Lecture Notes in Speech Production, Speech Coding, and Speech Recognition

Mark Hasegawa-Johnson
University of Illinois at Urbana-Champaign

February 17, 2000

Chapter 7

Speech Coding

7.1 What is Speech Coding?

"Speech coding" = finding a representation of speech which can be transmitted efficiently through a digital channel.

... usually *lossy* coding, meaning that the waveform can not be completely reproduced by the decoder – instead, only the information which is useful to a human listener is retained.

- Advantages:

  1. Nearly eliminates "generation loss" at repeaters.
  2. Digital encryption stronger than analog "scrambling."
  3. Multiplex several signals on one channel.
  4. Engineering tradeoffs (e.g. quality vs. bandwidth) are explicit and flexible.

- Disadvantages:

  1. Often (not always) greater BW for a given sound quality than analog transmission.
  2. Computational complexity.

7.2 Engineering Tradeoffs

- Speech Quality
- Bit Rate (bits/sample, bits/second)
- Complexity (multiplies/sample, multiplies/second)
- Delay

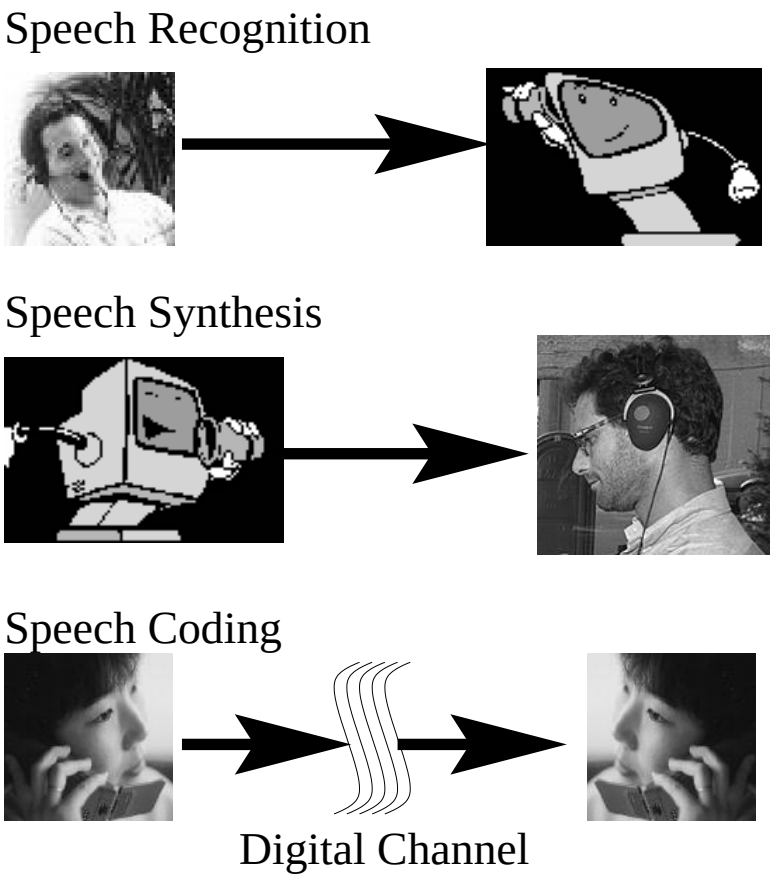

Figure 7.1: The three major engineering applications of speech signal processing.

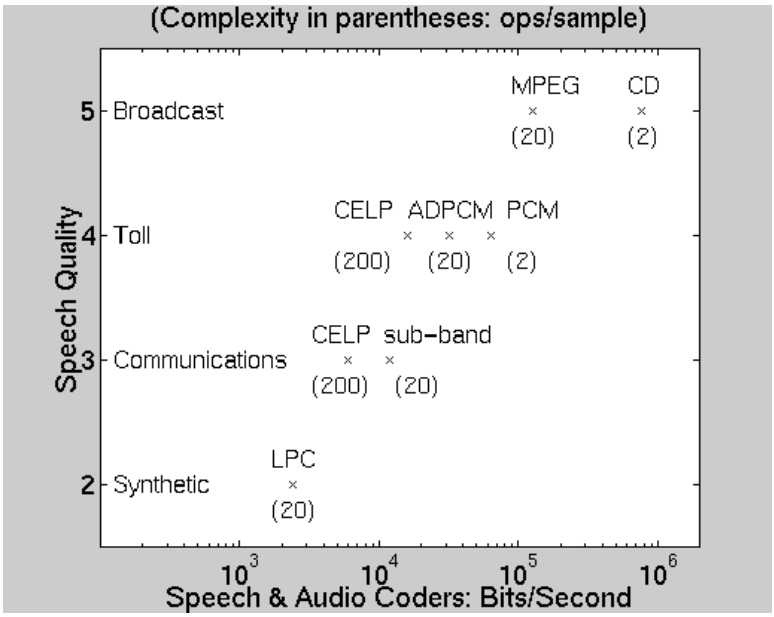

Figure 7.2: An approximate comparison of the speech qualities, bit rates, and computational complexities achieved by various speech and audio coding algorithms.

7.2.1 Applications and Standards

Application	Rate (kbps)	BW (kHz)	Standards Organization	Standard Number	Algorithm
Compact Disc	768/channel	24	ISO		Linear PCM
Land-line Telephone	64	3.2	ITU	G.711	μ-law or A-law PCM
	32	3.2	ITU	G.721, 723, 726	ADPCM
Tele-conferencing	64,56	7	ITU	G.722	Subband-ADPCM
	16	3.2	ITU	G.728	Low Delay CELP
Digital Cellular	13	3.2	GSM	Full-rate	RPE-LTP
	7.9	3.2	TIA	IS-54	VSELP
	6.5	3.2	GSM	Half-rate	VSELP
	6.5	3.2	JDC	Full-rate	VSELP
Satellite Telephony	4.1	3.2	INMARSAT		IMBE
Secure Communications	2.4	3.2	Dept of Defense	FS1015	LPC-10
	4.8	3.2	Dept of Defense	FS1016	CELP

ISO	International Standards Organization (http://www.iso.ch)
ITU	International Telecommunication Union (formerly CCITT) (http://www.itu.ch)
GSM	Groupe Speciale Mobile (of the ITU)
TIA	Telecommunications Industry Association
JDC	Japan Digital Cellular

ADPCM	Adaptive Differential Pulse Code Modulation (R&S, 5.7)
LPC-10	LPC Vocoder with 10 coefficients (R&S, 8.9)
CELP	Code Excited LPC
RPE-LTP	Regular Pulse Excited LPC with Long Term Prediction
VSELP	Vector Sum Excited LPC
IMBE	Improved Multi-Band Excitation

7.3 Subjective Quality Metrics

1. A-B discrimination test: tests "transparency" of the quantizer, for broadcast-quality coders. Force listeners to guess which of two signals was the original, and which was quantized.

2. Mean Opinion Score: tests subjective quality of coded speech. Ask listeners to rate signals on a five-point scale: (1=Bad), (2=Poor), (3=Fair), (4=Good), (5=Excellent). Average across listeners, and across sentences.

3. Diagnostic Rhyme Test: tests intelligibility of coded speech. Play a word, and give listeners two choices, e.g. was the word "bud" or "mud"? Was it "page" or "cage"? Was it "back" or "bat"?

