Speech Signal Processing

语音信号处理

School of Electronic Information, Wuhan University

Chapter 3 Speech Analysis

The previous chapter on speech science examined the production and perception of natural speech and described important aspects of speech for human communication. Applications of speech processing (e,g., coding, synthesis, recognition) exploit these properties. To store or recognize speech, reducing redundancy in the speech waveform permits efficient representation of the essential speech aspects in the form of parameters. The relevant parameters for speech recognition must be consistent across speakers, and should yield similar values for the same sounds uttered by various speakers, while exhibiting reliable variation for different sounds. Synthetic speech must replicate perceptually crucial properties of the speech, but may ignore other aspects.

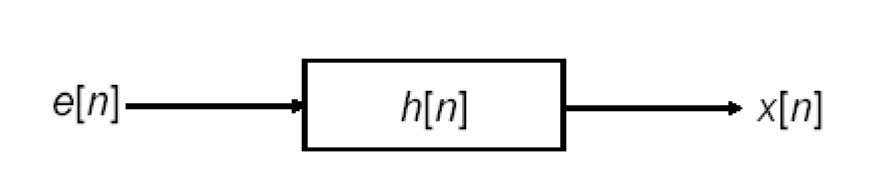
This chapter briefly examines methods of speech analysis, in both the time and frequency domains. There are tradeoffs when analyzing a time-varying signal such as speech; in examining a window of speech, the choice of duration and shape for the window reflects a compromise in time and frequency resolution. Accurate time resolution helps when segmenting speech signals and determining pitch periods, whereas good frequency resolution helps to identify different sounds. Time analysis is rapid but is limited to simple speech measures, e.g., energy and periodicity, while spectral analysis takes more effort but characterizes sounds more precisely.

3.1 Fundamental of Digital Speech signal

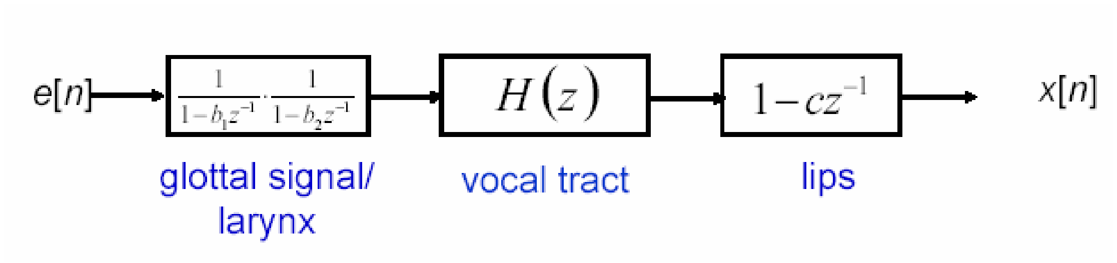
3.1.1 Overview

An underlying model of speech production involving an excitation source and a vocal tract filter is implicit in many analysis methods. Some techniques try to separate these two aspects of a speech signal. The excitation is often represented in terms of the periodicity and amplitude of the signal, while variations in the speech spectrum are assumed to derive from vocal tract variations.

A general model is as followed, where e[n] and x[n] denotes the excitation signal and speech signal respectively.


$$\begin{tikzcd} e[n] \arrow[r] & \boxed{h[n]} \arrow[r] & x[n] \end{tikzcd}$$

As produced in last chapter, the source-filter model, which decomposition of speech signals are described, is shown as followed. Phone classification mostly dependent on the characteristics of the filter. Once the filter has been estimated, the source can be obtained by passing the speech signal through the inverse filter.


$$\begin{tikzcd} e[n] \arrow[r] & \boxed{\frac{1}{1-b_1z^{-1}}\cdot\frac{1}{1-b_2z^{-1}}} \arrow[r] & \boxed{H(z)} \arrow[r] & \boxed{1-cz^{-1}} \arrow[r] & x[n] \end{tikzcd}$$

— **Speech recognizers estimate the filter characteristics and ignore the source**

- **Speech Production Model: Linear Prediction Coding, Cepstral Analysis**
- **Speech Perception Model: Mel-frequency Cepstrum**

— **Speech synthesis techniques use a source-filter model to allow flexibility in altering the pitch and filter**

— **Speech coders use a source-filter model to allow a low bit rate.**

Main Considerations in Feature Extraction:

• Perceptually Meaningful

– Parameters represent salient aspects of the speech signal

– Parameters are analogous to those used by human auditory system (perceptually meaningful)

• Robust Parameters

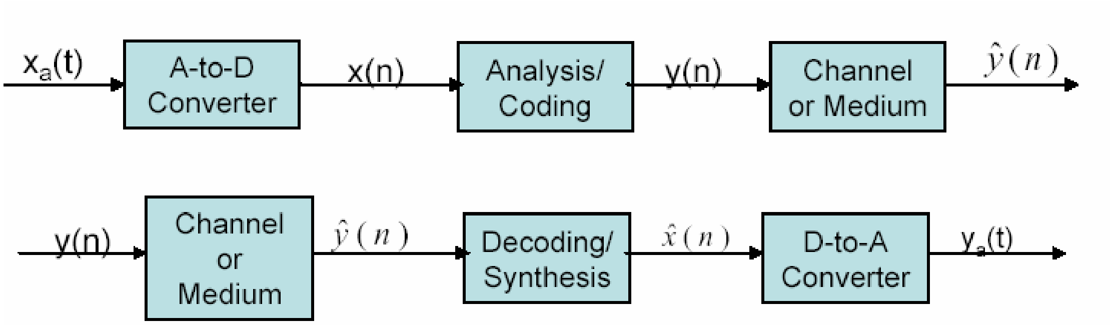
– Parameters are more robust to variations in environments such as the channels, speakers, and transducers

• Time-Dynamic Parameters

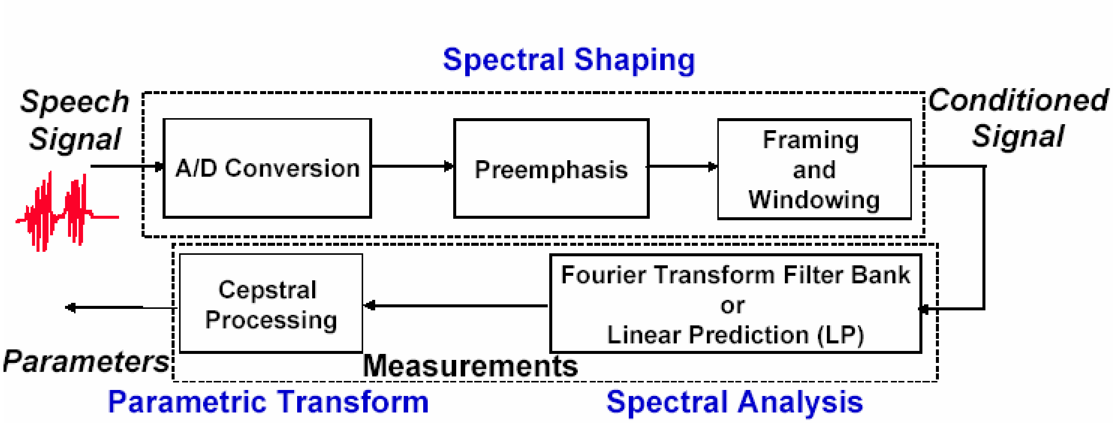
– Parameters can capture spectral dynamics, or changes of the spectrum with time (temporal correlation)

– Contextual information during articulation

A general procedure of speech processing is shown as followed:



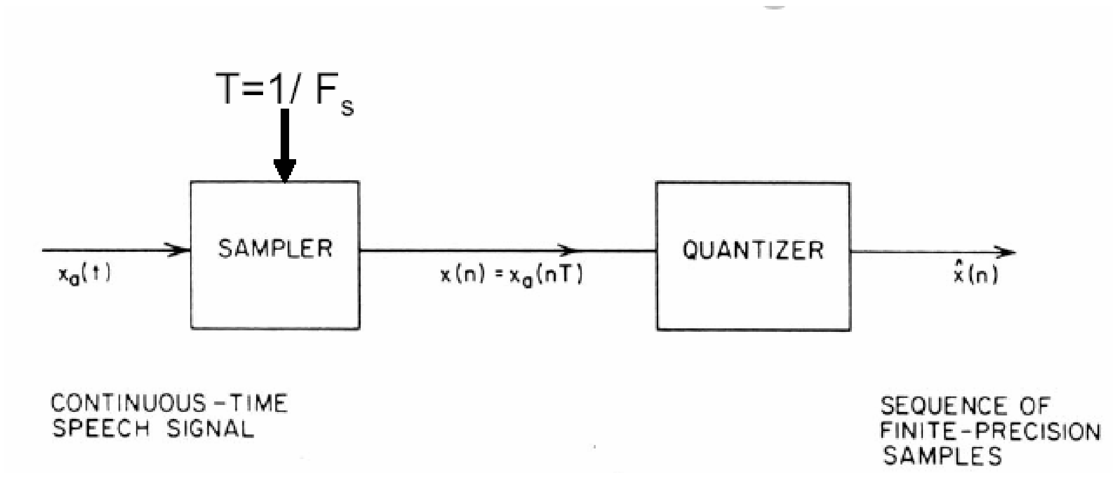
And typical procedures for feature extraction of speech signal are shown below.



3.1.2 Pre-processing

• A/D conversion
– Conversion of the signal from a sound pressure wave to a digital signal
• Digital Filtering (Pre-emphasis) – Emphasizing important frequency components in the signal
• Framing and Windowing – Short-term (short-time) processing

● A/D Conversion



$T=1/F_s$
SAMPLER
QUANTIZER
$x_a(t)$
$x(n) = x_a(nT)$
$\hat{x}(n)$
CONTINUOUS-TIME SPEECH SIGNAL
SEQUENCE OF FINITE-PRECISION SAMPLES

• $F_s = 1/T$ > twice the highest frequency in the input signal

• $X_a(t)$ must first be pre-filtered since speech is not inherently

lowpass. **The** pre-filtering is so called anti-aliasing filtering.

• telephone bandwidth-200-3200 Hz (pre-filtering range)

– F_s = 6400 Hz, 8000 Hz

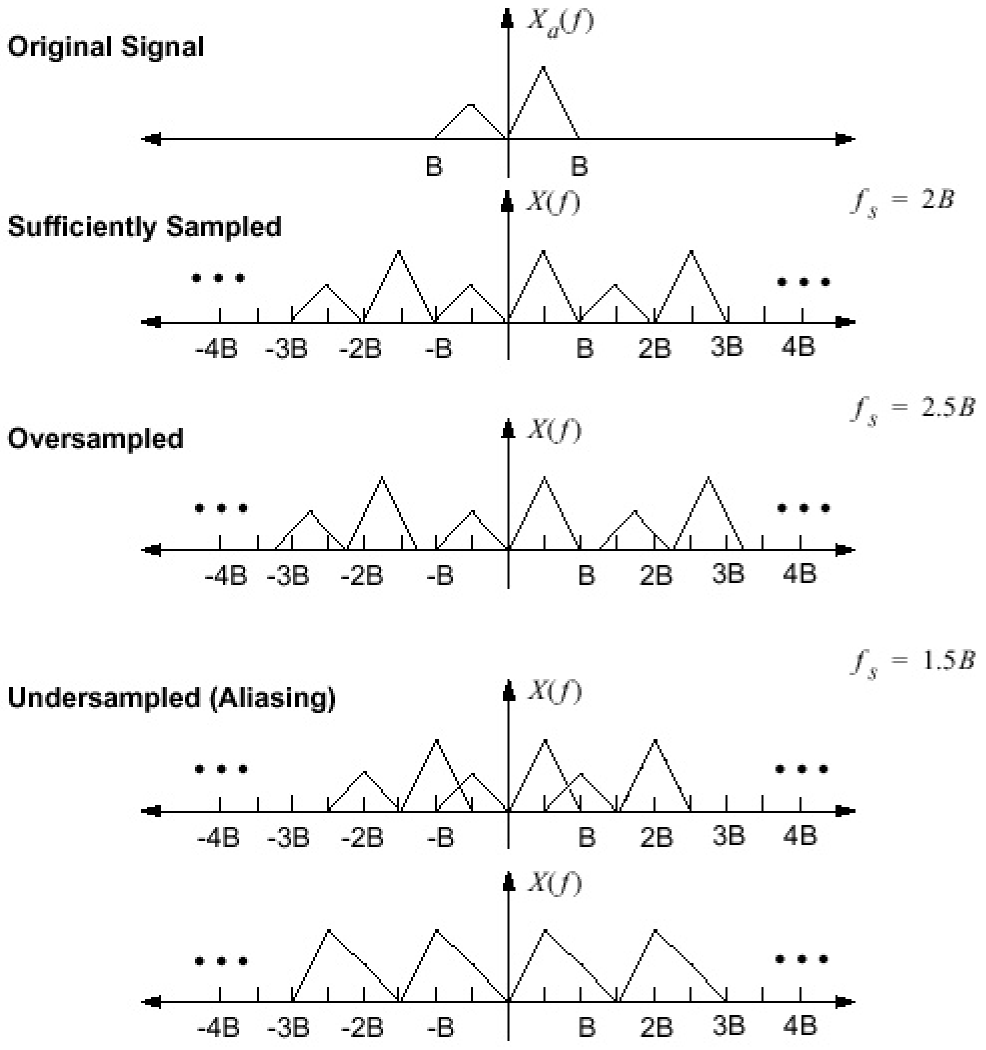
• wideband speech-100-7000 Hz (pre-filtering range)

– F_s = 16000 Hz

• Sampling Theorem: If the highest frequency contained in an analog signal $X_a(t)$ is $F_{max} = B$, and the signal is sampled at a frequency $F_s > 2B$, then the analog signal can be exactly recovered from its samples using the following reconstruction formula:

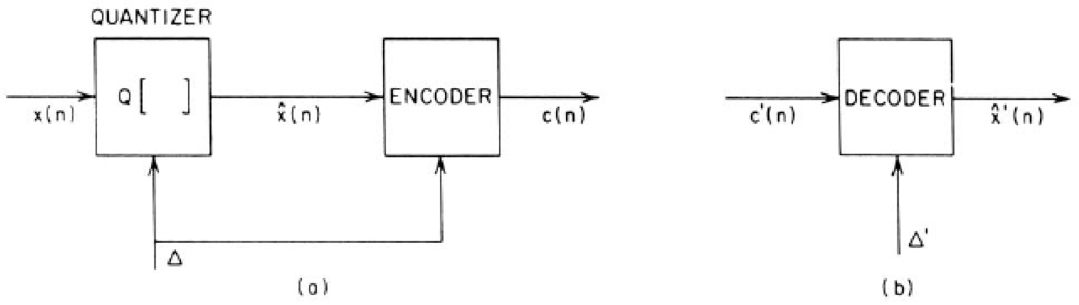
$$x_a(t) = \sum_{n=-\infty}^{\infty} x_a(nT) \frac{\sin((\pi/T)(t-nT))}{(\pi/T)(t-nT)}$$

Note that at the original sample instances (t = nT), the reconstructed analog signal is equal to the value of the original analog signal because the *sinc* functions take on values of zero at multiples of the sample period. At times between the sample instances, the signal is the weighted sum of shifted *sinc* functions.



• Instantaneous Quantization:

x(n) is known to infinite precision in amplitude, and need to be quantized in some suitable manner.



Process of quantization and coding; (a) coder; (b) decoder.

Coding is a two-stage process:

1. quantization process: $x(n) \rightarrow \hat{x}(n)$
2. encoding process: $\hat{x}(n) \rightarrow c(n)$

- where Δ is the (assumed fixed) quantization step size

Decoding is a single-stage process

1. decoding process: $c'(n) \rightarrow \hat{x}'(n)$

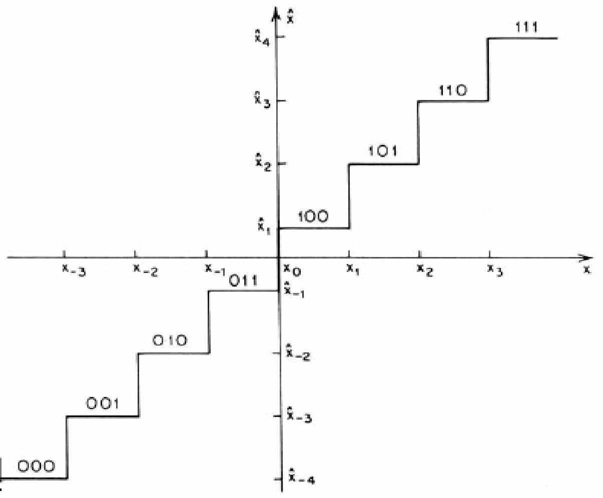
• if $c'(n) = c(n)$, (no errors in transmission) then $\hat{x}'(n) = \hat{x}(n)$

$\hat{x}'(n) \neq x(n) \Rightarrow$ coding and quantization loses information

Use B-bit binary numbers to represent the quantized samples => 2^B quantization levels, for example, 3-bit quantization:

3-bit quantizer => 8 levels

range	level	codeword
$0 = x_0 < x(n) \le x_1$	$\Rightarrow \hat{x}_1$	(100)
$x_1 < x(n) \le x_2$	$\Rightarrow \hat{x}_2$	(101)
$x_2 < x(n) \le x_3$	$\Rightarrow \hat{x}_3$	(110)
$x_3 < x(n) \le \infty$	$\Rightarrow \hat{x}_4$	(111)
$x_{-1} < x(n) \le x_0 = 0$	$\Rightarrow \hat{x}_{-1}$	(011)
$x_{-2} < x(n) \le x_{-1}$	$\Rightarrow \hat{x}_{-2}$	(010)
$x_{-3} < x(n) \le x_{-2}$	$\Rightarrow \hat{x}_{-3}$	(001)
$-\infty < x(n) \le x_{-3}$	$\Rightarrow \hat{x}_{-4}$	(000)



If it is a uniform quantizer:

$$x_i - x_{i-1} = \Delta$$

$$\hat{x}_i - \hat{x}_{i-1} = \Delta$$

$$\Delta = \text{quantizatbn step size}$$

The quantizers characterized by:

– number of levels—2^B (B bits)
– quantization step size— Δ

If we let $\hat{x}(n) = x(n) + e(n)$, with x(n) the unquantized speech sample, and e(n) the quantization error(noise), then $-\frac{\Delta}{2} \le e(n) \le \frac{\Delta}{2}$.

1. quantization noise is a zero-mean, stationary white noise process

$$E[e(n)e(n+m)] = \sigma_e^2, \quad m = 0$$
$$= 0 \quad otherwise$$

2. quantization noise is uncorrelated with the input signal

$$E[x(n)e(n+m)] = 0 \quad \forall m$$

3. distribution of quantization errors is uniform over each quantization interval

$$p_e(e) = 1/\Delta \quad -\Delta/2 \le e \le \Delta/2$$
$$= 0 \quad otherwise$$
$$\Rightarrow \bar{e} = 0, \ \sigma_e^2 = \frac{\Delta^2}{12}$$

For uniform quantizer, we can determine SNR for quantized speech as:

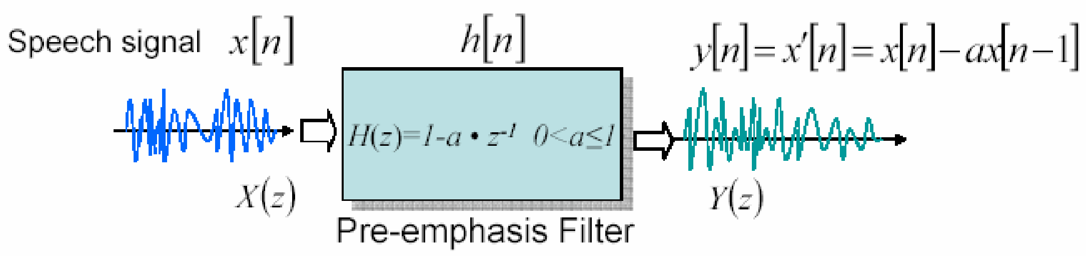
$$SNR(dB) = 10\log_{10}\left[\frac{\sigma_x^2}{\sigma_e^2}\right] = 6B + 4.77 - 20\log_{10}\left[\frac{X_{\max}}{\sigma_x}\right]$$

if we choose $X_{\max} = 4\sigma_x$, then $SNR = 6B - 7.2$

Experimental results shows that the variety range of speech waveform can reach 55dB, so we have B>10bit. In general, we have B=7bit to let SNR=35dB.

● Pre-emphasis

Normally an one coefficient digital filter (a high-pass filter called pre-emphasis filter) is used.


Speech signal $x[n]$
$h[n]$
$y[n] = x'[n] = x[n] - ax[n-1]$
$H(z) = 1 - a \cdot z^{-1} \quad 0 < a \le 1$
$X(z)$
$Y(z)$
Pre-emphasis Filter

$$H(z) = \frac{Y(z)}{X(z)} = 1 - az^{-1}$$

$$\Rightarrow Y(z) = X(z) - az^{-1}X(z)$$

$$\left(\begin{aligned} &\text{Notice that the } Z \text{ transform of } ax[n-1] \\ &= \sum_{n=-\infty}^{n=\infty} ax[n-1]z^{-n} = \sum_{n'=-\infty}^{n'=\infty} ax[n']z^{-(n'+1)} \\ &= az^{-1} \sum_{n'=-\infty}^{n'=\infty} x[n']z^{-n'} = az^{-1}X(z) \end{aligned} \right)$$

$$\Rightarrow y[n] = x[n] - ax[n-1]$$

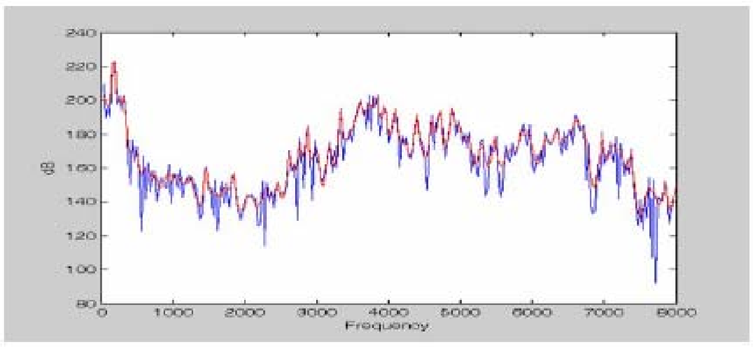
Reason 1: Analysis can be asserted to be seeking the parameters corresponding to the vocal tract only.

Reason 2: Physiological Characteristics

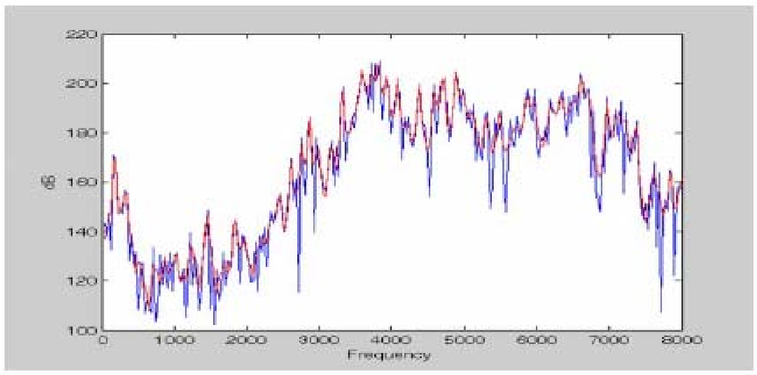
- **Voiced sections of the speech signal naturally have a negative spectral slope (attenuation) of approximately 20 dB per decade due to physiological characteristics of the speech production system**
- **High frequency formants have small amplitude with respect to low frequency formants. A pre-emphasis of high frequencies is therefore require to obtain similar amplitude for all formants**

Reason 3: Prevent Numerical Instability

An example:



No Pre-emphasis

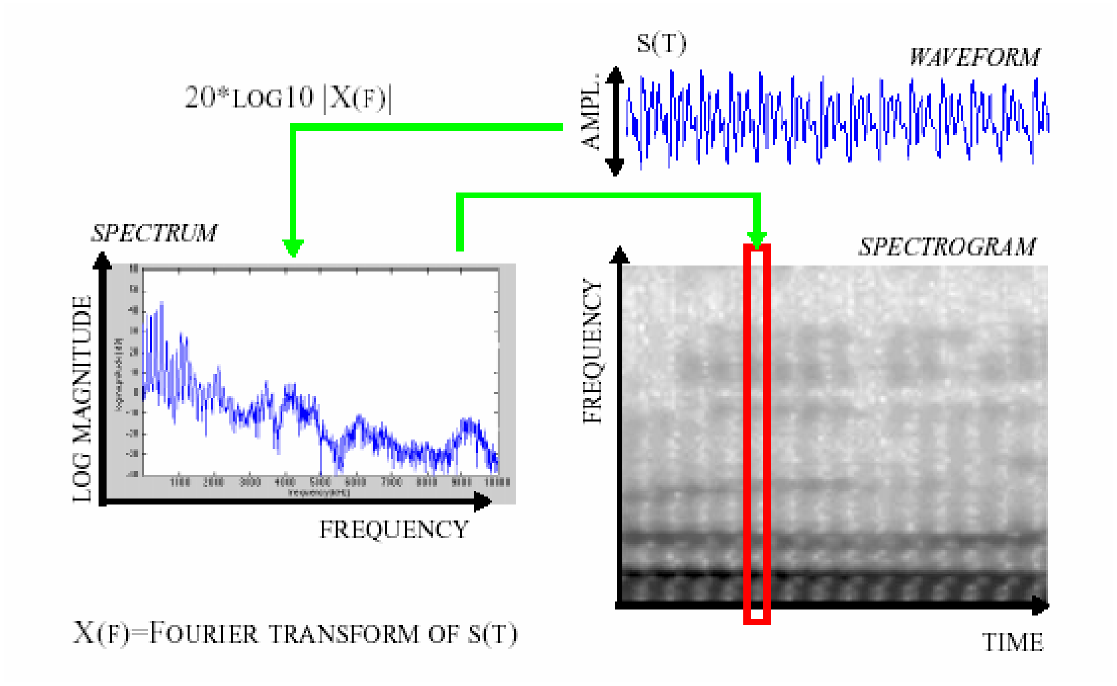


Pre-emphasis

$a_{pre}=0.975$

- **Framing and Windowing – Short-term processing**

A frame-based analysis is essential for speech signals.



