modify the script to demonstrate that a square wave can be constructed by summing odd harmonics with amplitudes  $1/\text{freq}$ .

```
dur=100;          y=zeros(1,n);
amp=[ ... ];     nharm=length(freq);
freq=[ ... ];    for i=1:nharm
phase = [ ... ];   harmi=amp(i).*cos(2*pi*T*(1:n)*freq(i)+phase(i));
Fs=10000;        y=y+harmi;
T=1/Fs;          end;
n=dur*Fs/1000;
```

## Group project

- Modify the script to synthesize a steady-state approximation to the vowel /i/. Estimate the amplitudes from the amplitude spectrum of a real vowel and set the frequencies to  $[100, 200, \dots, 100*\text{nharm}]$ .

```
dur=100;          y=zeros(1,n);
amp=[ ... ];     nharm=length(freq);
freq=[ ... ];    for i=1:nharm
phase = [ ... ];   harmi=amp(i).*cos(2*pi*T*(1:n)*freq(i)+phase(i));
Fs=10000;        y=y+harmi;
T=1/Fs;          end;
n=dur*Fs/1000;
```