7.3.1 Measures of Speech Quality: Mean Opinion Scores

1. Hire several highly trained listeners from an expensive consulting company.

2. Encode & decode a standard set of sentences.

3. Each listener rates each sentence with one of the following labels: (1=Bad), (2=Poor), (3=Fair), (4=Good), (5=Excellent).

4. Average the ratings across sentences, and across listeners.

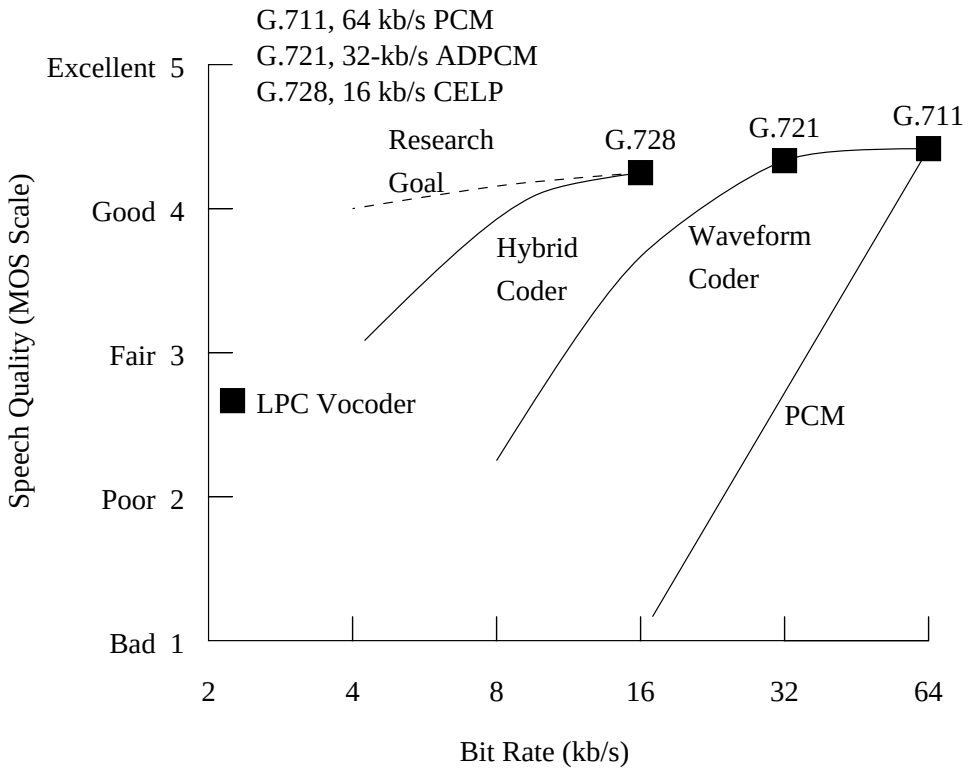

Figure 7.3: Mean opinion scores for various types of speech coders.

Advantage

MOS is an objective, repeatible measurement of a highly subjective quantity: "how good does it sound?"

Disadvantage

1. Expensive
2. MOS can't be calculated automatically, so it can't be used as an optimization criterion in a coding algorithm.

7.4 Objective Measures of Speech Quality

$$x(n) \rightarrow \boxed{\text{Coder}} \rightarrow \boxed{\text{Decoder}} \rightarrow \hat{x}(n) \tag{7.1}$$

7.4.1 Signal to Noise Ratio

- **Formula:**
$$SNR = \frac{\sigma_x^2}{\sigma_e^2} = \frac{\sum_{m=-\infty}^{\infty} x^2(m)}{\sum_{m=-\infty}^{\infty} e^2(m)}, \qquad e(n) = \hat{x}(n) - x(n) \tag{7.2}$$
- **Advantage:** Easy to compute.
- **Disadvantage:** Not a good measure of the perceived distortion. For example, it's much easier to hear quantization noise during quiet passages than during loud passages, but SNR gives more weight to loud passages!
- **Class of Coder:** No speech coders use this metric.

7.4.2 Segmental SNR

- **Formula:**
  Short-Time Analysis:

$$x_n(m) = x(m)w(n-m), \quad \hat{x}_n(m) = \hat{x}(m)w(n-m), \qquad w(m) = \text{windowing function} \tag{7.3}$$

$$\text{SEGSNR}(n) = 10\log_{10}\left(\frac{\sum_m x_n^2(m)}{\sum_m(\hat{x}_n(m) - x_n(m))^2}\right) \tag{7.4}$$

- **Characteristics of Audition Represented:**
  - Signal can mask quantization noise which occurs at about the same time.
- **Class of Coder:** Waveform Coder (R&S chapter 5)
- **Examples:** Companded PCM, DPCM, ADPCM
- Easy to compute (at the coder end; not at the decoder end).
- mean(SEGSNR(n)) — Represents short-time character of speech and hearing.

7.4.3 Perceptually-Weighted SEGSNR

- **Formula:**

$$PSNR(n) = \frac{\sigma_x^2(n)}{\sigma_{e*h}^2} = \frac{\sum_{m=0}^{M}\left[w(m)x(n-m)\right]^2}{\sum_{m=0}^{M}\left[w(m)\sum_{k=0}^{\infty}h(k)e(n-m-k)\right]^2} \tag{7.5}$$

  ... where $h(k)$ is the impulse response of a perceptual weighting filter. Usually, $H(e^{j\omega})$ emphasizes quantization noise at frequencies where $X(e^{j\omega})$ is small, and de-emphasizes noise at frequencies where $X(e^{j\omega})$ is large:

$$H(e^{j\omega}) \sim \frac{1}{X(e^{j\omega})} \tag{7.6}$$

- **Characteristics of Audition Represented:**
  - Signal can mask quantization noise which occurs at the same time.
  - Signal can mask quantization noise at the same frequency.
- **Class of Coder:** Hybrid Vocoder / LPC Analysis-by-Synthesis Coder
- **Examples:** CELP, VSELP, RPE-LTP

7.4.4 Spectral Amplitude Distortion

- **Formula**

$$D(n) \propto \int_{-\pi}^{\pi} \frac{|X_n(e^{j\omega})|^2}{|\hat{X}_n(e^{j\omega})/\hat{X}_n(e^0)|^2}\, d\omega \tag{7.7}$$

  For example, coefficients of the direct-form LPC filter $A(z) = 1 - \sum a_k z^{-k}$ are chosen to minimize (R&S 8.6.1):

$$E_n = \frac{1}{2\pi}\int_{-\pi}^{\pi}|E_n(e^{j\omega})|^2\, d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi}|X_n(e^{j\omega})|^2|A(e^{j\omega})|^2\, d\omega = \frac{G^2}{2\pi}\int_{-\pi}^{\pi}\frac{|X_n(e^{j\omega})|^2}{|\hat{X}_n(e^{j\omega})|^2}\, d\omega \tag{7.8}$$

- **Characteristics of Audition Represented:**
  - Signal can mask quantization noise which occurs at the same time.