• Framing: decompose the speech signal into a series of overlapping frames

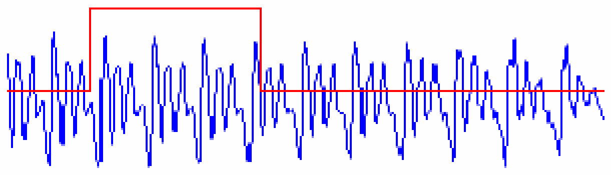
– Traditional methods for spectral evaluation are reliable in the case of <u>a stationary signal</u> (i.e., a signal whose statistical characteristics are invariant with respect to time)

- **Imply that the region is short enough for the behavior of (periodicity or noise-like appearance) the signal to be approximately constant**
- **In sense, the speech region has to be short enough so that it can reasonably be assumed to be stationary**
- **stationary in that region: i.e., the signal characteristics (whether periodicity or noise-like appearance) are uniform in that region. Frame duration ranges are between 10 ~ 25 ms in the case of speech processing.**

• Windowing:

Since speech is non-stationary, we are interesting in short-term estimates of parameters such as the Fourier spectrum. This requires that a speech segment be chosen for analysis. We are effectively cross-multiplying the signal by a window function.

–Rectangular window (w[n]=1 for 0≦n≦N-1):



$$w(n) = \begin{cases} 1, & 0 \le n \le N-1 \\ 0, & otherwise \end{cases}$$

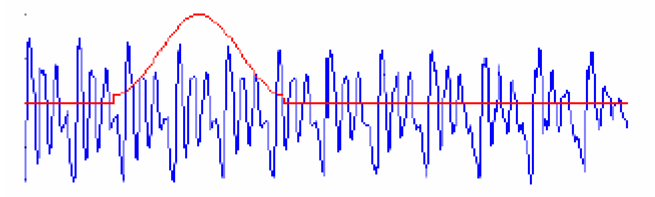
• Just extract the frame part of signal without further processing

• Whose frequency response has high side lobes

–Main lobe: spreads out in a wider frequency range the narrow band power of the signal, and thus reduces the local frequency resolution

–Side lobe: swaps energy from different and distant frequencies of $x_m[n]$, which is called leakage

However, it is desirable to use a tapered window (such as Hamming) instead:



$$w(n) = \begin{cases} 0.54 - 0.46\cos[2\pi n/(N-1)] & 0 \le n \le N-1 \\ 0 & otherwise \end{cases}$$

The reasons for this have to do with the assumption of periodicity made by the DFT, and become clearer in the frequency domain.

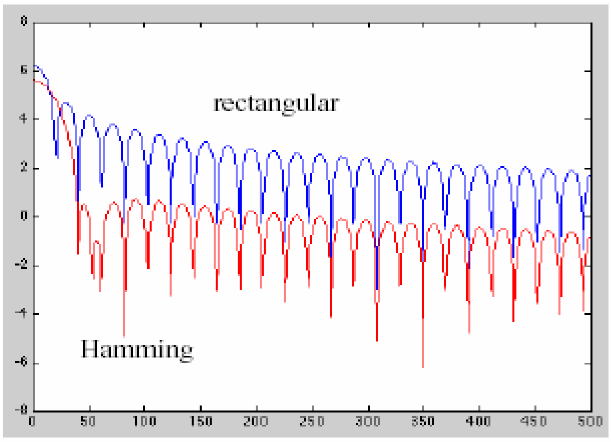
A window (on its own) tends to have an averaging effect. Thus it has a lowpass spectral characteristic. Ideally, we want

• to preserve spectral detail

• to produce little spectral distortion

The log magnitude spectrum of a rectangular window can be compared with that of a Hamming window:

• the Hamming has a wider main lobe, but much better attenuation of sidelobes (typically 20-30 dB better than rectangular).



For a designed window, we wish that :

- A narrow bandwidth main lobe

- Large attenuation in the magnitudes of the sidelobes

However, this is a trade-off!

Notice that:

1. **A narrow main lobe will resolve the sharp details of speech signal (the frequency response of the framed signal) as the convolution proceeds in frequency domain**
2. **The attenuated sidelobes prevents noise from other parts of the spectrum from corrupting the true spectrum at a given frequency**

other windows:

$$w[n] = \begin{cases} (1-\alpha) - \alpha \cos\left(\frac{2\pi n}{N-1}\right), & n = 0,1,\ldots\ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

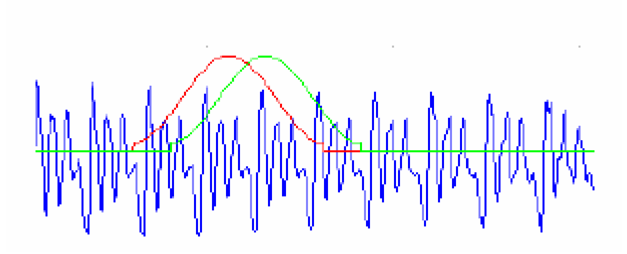
Generalized Hamming Window

Hanning window

$$w(n) = \begin{cases} 0.5 - 0.5\cos[2\pi n/(N-1)] & 0 \le n \le N-1 \\ 0 & \textit{otherwise} \end{cases}$$

• Frame shifting:

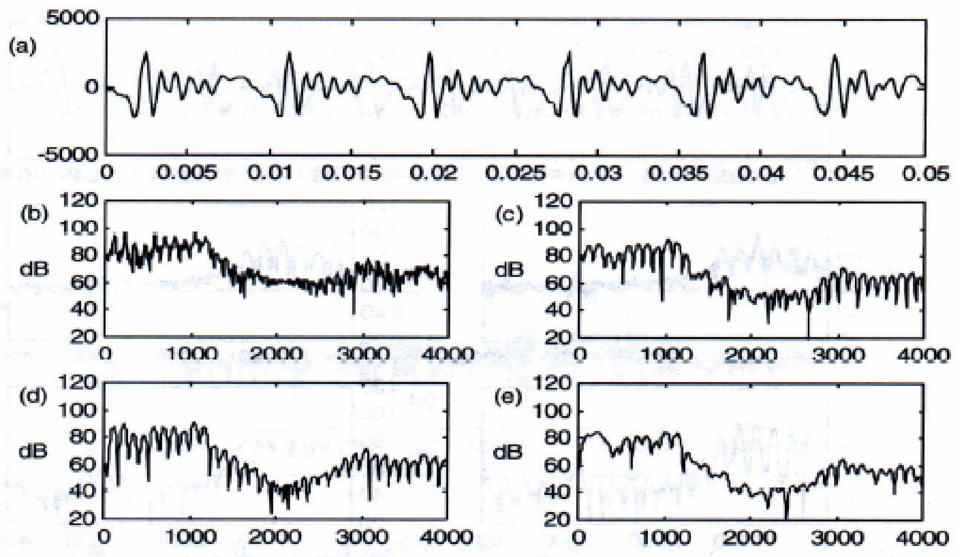
It is normal to use overlapping windows to ensure better temporal continuity in the transform domain. An overlap of half the window size (or less) is typical.



• Frame rate: the number of frames computed per second, in general 33 to 100 frames per second in sort-term speech processing.

- **Examples of short-term analysis of speech signals**

- Male Voiced Speech



As shown in above figure, (a) is the short-term waveform of a male voiced speech (Vowel /a:/) and (b)-(e) are its spectrum. Where (b) 30ms rectangular window and (c) 15ms rectangular window, (d) 30ms Hamming and (e) 15ms Hamming window. The window lobs are not visible in (e) since the window is too short (shorter than 2 times the pitch period). Note the spectral leakage present in (b).
We can read from (a) that there 6 periods in 50ms. So the pitch frequency can be estimated as about 120Hz. And we can read from (d) interestedly that there are 8 peaks within 1000Hz. So the pitch frequency can also be estimated as about 125Hz.
The short-term waveform and its spectrum analysis of female voiced speech are shown as below.

- Female Voiced Speech

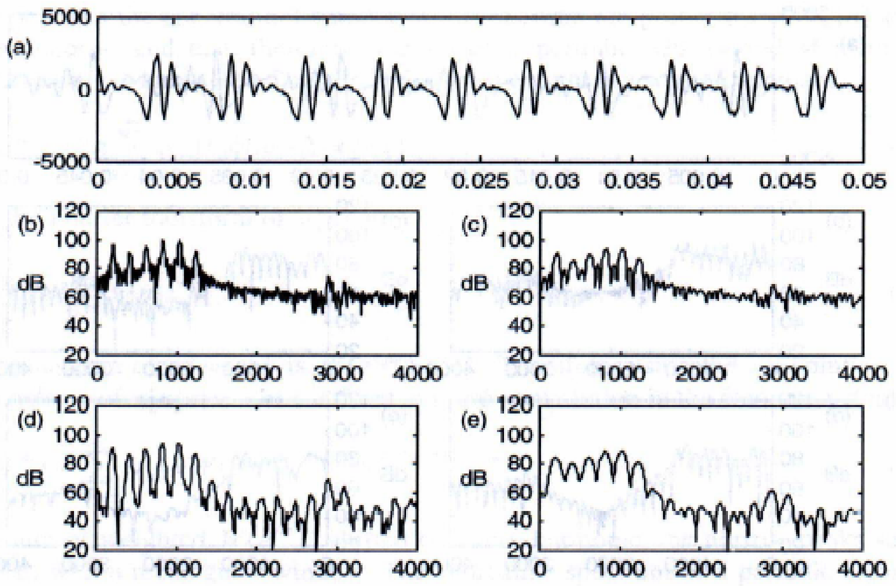


Figure 6.4 Short-time spectrum of female voiced speech (vowel /aa/ with local pitch of 200Hz): (a) time signal, spectra obtained with (b) 30 ms rectangular window and (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window. In all cases the window lobes are visible, since the window is longer than 2 times the pitch period. Note the spectral leakage present in (b) and (c).

And that of an unvoiced speech is shown as below.

- Unvoiced Speech

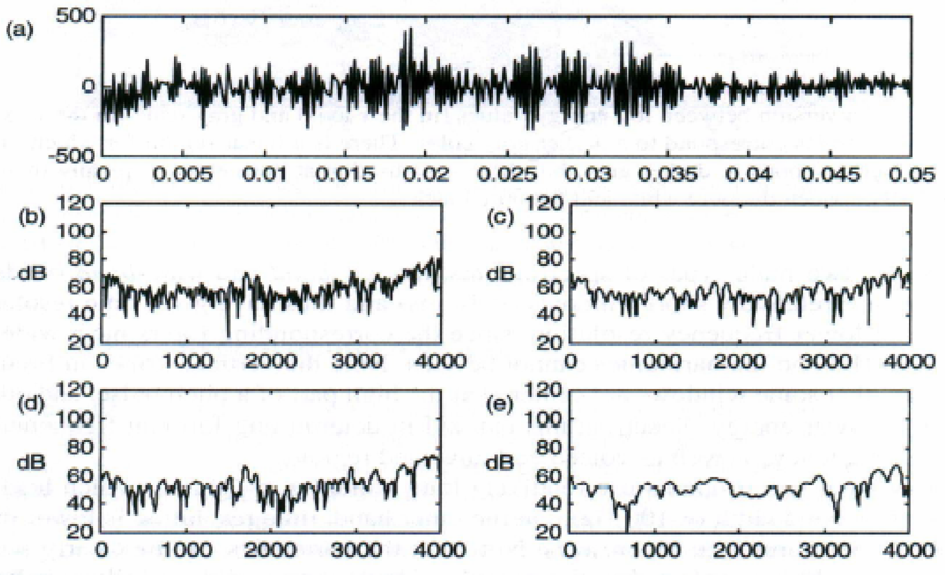
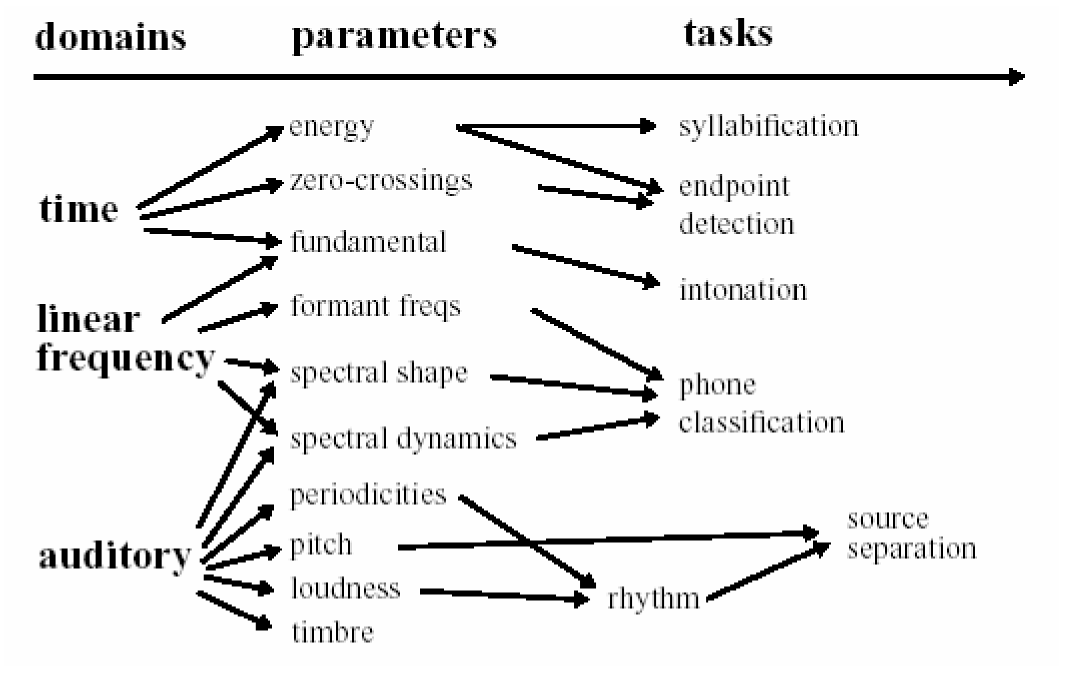


Figure 6.5 Short-time spectrum of unvoiced speech: (a) time signal, (b) 30 ms rectangular window, (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window.

3.2 Time-domain analysis

The main task of speech analysis is to extract features or parameters from speech signals, represented both in time-domain and frequency-domain, as shown below. And we have already shown in last section how to estimate pitch frequency (fundamental frequency or fundamental) roughly from speech waveform or its short-term spectrum. In this section we will focus on feature extraction via time-domain analysis techniques such as Short-term energy, Zero-crossing rate, Autocorrelation.



3.2.1 Short-term Energy

The long term definition of signal energy is as below:

$$E = \sum_{m=-\infty}^{\infty} x^2(m)$$

$$E_n = \sum_{m=n-N+1}^{n} x^2(m) = x^2(n-N+1) + \ldots + x^2(n)$$

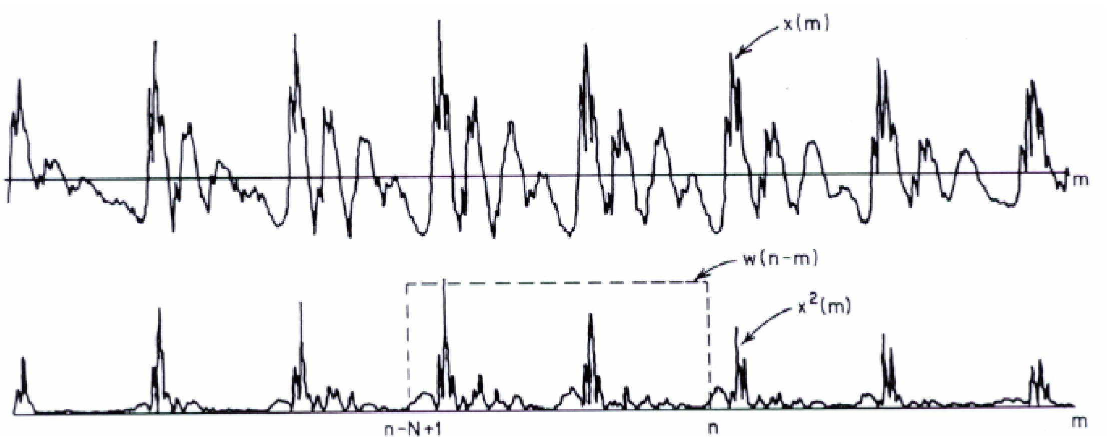
There is little or no utility of this definition for time-varying signals, say speech.
For a short-term speech signal (the n-th frame speech after framing and windowing):

$$x_n(m) = x(m)w(n-m) \qquad n-N+1 \le m \le n$$

where $w(\ )$ is window function and n=0,1T,2T,⋯, N is the window duration, T is the frame-shift.
Its short-term energy E_n is denoted:

$$E_n = \sum_{m=n-N+1}^{n} [x(m)w(n-m)]^2$$

where $w(\ )$is the window, n is the sample that the analysis window is centred on, and N is the window size. Window jumps/slides across sequence of squared values, selecting interval for processing, as shown below.

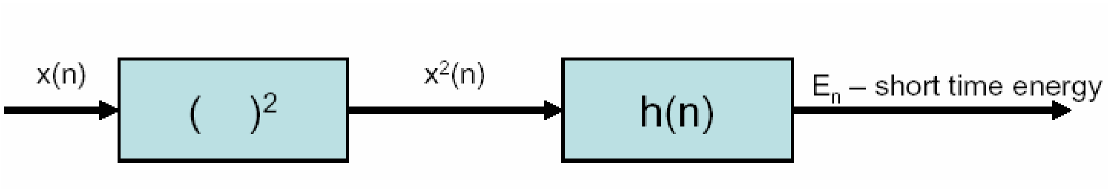


The equation above can be rewritten as:

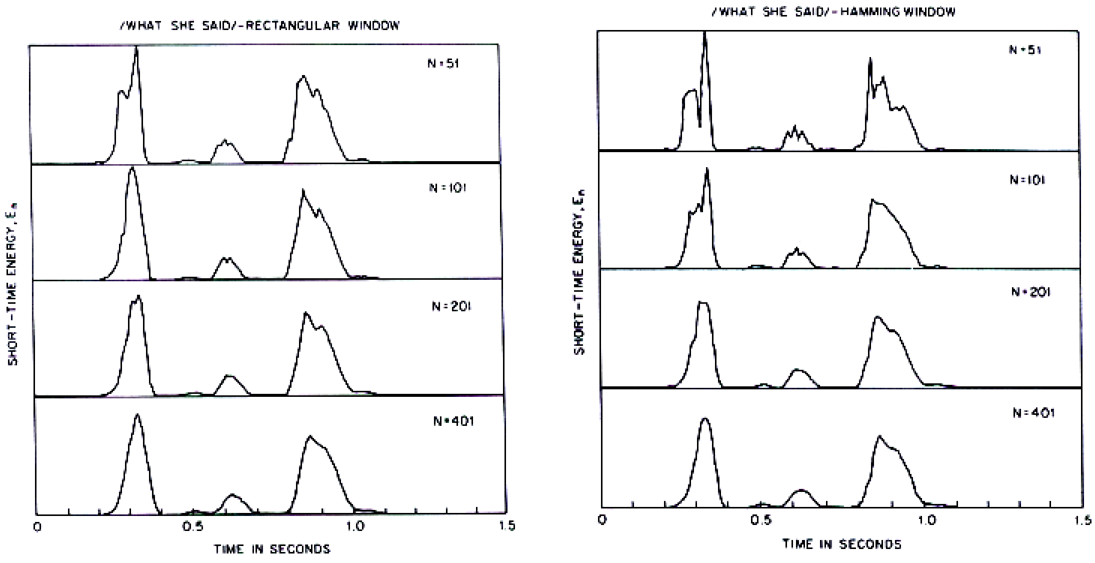
$$E_n = \sum_{m=-\infty}^{\infty} x^2(m)\ w^2(n-m) = \sum_{m=-\infty}^{\infty} x^2(m)\ h(n-m)$$

$$h(n) = w^2(n)$$

and can be sketched as below.



After Rabiner & Schafer, short-term energy of a sentence of “What she said” using a rectangular or a Hamming window are shown below.



It is shown that the short-term energy E_n can be used to

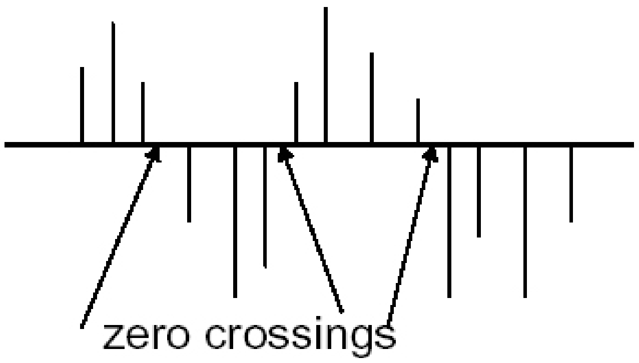
- **distinguish voiced/unvoiced speech: E_n of voiced sound has a much larger value than that of unvoiced sound.**
- **distinguish voiceful/voiceless sound.**
- **etc.**

E_n is a variety measure of squared signal magnitude. In some occasion, we can use another measure of variety in speech magnitude called average short-term magnitude M_n.

$$M_n = \sum_{m=n-N+1}^{n} |x(m)| w(n-m) = |x(n)| * w(n)$$

3.2.2 Zero-crossing rate

Zero crossing => successive samples have different algebraic signs.



- **zero crossing rate is a simple measure of the 'frequency content' of a signal, especially true for narrowband signals (e.g., sinusoids)**
- **sinusoid at frequency F_0 with sampling rate F_S has F_S/F_0 samples per cycle with two zero crossings per cycle, giving an average zero crossing rate of**

$Z=2F_0/F_S$ crossings/sample

Assume the sampling rate is F_s=10,000 Hz, F_0=100Hz sinusoid has F_0/F_S= 10,000 /100 =100 samples/cycle; or Z_1=2 /100 crossings/sample, or Z_{100}=2 /100*100= 2 crossings/10 msec interval.

How about F_0=1000Hz or 5000Hz?

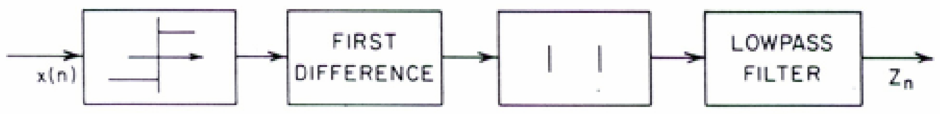
- **Definition of Zero-crossing Rate (ZC)**

$$Z_n = \sum_{m=-\infty}^{\infty} \left| \operatorname{sgn}[x(m)] - \operatorname{sgn}[x(m-1)] \right| w(n-m)$$

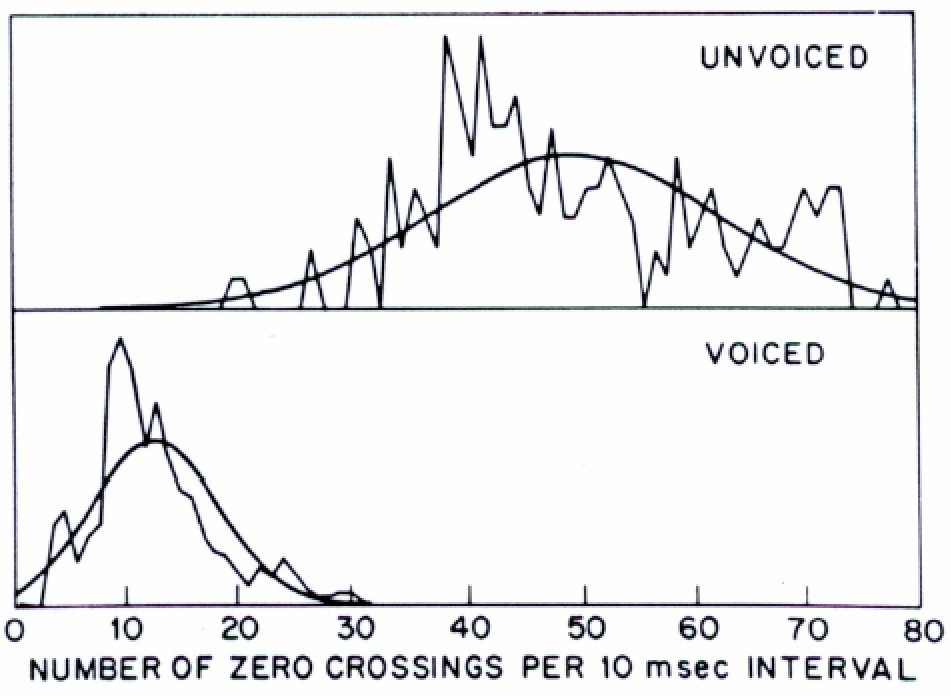
$$\begin{aligned} \operatorname{sgn}[x(n)] &= 1 \qquad x(n) \geq 0 \\ &= -1 \qquad x(n) < 0 \end{aligned}$$

- simple window,rectangular with

$$\begin{aligned} w(n) &= 1/(2N) \qquad 0 \leq n \leq N-1 \\ &= 0 \qquad \text{otherwise} \end{aligned}$$



In fact, it is the number of algebraic sign changing within the frame.



Distribution of zero-crossings for unvoiced and voiced speech.

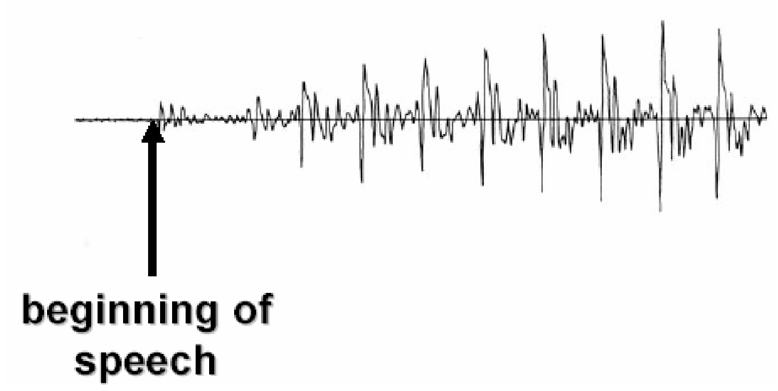
• for voiced speech, energy is mainly below 3 kHz
• for unvoiced speech, energy is mainly above 3 kHz
• mean ZC rate for unvoiced speech is 49 per 10 msec interval
• mean ZC rate for voiced speech is 14 per 10 msec interval

We can draw some feature from the features of M_n (E_n) or ZC, shown as table below. ZC is especially helpful for detecting speech from noisy background or detection of beginning-point or ending-point of speech.

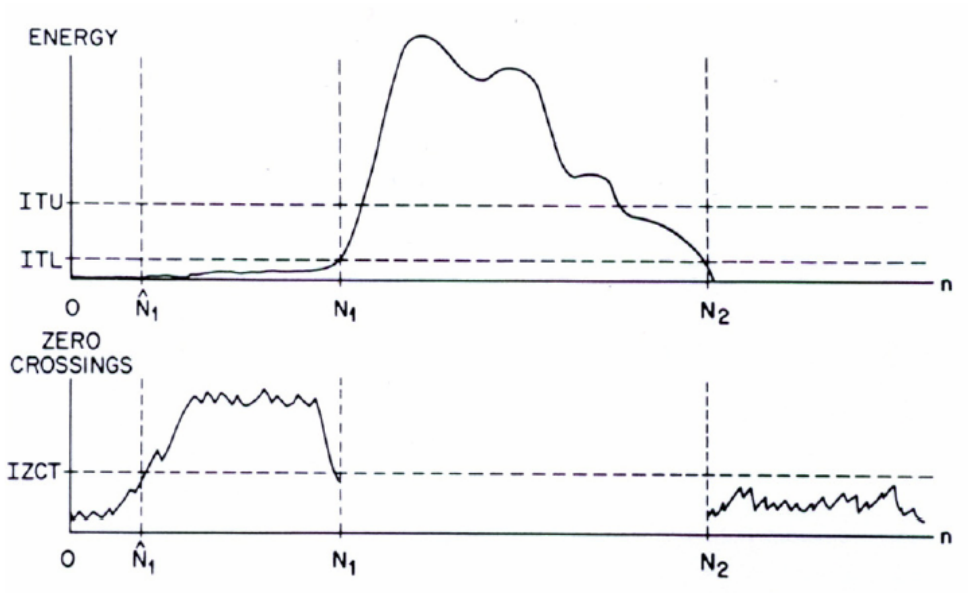
	Voiced	Unvoiced	Silience
M_n	High	Middle	low
Z_n	low	High	Middle

3.2.3 Endpoint Detection-Speech versus Silence

Key problem in speech processing is locating accurately the beginning and end of a speech utterance in noise/background signal.



1. Detect beginning and ending of speech intervals using short-time energy and short-time zero crossings
2. Find major concentration of signal (guaranteed to be speech) using region of signal energy around maximum value of short-time energy
3. Refine region of concentration of speech using reasonably tight short-time energy thresholds that separate speech from backgrounds — but may fail to find weak fricatives, low level nasals, etc
4. Refine endpoint estimates using zero crossing information outside intervals identified from energy concentrations-based on zero crossing rates commensurate with unvoiced speech



<u>Algorithm</u> **for endpoint detection:**

1. Compute mean and compute mean and σ of E_n and Z_n for first 100 msec of signal (assuming no speech in this interval).
2. Determine maximum value of E_n for entire recording.
3. Compute E_n thresholds based on results of steps 1 and 2—e.g., take some percentage of the peaks over the entire interval. Use threshold for zero crossings based on ZC distribution for unvoiced speech.
4. Find an interval of E_n that exceeds a high threshold ITU.
5. Find a putative starting point (N_1) where E_n crosses ITL from below; Find a putative ending point (N_2) where E_n crosses ITL from above.
6. Move backwards from N_1 by comparing Z_n to IZCT, and find the first point where Z_n exceeds IZCT; similarly move forward from N_2 by comparing Z_n to IZCT and finding last point where Z_n exceeds IZCT.

3.2.4 Short-term Autocorrelation

The Fourier transform of a speech signal provides both spectral magnitude and phase. Many applications ignore the phase since it is relatively unimportant perceptually. The time signal corresponding to the inverse Fourier transform of the energy spectrum (the square of the spectral magnitude) is called the *<u>autocorrelation</u>* of the original signal. It preserves information about a signal's harmonic and formant amplitudes as well as its periodicity, while ignoring phase. It has found application in pitch detection, voiced/unvoiced determination, and linear prediction.
The autocorrelation function of a signal s(n) is

$$R(k) = \sum_m s(m)s(m-k)$$

which measures the similarity between s(n) and a delayed version of

itself as a function of the time delay. The autocorrelation is large if at some delay the two signals have similar waveforms. The range of summation is usually limited (i.e., windowed), and the function can be normalized by dividing by the number of summed samples.

$$R(k) = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{m=-N}^{N} s(m)s(m+k)$$

A reasonable definition of short-term autocorrelation is:

$$R_n(k) = \sum_{m=-\infty}^{\infty} x(m)w(n-m)\, x(m+k)w(n-k-m)$$

Key points:

- **Select a segment of speech by windowing;**
- **Computer the deterministic autocorrelation of windowed speech;**
- **Feature of autocorrelation function:**
  1. **For a periodic input (with period *N_p*), its autocorrelation function is also periodic with the same period *N_p*, i.e.** $R(k) = R(k+N_p)$**;**
  2. **It is symmetrical, i.e.** $R(k) = R(-k)$
  3. $R(0) \geq |R(k)|$

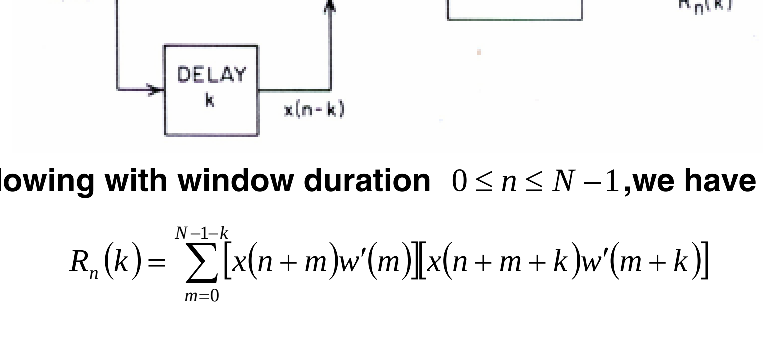
It can be shown that

$$R_n(k) = R_n(-k) = \sum_{m=-\infty}^{\infty} x(m)x(m-k)[w(n-m)w(n-m+k)]$$

define the filter of the form $h_k(n) = w(n)w(n+k)$

This enable us to rewrite the short-term autocorrelation in the form:

$$R_n(k) = \sum_{m=-\infty}^{\infty} [x(m)x(m-k)]h_k(n-m) = [x(n)x(n-k)] * h_k(n)$$



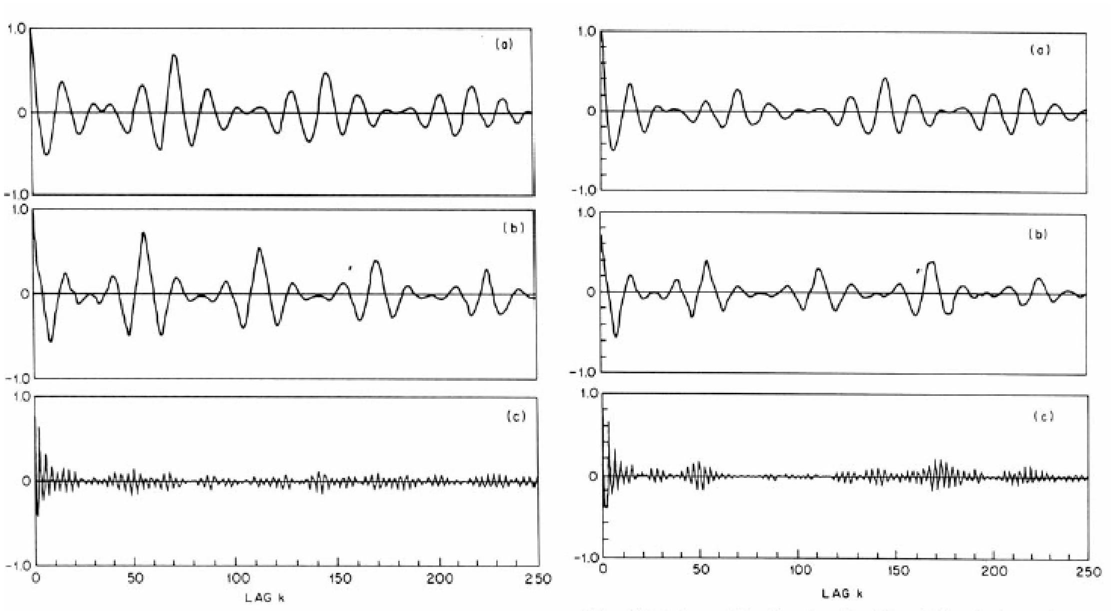
When windowing with window duration $0 \leq n \leq N-1$**,we have**

$$R_n(k) = \sum_{m=0}^{N-1-k} [x(n+m)w'(m)][x(n+m+k)w'(m+k)]$$

where $w'(n) = w(-n)$.

For rectangular window, we can calculate the autocorrelation by:

```
for(k=0; k<M; k++) {
  r[k] = 0.0;
  for(n=0; n<N-k; n++) {
    r[k] += s[n] * s[n+k]
  }
}
```



The left side of above figure shows autocorrelation function for 2 voiced speech [(a) and (b)], and (c) for unvoiced speech, using rectangular window with N=401 (40 msec of speech signal with sampling frequency of 10kHz). The autocorrelation function for the same segments of speech by using Hamming window with N=401 are shown in right side of above figure.

We can observe form the figure that:

• no strong peak for unvoiced speech

• autocorrelation peaks occur at k=72, 144, ... => 140 Hz pitch (See (a), left top of above figure)

• autocorrelation peaks occur at k=58, 116, ... => 180 Hz pitch (See (b), left middle of above figure)

－regularity of peaks for the voiced speech (vowel), shape for the low order lags indicates the pitch period.

• exponentially-decaying shape

• for autocorrelation based pitch detection, rectangular windowing is better than that of Hamming window

3.2.5 Autocorrelation Pitch Detection

• The autocorrelation representation of speech is just too rich

– it contains information that enables you to estimate the vocal tract transfer function (from the first 10 or so values)

– many peaks in autocorrelation in addition to pitch periodicity peaks

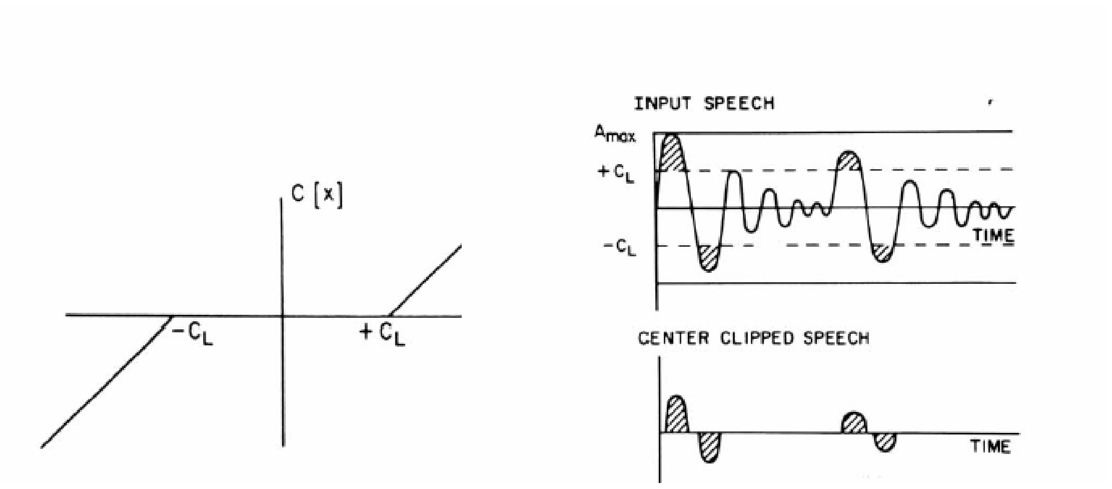
– some peaks due to rapidly changing formants

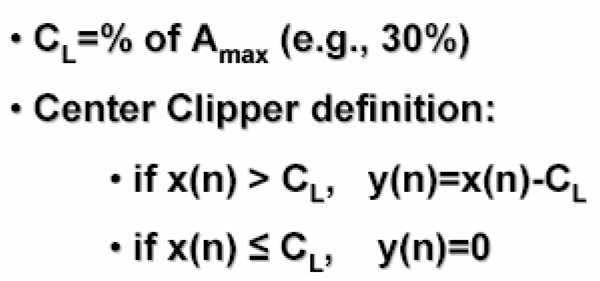
– some peaks due to window size interactions with the speech signal

• need some type of spectrum flattening so that the speech signal more closely approximates a periodic impulse train => center clipping spectrum flattener

● Center Clipping

$$y(n) = C[x(n)]$$



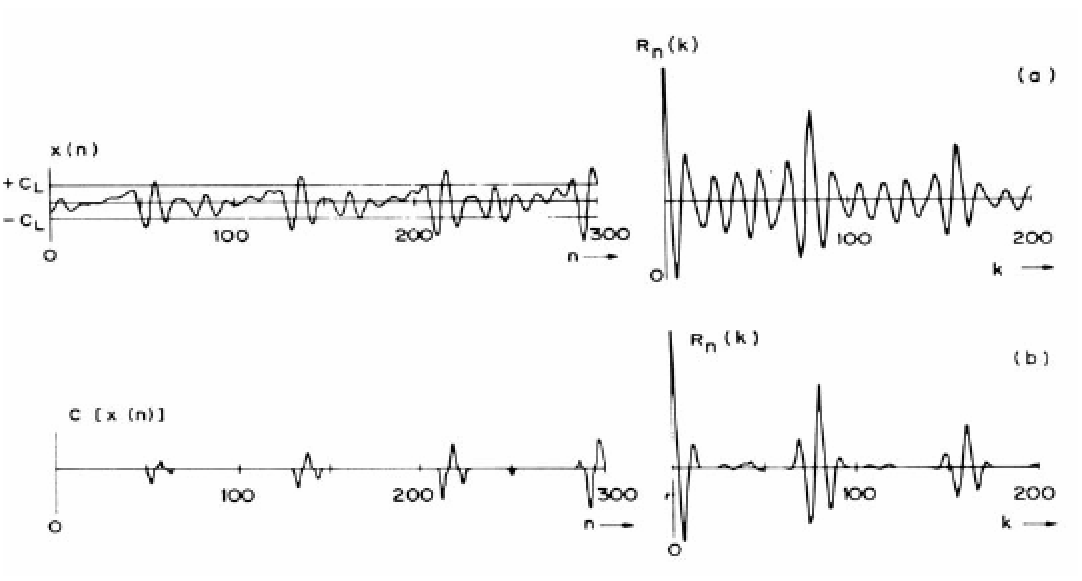

• C_L=% of A_{max} (e.g., 30%)

• Center Clipper definition:

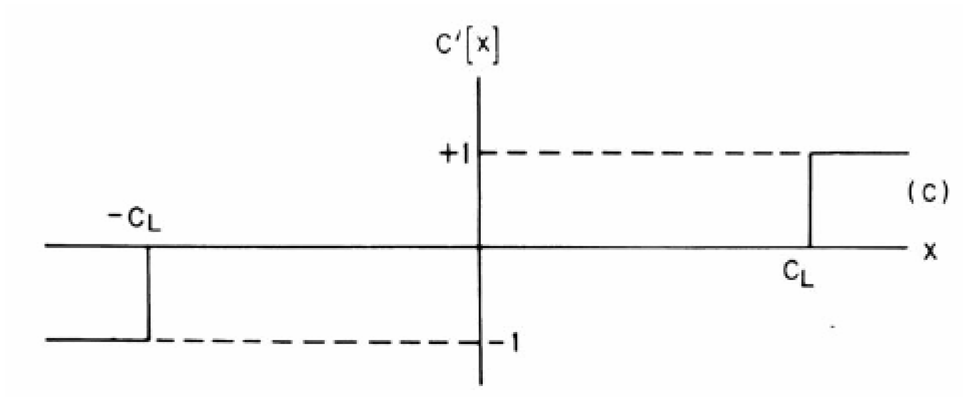
  • if $x(n) > C_L$, $y(n)=x(n)-C_L$

  • if $x(n) \le C_L$, $y(n)=0$

The example of pitch detection using center clipper is shown as followed.

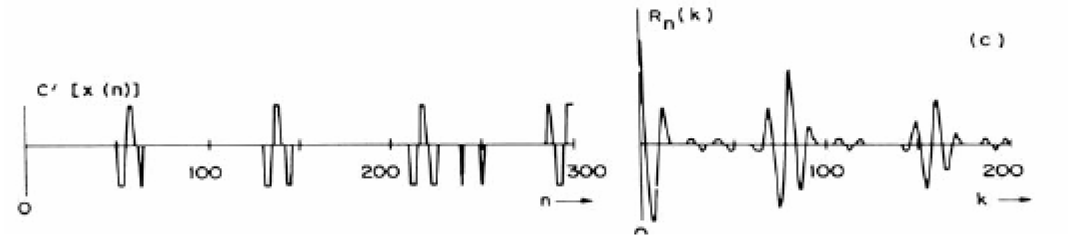


- **3-Level Center Clipper**



$$y(n) = +1 \quad \text{if } x(n) > C_L$$
$$= -1 \quad \text{if } x(n) < -C_L$$
$$= 0 \quad \text{otherwise}$$

• significantly simplified computation (no multiplications)
• autocorrelation function is very similar to that from a conventional center clipper => most of the extraneous peaks are eliminated and a clear indication of periodicity is retained



3.2.6 Non-Linear Smoothing

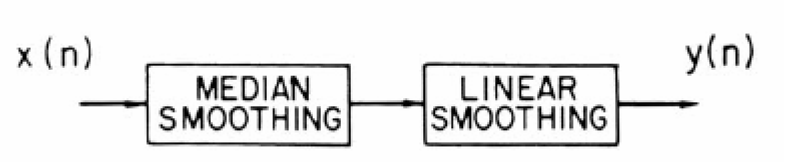
- **linear smoothers (filters) are not always appropriate for smoothing parameter estimates because of smearing and blurring discontinuities**
- **pitch period smoothing would emphasize errors and distort the contour**
- **use combination of non-linear smoother of running medians and linear smoothing**
- **linear smoothing => separation of signals based on non-overlapping frequency content**

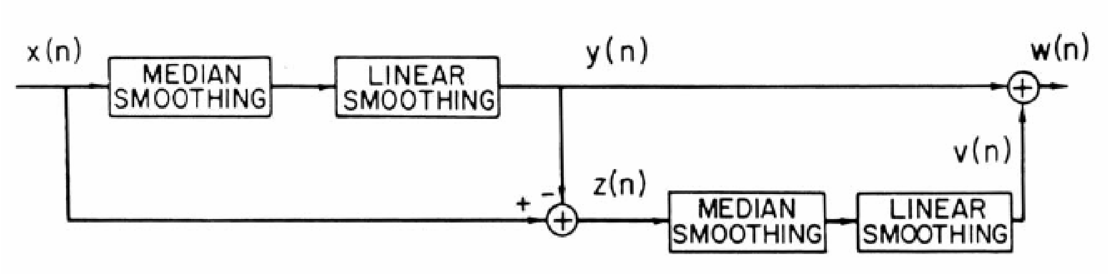
eg. Hanning filter: $h(n) = \begin{cases} 1/4 & n=0 \\ 1/2 & n=1 \\ 1/4 & n=2 \end{cases}$

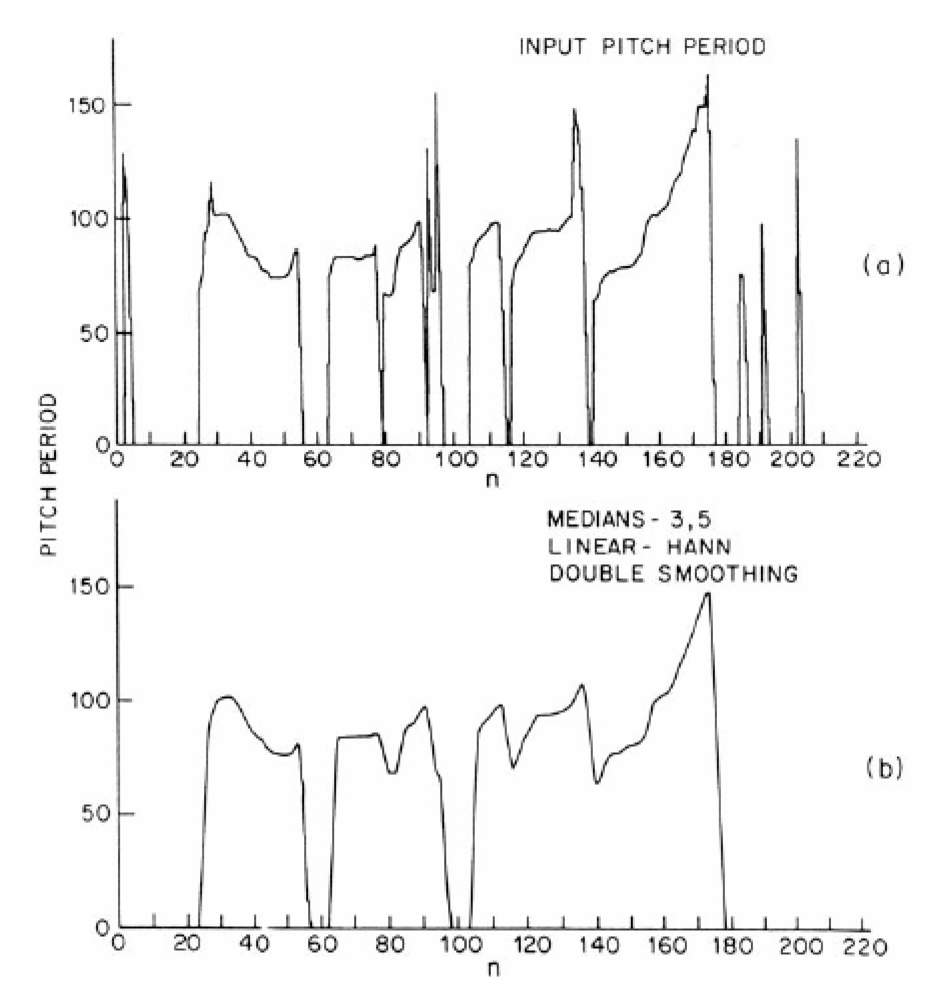
- **non-linear smoothing => separating signals based on their character (smooth or noise-like)**

$$x(n) = S[x(n)] + R[x(n)] \text{ - smooth + rough components}$$
$$y[x(n)] = \text{median}[x(n)] = M_L[x(n)]$$
$$M_L[x(n)] = \text{median of } x(n) \ldots x(n-L+1)$$

Median smoothers generally preserve sharp discontinuities in signal, but fail to adequately smooth noise-like components.

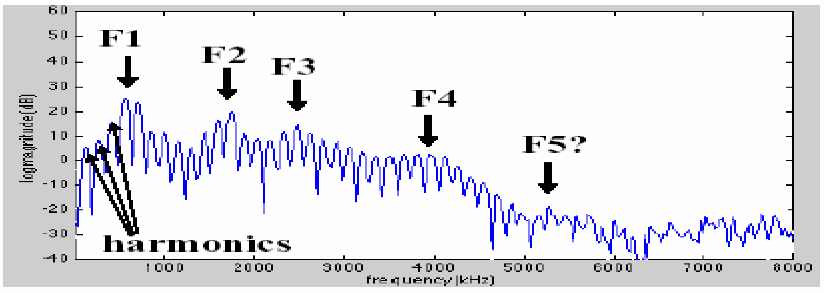






3.3 Frequency-domain Analysis

Represent signal by sum of sinusoids or complex exponentials as it leads to convenient solutions to problems (formant estimation, pitch period estimation, analysis-by-synthesis methods), and insight into the signal itself.
Such Fourier representations provide
– convenient means to determine response to a sum of sinusoids for linear systems
– clear evidence of signal properties that are obscured in the original signal



3.3.1 Short-time Fourier Transform

- **Discrete-time Fourier Transform (DTFT)**

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n} = DTFT\{x(n)\}$$

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

where ω **is the frequency variable of** $X(e^{j\omega})$

- Sufficient condition for convergence: $\sum_{n=-\infty}^{+\infty} |x[n]| < +\infty$
- Although $x[n]$ is discrete, $X(e^{j\omega})$ is continuous and periodic with period 2π.
- Convolution/multiplication duality:

$$\begin{cases} y[n] &= x[n] * h[n] \\ Y(e^{j\omega}) &= X(e^{j\omega}) H(e^{j\omega}) \end{cases}$$

$$\begin{cases} y[n] &= x[n] w[n] \\ Y(e^{j\omega}) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} W(e^{j\theta}) X(e^{j(\omega-\theta)}) d\theta \end{cases}$$

- **Definition of Short-time Fourier Transform (STFT)**

Discrete-time STFT at time n is given by

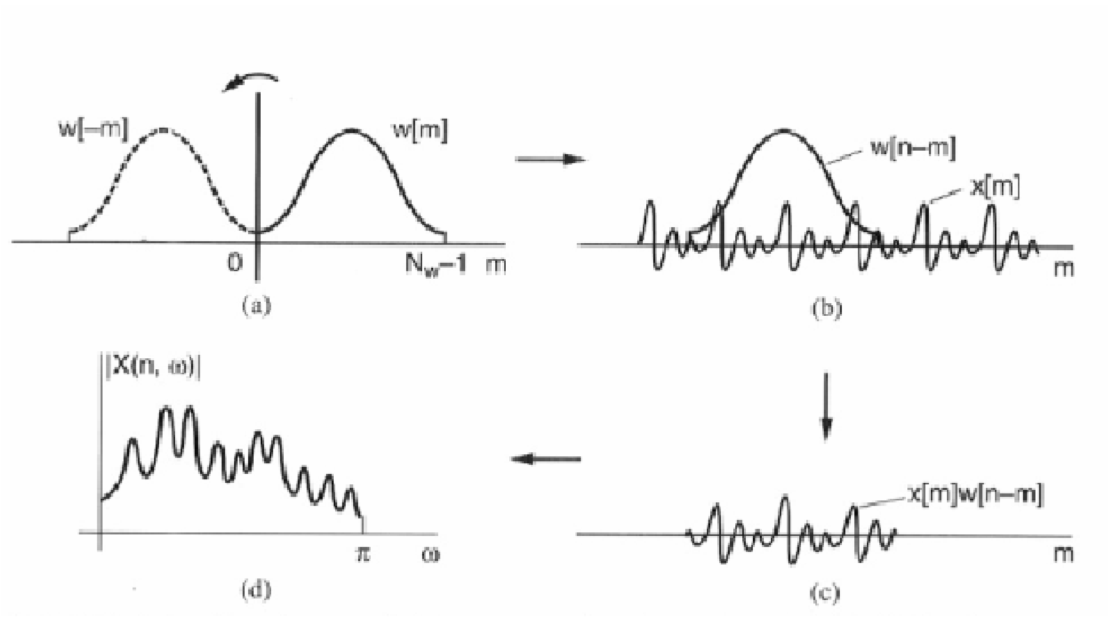
$$X_n(e^{j\varpi}) = \sum_{m=-\infty}^{\infty} x(m) w(n-m) e^{-j\varpi m}$$

where w[n] is is referred to as the analysis window, assumed to be non-zero only in the interval [0,N_w-1].