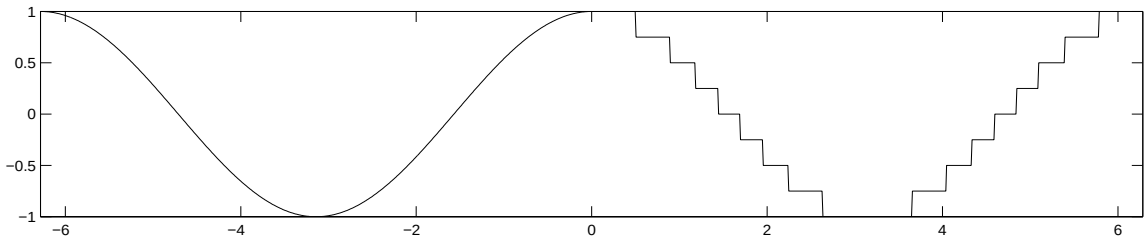

Figure 7.4: Pulse-code modulation (PCM) quantizes each sample of a signal by rounding it to the nearest of a set of fixed quantization levels.

  - Signal can mask quantization noise at the same frequency.
  - Most phase distortion is inaudible.

- **Class of Coder:** Vocoder
- **Examples:** LPC-10, MBE

7.4.5 Noise-to-Masker Ratio

- **Formula:**

$$NMR = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{|E(e^{j\omega})|^2}{|M(e^{j\omega})|^2}\, d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{|\hat{X}(e^{j\omega}) - X(e^{j\omega})|^2}{|M(e^{j\omega})|^2}\, d\omega \tag{7.9}$$

  ... where $M(e^{j\omega})$ is a "masking function" which shows how well $X(e^{j\omega})$ masks the noise at each frequency.

- **Characteristics of Audition Represented:**
  - Signal can mask quantization noise which occurs at the same time.
  - Signal can mask quantization noise at the same frequency.
  - Signal can mask quantization noise at nearby frequencies.
  - Quantization noise is less audible at high frequencies.
- **Class of Coder:** Adaptive Transform Coder, Sub-Band Coder.
- **Examples:** ATC, SBC, MPEG

7.5 Memoryless Quantization ("Pulse Code Modulation" PCM)

$$\boxed{\hat{x}(n) = \hat{x}_i \quad \text{for all} \quad x_{i-1} < x(n) < x_i} \tag{7.10}$$

$$\text{Reconstruction Level} = \hat{x}_i \tag{7.11}$$

$$\text{Decision Boundary} = x_i = \frac{1}{2}(\hat{x}_i + \hat{x}_{i+1}) \tag{7.12}$$

$$\text{Quantization Noise} = q(n) = \hat{x}(n) - x(n) \tag{7.13}$$

Type of Quantizer	Zero is a:	# Reconstruction Levels
Mid-Tread	Reconstruction Level	$2^B - 1$
Mid-Riser	Decision Boundary	2^B

7.5.1 Uniform Quantization

Uniformly spaced reconstruction levels:

$$x_i = i\delta - X_{max}, \quad \delta = \begin{cases} 2X_{max}/(2^B - 1) & \text{mid-tread} \\ 2X_{max}/2^B & \text{mid-riser} \end{cases} \tag{7.14}$$

This is like quantizing by just rounding to the nearest integer using the round(x) function:

$$\hat{x}(n) = \delta \times \text{round}(x(n)/\delta) \qquad \text{(mid-tread)} \tag{7.15}$$

$q(n)$ is uniformly distributed between $\delta/2$ and $-\delta/2$, so

$$\sigma_q^2 = E[q^2(n)] = \frac{\delta^2}{12} = \frac{1}{12}\left(\frac{4X_{max}^2}{2^{2B}}\right) \tag{7.16}$$

which means that the expected SNR, in decibels, is directly proportional to the bit rate B!!

$$SNR = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_q^2}\right) = 6B + 20\log_{10}\left(\frac{\sigma_x}{X_{max}}\right) - 1.2 \tag{7.17}$$

Examples

$$x(n) \in [-32, 32] \Rightarrow RANGE(x) = 64 \tag{7.18}$$

$B = 5$ $B = 6$ (7.19)

$\Delta = \frac{64}{2^5} = 2.0$ $\Delta = \frac{64}{2^6} = 1.0$ (7.20)

(7.21)

$$x(n) = [0.3, 5.5, 15.3, -2, -16.6] \tag{7.22}$$

$\hat{x}(n) = [0, 6, 16, -2, -16]$ $\hat{x}(n) = [0, 6, 15, -2, -17]$ (7.23)

$e(n) = [-0.3, 0.5, 0.7, 0, 0.6]$ $e(n) = [-0.3, 0.5, -0.3, 0, -0.4]$ (7.24)

(7.25)

$SNR = 10log_{10}\frac{\sum x^2(n)}{\sum e^2(n)} = 26.6dB$ $SNR = 10log_{10}\frac{\sum x^2(n)}{\sum e^2(n)} = 29.6dB$ (7.26)

(7.27)

7.5.2 Companded PCM

$$x(n) \rightarrow \boxed{\text{Compress}} \rightarrow t(n) \rightarrow \boxed{\text{Linear PCM}} \rightarrow \hat{t}(n) \rightarrow \boxed{\text{Expand}} \rightarrow \hat{x}(n) \tag{7.28}$$

Usually, $t(n) \approx \log(x(n))$.

$$\begin{aligned} \hat{t}(n) &= \Delta\text{round}(\frac{t(n)}{\Delta}) && (7.29)\\ \hat{x}(n) &= e^{\hat{t}(n)} && (7.30)\\ &= e^{t(n)+\epsilon(n)} && (7.31)\\ &= x(n)e^{\epsilon(n)} && (7.32)\\ &\approx x(n)[1+\epsilon(n)] && (7.33)\\ &= x(n) + x(n)\epsilon(n) && (7.34) \end{aligned}$$

so

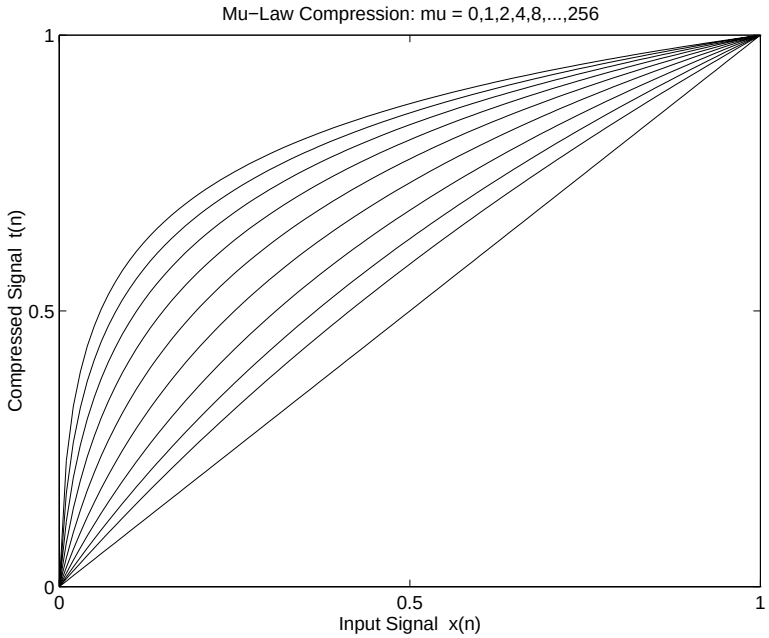

Figure 7.5: Mu-law companding function.

$$\sigma_e^2 = \sigma_x^2 \sigma_\epsilon^2 = \frac{(\sigma_x \Delta)^2}{12} \tag{7.35}$$

$$SNR = \frac{\sigma_x^2}{\sigma_e^2} = \frac{1}{\sigma_\epsilon^2} = 2^{2B} \left(\frac{\sqrt{12}}{RANGE(t)} \right)^2 \tag{7.36}$$

Mu-Law PCM

Mu-law PCM (for example, in the .au audio file format on unix workstations) uses the following companding function:

$$t(n) = X_{max} \frac{\log\left(1 + \mu |x(n)/X_{max}|\right)}{\log(1+\mu)} \text{sign}(x(n)) \tag{7.37}$$

Why Companded PCM is better than Linear PCM

1. SNR is independent of signal level σ_x^2, so microphone settings are not as critical.
2. Quantization noise is reduced in periods of low signal energy, when it would be more audible.
3. Error variance σ_e^2 is sometimes lower in companded PCM. If $p_x(x)$ is the probability distribution of the samples $x(n)$:

$$\sigma_e^2 = E[e^2(n)] = \int_{-\infty}^{\infty} e^2 p_x(x) dx \tag{7.38}$$

So σ_e^2 is minimized if e^2 is low for all likely values of x. In speech, small values of x are more likely than large values, so e^2 should be smaller for small values of x than large values. This is exactly what companding does.