• The sequence f_n[m] = x[m]w[n - m] is called a short-time section of x[m] at time n.

• f_n[m] is obtained by time-reversing the analysis window, w[m], shifting the result by n points, and multiplying it with x[m].

• $X_n(e^{j\varpi})$ **is the Fourier transform of f_n[m].**



• STFT is a function of two variables, the time index, *n* , which is discrete, and the frequency variable, ω , which is continuous.

• there are 2 distinct interpretations of $X_n(e^{j\omega})$

1. assume n is fixed, then $X_n(e^{j\omega})$ is simply the normal Fourier transform of the sequence $w(n-m)x(m), -\infty < m < \infty$ => for <u>fixed n</u>, $X_n(e^{j\omega})$ has the same properties as a normal Fourier transform

2. consider $X_n(e^{j\omega})$ as a function of the time index n with ω fixed. Then $X_n(e^{j\omega})$ is in the form of a convolution of the signal $x(n)e^{-j\omega n}$ with the window $w(n)$. This leads to an interpretation in the form of linear filtering of the frequency modulated signal $x(n)e^{-j\omega n}$ by $w(n)$.

● **Signal Recovery From STFT**

Since for a given value of n, $X_n\left(e^{j\varpi}\right)$ **has the same properties as a normal Fourier transform, we can recover the input sequence exactly. Since** $X_n\left(e^{j\varpi}\right)$ **is the normal Fourier transform of the windowed sequence** $w(n-m)x(m)$**, then**

$$w(n-m)x(m) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X_n(e^{j\omega})e^{j\omega m}\,d\omega$$

assuming the window satisfies the property that $w(0) \neq 0$, which is a trivial requirement, we obtain

$$x(n) = \frac{1}{2\pi\, w(0)} \int_{-\pi}^{\pi} X_n(e^{j\omega}) e^{j\omega n} d\omega$$

With the requirement that $w(0) \neq 0$, the sequence $x(n)$can be recovered exactly from $X_n(e^{j\varpi})$, if $X_n(e^{j\varpi})$ is known for all values of ϖ over complete period.
Assume that

$$X(e^{j\omega}) = DTFT\left[x(m)\right] = \sum_{m=-\infty}^{\infty} x(m) e^{-j\omega m}$$

$$W\left(e^{j\varpi}\right) = \sum_{m=-\infty}^{\infty} w(m) e^{-j\varpi m}$$

we have

$$X_n\left(e^{j\varpi}\right) = X\left(e^{j\varpi}\right) * \left[e^{j\varpi n} W\left(e^{-j\varpi}\right)\right]$$

$$X_n(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} W(e^{-j\theta}) X(e^{j(\omega-\theta)}) e^{-j\theta n} d\theta$$

limiting case

$$w(n) = 1 \quad -\infty < n < \infty \Leftrightarrow W(e^{j\omega}) = 2\pi\delta(\omega)$$

$$X_n(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} 2\pi\delta(-\theta) X(e^{j(\omega-\theta)}) e^{-j\theta n} d\theta = X(e^{j\omega})$$

• $X_n(e^{j\omega})$ is the convolution of $X(e^{j\omega})$ with the FT of the shifted window sequence $W(e^{-j\omega})\, e^{-j\omega}$

• $X\left(e^{j\varpi}\right)$ **really does not have any meaning since *X(n)* varies with time. Consider *X(n)* defined for window duration and extended for all time to have the same properties, then $X\left(e^{j\varpi}\right)$ does exist with the properties**

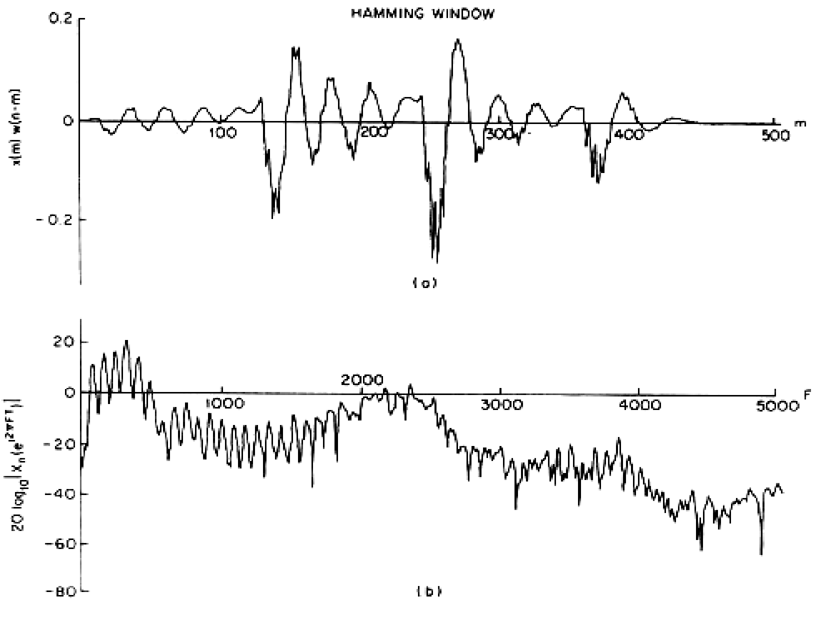
that reflect the sound within the window. (We can also consider X(n)=0 outside the window and define $X\left(e^{j\varpi}\right)$ appropriately.)

• Or we can consider that $X_n\left(e^{j\varpi}\right)$ is a smoothed version of the FT of the part of X(n) that is within the window.

- **Windows in STFT**

For $X_n\left(e^{j\varpi}\right)$ to represent the short-time spectral properties of *X(n)* inside the window, $W\left(e^{j\varpi}\right)$ should be much narrower in frequency than significant spectral regions of $X\left(e^{j\varpi}\right)$--i.e., almost an impulse in frequency. Consider rectangular and Hamming windows, where width of the main spectral lobe is inversely proportional to window length, and side lobe levels are essentially independent of window length.

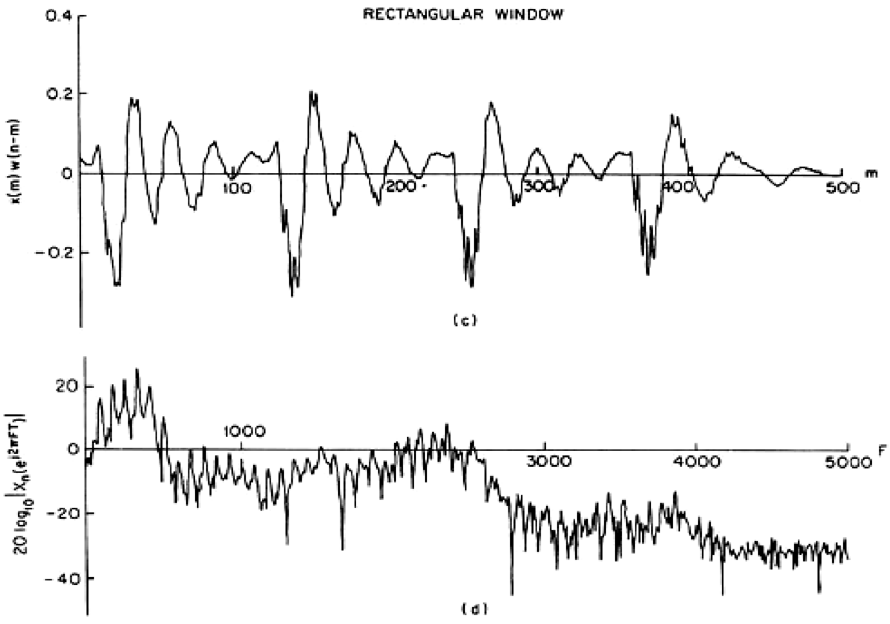
• Rectangular Window: flat window of length N samples; first zero in frequency response occurs at F_S/N, with sidelobe levels of -14 dB or lower.
• Hamming Window: raised cosine window of length L samples; first zero in frequency response occurs at $2F_S/N$, with sidelobe levels of -40 dB or lower.



• 500 sample windows (50 msec)

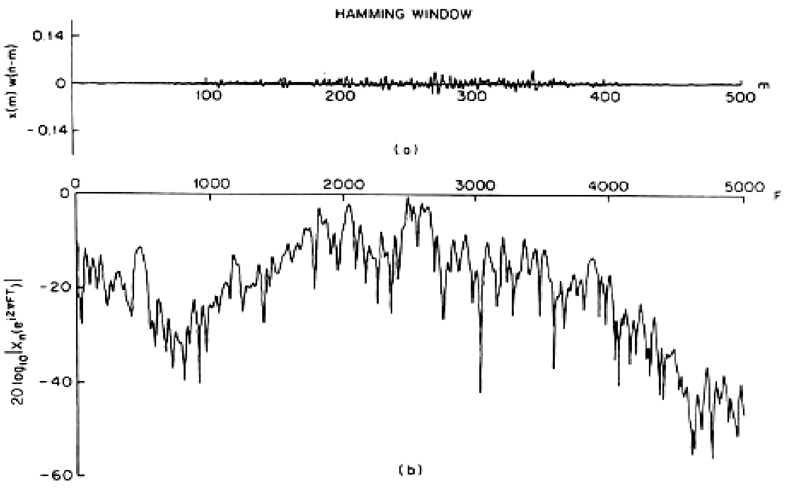
• can see periodicity in time and in frequency

• can see strong first formant (300-400 Hz), strong resonance at 2200 Hz, resonance at 3800 Hz



• spectrum falloff due to glottal pulse shape

• greater sharpness of pitch harmonics due to RW having narrower main lob
• ragged/noisy spectrum using RW due to interharmonic interference because RW sidelobes down only 14 dB

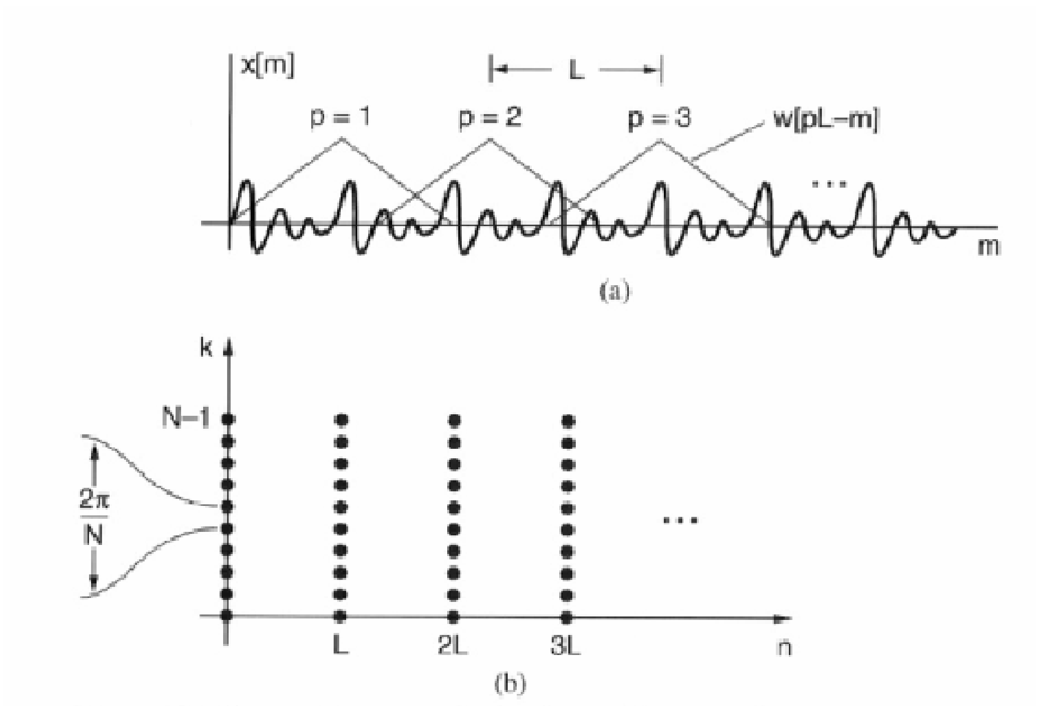


• spectrum shows a slowly varying trend with noise-like set of peaks and valleys
• Time-Frequency Resolution Tradeoffs: The selection of an analysis window is the compromise required between a long window for showing signal detail in frequency and a short window for representing fine temporal structure. However, for a given window, frequency resolution varies inversely with the effective length of the window. Thus, good frequency resolution requires long analysis windows, whereas the desire

for short-time analysis, and thus good time resolution, requires short analysis windows.

- **Time-frequency sampling—Sampling rates of STFT**

• need to sample STFT in both time and frequency to produce an unaliased representation from which x(n) can be exactly recovered
– this is useful for spectral estimation, pitch estimation, formant estimation, speech spectrograms, vocoders
– for applications where the signal is modified, e.g., speech enhancement, cannot under-sample STFT and still recover modified signal exactly



• Discrete STFT
The discrete STFT is obtained from the discrete-time STFT through

$$X(n,k) = X(n,\omega)|_{\omega=\frac{2\pi}{N}k},$$

where we have sampled the discrete-time STFT with a frequency sampling interval of $2\pi/N$ in order to obtain the discrete STFT.
That is,

$$X(n,k) = \sum_{m=-\infty}^{\infty} x[m]w[n-m]e^{-j\frac{2\pi}{N}km}.$$

In many applications, the time variation (the n dimension) of X(n,k) is decimated by a temporal decimation factor, L, to yield the function

X(nL,k). X(nL,k) is the Fourier transform of $f_nL[m]$ = x[m]w[nL - m]

• to determine the sampling rate in time, we take a linear filtering view

1. $X_n(e^{j\omega})$ is the output of a filter with impulse response $w(n)$

2. $W(e^{j\omega})$ is a lowpass response with effective bandwidth of B Hertz

• thus the effective bandwidth of $X_n(e^{j\omega})$ is B Hertz => $X_n(e^{j\omega})$ has to be sampled at a rate of $2B$ samples/second to avoid aliasing

<u>Example</u>: Hamming Window

$$\begin{aligned} w(n) &= 0.54 - 0.46\cos(2\pi n/(L-1)) \quad && 0 \le n \le L-1 \\ &= 0 && \text{otherwise} \end{aligned}$$

$\Rightarrow B \approx \dfrac{2F_s}{L}$(Hz); for $L = 100$, $F_s = 10,000$ Hz => $B = 200$ Hz => need rate of 400/sec (<u>every 25 samples</u>) for sampling rate in time

• Sampling rate in frequency

• since $X_n(e^{j\omega})$ is periodic in ω with period 2π, it is only necessary to sample over an interval of length 2π

• need to determine an appropriate finite set of frequencies, $\omega_k = 2\pi k/N$, $k = 0,1,\ldots,N-1$ at which $X_n(e^{j\omega})$ must be specified to exactly recover $x(n)$

• use the Fourier transform interpretation of $X_n(e^{j\omega})$

1. if the window $w(n)$ is time-limited, then the inverse transform of $X_n(e^{j\omega})$ is time-limited

2. the sampling theorem requres that we sample $X_n(e^{j\omega})$ in the frequency dimension at a rate of at least twice its ('symmetric') "time width"

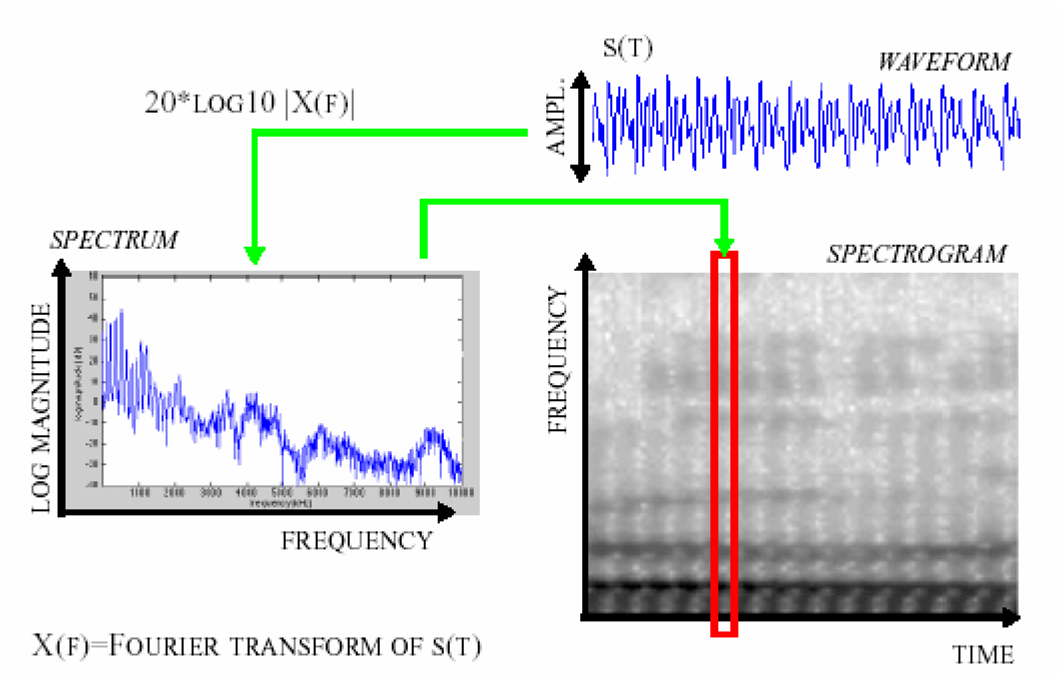
3. since the inverse Fourier transform of $X_n(e^{j\omega})$ is the signal $x(m)w(n-m)$ and this signal is of duration L samples (the duration of $w(n)$), then according to the sampling theorem $X_n(e^{j\omega})$ must be sampled (in frequency) at the set of frequencies

$$\omega_k = \frac{2\pi k}{L}, \quad k = 0,1,\ldots,L-1 \ \ \text{(where } L/2 \text{ is the effective width of the window)}$$

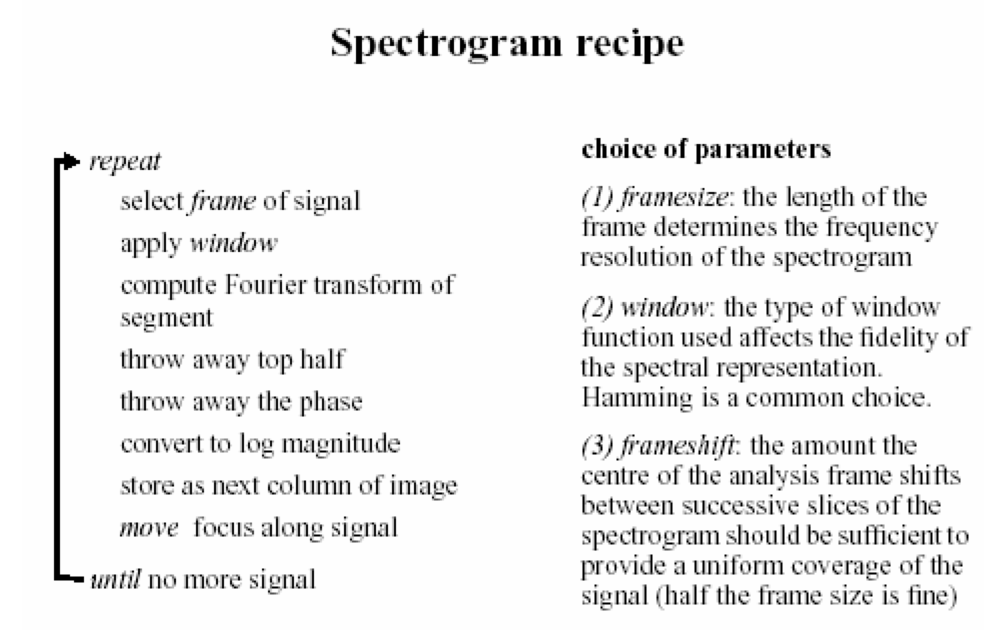
in order to exactly recover $x(n)$ from $X_n(e^{j\omega_k})$

3.3.2 Spectrogram Representation

Spectrogram: The two-dimensional function $|X(n,\varpi)|^2$ is called the spectrogram.



– A spectrogram of a time signal is a two-dimension representation that display time in its horizontal axis and frequency in its vertical axis
– A gray scale is typically used to indicate the energy at each point (t,f)
• "white": low energy, "black": high energy



Spectrogram recipe

```
repeat
    select frame of signal
    apply window
    compute Fourier transform of
    segment
    throw away top half
    throw away the phase
    convert to log magnitude
    store as next column of image
    move  focus along signal
until no more signal
```

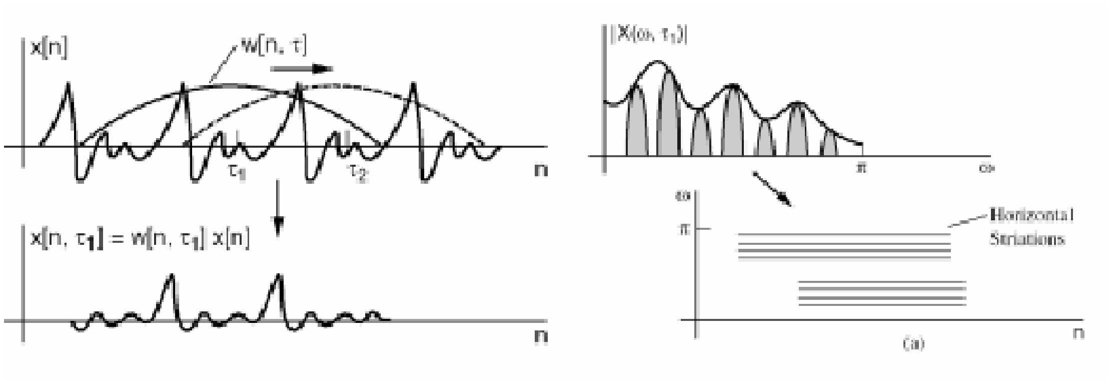
choice of parameters

(1) framesize: the length of the frame determines the frequency resolution of the spectrogram

(2) window: the type of window function used affects the fidelity of the spectral representation. Hamming is a common choice.

(3) frameshift: the amount the centre of the analysis frame shifts between successive slices of the spectrogram should be sufficient to provide a uniform coverage of the signal (half the frame size is fine)

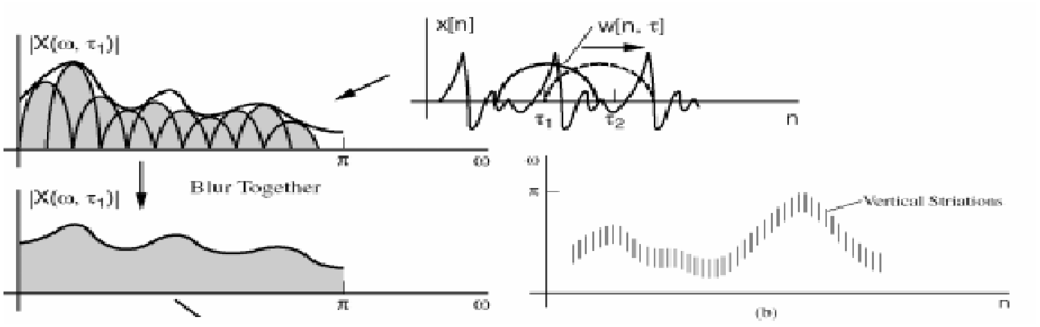
• Narrowband spectrogram:
When the window w[n] is "long" in duration, e.g., a few pitch periods in duration, the spectrogram is referred to as a narrowband spectrogram.
Narrowband spectrograms exhibit the harmonic structure in x[n] as "horizontal striations".

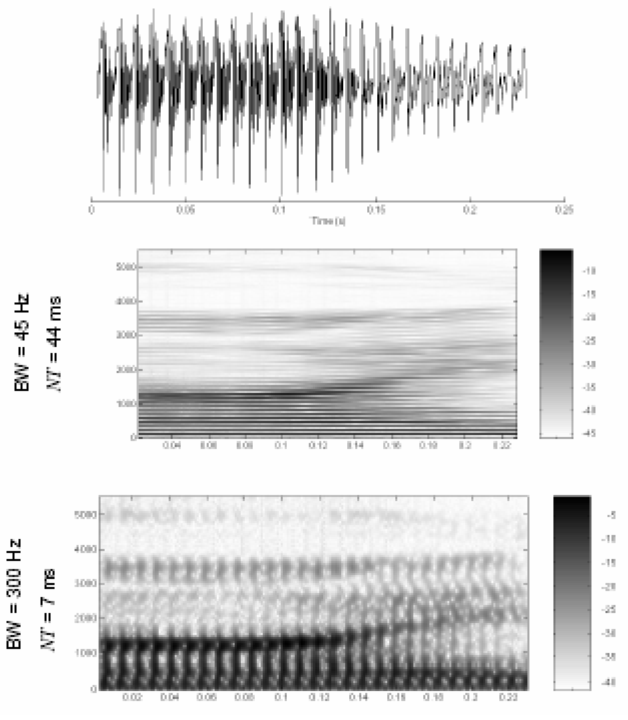


• Wideband spectrogram

For a "short" window w[n], the spectrogram is referred to as wideband spectrogram.

Wideband spectrograms exhibit the periodic temporal structure in x[n] as "vertical striations”.





3.3.3 Filtering View of the STFT

The STFT can also be viewed as the output of a filtering operation where the analysis window w[n] plays the role of the filter impulse response, and hence the alternative name analysis filter for w[n].

For the filtering view of the STFT, we fix the value of ω at ω_o (in the Fourier transform view, we had fixed the value of n).

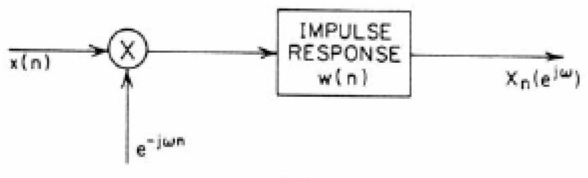
1. modulation-lowpass filter form:

$$X_n(e^{j\omega}) = \sum_{m=-\infty}^{\infty} x(m)e^{-j\omega m}w(n-m), \quad n \text{ variable, } \omega \text{ fixed}$$

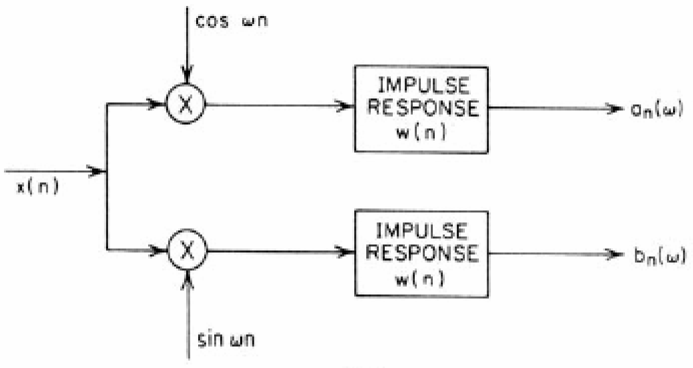
$$= \left(x(n)e^{-j\omega n}\right) * w(n)$$

$$= \left(x(n)\cos(\omega n)\right) * w(n) - j\left(x(n)\sin(\omega n)\right) * w(n)$$

$$= a_n(\omega) - jb_n(\omega)$$



(a)



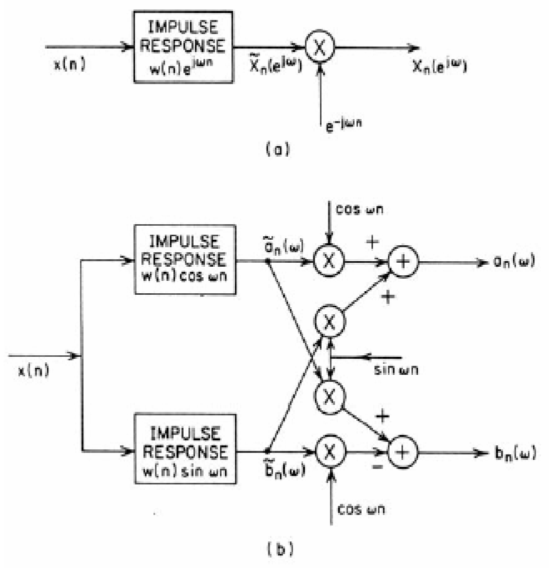
(b)

2. bandpass filter-demodulation

$$X_n(e^{j\omega}) = \sum_{m=-\infty}^{\infty} w(m)x(n-m)e^{-j\omega(n-m)}$$

$$= e^{-j\omega n}\sum_{m=-\infty}^{\infty} (w(m)e^{j\omega m})x(n-m)$$

$$= e^{-j\omega n}[(w(m)e^{j\omega m}) * x(n)], \quad n \text{ variable, } \omega \text{ fixed}$$



- complex bandpass filter output modulated by signal $e^{-j\omega n}$
- if $W(e^{j\theta})$ is lowpass, then filter is bandpass around $\theta = \omega$
- all real computation for lower half structure

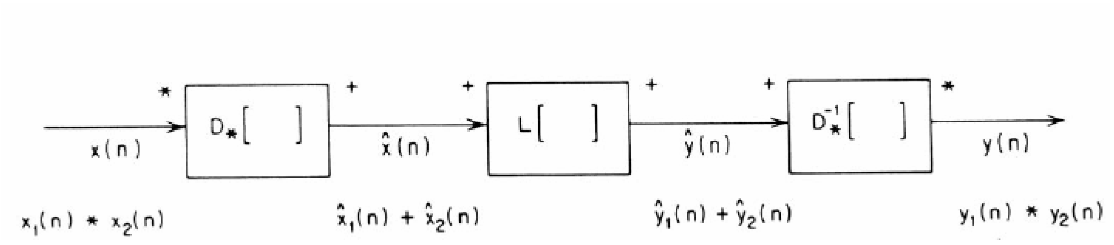
Conclusion:

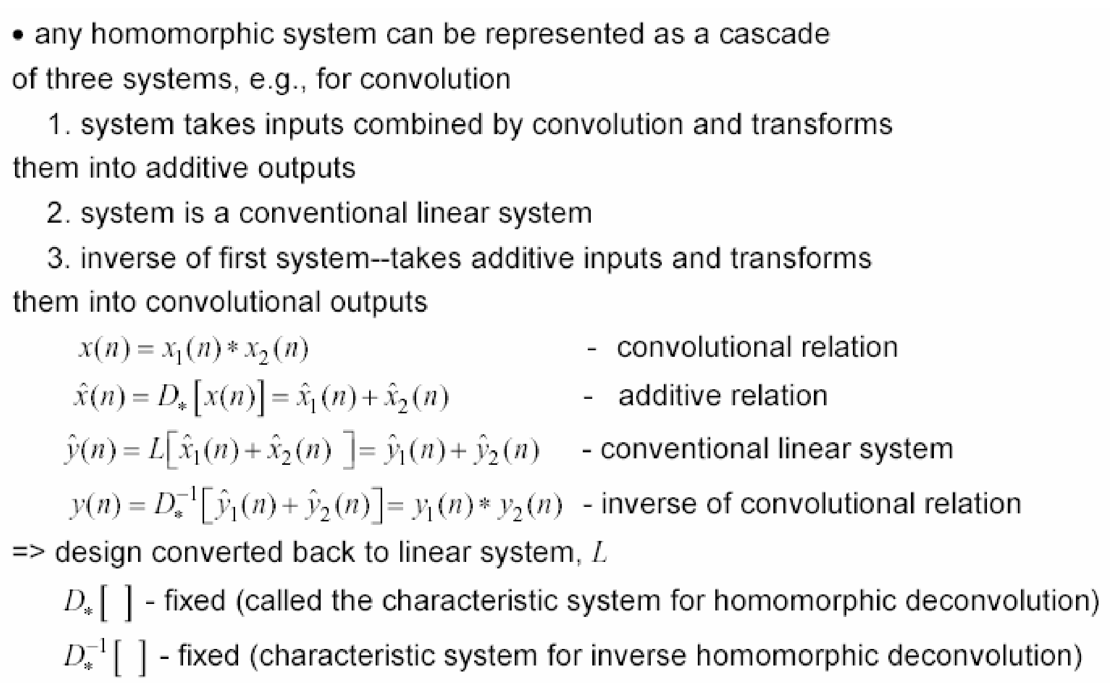
The speech signal is usually analyzed using spectral features, instead of directly using its waveform. There are two reasons for this. One is that the speech signal is considered to be reproducible by summing the sinusoidal waves, the amplitude and phase of which change slowly. The other is that the critical features for perceiving speech by the human ear are mainly included in the spectral information (with the phase information not usually playing a key role).
The short-time spectrum of speech can be regarded as the product of two elements: the spectral envelope, which slowly changes as a function of frequency, and the spectral fine structure, which changes rapidly. The spectral fine structure produces periodic patterns for voiced sounds. The spectral envelope, or the overall spectral feature, reflects not only the resonance and antiresonance characteristics of the articulation organs, but also the overall shape of the glottal source spectrum and radiation characteristics at the lips and nostrils. On the other hand, the spectral fine structure corresponds to the periodicity of the sound source, that is, pitch structure for voiced sounds.

3.4 Cepstral Analysis

The cepstrum is defined as the inverse Fourier transform of the short-time logarithmic amplitude spectrum. It was invented by Bogert et al. in 1960's. The term cepstrum is essentially a coined word by reversing the first syllable of the word spectrum. The independent parameter (variable n in c(n)) for the cepstrum is called quefrency, which is obviously formed from the word frequency. Since the cepdtrum is the inverse transform of the frequency domain function, the quefrency becomes the time domain parameter. The special feature of the cepstrum is that it allows for the separate representation of the spectral envelope and fine structure.
The cepstral analysis of speech signal is in terms of homomorphic systems and deconvolution.

- **Homomorphic system**





• any homomorphic system can be represented as a cascade of three systems, e.g., for convolution

1. system takes inputs combined by convolution and transforms them into additive outputs
2. system is a conventional linear system
3. inverse of first system--takes additive inputs and transforms them into convolutional outputs

$x(n) = x_1(n) * x_2(n)$ - convolutional relation

$\hat{x}(n) = D_*[x(n)] = \hat{x}_1(n) + \hat{x}_2(n)$ - additive relation

$\hat{y}(n) = L[\hat{x}_1(n) + \hat{x}_2(n)] = \hat{y}_1(n) + \hat{y}_2(n)$ - conventional linear system

$y(n) = D_*^{-1}[\hat{y}_1(n) + \hat{y}_2(n)] = y_1(n) * y_2(n)$ - inverse of convolutional relation

=> design converted back to linear system, L

$D_*[\ ]$ - fixed (called the characteristic system for homomorphic deconvolution)

$D_*^{-1}[\ ]$ - fixed (characteristic system for inverse homomorphic deconvolution)

In frequency domain, we have

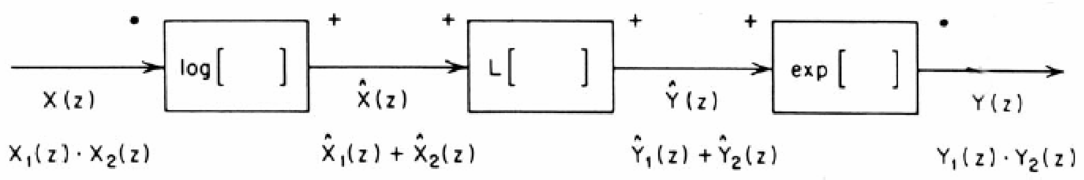
$$x(n) = x_1(n) * x_2(n)$$
$$X(z) = X_1(z) \cdot X_2(z)$$

- since

$$D_*\left[x(n)\right] = \hat{x}_1(n) + \hat{x}_2(n) = \hat{x}(n)$$
$$D_*\left[X(z)\right] = \hat{X}_1(z) + \hat{X}_2(z) = \hat{X}(z)$$

=> use log function which converts products to sums

$$\hat{X}(z) = \log\left[X(z)\right] = \log\left[X_1(z) \cdot X_2(z)\right]$$
$$= \log\left[X_1(z)\right] + \log\left[X_2(z)\right] = \hat{X}_1(z) + \hat{X}_2(z)$$
$$\hat{Y}(z) = L\left[\hat{X}_1(z) + \hat{X}_2(z)\right] = \hat{Y}_1(z) + \hat{Y}_2(z)$$
$$Y(z) = \exp\left[\hat{Y}_1(z) + \hat{Y}_2(z)\right] = Y_1(z) \cdot Y_2(z)$$



$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}$$
$$\hat{X}(z) = \log\left[X(z)\right] = \log\left|X(z)\right| + j\arg\left[X(z)\right]$$
$$\hat{x}(n) = \frac{1}{2\pi j}\oint_C \hat{X}(z) z^{n-1} dz$$

$$\hat{Y}(z) = \sum_{n=-\infty}^{\infty} \hat{y}(n) z^{-n}$$
$$Y(z) = \exp\left[\hat{Y}(z)\right]$$
$$y(n) = \frac{1}{2\pi j}\oint_C Y(z) z^{n-1} dz$$

- **Complex and Real Complex and Real Cepstrum**

• define the inverse transform of $\hat{X}(e^{j\omega})$ as

$$\hat{x}(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{X}(e^{j\omega}) e^{j\omega n}\, d\omega$$

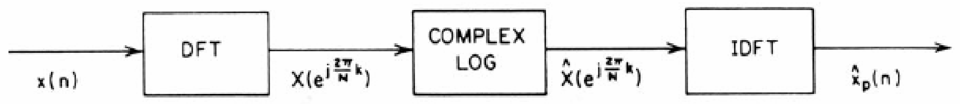
• where $\hat{x}(n)$ called the "complex cepstrum" since a complex logarithm is involved in the computation

• can also define a "real cepstrum" using just the real part of the logarithm, giving

$$c(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{Re}\left[\hat{X}(e^{j\omega})\right] e^{j\omega n}\, d\omega$$
$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} \log|X(e^{j\omega})| e^{j\omega n}\, d\omega$$

• can show that $c(n)$ is the even part of $\hat{x}(n)$

The Complex Cepstrum



$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n}$$

$$X_p(k) = X(e^{j\frac{2\pi}{N}k}) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\frac{2\pi}{N}kn} \quad - N \text{ point DFT}$$

$$\hat{X}(e^{j\omega}) = \log\left[X(e^{j\omega})\right] = \log\left|X(e^{j\omega})\right| + j\arg\left[X(e^{j\omega})\right]$$

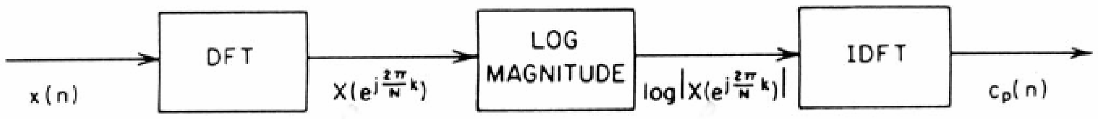
$$\hat{X}_p(k) = \log\left\{X_p(k)\right\} = \log\left|X_p(k)\right| + j\arg\left\{X_p(k)\right\}$$

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

$$\hat{x}_p(n) = \frac{1}{N}\sum_{k=0}^{N-1} \hat{X}_p(k) e^{j\frac{2\pi}{N}kn} = \sum_{r=-\infty}^{\infty} \hat{x}(n+rN)$$

• $\hat{x}_p(n)$ is an aliased version of $\hat{x}(n) \Rightarrow$ use as large a value of N as possible to minimize aliasing

The Cepstrum



$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$$

$$X_p(k) = X(e^{j\frac{2\pi}{N}k}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\frac{2\pi}{N}kn} \quad - N \text{ point DFT}$$

$$C(e^{j\omega}) = \log\left|X(e^{j\omega})\right|$$

$$C_p(k) = \log\left|X_p(e^{j\frac{2\pi}{N}k})\right|$$

$$c(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} C(e^{j\omega})e^{j\omega n}d\omega = \left[\hat{x}(n) + \hat{x}(-n)\right]/2$$

$$c_p(n) = \frac{1}{N}\sum_{k=0}^{N-1} C_p(k)e^{j\frac{2\pi}{N}kn} = \sum_{r=-\infty}^{\infty} c(n+rN)$$

- $c_p(n)$ is an aliased version of $c(n) \Rightarrow$ use as large a value of N as possible to minimize aliasing

Complex Cepstrum Examples

- single pole inside the unit circle ($|a| < 1$)

$$X(z) = \frac{1}{1 - az^{-1}}$$

$$\hat{X}(z) = \log\left[X(z)\right] = -\log\left(1 - a\,z^{-1}\right) = \sum_{n=1}^{\infty} \frac{a^n}{n} z^{-n}$$

$$\begin{aligned} \hat{x}(n) &= \frac{a^n}{n} & n \geq 1 \\ &= 0 & n < 1 \end{aligned}$$

$$\hat{x}(n) = \sum_{r=1}^{\infty} \frac{a^r}{r}\delta(n-r)$$

- single zero outside the unit circle ($|b| < 1$)

$$X(z) = 1 - bz \quad \text{(zero at } z = 1/b\text{)}$$

$$\hat{X}(z) = \log\left[X(z)\right] = \log\left(1 - b\,z\right)$$

$$= -\sum_{n=1}^{\infty} \frac{b^n}{n} z^n = \sum_{n=-\infty}^{1} \frac{b^{-n}}{n} z^{-n}$$

$$\hat{x}(n) = \frac{b^{-n}}{n} \qquad n \leq -1$$

$$= 0 \qquad n > 1$$

$$\hat{x}(n) = -\sum_{r=1}^{\infty} \frac{b^r}{r} \delta(n + r)$$

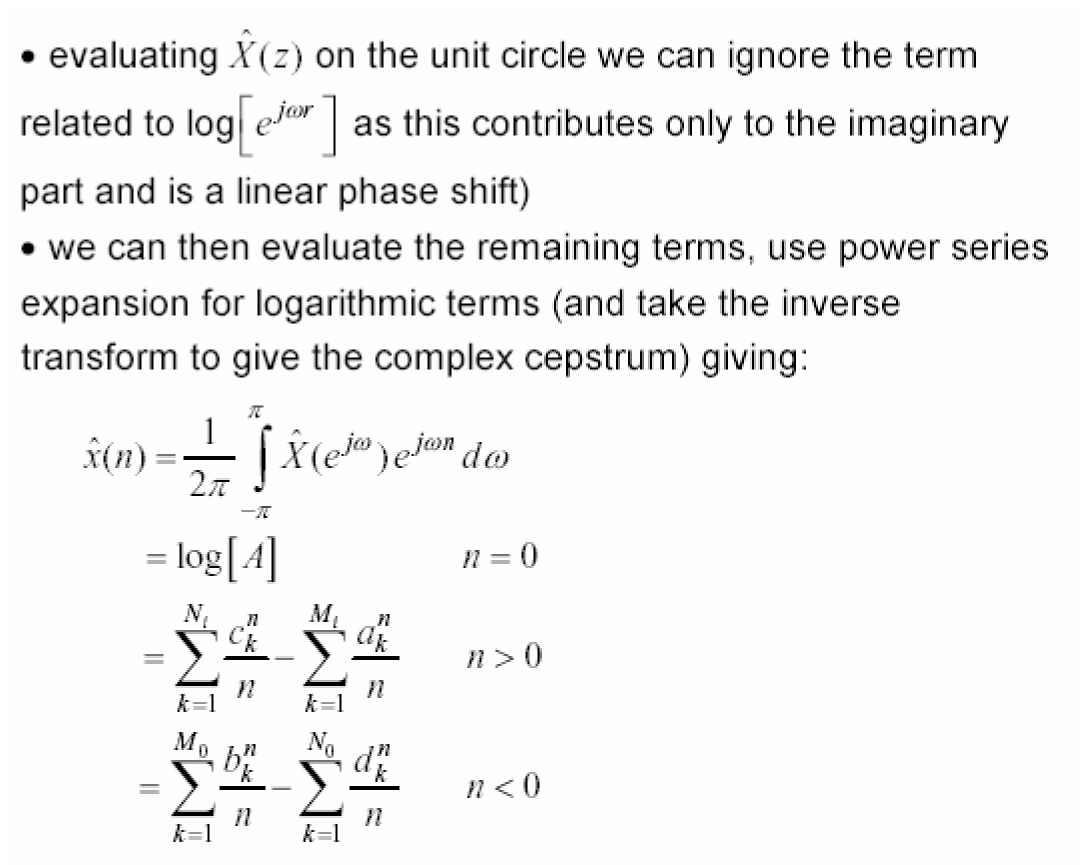
- consider rational z-transforms of the general type

$$X(z) = \frac{Az^r \prod_{k=1}^{M_i}\left(1 - a_k z^{-1}\right)\prod_{k=1}^{M_0}\left(1 - b_k z\right)}{\prod_{k=1}^{N_i}\left(1 - c_k z^{-1}\right)\prod_{k=1}^{N_0}\left(1 - d_k z\right)}$$

- with all coefficients $a_k, b_k, c_k, d_k < 1$ => all c_k poles and a_k zeros are inside the unit circle; all d_k poles and b_k zeros are outside the unit circle; the factor z^r is just a shift of the time origin and has no effect
- the complex logarithm of $X(z)$ is

$$\hat{X}(z) = \log\left[X(z)\right] = \log\left[A\right] + \log\left[z^r\right] + \sum_{k=1}^{M_i} \log\left(1 - a_k z^{-1}\right)$$

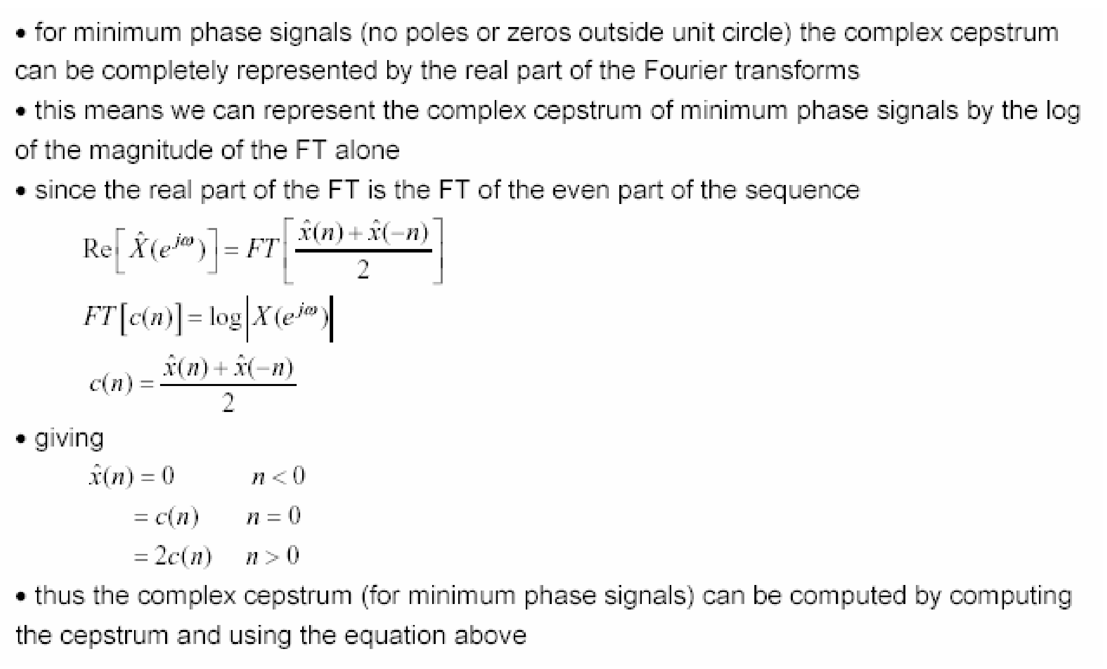
$$+ \sum_{k=1}^{M_0} \log\left(1 - b_k z\right) - \sum_{k=1}^{N_i} \log\left(1 - c_k z^{-1}\right) - \sum_{k=1}^{N_0} \log\left(1 - d_k z\right)$$



• evaluating $\hat{X}(z)$ on the unit circle we can ignore the term related to $\log\left[e^{j\omega r}\right]$ as this contributes only to the imaginary part and is a linear phase shift)

• we can then evaluate the remaining terms, use power series expansion for logarithmic terms (and take the inverse transform to give the complex cepstrum) giving:

$$\begin{aligned}
\hat{x}(n) &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{X}(e^{j\omega})e^{j\omega n}\,d\omega \\
&= \log[A] && n = 0 \\
&= \sum_{k=1}^{N_i}\frac{c_k^n}{n} - \sum_{k=1}^{M_i}\frac{a_k^n}{n} && n > 0 \\
&= \sum_{k=1}^{M_0}\frac{b_k^n}{n} - \sum_{k=1}^{N_0}\frac{d_k^n}{n} && n < 0
\end{aligned}$$



• for minimum phase signals (no poles or zeros outside unit circle) the complex cepstrum can be completely represented by the real part of the Fourier transforms

• this means we can represent the complex cepstrum of minimum phase signals by the log of the magnitude of the FT alone

• since the real part of the FT is the FT of the even part of the sequence

$$\mathrm{Re}\left[\hat{X}(e^{j\omega})\right] = FT\left[\frac{\hat{x}(n)+\hat{x}(-n)}{2}\right]$$

$$FT\left[c(n)\right] = \log\left|X(e^{j\omega})\right|$$

$$c(n) = \frac{\hat{x}(n)+\hat{x}(-n)}{2}$$

• giving

$$\begin{aligned}
\hat{x}(n) &= 0 && n < 0 \\
&= c(n) && n = 0 \\
&= 2c(n) && n > 0
\end{aligned}$$

• thus the complex cepstrum (for minimum phase signals) can be computed by computing the cepstrum and using the equation above

● Homomorphic Filtering of Voiced Speech

• **Goal is to separate out the excitation impulses from the remaining components of the complex cepstrum.**

• **cepstral window, l(n), to separate excitation pulses from combined vocal tract**

– l(n)=1 for $|n|<n_0<N_p$
– l(n)=0 for $|n| \geqslant n_0$
– this window removes excitation pulses
– l(n)=0 for $|n|<n_0<N_p$
– l(n)=1 for $|n| \geqslant n_0$
– this window removes combined vocal tract