Example

$$\Delta = 1.0, \quad x(n) = [0.3, 5.5, 15.3, -2, -16.6] \tag{7.39}$$

Using μ-law coding, with $X_{max} = 32$, $\mu = 255$:

$$x(n) = [0.3, 5.5, 15.3, -2, -16.6] \tag{7.40}$$
$$\tag{7.41}$$
$$t(n) = [7, 22, 27.8, -16.3, -28.2] \tag{7.42}$$
$$\hat{t}(n) = [7, 22, 28, -16, -28] \tag{7.43}$$
$$\tag{7.44}$$
$$\hat{x}(n) = [0.3, 5.6, 15.9, -1.9, -15.9] \tag{7.45}$$
$$e(n) = [0, 0.1, 0.6, 0.1, 0.7] \tag{7.46}$$
$$\tag{7.47}$$
$$SNR = 10log_{10}\frac{\sum x^2(n)}{\sum e^2(n)} = 28.0dB \tag{7.48}$$
$$\tag{7.49}$$

7.6 Quantization: Basic Principles

$$x(n) \rightarrow \boxed{\text{Coder}} \rightarrow \boxed{\text{Decoder}} \rightarrow \hat{x}(n) \tag{7.50}$$

$$\begin{aligned} \text{Original Vector } \mathbf{x}(n) &= [x(n) \dots x(n+L-1)]' && (7.51)\\ \text{Quantized Vector } \hat{\mathbf{x}}(n) &= [\hat{x}(n) \dots \hat{x}(n+L-1)]' && (7.52)\\ \text{Codebook Vector/Reconstruction Vector } \hat{\mathbf{x}}_i &= [\hat{x}_i(0) \dots \hat{x}_i(L-1)]' && (7.53)\\ \text{Quantization Noise } \mathbf{q}(n) &= \hat{\mathbf{x}}(n) - \mathbf{x}(n) && (7.54)\\ \text{Distortion Metric } D &= D(\hat{\mathbf{x}}(n), \mathbf{x}(n)) && (7.55)\\ & && (7.56) \end{aligned}$$

where $\mathbf{x}'$ is the transpose of $\mathbf{x}$.

7.6.1 Minimum Distortion Rule

Given a codebook of reconstruction vectors $\hat{\mathbf{x}}_i$, the distortion metric $D(\hat{\mathbf{x}}(n), \mathbf{x}(n))$ is minimized by following the rule

$$\boxed{\hat{\mathbf{x}}(n) = \hat{\mathbf{x}}_i \text{ only if } D(\hat{\mathbf{x}}_i, \mathbf{x}(n)) < D(\hat{\mathbf{x}}_j, \mathbf{x}(n)) \text{ for all } i \neq j} \tag{7.57}$$

The set of all vectors $\mathbf{x}$ which should be quantized as $\hat{\mathbf{x}}_i$ is called the ith *Voronoi region*. If D is an L_n norm (for example, if it is a Euclidean distance), the boundaries between Voronoi regions are piece-wise linear, as shown in figure 7.6.1

7.6.2 Mean-Squared Error, SEGSNR

The segmental SNR for a vector of length L can be defined as

$$\text{SEGSNR}(n) = 10\log_{10}\left(\frac{\sum_m x^2(n+m)}{\sum_m (\hat{x}(n+m) - x(n+m))^2}\right) = 10\log_{10}\frac{|\mathbf{x}|^2}{|\mathbf{q}|^2} \tag{7.58}$$

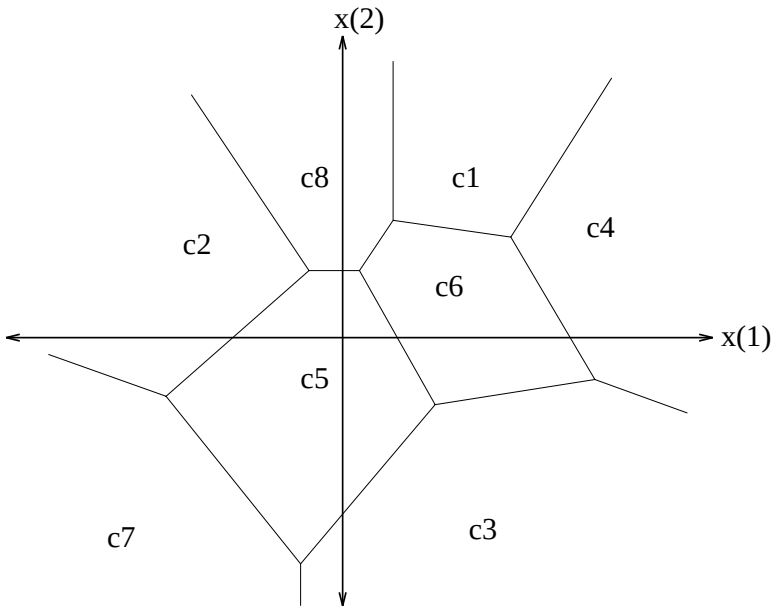

Figure 7.6: The ith Voronoi region is defined as the set of points in N-space which are closer to the ith codebook vector than to any other codebook vector.

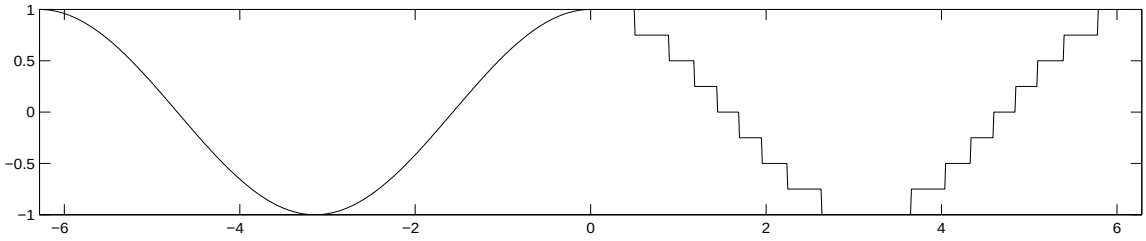

Figure 7.7: Scalar quantization involves quantizing each sample independently of the previous and following samples.