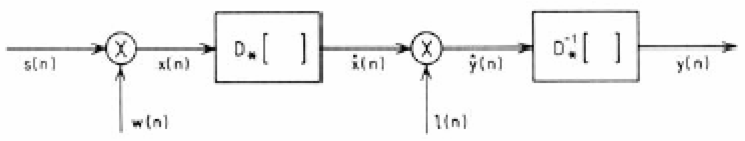
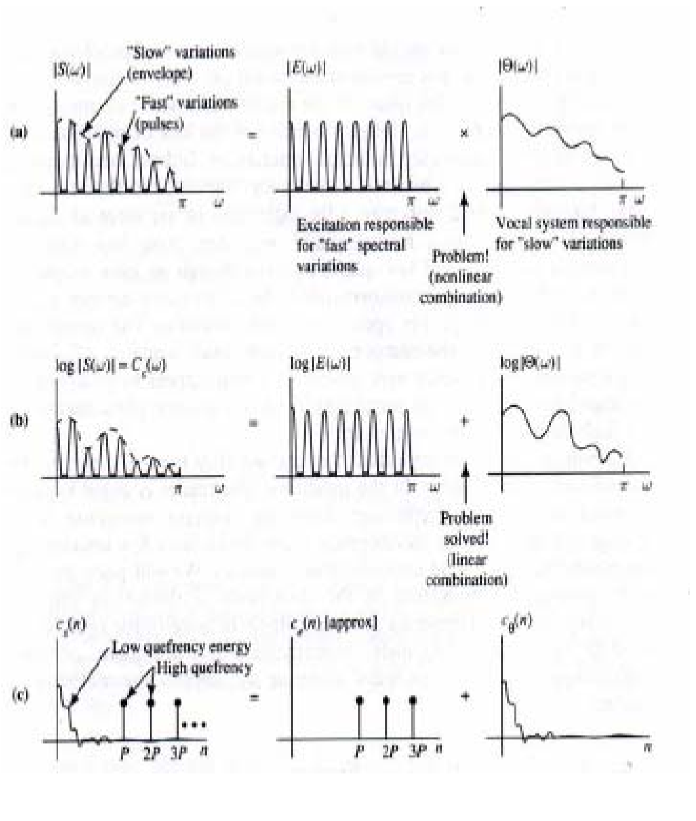


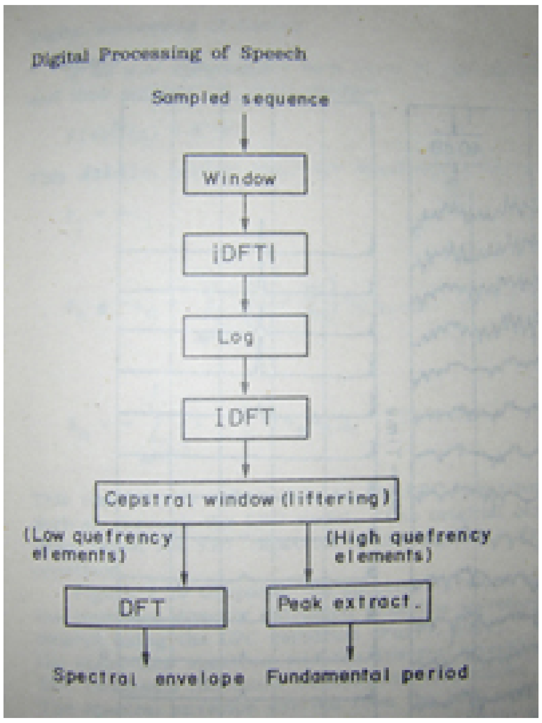
Fig. 7.9 Implementation of a system for homomorphic filtering of

$$\hat{y}(n) = l(n) \cdot \hat{x}(n)$$

$$\hat{Y}(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{X}(e^{j\theta}) L(e^{j(\omega-\theta)}) d\theta$$

• the filtered signal is processed by the inverse characteristic system to recover the combined vocal tract component



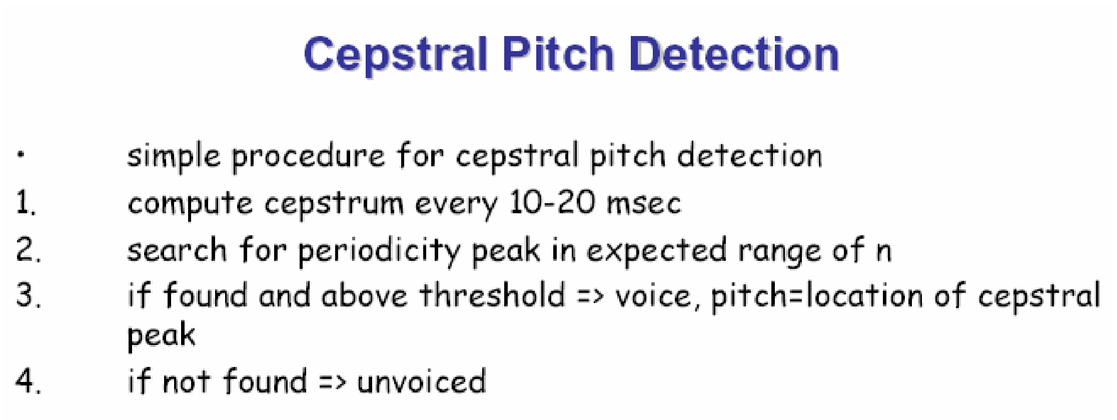


<u>Explanation of above figure</u>:

(a) **In speech magnitude spectrum, $|S(\varpi)|$, two components can be identified: envelope (slowing varying) due to the speech system, $|\Theta(\varpi)|$, and a "quickly varying" part due to excitation, $|E(\varpi)|$. These components are combined by addition. Their time domain counterparts, $\theta(n)\, and\ e(n)$, are convolved.**

(b) **Once the logarithm of the spectral magnitude is taken, the two convolved signal components, $\theta(n)\, and\ e(n)$, have additive correlates in the new signal, $C_s(\varpi)$. The former corresponds to a slowly varying component and the latter to a quickly varying component.**

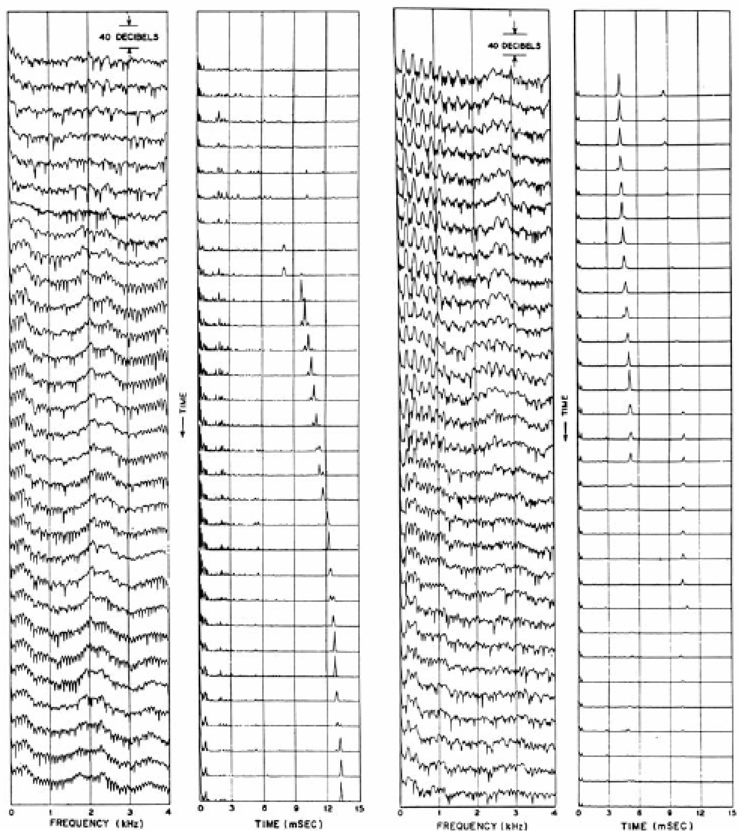
(c) **When the IDTFT is taken, the solely varying part yields a cepstral component at low quefrencies (smaller values on the time axis), and the component with fast variations results in a cepstral component at high quefrencies (larger values on the time axis). The low-quefrency part of the cepstrum therefore represents an approximation to the cepstrum of the vocal system impulse response, $c_\theta(n)$, and the high-quefrency part corresponds to the cepstrum of the excitation, $c_e(n)$.**



Cepstral Pitch Detection

- simple procedure for cepstral pitch detection

1. compute cepstrum every 10-20 msec
2. search for periodicity peak in expected range of n
3. if found and above threshold => voice, pitch=location of cepstral peak
4. if not found => unvoiced



- male speaker (left), female speaker (right)
- log spectra and cepstra $[c(n)]^2$
- 40 msec HW with 10 msec jumps
- for male-first 7 intervals are unvoiced; for female voiced at beginning, unvoiced at end
- pitch period increases with time for male; pitch period much shorter for female
- pitch period doubling at end of voiced regions for female

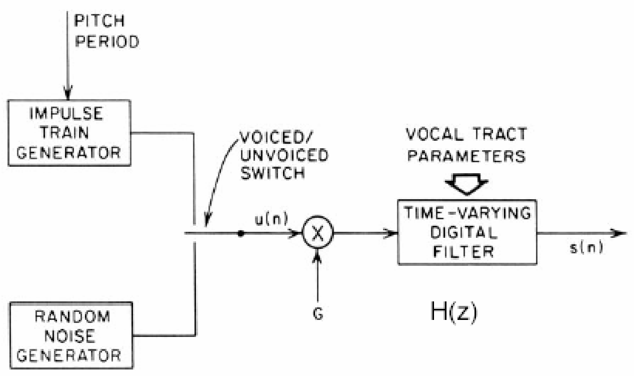
3.5 Linear prediction analysis

A very powerful method for speech analysis is based on linear predictive coding (LPC), also known as LPC analysis or auto-regressive (AR) modeling. This method is widely used because it is fast and simple, yet an effective way of estimating the main parameters of speech signals.
Basic idea of Linear Prediction: current speech sample can be closely approximated as a linear combination of past samples, i.e.

$$x(n) = \sum_{k=1}^{p} \alpha_k \, x(n-k) \text{ for some value of } p, \alpha_k\text{'s}$$

The predictor coefficients (the a_k's) are determined (computed) by minimizing the sum of squared differences (over the finite interval) between the actual speech samples and the linearly predicted ones

3.5.1 Basic Principle of LPC



$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + G\,u(n)$$

- use the time-varying digital filter to represent the glottal pulse shape, the vocal tract IR and the radiation effects, i.e.,

$$\boxed{H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}}}$$

- system excited by an impulse train for voiced speech or a random sequence for unvoiced speech
- already know how to estimate pitch period and V/UV
- this "all-pole model" is a natural representation for non-nasal voiced speech, but also works reasonably well for nasals and unvoiced sounds (even without explicit zeros) => recall that a lot of poles can approximate zeros

- a p^{th} order linear predictor is a system of the form

$$\tilde{s}(n) = \sum_{k=1}^{p} \alpha_k s(n-k) \Leftrightarrow P(z) = \sum_{k=1}^{p} \alpha_k z^{-k} = \frac{\tilde{S}(z)}{S(z)}$$

- the prediction error, $e(n)$, is of the form

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

- the prediction error is the output of a system with transfer function

$$A(z) = \frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{p} \alpha_k z^{-k}$$

- if the speech signal obeys the production model exactly, and if $\alpha_k = a_k, 1 \le k \le p$

$\Rightarrow e(n) = Gu(n)$ and $A(z)$ is an inverse filter for $H(z)$, i.e.,

$$H(z) = \frac{G}{A(z)}$$

- **The Orthogonality Principle**

We know that the prediction is optimum when we determined (computed) The predictor coefficients (the a_k's) by minimizing the sum of squared differences (Principle of Minimum Mean Squared Error). i.e.

$$\varepsilon = E\left[e^2(n)\right] = \min$$

where $\hat{x}(n) = -\sum_{k=1}^{p} a_k x(n-k)$**;**

$$e(n) = x(n) - \hat{x}(n) = \sum_{k=0}^{p} a_k x(n-k),\ a_0 = 1$$

Taking the derivative of ε **with respect to** a_k **and equating to 0, we obtain:**

$$\frac{\partial \varepsilon}{\partial a_k} = 2E\left[e(n)\frac{\partial e(n)}{\partial a_k}\right] = 0$$

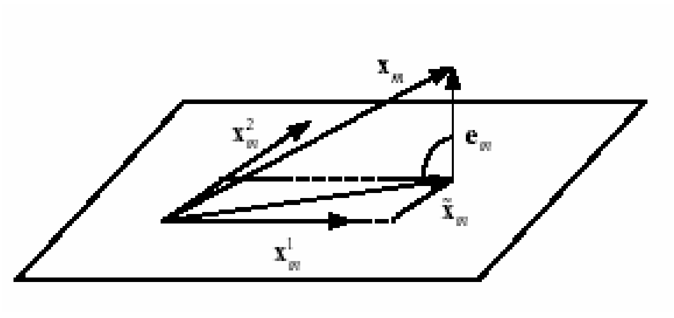
Notice that $\frac{\partial e(n)}{\partial a_k} = x(n-k), \quad k = 1,2,\cdots p$

So we have

$$E\left[e(n)x(n-k)\right] = 0, \quad k = 1,2,\cdots p$$

The above equation is so called orthogonality equation, or orthogonality principle, says that the predictor coefficients that minimize the prediction error are such that the error must be orthogonal to the past sample

vectors.



The orthogonality equation can also be rewritten as:

$$E\left[\sum_{l=0}^{p} a_l x(n-l)x(n-k)\right] = 0, \quad k = 1,2,\cdots p; \quad a_0 = 1$$

or

$$\sum_{l=0}^{p} a_l E[x(n-l)x(n-k)] = 0, \quad k = 1,2,\cdots p; \quad a_0 = 1$$

For convenience, we can define the autocorrelation coefficients as

$$R(k-l) = E[x(n-l)x(n-k)], \quad l,k = 1,2,\cdots p$$

Hence we have

$$\sum_{l=0}^{p} a_l R(k-l) = 0, \quad k = 1,2,\cdots p; \quad a_0 = 1$$

The above equation is known as normal equation. We can obtain a set of p equations in p unknowns that can be solved in an efficient manner for the $\{a_k\}$.

In general, we can define the correlation coefficients as

$$\phi_n(i,k) = \sum_m s_n(m-i)s_n(m-k)$$

so that we obtain the so-called Yule-Walker equations:

$$\sum_{k=1}^{p} \alpha_k \phi_n(i,k) = \phi_n(i,0), \quad i = 1,2,\ldots,p$$

Solution of the set of p linear equations results in the p LPC coefficients that minimize the prediction error. With a_i satisfying the Yule-Walker equation, the total prediction error takes on the following value:

$$E_n = \sum_m s_n^2(m) - \sum_{k=1}^{p} \alpha_k \sum_m s_n(m) s_n(m-k)$$

which can be written in the form

$$E_n = \phi_n(0,0) - \sum_{k=1}^{p} \alpha_k \phi_n(0,k)$$

For expressing in the form of autocorrelation, we can derive form the equation

$$\varepsilon = E\left[e(n)\left(x(n) + \sum_{l=1}^{p} a_l x(n-l) \right) \right]$$

substituting the above equation by orthogonality equation, we have

$$\begin{aligned} \varepsilon_{\min} &= E[e(n)x(n)] \\ &= E\left[\left(\sum_{l=0}^{p} a_l x(n-l) \right) x(n) \right] \\ &= \sum_{l=0}^{p} a_l E[x(n-l)x(n)], \quad a_0 = 1 \end{aligned}$$

Considering the definition of autocorrelation, we can write the above equation in the form of

$$\varepsilon_{\min} = \sum_{l=0}^{p} a_l R(k-l), \quad k = 0, \quad a_0 = 1$$

Combine the above equation with the normal equation, we have

$$\sum_{l=0}^{p} a_l R(k-l) = \begin{cases} \varepsilon_{\min} & , k = 0 \\ 0 & , k = 1,2,\cdots p \end{cases}$$

where it also has $a_0 = 1$.

The above equation is another form of Yule-Walker equation, which is a set of *p*+1 linear equations in *p*+1 unknowns, say $\{a_k\}$ and $\varepsilon_{\min}$.

It is convenient to define a normalized prediction error u[n] with unity energy

$$\sum_n u_m^2[n] = 1$$

and a gain G, such that

$$e_m[n] = G u_m[n]$$

The gain G can be computed from the short-term prediction error

$$E_m = \sum_n e_m^2[n] = G^2 \sum_n u_m^2[n] = G^2$$

- **Solution of the LPC Equations**

The solution of the Yule-Walker equations can be achieved with any standard matrix inversion package. Because of the special form of the matrix here, some efficient solutions are possible, as described below. Also, each solution offers a different insight so we present three different algorithms: the covariance method, the autocorrelation method, and the lattice method.

1. compute $\phi_n(i,k)$ for $1 \le i \le p,\ 0 \le k \le p$
2. solve matrix equation for α_k

- need to specify $\underline{\text{range of } m}$ to compute $\phi_n(i,k)$
- need to specify $s_n(m)$

3.5.2 Solution for LPC Coefficients

- **Autocorrelation Method**

• assume $s_n(m)$ exists for $0 \le m \le N-1$ and is exactly zero everywhere else (i.e., window of length N samples)

$$s_n(m) = s(m+n)\,w(m)$$

• where $w(m)$ is a finite length window of length N samples

• if $s_n(m)$ is non-zero only for $0 \le m \le N-1$ then $e_n(m) = s_n(m) - \sum_{k=1}^{p} \alpha_k s_n(m-k)$ is non-zero only over the interval $0 \le m \le N-1+p$, giving

$$E_n = \sum_{m=-\infty}^{\infty} e_n^2(m) = \sum_{m=0}^{N-1+p} e_n^2(m)$$

• at values of m near 0 (i.e., $m = 0,1,\ldots,p-1$) we are predicting signal from zero-valued samples outside the window range => $e_n(m)$ will be (relatively) large

• at values near $m = N$ (i.e., $m = N, N+1, \ldots, N+p-1$) we are predicting zero-valued samples (outside window range) from non-zero samples => $e_n(m)$ will be (relatively) large

- for these reasons, normally use windows that taper the segment to zero (e.g., Hamming window)

- for calculation of $\phi_n(i,k)$ since $s_n(m)=0$ outside the range $0 \le m \le N-1$, then

$$\phi_n(i,k) = \sum_{m=0}^{N-1+p} s_n(m-i)s_n(m-k), \quad 1 \le i \le p, 0 \le k \le p$$

- which is equivalent to the form

$$\phi_n(i,k) = \sum_{m=0}^{N-1+(i-k)} s_n(m)s_n(m+i-k), \; 1 \le i \le p, 0 \le k \le p$$

- there are $N-|i-k|$ non-zero terms in the computation of $\phi_n(i,k)$ for each value of i and k; can easily show that

$$\phi_n(i,k) = f(i-k) = R_n(i-k)$$

- where $R_n(i-k)$ is the short-time autocorrelation of $s_n(m)$ evaluated at $i-k$ where

$$R_n(k) = \sum_{m=0}^{N-1-k} s_n(m)s_n(m+k)$$

- since $R_n(k)$ is even, then

$$\phi_n(i,k) = R_n(|i-k|), 1 \le i \le p, \; 0 \le k \le p$$

- thus the basic equation becomes

$$\sum_{k=1}^{p} \alpha_k \phi_n(i-k) = \phi_n(i,0), \quad 1 \le i \le p$$

$$\sum_{k=1}^{p} \alpha_k R_n(|i,k|) = R_n(i), \quad 1 \le i \le p$$

- with the minimum mean-squared prediction error of the form

$$E_n = \phi_n(0,0) - \sum_{k=1}^{p} \alpha_k \phi_n(0,k)$$

$$= R_n(0) - \sum_{k=1}^{p} \alpha_k R_n(k)$$

- or expressed in matrix form

$$\begin{bmatrix} R_n(0) & R_n(1) & . & . & R_n(p-1) \\ R_n(1) & R_n(0) & . & . & R_n(p-2) \\ . & . & . & . & . \\ . & . & . & . & . \\ R_n(p-1) & R_n(p-2) & . & . & R_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ . \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ . \\ . \\ R_n(p) \end{bmatrix}$$

- this is a $p \times p$ Toeplitz Matrix => symmetric with all diagonal elements equal => there exist efficient algorithms to solve for $\{\alpha_k\}$

- **Covariance Method**

- there is a second basic approach to defining the speech segment $s_n(m)$ and the limits on the sums, namely fix the interval over which the mean-squared error is computed, giving

$$E_n = \sum_{m=0}^{N-1} e_n^2(m) = \sum_{m=0}^{N-1} \left[s_n(m) - \sum_{k=1}^{p} \alpha_k s_n(m-k) \right]^2$$

$$\phi_n(i,k) = \sum_{m=0}^{N-1} s_n(m-i) s_n(m-k), \quad 1 \le i \le p, \; 0 \le k \le p$$

- changing the summation index gives

$$\phi_n(i,k) = \sum_{m=-i}^{N-i-1} s_n(m) s_n(m+i-k), \quad 1 \le i \le p, \; 0 \le k \le p$$

$$\phi_n(i,k) = \sum_{m=-k}^{N-k-1} s_n(m) s_n(m+k-i), \quad 1 \le i \le p, \; 0 \le k \le p$$

- key difference from Autocorrelation Method is that limits of summation include terms before $m = 0$ => window extends p samples backwards from $s(n-p)$ to $s(n+N-1)$
- since we are extending window backwards, don't need to taper it using HW- since there is no transition at window edges

- cannot use autocorrelation formulation => this is a true cross correlation
- need to solve set of equations of the form

$$\sum_{k=1}^{p} \alpha_k \phi_n(i,k) = \phi_n(i,0), \quad i = 1,2,\dots,p, \; E_n = \phi_n(0,0) - \sum_{k=1}^{p} \alpha_k \, \phi_n(0,k)$$

$$\begin{bmatrix} \phi_n(1,1) & \phi_n(1,2) & . & . & \phi_n(1,p) \\ \phi_n(2,1) & \phi_n(2,2) & . & . & \phi_n(2,p) \\ . & . & . & . & . \\ . & . & . & . & . \\ \phi_n(p,1) & \phi_n(p,2) & . & . & \phi_n(p,p) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ . \\ \alpha_p \end{bmatrix} = \begin{bmatrix} \phi_n(1,0) \\ \phi_n(2,0) \\ . \\ . \\ \phi_n(p,0) \end{bmatrix}$$

- we have $\phi_n(i,k) = \phi_n(k,i)$ => symmetric but not Toeplitz matrix whose diagonal elements are related as

  $$\phi_n(i+1,k+1) = \phi_n(i,k) + s_n(-i-1)s_n(-k-1) - s_n(N-1-i)s_n(N-1-k)$$
  $$\phi_n(2,2) = \phi_n(1,1) + s_n(-2)s_n(-2) - s_n(N-2)s_n(N-2)$$
- all terms $\phi_n(i,k)$ have a fixed number of terms contributing to the computed values (N terms)
- $\phi_n(i,k)$ is a covariance matrix => solutions for $\{\alpha_k\}$ called Covariance Method

The equations

$$\begin{bmatrix} \phi_n(1,1) & \phi_n(1,2) & . & . & \phi_n(1,p) \\ \phi_n(2,1) & \phi_n(2,2) & . & . & \phi_n(2,p) \\ . & . & . & . & . \\ . & . & . & . & . \\ \phi_n(p,1) & \phi_n(p,2) & . & . & \phi_n(p,p) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ . \\ . \\ \alpha_p \end{bmatrix} = \begin{bmatrix} \phi_n(1,0) \\ \phi_n(2,0) \\ . \\ . \\ \phi_n(p,0) \end{bmatrix}$$

can be expressed as the flowing matrix equation

$$\mathbf{\Phi a} = \mathbf{\Psi}$$

where the matrix Φ is symmetric and positive definite, for which efficient methods are available, such as Cholesky decomposition.

3.5.3 The Levinson-Durbin Recursions

$$\begin{pmatrix} R_m[0] & R_m[1] & R_m[2] & \cdots & R_m[p-1] \\ R_m[1] & R_m[0] & R_m[1] & \cdots & R_m[p-2] \\ R_m[2] & R_m[1] & R_m[0] & \cdots & R_m[p-3] \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ R_m[p-1] & R_m[p-2] & R_m[p-3] & \cdots & R_m[0] \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \cdots \\ a_p \end{pmatrix} = \begin{pmatrix} R_m[1] \\ R_m[2] \\ R_m[3] \\ \cdots \\ R_m[p] \end{pmatrix}$$

The $p \times p$ matrix R_x of above equation (Yule-Walker equation is a Hermitian Toeplitx matrix, that is the elements of the matrix obey the property:

$$a_{ij} = a_{i+i,j+1} \quad i < p, j < p \text{ (Toeplitz property) and } a_{ij} = a_{ji}^{*} \text{ (Hermitian property)}$$

Solution of this equation can be carried by the usual matrix inversion . However the special Hermitian Toeplitz property of R_x can be exploited to yield a more efficient order-recursive solution based on the Levinson-Durbin recursions. For convenience, we omit the subscript m of the autocorrelation function. The proof is beyond the scope of our course.
The procedure of Levinson-Durbin recursions can be derived as followed steps:

$$E^{(0)} = R(0)$$

$$k_i = \left[R(i) - \sum_{j=1}^{i-1} \alpha_j^{i-1} R(i-j) \right] / E^{(i-1)} \qquad 1 \le i \le p$$

$$\alpha_i^{(i)} = k_i$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \qquad 1 \le j \le i-1$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)}$$

- solve equations recursively for $i = 1, 2, ..., p$ with the final solution

$$\alpha_j = \alpha_j^{(p)} \qquad 1 \le j \le p$$

- all predictors of orders $1, 2, 3, ..., p$-1 are found in Durbin's method, where $\alpha_j^{(i)}$ is the j^{th} predictor coefficient of an i^{th} order predictor

As an example,

- consider a simple $p = 2$ solution of the form

$$\begin{bmatrix} R(0) & R(1) \\ R(1) & R(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \end{bmatrix}$$

- with solution

$$E^{(0)} = R(0)$$

$$k_1 = R(1)/R(0)$$

$$\alpha_1^{(1)} = R(1)/R(0)$$

$$E^{(1)} = \frac{R^2(0) - R^2(1)}{R(0)}$$

$$k_2 = \frac{R(2)R(0) - R^2(1)}{R^2(0) - R^2(1)}$$

$$\alpha_2^{(2)} = \frac{R(2)R(0) - R^2(1)}{R^2(0) - R^2(1)}$$

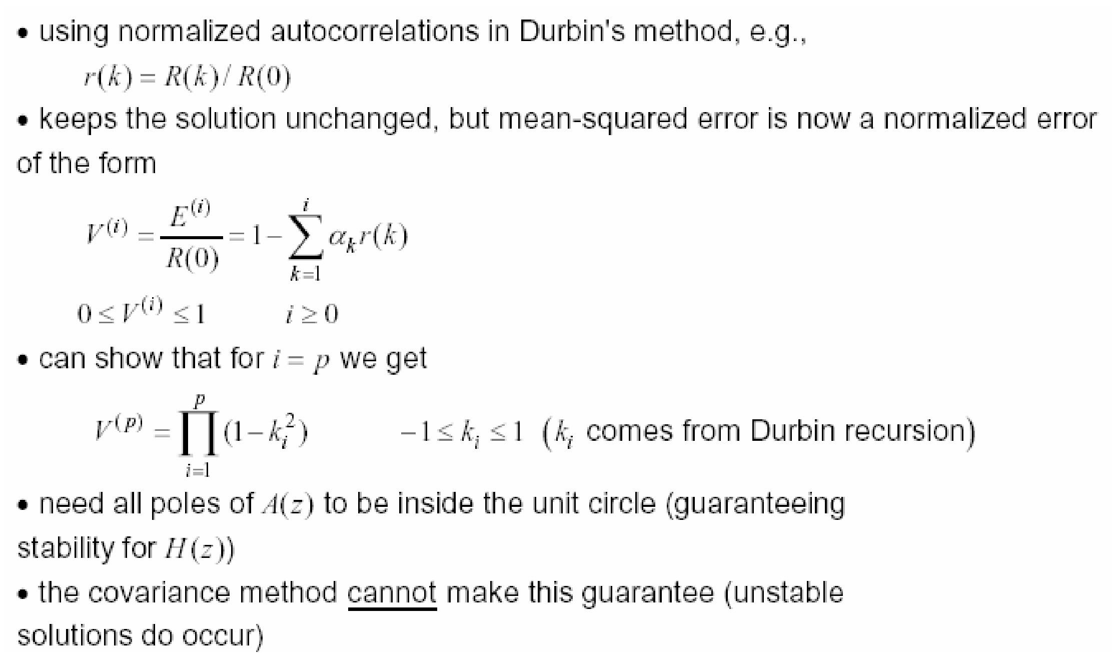
$$\alpha_1^{(2)} = \frac{R(1)R(0) - R(1)R(2)}{R^2(0) - R^2(1)}$$

- with final coefficients

$$\alpha_1 = \alpha_1^{(2)}$$

$$\alpha_2 = \alpha_2^{(2)}$$

$E^{(i)}$ = prediction error for predictor of order i



- using normalized autocorrelations in Durbin's method, e.g.,

  $r(k) = R(k)/R(0)$
- keeps the solution unchanged, but mean-squared error is now a normalized error of the form

$$V^{(i)} = \frac{E^{(i)}}{R(0)} = 1 - \sum_{k=1}^{i} \alpha_k r(k)$$

$$0 \le V^{(i)} \le 1 \qquad i \ge 0$$

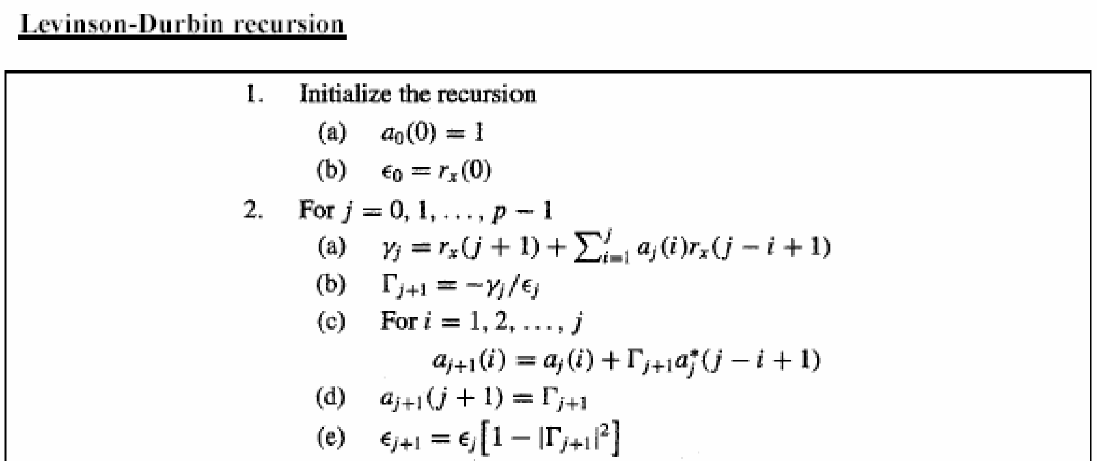
- can show that for $i = p$ we get

$$V^{(p)} = \prod_{i=1}^{p} (1 - k_i^2) \qquad -1 \le k_i \le 1 \ \ (k_i \text{ comes from Durbin recursion})$$

- need all poles of $A(z)$ to be inside the unit circle (guaranteeing stability for $H(z)$)
- the covariance method cannot make this guarantee (unstable solutions do occur)

We note that for the 0th order model we have $a_0(0) = 1$ and $\varepsilon_0 = R(0)$.

Then we can derive the 1st, 2nd, ..., pth order model solutions recursively, this defines the Levinson-Durbin recursion as detailed in below. Notice that we denote the normalized autocorrelation as $r_x(k)$.



Levinson-Durbin recursion

```
1.  Initialize the recursion
    (a)  $a_0(0) = 1$
    (b)  $\epsilon_0 = r_x(0)$
2.  For $j = 0, 1, \ldots, p-1$
    (a)  $\gamma_j = r_x(j+1) + \sum_{i=1}^{j} a_j(i) r_x(j-i+1)$
    (b)  $\Gamma_{j+1} = -\gamma_j / \epsilon_j$
    (c)  For $i = 1, 2, \ldots, j$
             $a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j-i+1)$
    (d)  $a_{j+1}(j+1) = \Gamma_{j+1}$
    (e)  $\epsilon_{j+1} = \epsilon_j \left[1 - |\Gamma_{j+1}|^2\right]$
```

Figure 1-1 The Levinson-Durbin Recursion ([1], Table 5.1, pg. 219)

The Matlab code for implementing the Levinson-Durbin Recursion is as below:

The Levinson-Durbin Recursion

```
function [a,epsilon]=rtoa(r)
%
   r=r(:);
   p=length(r)-1;
   a=1;
   epsilon=r(1);
   for j=2:p+1;
       gamma=-r(2:j)'*flipud(a)/epsilon;
       a=[a;0] + gamma*[0;conj(flipud(a))];
       epsilon=epsilon*(1 - abs(gamma)^2);
       end
```

Property 1

The *reflection co-efficients*, Γ_j, that are generated by the Levinson-Durbin recursion, assuming a valid autocorrelation sequence such that $|r_x(0)| \geq |r_x(1)|$, are bounded by one in magnitude, $|\Gamma_j| \leq 1$

The all-pole model using Prony's method is of the form:

$$H(z) = \frac{b(0)}{A_p(z)} = \frac{\sqrt{\varepsilon_p}}{1 + \sum_{k=1}^{p} a_p(k) z^{-k}}$$

Property 2

If and only if $|\Gamma_j| < 1$ for all j will all of the roots of $A_p(z)$ be inside the unit circle, that is the all-pole model will be stable. Furthermore, if $|\Gamma_j| \leq 1$ for all j then all of the roots of $A_p(z)$ will be either on or inside the unit circle.

Property 3

If $\mathbf{a}_p$ is the solution to $\mathbf{R}_p \mathbf{a}_p = \varepsilon_p \mathbf{u}_1$ where $\mathbf{R}_p$ is any $(p+1) \times (p+1)$ Hermitian Toeplitz matrix, then the all-pole model will be minimum-phase and stable if $\mathbf{R}_p$ is positive definite.

Property 4

All-pole modelling using the autocorrelation method ensures $\mathbf{R}_p$ is a positive definite Hermitian Toeplitz matrix and hence the model is guaranteed to be stable.

Example: Use the Levinson-Durbin recursion to find the 3rd order all-pole model for a signal having autocorrelation values:

$$r_x(0) = 1, \quad r_x(1) = 0.5, \quad r_x(2) = 0.5, \quad r_x(3) = 0.25 .$$

Solution:

$$H(z)=\frac{\sqrt{42}/8}{1-\frac{3}{8}z^{-1}-\frac{3}{8}z^{-2}+\frac{1}{8}z^{-3}}$$

- **Step-Up and Step-Down Recursions**

The Levinson-Durbin recursion may be viewed as the mapping:

$$\{r_x(0),r_x(1),\dots,r_x(p)\}\xrightarrow{Levinson-Durbin}\begin{cases}a_p(1),a_p(2),\dots,a_p(p)\\ \Gamma_1,\Gamma_2,\dots,\Gamma_p,\varepsilon_p\end{cases}$$

That is, we can obtain the all-pole filter or forward linear predictor coefficients, $a_p(k)$, and the reflection coefficients, Γ_k , <u>given</u> the autocorrelation sequence, rx(k) . Other recursions can be derived from the Levinson-Durbin as follows.

Step-Up Recursion:

The Step-Up Recursion can be viewed as the mapping:

$$\{\Gamma_1,\Gamma_2,\dots,\Gamma_p\}\xrightarrow{Step-Up}\{a_p(1),a_p(2),\dots,a_p(p)\}$$

That is, we can obtain the all-pole filter or forward linear predictor coefficients, $a_p(k)$,directly <u>given</u> the reflection coefficients, Γ_k .

$$a_{j+1}(i)=a_j(i)+\Gamma_{j+1}a_j^*(j-i+1)\qquad i=0,1,\dots,j+1$$

we obtain the required recursion:

```
1. Initialize the recursion: $a_0(0)=1$
2. For $j=0,1\dots,p-1$
   (a) For $i=1,2,,\dots,j$
       $a_{j+1}(i)=a_j(i)+\Gamma_{j+1}a_j^*(j-i+1)$
   (b) $a_{j+1}(j+1)=\Gamma_{j+1}$
```

Figure 1-3 Step-Up Recursion ([1], Table 5.2, pg. 233)

The Step-Up Recursion

```
function a=gtoa(gamma)
%
   a=1;
   gamma=gamma(:);
   p=length(gamma);
   for j=2:p+1;
      a=[a;0] + gamma(j-1)*[0;conj(flipud(a))];
      end
```

MATLAB code for implementing the Step-Up Recursion

Step-Down (backward Levinson-Durbin) Recursion:

The Step-Down Recursion can be viewed as the mapping:

$$\{a_p(1), a_p(2), \ldots, a_p(p)\} \xrightarrow{Step-Down} \{\Gamma_1, \Gamma_2, \ldots, \Gamma_p\}$$

That is, we can obtain the reflection coefficients, Γ_k, directly <u>given</u> the all-pole filter or forward linear predictor coefficients, $a_p(k)$.

From above, if we have the jth order coefficients, $a_j(k)$, then we can obtain the jth reflection co-efficient by simply:

$$\Gamma_j = a_j(j)$$

So what remains to be done is to derive the jth order coefficients, $a_j(k)$,) recursively for j=p-1 , p-2, … , 1 <u>given</u> the pth order coefficients, that is we need a recursion to derive the jth order coefficients given the (j+1)th order coefficients. To do this we consider:

$$a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j-i+1)$$

Let $i \equiv j - i + 1$ and then take the complex conjugate of the expression to yield:

$$a_{j+1}^*(j-i+1) = a_j^*(j-i+1) + \Gamma_{j+1}^* a_j(i)$$

We can express these two equations in matrix form as:

$$\tilde{\mathbf{a}}_{j+1}(i) = \begin{bmatrix} a_{j+1}(i) \\ a_{j+1}^*(j-i+1) \end{bmatrix} = \begin{bmatrix} 1 & \Gamma_{j+1} \\ \Gamma_{j+1}^* & 1 \end{bmatrix} \begin{bmatrix} a_j(i) \\ a_j^*(j-i+1) \end{bmatrix} = \begin{bmatrix} 1 & \Gamma_{j+1} \\ \Gamma_{j+1}^* & 1 \end{bmatrix} \tilde{\mathbf{a}}_j(i)$$

from which we can solve for the jth order co-efficient values given the (j+1)th order values:

$$\tilde{\mathbf{a}}_j(i) = \begin{bmatrix} 1 & \Gamma_{j+1} \\ \Gamma_{j+1}^* & 1 \end{bmatrix}^{-1} \tilde{\mathbf{a}}_{j+1}(i)$$

and the solution for $a_j(i)$ is:

$$a_j(i) = \frac{1}{1 - |\Gamma_{j+1}|^2} \left[a_{j+1}(i) - \Gamma_{j+1} a_{j+1}^*(j-i+1) \right]$$

from which we obtain the required recursion:

```
1. Set $\Gamma_p = a_p(p)$
2. For $j = p-1, p-2, \ldots, 1$
   (a) For $i = 1, 2, \ldots, j$
         $a_j(i) = \frac{1}{1 - |\Gamma_{j+1}|^2} \left[ a_{j+1}(i) - \Gamma_{j+1} a_{j+1}^*(j-i+1) \right]$
   (b) Set $\Gamma_j = a_j(j)$
   (c) If $|\Gamma_j| = 1$, Quit.
```

Figure 1-5 Step-Down Recursion ([1], Table 5.3, pg. 236)

For <u>Example</u>: verify that if we are given $H(z) = 1 + 0.5z^{-1} - 0.1z^{-2} - 0.5z^{-3}$ then

the reflection coefficients are $\Gamma = [0.5, 0.2, -0.5]^T$

MATLAB code for implementing the Step-Down Recursion is as follows:

The Step-Down Recursion

```
function gamma=atog(a)
%
   a=a(:);
   p=length(a);
   a=a(2:p)/a(1);
   gamma(p-1)=a(p-1);
   for j=p-1:-1:2;
      a=(a(1:j-1) - gamma(j)*flipud(conj(a(1:j-1)))./ ...
        (1 - abs(gamma(j))^2);
      gamma(j-1)=a(j-1);
      end
```

Schur-Cohn stability test:

Assume a causal, linear shift-invariant filter with rational function:

$$H(z)=\frac{B(z)}{A(z)}$$

From Property 2 we know that if $\left|\Gamma_j\right|<1$ for all j, where the Γ_j are the reflection co-efficients associated with the filter co-efficients of $A(z)$, then the roots of $A(z)$ (i.e. the poles of $H(z)$) will lie within the unit circle. That is the filter $H(z)$ is stable. The *Schur-Cohn stability test* consists of running the step-down recursion to derive the Γ_i from the filter co-efficients of $A(z)$ and checking whether any of the $\left|\Gamma_j\right|\geq 1$, if they are then the filter is unstable.

<u>Example</u>: use the Schur-Cohn stability test to check the stability of the filter:

$$H(z)=\frac{1}{2+4z^{-1}-3z^{-2}+z^{-3}}$$

- **Inverse Levinson-Durbin Recursion**

The *Inverse Levinson-Durbin* recursion can be viewed as the mapping:

$$\left\{\Gamma_1,\Gamma_2,\ldots,\Gamma_p,\varepsilon_p\right\}\xrightarrow{\textit{Inverse Levinson-Durbin}}\left\{r_x(0),r_x(1),\ldots,r_x(p)\right\}$$

That is, we can obtain the autocorrelation sequence, $r_x(k)$, directly <u>given</u> the reflection co-efficients, Γ_k, and ε_p.

The recursion from

$$\varepsilon_{j+1}=\varepsilon_j+\Gamma_{j+1}\gamma_j^*=\varepsilon_j[1-|\Gamma_{j+1}|^2]$$

can be expressed as:

$$\varepsilon_{j+1}=\varepsilon_j[1-|\Gamma_i|^2]=r_x(0)\prod_{i=1}^{j+1}(1-|\Gamma_i|^2)$$

Thus given ε_p and the Γ_i for $i = 1,2,\ldots,p$ we can derive:

$$r_x(0) = \frac{\varepsilon_p}{\prod_{i=1}^{p}(1-|\Gamma_i|^2)}$$

and together with $a_0(0) = 1$ this initialises the recursion. Using the Step-Up recursion the all-pole filter co-efficients, $a_j(k)$ are derived and from these we note that the matrix equation of

$$\begin{pmatrix} R_m[0] & R_m[1] & R_m[2] & \cdots & R_m[p-1] \\ R_m[1] & R_m[0] & R_m[1] & \cdots & R_m[p-2] \\ R_m[2] & R_m[1] & R_m[0] & \cdots & R_m[p-3] \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ R_m[p-1] & R_m[p-2] & R_m[p-3] & \cdots & R_m[0] \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \cdots \\ a_p \end{pmatrix} = \begin{pmatrix} R_m[1] \\ R_m[2] \\ R_m[3] \\ \cdots \\ R_m[p] \end{pmatrix}$$

can be rewritten in the form:

$$r_x(k) + \sum_{i=1}^{p} a_p(i) r_x(k-i) = 0 \quad \text{for} \quad k = 1,2,\ldots,p$$

By setting $k \equiv j+1$ and $p \equiv j+1$ we obtain an expression for $r(j+1)$ based on the previous (calculated) values $\{r_x(0), r_x(1), \ldots r_x(j)\}$:

$$r_x(j+1) = -\sum_{i=1}^{j+1} a_{j+1}(i) r_x(j+1-i)$$

and we obtain the required recursion:

```
1.  Initialize the recursion
      (a)  $r_x(0) = \epsilon_p / \prod_{i=1}^{p}(1 - |\Gamma_i|^2)$
      (b)  $a_0(0) = 1$
2.  For $j = 0, 1, \ldots, p-1$
      (a)  For $i = 1, 2, \ldots, j$
             $a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j - i + 1)$
      (b)  $a_{j+1}(j+1) = \Gamma_{j+1}$
      (c)  $r_x(j+1) = -\sum_{i=1}^{j+1} a_{j+1}(i) r_x(j + 1 - i)$
```

Figure 1-7 Inverse Levinson-Durbin Recursion ([1], Table 5.4, pg. 239)

MATLAB code for implementing the Inverse Levinson-Durbin Recursion is as follows:

The Inverse Levinson-Durbin Recursions

```
function r=gtor(gamma,epsilon)
%
   p=length(gamma);
   aa=gamma(1);
   r=[1 -gamma(1)];
   for j=2:p;
      aa=[aa;0]+gamma(j)*[conj(flipud(aa));1];
      r=[r -fliplr(r)*aa];
      end;
   if nargin == 2,
   r = r*epsilon/prod(1-abs(gamma).^2);
   end;
```

For <u>Example</u>: to derive the autocorrelation sequence, $r_x(k)$, for k=0,1,2,3 <u>given</u> the reflection coefficients $\Gamma = [0.5,0.5,0.5]^T$ and $\varepsilon_3 = 2(0.75)^3$

<u>Solution:</u>

$$\mathbf{r}_x = [2 \quad -1 \quad -1/4 \quad 1/8]^T$$

- **Schur Recursion**

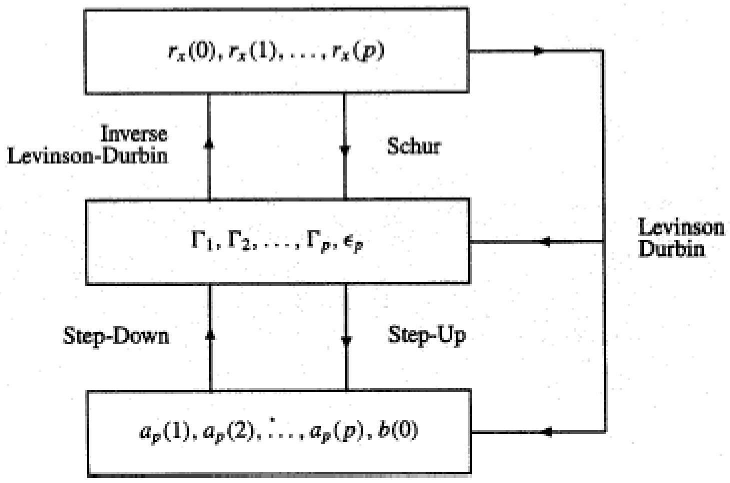
The Schur recursion can be viewed as the mapping:

$$\{r_x(0), r_x(1), \ldots, r_x(p)\} \xrightarrow{Schur} \{\Gamma_1, \Gamma_2, \ldots, \Gamma_p, \varepsilon_p\}$$

The recursion is simply stated as shown below

```
1.  Set $g_0(k) = g_0^R(k) = r_x(k)$ for $k = 0, 1, \ldots, p$.
2.  For $j = 0, 1, \ldots, p-1$
    (a)  Set $\Gamma_{j+1} = -g_j(j+1)/g_j^R(j)$
    (b)  For $k = j+2, \ldots, p$
             $g_{j+1}(k) = g_j(k) + \Gamma_{j+1} g_j^R(k-1)$
    (c)  For $k = j+1, \ldots, p$
             $g_{j+1}^R(k) = g_j^R(k-1) + \Gamma_{j+1}^* g_j(k)$
3.  $\epsilon_p = g_p^R(p)$
```

- **Relationship between Levinson-Durbin Recursions**


$r_x(0), r_x(1), \ldots, r_x(p)$
Inverse Levinson-Durbin
Schur
$\Gamma_1, \Gamma_2, \ldots, \Gamma_p, \epsilon_p$
Levinson Durbin
Step-Down
Step-Up
$a_p(1), a_p(2), \ldots, a_p(p), b(0)$

3.5.4 Lattice Formulations of LPC

Lattice filters are popular alternative implementation structures to the standard direct form realisations for FIR and IIR digital filters. Interestingly their derivation is tied directly to all-pole modeling and the Levinson-Durbin recursion.

Let $\hat{x}(n)$ be an estimate of $x(n)$ based on the past p samples $\{x(n-1), x(n-2), \ldots, x(n-p)\}$. In pth order *forward linear prediction* we form this estimate by:

$$\hat{x}(n) = -\sum_{k=1}^{p} a_p(k)x(n-k)$$

and the pth order *forward prediction error* is given by:

$$e_p^+(n) = x(n) - \hat{x}(n) = x(n) + \sum_{k=1}^{p} a_p(k)x(n-k) = \sum_{k=0}^{p} a_p(k)x(n-k)$$

Equation 2.1

where $a_p(0) = 1$. The + notation is necessary to differentiate the forward prediction error from the backward prediction which we will later define as $e_p^-(n)$.

The co-efficients, $a_p(k)$, are obtained as a consequence of minimising the squared error:

$$\varepsilon_p^+ = \sum_{n=0}^{\infty} \left| e_p^+(n) \right|^2$$

Taking the z-transform of Equation 2.1 we obtain:

$$E_p^+(z) = A_p(z)X(z)$$

Equation 2.2

where:

$$A_p(z) = 1 + \sum_{k=1}^{p} a_p(k)z^{-k}$$

is the *forward prediction error filter*.

If we let $Y(z) \equiv E_p^+(z)$ (i.e. $y(n) \equiv e_p^+(n)$) then $H(z) = A_p(z)$ is an FIR filter. Noting that the co-efficients, $a_p(k)$, can be derived from the Levinson-Durbin recursion we can develop a lattice structure to implement $H(z) = A_p(z)$ as follows:

From Chapter 1 we had:

$$a_{j+1}(i) = a_j(i) + \Gamma_{j+1} a_j^*(j-i+1)$$

Equation 2.3

Taking the z-transform of Equation 2.3 and remembering the $Z\{x(n-k)\} = z^{-k}X(z)$ and $Z\{x(-n)\} = X(1/z)$ z-transform properties we have:

$$A_{j+1}(z) = A_j(z) + \Gamma_{j+1}[z^{-(j+1)} A_j^*(1/z^*)]$$

Equation 2.4

Multiplying both sides of Equation 2.4 by $X(z)$ and using Equation 2.2:

$$E_j^+(z) = A_j(z)X(z)$$

Equation 2.5

we get:

$$E_{j+1}^+(z) = E_j^+(z) + z^{-1}\Gamma_{j+1} E_j^-(z)$$

Equation 2.6

where:

$$E_j^-(z) = z^{-j}X(z)A_j^*(1/z^*)$$

Equation 2.7

Let $a_j^R(i) = a_j^*(j-i)$ be the complex conjugate of the reversed $a_j(i)$. Taking z-transforms we obtain $A_j^R(z) = z^{-j}A_j^*(1/z^*)$ and hence we can also express Equation 2.7 in the form:

$$E_j^-(z) = A_j^R(z)X(z)$$

Equation 2.8

Taking the inverse z-transform of Equation 2.6 yields the **first** of our **important** relations for the lattice filter:

$$e_{j+1}^+(n) = e_j^+(n) + \Gamma_{j+1}e_j^-(n-1)$$

Equation 2.9

Now take the complex conjugate of Equation 2.3 and substitute $i \equiv j-i+1$ to yield:

$$a_{j+1}^*(j-i+1) = a_j^*(j-i+1) + \Gamma_{j+1}^*a_j(i)$$

Equation 2.10

Taking the z-transform of Equation 2.10 gives:

$$z^{-(j+1)}A_{j+1}^*(1/z^*) = z^{-(j+1)}A_j^*(1/z^*) + \Gamma_{j+1}^*A_j(z)$$

Equation 2.11

Multiplying both sides of Equation 2.11 by $X(z)$ and using Equation 2.5 and Equation 2.7 yields:

$$E_{j+1}^-(z) = z^{-1}E_j^-(z) + \Gamma_{j+1}^*E_j^+(z)$$

Equation 2.12

Taking the inverse z-transform of Equation 2.12 yields the **second** of our **important** relations for the lattice filter:

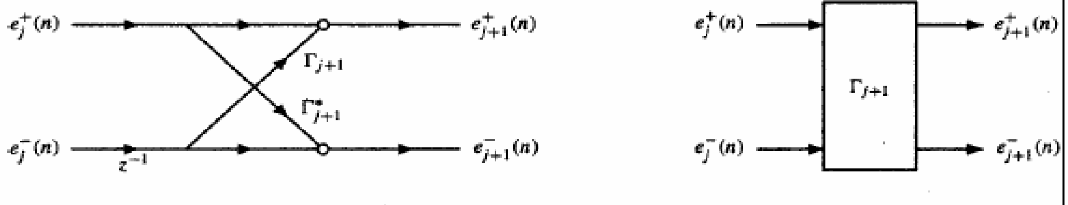
$$e_{j+1}^-(n) = e_j^-(n-1) + \Gamma_{j+1}^*e_j^+(n)$$

Equation 2.13

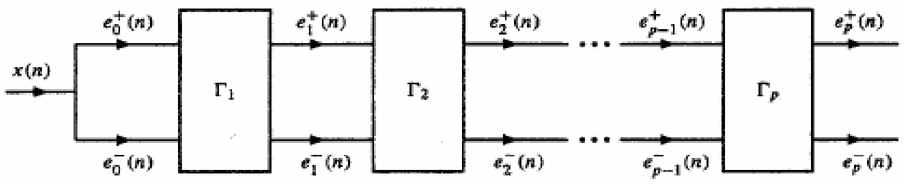
Equation 2.9 and Equation 2.13 represent a pair of coupled difference equations that correspond to the two-port network shown in Figure 2-1(a). And with a cascade of two-port networks having reflection co-efficients Γ_j we have the pth order FIR lattice filter as shown in Figure 2-1(b) noting that:

$$e_0^+(n) = e_0^-(n) = x(n)$$

Equation 2.14



(a) Single stage of an FIR lattice filter.