Maximizing SEGSNR for a given segment is equivalent to minimizing the mean squared error:

$$D(\hat{\mathbf{x}}, \mathbf{x}) = |\hat{\mathbf{x}} - \mathbf{x}|^2 = |\mathbf{q}|^2 \tag{7.59}$$

- SEGSNR or MSE is an easy distortion measure to compute (at the coder end; not at the decoder end).
- If you want to measure the quality of a whole sentence, mean(SEGSNR(n)) is a better metric than SNR, because it represents the short-time character of speech and hearing.
- SEGSNR does not represent perceptual effects such as warping of the frequency scale, equal-loudness curves, masking.

7.7 Scalar Quantization ($L = 1$)

$$\text{Reconstruction Level} = \hat{x}_i \tag{7.60}$$

$$\text{Distortion Metric} = q^2(n) = (\hat{x}(n) - x(n))^2 \tag{7.61}$$

$$\tag{7.62}$$

Minimum distortion rule:

$$\boxed{\hat{x}(n) = \hat{x}_i \quad \text{for all} \quad x_{i-1} < x(n) < x_i} \tag{7.63}$$

where

$$x_i = \frac{1}{2}(\hat{x}_i + \hat{x}_{i+1}) \tag{7.64}$$

Type of Quantizer	Zero is a:	# Reconstruction Levels	Advantage
Mid-Tread	Reconstruction Level	$2^B - 1$	Accurately represent silence
Mid-Riser	Decision Boundary	2^B	Maximize total SNR

7.7.1 Linear PCM

In linear PCM, the decision boundaries are uniformly spaced:

$$x_i = i\Delta - X_{max}, \quad \Delta = \begin{cases} 2X_{max}/(2^B - 1) & \text{mid-tread} \\ 2X_{max}/2^B & \text{mid-riser} \end{cases} \tag{7.65}$$

$q(n)$ is uniformly distributed between $\Delta/2$ and $-\Delta/2$, so

$$\sigma_q^2 = E[q^2(n)] = \frac{\Delta^2}{12} = \frac{1}{12}\left(\frac{4X_{max}^2}{2^{2B}}\right) \tag{7.66}$$

so, in decibels,

$$SNR = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_q^2}\right) = 6B + 20\log_{10}\left(\frac{\sigma_x}{X_{max}}\right) - 1.2 \tag{7.67}$$

7.7.2 Minimum-MSE Scalar Quantizer

If we know that x is distributed according to $p_x(x)$, then it is possible to calculate the expected value of the squared error, as a function of x_i and $\hat{x}_i$:

$$\sigma_q^2 = \int_{-\infty}^{\infty} (\hat{x} - x)^2 p_x(x)dx = \sum_i \int_{x_{i-1}}^{x_i} (\hat{x}_i - x)^2 p_x(x)dx \tag{7.68}$$

Differentiating with respect to $\hat{x}_i$ and x_i yields

$$\frac{\partial \sigma_q^2}{\partial \hat{x}_i} = \int_{x_{i-1}}^{x_i} (\hat{x}_i - x)p_x(x)dx \tag{7.69}$$

$$\frac{\partial \sigma_q^2}{\partial x_i} = p_x(x_i)((\hat{x}_{i+1} - x_i)^2 - (\hat{x}_i - x_i)^2) \tag{7.70}$$

Setting these to zero and solving, we get

$$\hat{x}_i = \frac{\int_{x_{i-1}}^{x_i} x p_x(x)dx}{\int_{x_{i-1}}^{x_i} p_x(x)dx} = \int_{x_{i-1}}^{x_i} x p_x(x|x_{i-1} < x < x_i)dx \tag{7.71}$$

$$x_i = \frac{1}{2}(\hat{x}_i + \hat{x}_{i+1}) \tag{7.72}$$

$$\tag{7.73}$$

- For many distributions $p_x(x)$, there is more than one solution.
- For most distributions $p_x(x)$, there is no analytic solution, and solutions must be calculated numerically using an iterative algorithm.
- If $p_x(x) = p_x(-x)$ (symmetric distribution), one of the boundary levels should be $x_i = 0$ (*mid-riser quantizer*).

Example: Laplacian Probability Density

According to Rabiner & Schafer, speech samples $x(n)$ are roughly distributed according to the Laplacian probability density:

$$p_x(x) = \frac{1}{\sigma_x\sqrt{2}} e^{-|x|\frac{\sqrt{2}}{\sigma_x}} \tag{7.74}$$

With one bit per sample, the optimum decision boundary is

$$x_1 = 0 \tag{7.75}$$

The optimum reconstruction levels are $\hat{x}_1 = -\hat{x}_2$,

$$\hat{x}_2 = \frac{\int_0^\infty x p_x(x) dx}{\int_0^\infty p_x(x) dx} = \frac{\frac{1}{\sigma_x\sqrt{2}} \int_0^\infty x e^{-x\frac{\sqrt{2}}{\sigma_x}} dx}{0.5} = \frac{\sigma_x}{\sqrt{2}} \tag{7.76}$$

7.7.3 Semilog Companded Quantization

Define a set of semi-exponentially spaced decision boundaries:

$$x_i \approx \begin{cases} e^{iC_1 - C_2} & \hat{x}_i \gg 0 \\ iC_3 & \hat{x}_i \text{ small} \end{cases} \tag{7.77}$$

Quantizing using x_i above is the same as compressing the dynamic range of $x(n)$ using a semilog compresser, then quantizing using linear PCM:

$$v_i = i\Delta - V_{max}, \quad \Delta = \begin{cases} 2V_{max}/(2^B - 1) & \text{mid-tread} \\ 2V_{max}/2^B & \text{mid-riser} \end{cases} \tag{7.78}$$

$$x(n) \rightarrow \boxed{\text{Semilog-Compress}} \rightarrow v(n) \rightarrow \boxed{\text{Linear PCM}} \rightarrow \hat{v}(n) \rightarrow \boxed{\text{Exponential-Expand}} \rightarrow \hat{x}(n) \tag{7.79}$$

Example: Mu-Law PCM

North American telephone transmission uses 8-bit μ-law PCM. European telephone transmission uses a similar scheme called A-law companding. Mu-law companding uses the rule:

$$v(n) = X_{max} \frac{\log\left(1 + \mu|x(n)/X_{max}|\right)}{\log(1+\mu)} \text{sign}(x(n)) \tag{7.80}$$

Why Companded PCM is better than Linear PCM (for small word lengths)

1. Semilog-companding approximates the maximum-SNR quantizer for speech.
2. Semilog-companding sounds *better* than the maximum-SNR quantizer for speech.
3. SNR is independent of signal level σ_x^2, so microphone settings are not as critical:

$$\hat{x}(n) \approx e^{C_1 v(n)} e^{C_1(\hat{v}(n) - v(n))} = x(n) e^{C_1(\hat{v}(n) - v(n))} \approx x(n)[1 + \epsilon(n)] \tag{7.81}$$

$$q(n) = x(n)\epsilon(n) \tag{7.82}$$

If $x(n)$, $n\epsilon(n)$ are independent Gaussian random variables,

$$\text{SNR} = 10\log_{10}\frac{\sigma_x^2}{\sigma_x^2\sigma_\epsilon^2} = 10\log_{10}\frac{1}{\sigma_\epsilon^2} \tag{7.83}$$