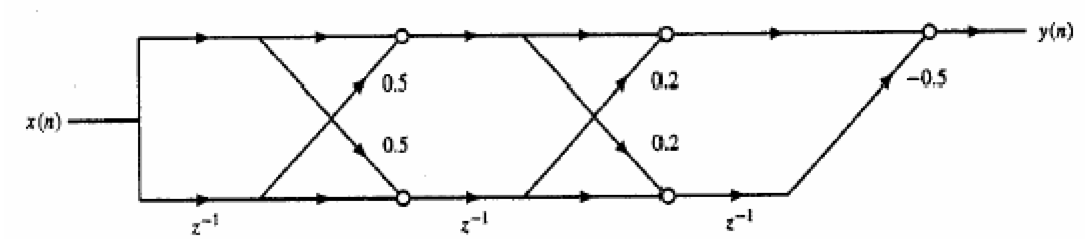


(b) A pth-order FIR lattice filter.

For Example: we want to derive the FIR lattice filter structure for the 3rd order FIR filter:

$$H(z) = 1 + 0.5z^{-1} - 0.1z^{-2} - 0.5z^{-3}$$

Solution: We use the step-down recursion on $a_3 = [1, 0.5, -0.1, -0.5]^T$ to derive the reflection coefficients $\Gamma = [0.5, 0.2, -0.5]^T$ and hence the FIR lattice filter



Advantages of FIR lattice filter implementations over direct form

- modularity of the filter, increase or decrease the order of the filter by simply adding or removing stages from the lattice (in direct form all of the filter coefficients would have to be recalculated)
- easy to confirm that the filter is minimum-phase by simply noting that $|\Gamma_j| < 1$ (i.e. lattice gain parameters are all less than unity)
- Less sensitive to parameter (co-efficient) quantisation effects

From Equation 2.2 and Equation 2.14 we can state that $E_p^+(z) = A_p(z)E_0^+(z)$ and hence the all-pole filter is given by:

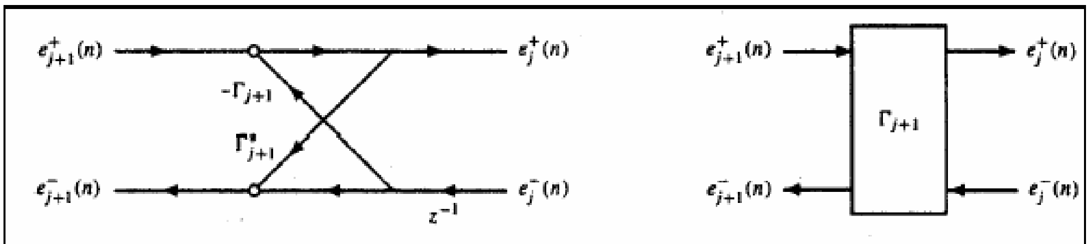
$$H(z) = \frac{1}{A_p(z)} = \frac{E_0^+(z)}{E_p^+(z)} \text{ or } E_0^+(z) = H(z)E_p^+(z)$$

That is, the all-pole filter would produce a response of $y(n) \equiv e_0^+(n)$ to an input of $x(n) \equiv e_p^+(n)$. To derive the required lattice structure we rewrite Equation 2.9 and together with Equation 2.13 we get:

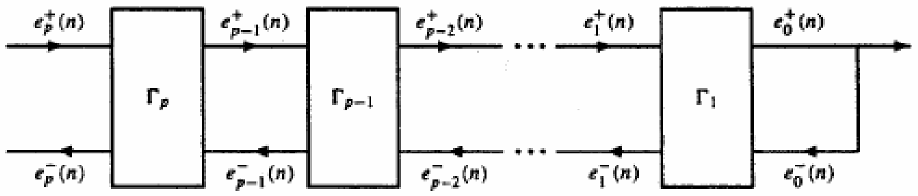
$$e_j^+(n) = e_{j+1}^+(n) - \Gamma_{j+1}e_j^-(n-1)$$
$$e_{j+1}^-(n) = e_j^-(n-1) + \Gamma_{j+1}^*e_j^+(n)$$

Equation 2.19

Equation 2.19 represents a pair of coupled equations which can be represented by the two-port network of Figure 2-4(a). With a cascade of *p* such stages we have the *p*th order all-pole lattice filter shown in Figure 2-4(b) noting that since $e_0^+(n) = e_0^-(n) = x(n)$ we feed the output of the last module, $e_0^+(n)$, back into its lower input, $e_0^-(n)$.



(a) Single stage of an all-pole lattice filter.



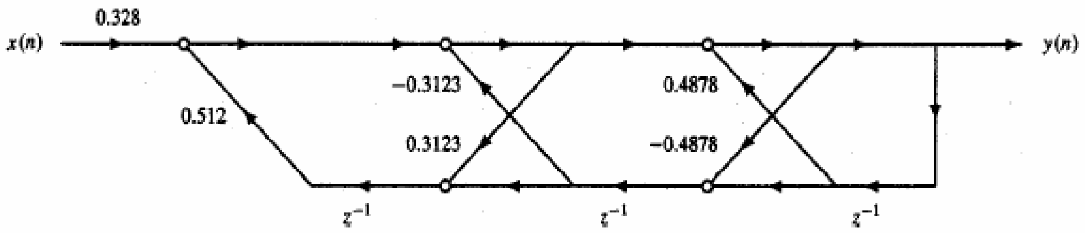
(b) A pth-order all-pole lattice filter and a pth-order allpass filter that is formed by cascading p modules of the form given in (a).

For Example: we want to implement the following all-pole filter as a lattice structure:

$$H(z) = \frac{0.328}{1 - 0.8z^{-1} + 0.64z^{-2} - 0.512z^{-3}}$$

Solution:

We use the step-down recursion on $\mathbf{a}_3 = [1 \quad -0.8 \quad 0.64 \quad -0.512]^T$ to derive the reflection co-efficients $\mathbf{\Gamma} = [-0.4878 \quad 0.3123 \quad -0.512]^T$ and hence the all-pole IIR lattice filter where $x(n) \equiv e^+_p(n)$ and $y(n) \equiv e^+_0(n)$:



3.5.5 Burg's Method

In signal modeling with finite data records an error expression is formulated, in terms of e(n) , and is minimized with respect to the filter coefficients $a_p(k)$. Two different approaches to this are the autocorrelation method and covariance method. Using the lattice filter derivations we can develop alternative lattice methods for signal modeling based on minimization of expressions involving $e^+_j(n)$ and $e^+_j(n)$, but with respect to the reflection coefficients, Γ_j :

Forward covariance method

Minimise $$\varepsilon_j^+ = \sum_{n=j}^{N} \left| e_j^+(n) \right|^2$$

Backward covariance method

Minimise $$\varepsilon_j^- = \sum_{n=j}^{N} \left| e_j^-(n) \right|^2$$

Burg's method

Minimise $$\varepsilon_j^B = \varepsilon_j^+ + \varepsilon_j^- = \sum_{n=j}^{N} \left| e_j^+(n) \right|^2 + \sum_{n=j}^{N} \left| e_j^-(n) \right|^2$$

Equation 2.28

Both the forward and backward covariance methods are related to the standard covariance method and exhibit the same property that unstable models may be produced.

Of particular interest is Burg's method. To derive the subsequent expressions for the reflection co-efficients, Γ_j^B, that minimise Equation 2.28 we form the partial derivative of Equation 2.28 with respect to Γ_j^B and set this to zero:

$$\frac{\partial}{\partial(\Gamma_j^B)^*}\varepsilon_j^B = \sum_{n=j}^{N} \{ e_j^+(n)[e_{j-1}^-(n-1)]^* + [e_j^-(n)]^* e_{j-1}^+(n) \} = 0$$

Equation 2.29

where we have used Equation 2.9 and Equation 2.13 and using these in Equation 2.29 to solve for the Γ_j^B that minimises Equation 2.28 we get:

$$\boxed{\Gamma_j^B = -\frac{2\sum_{n=j}^{N} e_{j-1}^+(n)[e_{j-1}^-(n-1)]^*}{\sum_{n=j}^{N} \{ \left| e_{j-1}^+(n) \right|^2 + \left| e_{j-1}^-(n-1) \right|^2 \}} = -\frac{2\langle \mathbf{e}_{j-1}^+, \mathbf{e}_{j-1}^- \rangle}{\left\| \mathbf{e}_{j-1}^+ \right\|^2 + \left\| \mathbf{e}_{j-1}^- \right\|^2}}$$

Equation 2.30

where:

$$\mathbf{e}_{j-1}^+ = \left[e_{j-1}^+(j) \quad e_{j-1}^+(j+1) \quad \cdots \quad e_{j-1}^+(N) \right]^T \text{ and } \mathbf{e}_{j-1}^- = \left[e_{j-1}^-(j-1) \quad e_{j-1}^-(j) \quad \cdots \quad e_{j-1}^-(N-1) \right]^T$$

- **Burg's method yields stable models:**

For any two vectors, a and b, the Cauchy-Schwarz inequality states:

$$2\left|\langle \mathbf{a}, \mathbf{b} \rangle\right| \le \left\| \mathbf{a} \right\|^2 + \left\| \mathbf{b} \right\|^2$$

- **Burg's recursion**

From Equation 2.30 given the j-1 reflection coefficients, $\{\Gamma_1^B, \Gamma_2^B, \cdots, \Gamma_{j-1}^B\}$ we can use the pth order FIR lattice filter to compute the j-1 backward and forward prediction errors given the N data inputs $\{x(n)\}_{n=0}^{N}$ and then calculate the jth reflection coefficient Γ^B_j using Equation 2.30. This yields a recursive algorithm for j=1,2,⋯p.

The main computational requirements in the Burg recursion just described is the evaluation of the inner product and norms in Equation 2.30. We can derive a recursion for the denominator, D$_j$, as follows:

$$D_j = \sum_{n=i}^{N} \{ \left| e_{j-1}^{+}(n) \right|^2 + \left| e_{j-1}^{-}(n-1) \right|^2 \} = \varepsilon_{j-1}^{B} - \left| e_{j-1}^{+}(j-1) \right|^2 - \left| e_{j-1}^{-}(N) \right|^2$$

where from Equation 2.28 we have

$$\varepsilon_{j-1}^{B} = \sum_{n=j-1}^{N} \{ \left| e_{j-1}^{+}(n) \right|^2 + \left| e_{j-1}^{-}(n) \right|^2 \}$$

It can be shown that

$$\varepsilon_j^{B} = D_j \left[1 - \left| \Gamma_j^{B} \right|^2 \right] \qquad \textit{Equation 2.32}$$

Since:

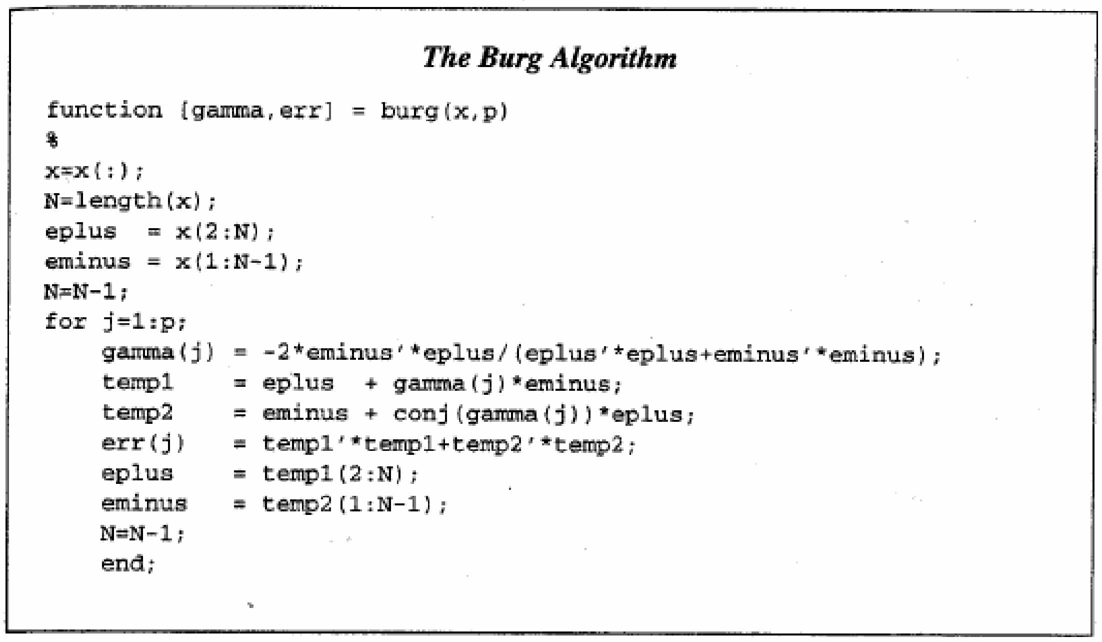
$$D_{j+1} = \varepsilon_j^{B} - \left| e_j^{+}(j) \right|^2 - \left| e_j^{-}(N) \right|^2$$

Then using Equation 2.32 we obtain the desired recursion:

$$D_{j+1} = D_j \left[1 - \left| \Gamma_j^{B} \right|^2 \right] - \left| e_j^{+}(j) \right|^2 - \left| e_j^{-}(N) \right|^2$$

The Burg recursion is summarized and the MATLAB code implementation is given by :

```
1. Initialize the recursion
   (a) $e_0^{+}(n) = e_0^{-}(n) = x(n)$
   (b) $D_1 = \sum_{n=1}^{N} \left\{ |x(n)|^2 + |x(n-1)|^2 \right\}$
2. For $j = 1$ to $p$
   a) $\Gamma_j^{B} = -\frac{2}{D_j} \sum_{n=j}^{N} e_{j-1}^{+}(n) \left[ e_{j-1}^{-}(n-1) \right]^{*}$
   b) For $n = j$ to $N$
          $e_j^{+}(n) = e_{j-1}^{+}(n) + \Gamma_j^{B} e_{j-1}^{-}(n-1)$
          $e_j^{-}(n) = e_{j-1}^{-}(n-1) + (\Gamma_j^{B})^{*} e_{j-1}^{+}(n)$
   c) $D_{j+1} = D_j \left( 1 - |\Gamma_j^{B}|^2 \right) - |e_j^{+}(j)|^2 - |e_j^{-}(N)|^2$
   d) $\mathcal{E}_j^{B} = D_j \left[ 1 - |\Gamma_j^{B}|^2 \right]$
```



The Burg Algorithm

```
function [gamma,err] = burg(x,p)
%
x=x(:);
N=length(x);
eplus  = x(2:N);
eminus = x(1:N-1);
N=N-1;
for j=1:p;
    gamma(j)  = -2*eminus'*eplus/(eplus'*eplus+eminus'*eminus);
    temp1     = eplus  + gamma(j)*eminus;
    temp2     = eminus + conj(gamma(j))*eplus;
    err(j)    = temp1'*temp1+temp2'*temp2;
    eplus     = temp1(2:N);
    eminus    = temp2(1:N-1);
    N=N-1;
    end;
```

Example: we want to investigate the Burg recursion for the signal, x (n) , generated as the unit sample response (impulse response sequence) of the third-order system:

$$H(z) = \frac{1}{1 - 0.12z^{-1} - 0.456z^{-2} + 0.6z^{-3}}$$

Using the step-down recursion we obtain the lattice filter coefficients:

$$\mathbf{\Gamma} = [0.6 \quad -0.6 \quad 0.6]^T$$

Now using the MATLAB program for the Burg recursion with p=3 and N= 60 samples of x(n) , the reflection coefficients are estimated to be:

$$\mathbf{\Gamma}^B = [0.6753 \quad -0.6653 \quad 0.8920]^T$$

which corresponds (by using the step-up recursion) to a filter having system function:

$$\hat{H}(z) = \frac{1}{1 - 0.3675z^{-1} - 0.4637z^{-2} + 0.8920z^{-3}}$$

with the following sequence of squared errors:

$$\underline{\varepsilon}^B = [3.4371 \quad 1.5627 \quad 0.3134]^T$$

Again, if we increase the model order to p=5, the next two reflection coefficients are:

$$\Gamma_4^B = -0.5324 \,;\, \Gamma_5^B = 0.0390$$

and the corresponding errors are:

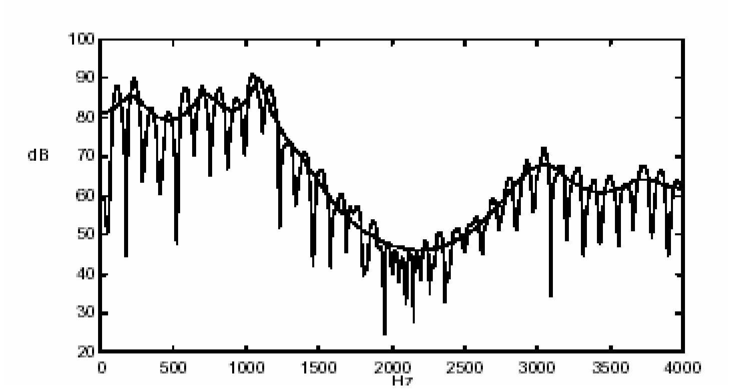
$$\varepsilon_4^B = 0.2027 \,;\, \varepsilon_5^B = 0.2022$$

3.5.6 Spectral Analysis via LPC

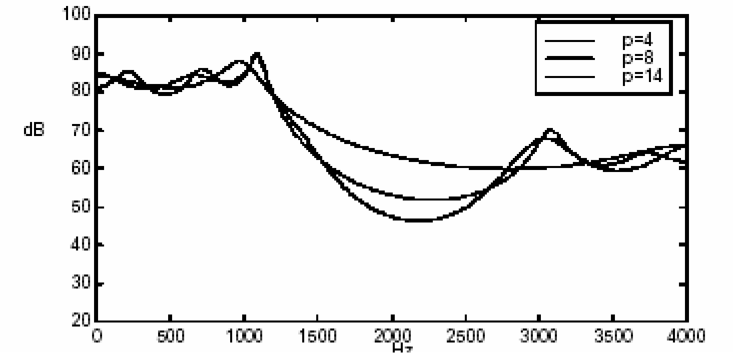
Let's now analyze the frequency-domain behavior of the LPC analysis by evaluating

$$H(e^{j\omega}) = \frac{G}{1-\sum_{k=1}^{p} a_k e^{-j\omega k}} = \frac{G}{A(e^{j\omega})}$$

which is an all-pole or IIR filter. If we plot $H(e^{j\varpi})$, we expect to see peaks at the roots of the denominator. Figure below shows the 14-order LPC spectrum of the vowel /ah/.



Used here are a 30-ms Hamming window and the autocorrelation method with p = 14. The short-time spectrum is also shown.
Even nasals, which have zeros in addition to poles, can be represented with an infinite number of poles. In practice, if p is large enough we can approximate the signal spectrum with arbitrarily small error. Figure below shows different fits for different values of p. The higher p, the more details of the spectrum are preserved.

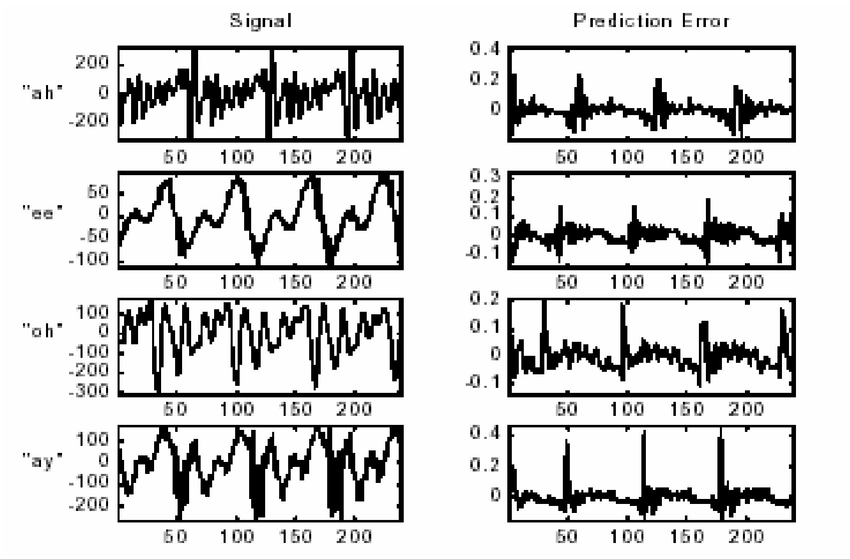


The prediction order is not known for arbitrary speech, so we need to set it to balance spectral detail with estimation errors.
How do we choose p? This is an important design question. Larger values of p lead to lower prediction errors. Unvoiced speech has higher error than voiced speech, because the LPC model is more accurate for voiced speech. In general, the normalized error rapidly decreases, and

then converges to a value of around 12 - 14 for 8 kHz speech. If we use a large value of p, we are fitting the individual harmonics; thus the LPC filter is modeling the source, and the separation between source and filter is not going to be so good. The more coefficients we have to estimate, the larger the variance of their estimates, since the number of available samples is the same.

3.5.6 The Prediction Error

So far, we have concentrated on the filter component of the source-filter model. We can compute the prediction error signal, also called the excitation, or residual signal. For unvoiced speech synthetically generated by white noise following an LPC filter we expect the residual to be approximately white noise. In practice, this approximation is quite good, and replacement of the residual by white noise followed by the LPC filter typically results in no audible difference. For voiced speech synthetically generated by an impulse train following an LPC filter, we expect the residual to approximate an impulse train. In practice, this is not the case, because the all-pole assumption is not altogether valid; thus, the residual, although it contains spikes, is far from an impulse train. Replacing the residual by an impulse train, followed by the LPC filter, results in speech that sounds somewhat robotic, partly because real speech is not perfectly periodic (it has a random component as well), and because the zeroes are not modeled with the LPC filter. Residual signals computed from inverse LPC filters for several vowels are shown in Figure below.



3.5.7 Equivalent Representations

There are a number of alternate useful representations of the predictor coefficients. The most important are the line spectrum pairs, reflection coefficients, log-area ratios, LPC cepstrum, and the roots of the predictor polynomial.

- **Log-area ratio coefficients can be drawn from reflection coefficients**

$$g_i = \log\left[\frac{A_{i+1}}{A_i}\right] = \log\left[\frac{1-k_i}{1+k_i}\right] \quad 1 \le i \le p$$

or with the inverse form

$$k_i = \frac{1-e^{g_i}}{1+e^{g_i}} \quad 1 \le i \le p$$

The log-area ratio coefficients are equal to the natural logarithm of the ratio of the areas of adjacent sections of a lossless tube equivalent of the vocal tract having the same transfer function. Since for stable predictor filters $1< k_i <1$, we have $\infty<g_i< \infty$. For speech signals, it is not uncommon to have some reflection coefficients close to 1, and quantization of those values can cause a large change in the predictor's transfer function. On the other hand, the log-area ratio coefficients have relatively flat spectral sensitivity (i.e., a small change in their values causes a small change in the transfer function) and thus are useful in coding.

- **line spectrum pairs**

• from the Levinson recursion we have

$$A(z) = A^{(p)}(z) = A^{(p-1)}(z) - k_p z^{-p} A^{(p-1)}(z^{-1})$$

• the LSP polynomials are defined as

$$P(z) = A(z) + z^{-(p+1)} A(z^{-1})$$

$$Q(z) = A(z) - z^{-(p+1)} A(z^{-1})$$

from which it follows that

$$A(z) = \frac{P(z)+Q(z)}{2}$$

- properties of LSP parameters
  1. $P(z)$ corresponds to a lossless tube, open at the lips and open ($k_{p+1} = 1$) at the glottis
  2. $Q(z)$ corresponds to a lossless tube, open at the lips and closed ($k_{p+1} = -1$) at the glottis
  3. all the roots of $P(z)$ and $Q(z)$ are on the unit circle
  4. if p is an even integer, then $P(z)$ has a root at $z = +1$ and $Q(z)$ has a root at $z = -1$
  5. a necessary and sufficient condition for $|k_i| < 1,\ i = 1, 2, \ldots, p$ is that the roots of $P(z)$ and $Q(z)$ alternate on the unit circle
  6. the LSP frequencies get close together when roots of $A(z)$ are close to the unit circle
  7. the roots of $P(z)$ are approximately equal to the formant frequencies