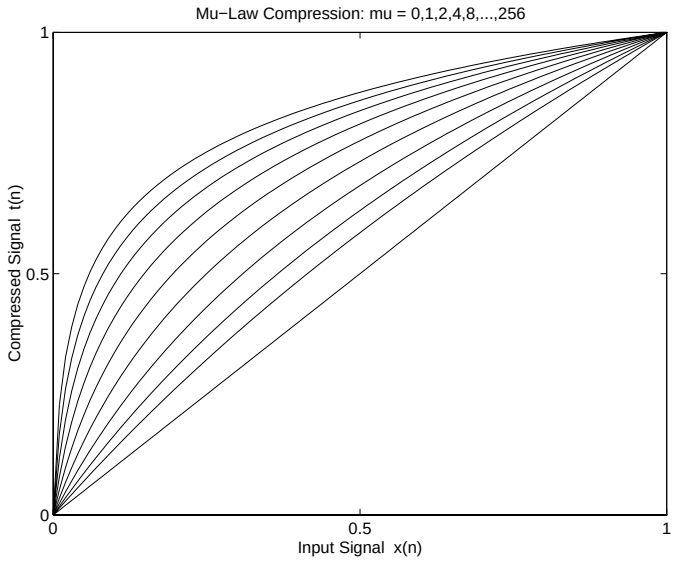

Figure 7.8: The mu-law companding function.

7.8 Vector Quantization ($L > 1$)

7.8.1 Minimum-MSE Vector Quantizer

If we know that the vector $\mathbf{x}$ is distributed according to $p_x(\mathbf{x})$, then it is possible to calculate the expected value of the squared error, as a function of the Voronoi regions V_i and codebook vectors $\hat{\mathbf{x}}_i$:

$$E[|\mathbf{q}|^2] = \sum_i \int_{V_i} |\hat{\mathbf{x}}_i - \mathbf{x}|^2 p_x(\mathbf{x}) dx \tag{7.84}$$

Minimizing with respect to V_i and $\hat{\mathbf{x}}_i$ yields

$$\hat{\mathbf{x}}_i = \int_{V_i} \mathbf{x} p_x(\mathbf{x}|\mathbf{x} \in V_i) dx \tag{7.85}$$

$$V_i = \left\{\mathbf{x} \| (\hat{\mathbf{x}}_i - \mathbf{x})^2 < (\hat{\mathbf{x}}_j - \mathbf{x})^2 \text{ for } i \neq j\right\} \tag{7.86}$$

$$\tag{7.87}$$

In most practical systems, $p_x(\mathbf{x})$ is approximated using a set of N training vectors $\mathbf{y}_n$, where N should be at least ten times the dimension of $\mathbf{x}$:

$$\int_{V_i} f(\mathbf{x}) p_x(\mathbf{x}) dx \approx \frac{1}{N(V_i)} \sum_{V_i} f(\mathbf{y_n}), \quad N(V_i) \equiv \text{Number of } \mathbf{y}_n \in V_i \tag{7.88}$$

With this approximation, the minimum-MSE quantizer becomes

$$\hat{\mathbf{x}}_i = \frac{1}{N(V_i)} \sum_{V_i} \mathbf{y}_n \tag{7.89}$$

$$V_i = \left\{\mathbf{x} \| (\hat{\mathbf{x}}_i - \mathbf{x})^2 < (\hat{\mathbf{x}}_j - \mathbf{x})^2 \text{ for } i \neq j\right\} \tag{7.90}$$

$$\tag{7.91}$$

The equations above give rise to the following iterative procedure for finding a minimum-MSE vector codebook. If we are given some initial set of codebook vectors $\hat{\mathbf{x}}_i$, then the MSE (calculated over the training tokens) is always either improved or not changed using the following iteration, called alternatively the LBG algorithm, the generalized Lloyd algorithm, or the K-means algorithm:

1. Assign each of the training tokens $\mathbf{y}_i$ to a Voronoi region based on the minimum-distortion rule.
2. Implementation issue: check to make sure that each of the Voronoi regions contains at least one training token. If there is a region without training tokens, eliminate it, and replace it by splitting the largest Voronoi region in half.
3. Update $\hat{\mathbf{x}}_i$ by calculating the centroid of each Voronoi region.
4. If any of the training tokens have moved from one region to another since the previous iteration, repeat the procedure.

Characteristics:

- Guaranteed to converge.
- Usually has more than one solution.
- The solution to which it converges for any given set of starting vectors may not be very good.
- Therefore, it is usually good to run the algorithm a few times, with different starting vectors.

K-Means Algorithm

In the classic K-means algorithm, the codebook vectors $\hat{\mathbf{x}}_i$ are initialized using random values.

Binary Splitting

The binary splitting algorithm initializes a codebook of size K by "splitting" a codebook of size $K/2$. If ξ_i are the vectors in a minimum-MSE codebook of size $K/2$, then the vectors in a codebook of size K are initialized as

$$\hat{\mathbf{x}}_i = \xi_i(1+\epsilon) \tag{7.92}$$
$$\hat{\mathbf{x}}_{K/2+i} = \xi_i(1-\epsilon) \tag{7.93}$$
$$\tag{7.94}$$

where $\epsilon \approx 0.01 - 0.05$.

The binary splitting algorithm is initialized using a minimum-MSE codebook of size $K = 1$. The minimum-MSE codebook of size 1 is just the centroid of all of the training data:

$$\hat{\mathbf{x}}_1 = \frac{1}{N}\sum_{n=1}^{N} \mathbf{y}_n \tag{7.95}$$

7.8.2 Product Coding: VQ with the Complexity of Scalar Quantization

If the samples of $\mathbf{x}$ are statistically independent, one optimum codebook consists of independent scalar quantizers for each of the elements of $\mathbf{x}$:

$$\hat{x}(n+m) = \hat{x}_i(m) \quad \text{if} \quad x_{i-1}(m) < x(n+m) < x_i(m) \tag{7.96}$$

where

$$x_i(m) = \frac{1}{2}(\hat{x}_i(m) + \hat{x}_{i+1}(m)) \tag{7.97}$$

A codebook created in this way is called a product code, because if $\hat{x}(m)$ has K_m possible quantization levels, the total number of codebook vectors is

$$K = K_0 \times K_2 \times \ldots K_{L-1} \tag{7.98}$$

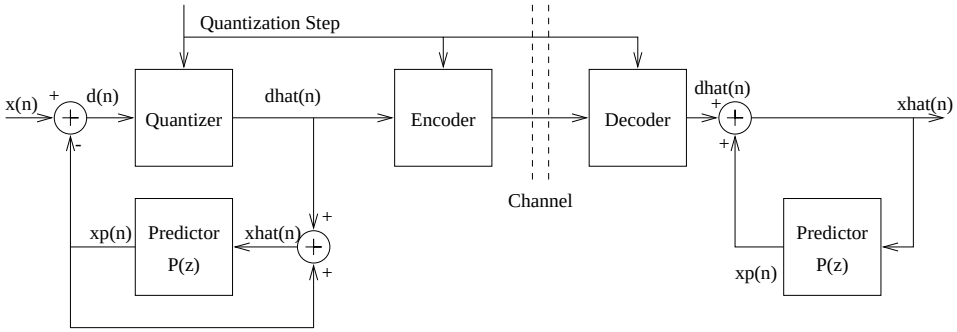

Figure 7.9: Schematic of a DPCM coder.

Product coding has most of the features of normal scalar quantization (e.g. low complexity), but it has one feature of vector quantization: the number of bits per sample can be a fraction:

$$B = \frac{\log_2(K)}{L} = \frac{1}{L}\sum_{m=0}^{L-1}\log_2(K_m) \tag{7.99}$$

This is especially useful in schemes such as sub-band coding, where a bit allocation algorithm is used to decide how many bits to use per channel. With product coding, if the bit allocation algorithm tells you to use 2.3 bits in one channel and 2.6 bits in another channel, you can do it.

7.9 Adaptive Step Size Quantization (APCM)

Energy σ_x^2 of the speech signal varies over a huge dynamic range (human hearing covers a range of up to about 120dB, or 12 orders of magnitude of σ_x^2). In order to represent loud sounds clearly, PCM coders need to have large upper bounds X_{max}, even though the larger levels are almost never used. A simple solution to these problems is this: adapt the step size Δ to match short-time energy E_n.

1. Forward-adaptive:

$$\Delta = \Delta_{ref}\sqrt{\sum_{m=1}^{N} x_n^2(m)} \tag{7.100}$$

   - Δ based on samples of $x(n)$, so can adapt quickly to changes in spectral shape.
   - Δ must be transmitted as side information, so bit rate goes up.
   - SNR improves by up to 8dB, so we could cut the bit rate by more than 1 bit/sample.

2. Backward-adaptive:

$$\Delta = \Delta_{ref}\sqrt{\sum_{m=n-N}^{n-1} \hat{x}^2(m)} \tag{7.101}$$

   - Δ based on previous samples of $\hat{x}(n)$, so adaptation is slower.
   - Δ don't need to be transmitted, so the total bit rate is less.

7.10 Differential PCM (DPCM)

$$d(n) = x(n) - \sum_{k=1}^{p} a_k \hat{x}(n-k) \tag{7.102}$$

$$\hat{x}(n) = \hat{d}(n) + \sum_{k=1}^{p} a_k \hat{x}(n-k) \tag{7.103}$$

$$q(n) = \hat{x}(n) - x(n) = \hat{d}(n) - d(n) \tag{7.104}$$

Remember that the SNR of a linear PCM coder is

$$\text{SNR}_{PCM} = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_q^2}\right) = 6B + 10\log_{10}\left(\frac{\sigma_x^2}{X_{max}^2}\right) - 1.2 \tag{7.105}$$

By quantizing $d(n)$ instead of $x(n)$, we get

$$\text{SNR}_{DPCM} = 6B + 10\log_{10}\left(\frac{\sigma_x^2}{D_{max}^2}\right) - 1.2 = \text{SNR}_{PCM} + 10\log_{10}\left(\frac{X_{max}^2}{D_{max}^2}\right) \tag{7.106}$$

Many derivations assume that the maximum clipping threshold is set proportional to the signal standard deviation, so that

$$\text{SNR}_{DPCM} = \text{SNR}_{PCM} + 10\log_{10} G_p, \quad G_p \equiv \frac{\sigma_x^2}{\sigma_d^2} \tag{7.107}$$

G_p is called the "prediction gain." The prediction gain is determined by the prediction filter as follows:

$$X(z) \approx \frac{D(z)}{1 - P(z)} \tag{7.108}$$

$$G_p = \frac{\sigma_x^2}{\sigma_d^2} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{|D(e^{j\omega})/\sigma_d|^2}{|1 - P(e^{j\omega})|^2} d\omega \approx \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{1}{|1 - P(e^{j\omega})|^2} d\omega \tag{7.109}$$

where the last approximation assumes that $D(e^{j\omega})$ is relatively flat as a function of frequency.

7.10.1 Fixed Differential PCM (DPCM)

In fixed-predictor DPCM, the predictor models the long-term correlation of speech samples, for example

$$a_1 = \frac{E[x(n)x(n-1)]}{E[x^2(n)]} \tag{7.110}$$

For standard long-term spectra, $a_1 \approx 0.8 - 0.9$, so

$$G_p = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{1}{|1 - a_1 e^{-j\omega}|^2} d\omega \approx 4 - 7\text{dB} \tag{7.111}$$

In other words, with $p = 1$, we get a savings of about 1 bit/sample over regular PCM. $p = 2$ shows another dB or so of improvement, and then there is little additional improvement for higher orders.

7.10.2 Adaptive Differential PCM (ADPCM)

In adaptive differential PCM, the quantizer step Δ and the predictor coefficients a_k may be updated once per frame (e.g. once per 20ms) in order to match the spectral characteristics of the signal, e.g.

1. Forward-adaptive: $P(z)$ is calculated using standard LPC normal equations, and transmitted as side information.
2. Backward-adaptive: $P(z)$ is calculated using backward-adaptive LPC equations.

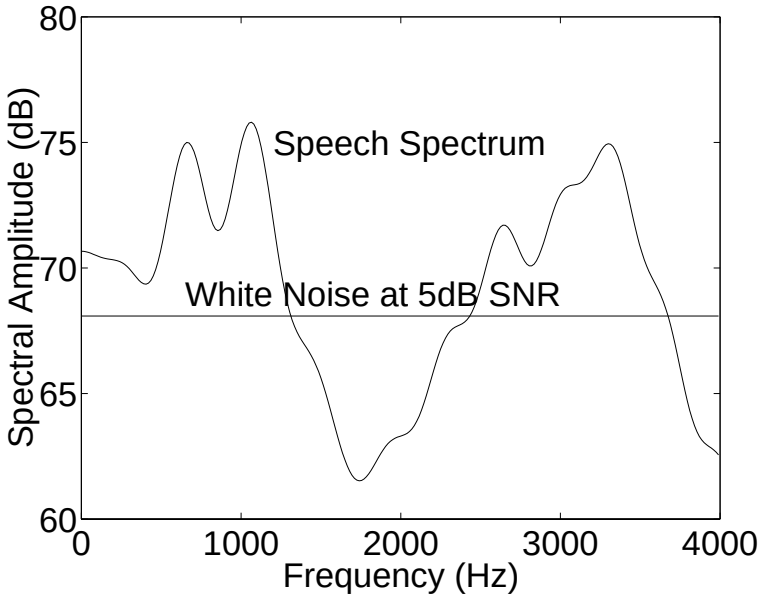

Figure 7.10: The minimum-energy quantization noise is usually white noise.

7.11 Perceptual Error Weighting

7.11.1 Error Spectrum is Nearly White

Quantization is designed to minimize the sum squared error,

$$E_n = \frac{1}{2\pi}\int_{\pi}^{\pi} |\hat{X}(e^{j\omega}) - X(e^{j\omega})|^2 d\omega \tag{7.112}$$

In DPCM, for example,

$$\hat{X}(e^{j\omega}) = \frac{\hat{D}(e^{j\omega})}{1 - P(e^{j\omega})} = H(e^{j\omega})\hat{D}(e^{j\omega}) \tag{7.113}$$

It turns out that

$$E_n \text{ is minimized if } \quad Q(e^{j\omega}) = \text{constant} \tag{7.114}$$

that is, if $q(n)$ is an uncorrelated random noise signal, as shown in figure 7.11.1.

7.11.2 Using the Signal to Mask the Noise

Noise near peaks of the spectrum $X(e^{j\omega})$ is masked by X. Therefore, quantization noise is less audible if it is shaped to look a little bit like the signal spectrum:

We can shape the noise spectrum by minimizing a "noise-to-masker ratio" of approximately the following form:

$$\text{NMR} = \frac{1}{2\pi}\int_{\pi}^{\pi} \frac{|Q(e^{j\omega})|^2}{|M(e^{j\omega})|^2} d\omega \tag{7.115}$$

For practical reasons, we use the masking spectrum

$$M(e^{j\omega}) \approx \frac{|H(e^{j\omega})|^2}{|H(e^{j\omega}/\theta)|^2} = \frac{|1 - P(e^{j\omega}/\theta)|^2}{|1 - P(e^{j\omega})|^2}, \qquad 0 \le \theta \le 1 \tag{7.116}$$

At first glance, this seems to be a really odd masking spectrum to use. We make use of this spectrum for three reasons:

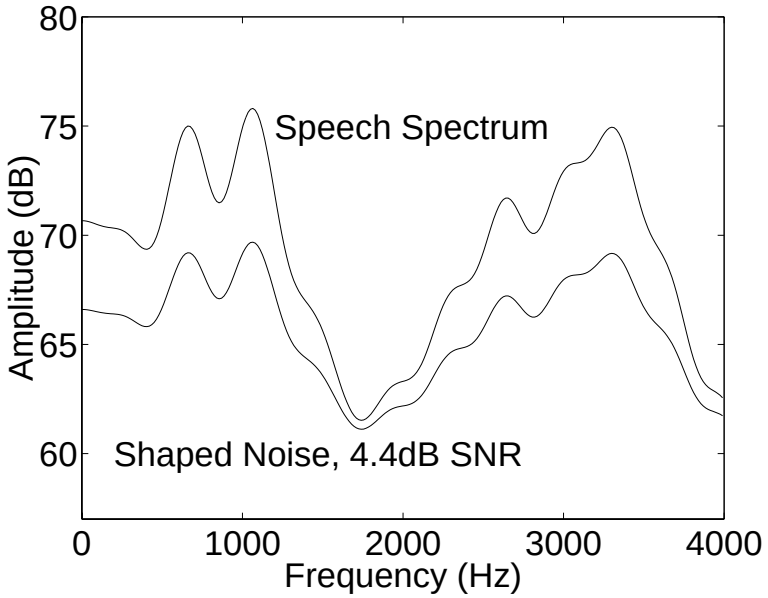

Figure 7.11: Shaped quantization noise may be less audible than white quantization noise, even at slightly higher SNR.

1. The numerator and denominator have roots at exactly the same frequencies, so that the total spectral tilt of $M(e^{j\omega})$ is zero.
2. The roots of the denominator have narrower bandwidth than the roots of the numerator, which means that $M(e^{j\omega})$ peaks at the formant frequencies. It makes sense that a "masking spectrum" should have peaks at the same frequencies as the speech spectrum does.
3. The fractional form of $M(e^{j\omega})$ results in computationally efficient error weighting algorithms, as we'll see later (both this lecture and next lecture).

Given this particular choice of $M(e^{j\omega})$, the NMR computation becomes

$$\text{NMR} = \frac{1}{2\pi}\int_{\pi}^{\pi} |Q_w(e^{j\omega})|^2 = \sum_n q_w^2(n) \tag{7.117}$$

where

$$Q_w(z) = Q(z)\frac{H(z/\theta)}{H(z)} = Q(z)\frac{1-P(z)}{1-P(z/\theta)} \tag{7.118}$$

The spectrum $H(z/\theta)$ is like the transfer function $H(z)$, but with the bandwidths of all poles expanded:

$$H(z) = \prod_{i=1}^{p}\frac{1}{(1-r_i z^{-1})}, \qquad H(z/\theta) = \prod_{i=1}^{p}\frac{1}{(1-r_i\theta z^{-1})} \tag{7.119}$$

"Bandwidth expansion" means moving pole frequencies away from the unit circle, as shown in figure 7.11.2.

7.12 DPCM with Noise Feedback

7.12.1 An Alternative Representation of DPCM

Remember that $D(z)$ is

$$D(z) = X(z) - P(z)\hat{X}(z) = X(z) - \frac{P(z)\hat{D}(z)}{1-P(z)}, \qquad P(z) = \sum_{k=1}^{p} a_k z^{-k} \tag{7.120}$$

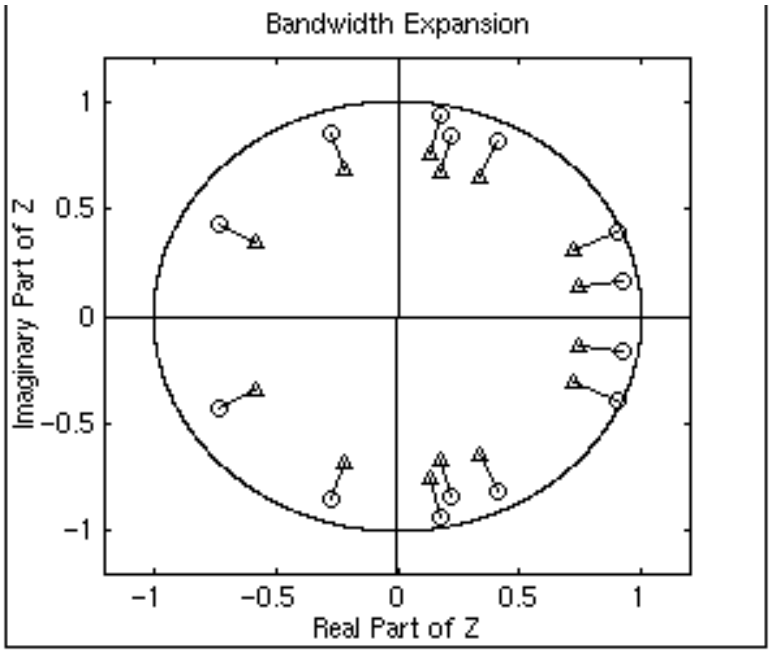

Figure 7.12: Bandwidths of the LPC poles are expanded by moving the poles away from the unit circle (poles ar r_i are shown as circles, poles at θr_i as triangles).

By multiplying both sides of the equation by $1 - P(z)$, we get

$$\begin{aligned} D(z) &= (1 - P(z))X(z) - P(z)\hat{D}(z) + P(z)D(z) && (7.121) \\ &= (1 - P(z))X(z) - P(z)Q(z) && (7.122) \end{aligned}$$

This equation represents an alternate implementation of DPCM (shown in figure 7.12.1, which produces exactly the same output as the form we considered before. The advantage of this representation is that the noise $q(n)$ is explicitly represented.

7.12.2 DPCM with Noise Shaping

The figure above shows a structure designed to minimize the total noise power. Noise shaping can be achieved by designing a coder which minimizes the weighted quantization noise spectrum,

$$Q_w(z) = Q(z)\frac{1 - P(z)}{1 - F(z)}, \qquad F(z) = P(z/\theta) \tag{7.123}$$

It can be shown that $Q_w(z)$ is minimized if we weight the error with $F(z)$ instead of $P(z)$ in the DPCM feedback loop, as shown:

7.13 LPC Vocoder

7.13.1 A Simple Model of Speech Production

Most of the sounds of speech can be produced using a synthesizer with the form shown in figure 7.13.1. In order to code speech for digital transmission, this model is often simplified to the form shown in figure 7.13.1. Figure 7.13.1 is derived from figure 7.13.1 by making the following two assumptions:

1. Either $G_v = 0$ or $G_h = 0$, i.e. you can't have voicing and aspiration at the same time.

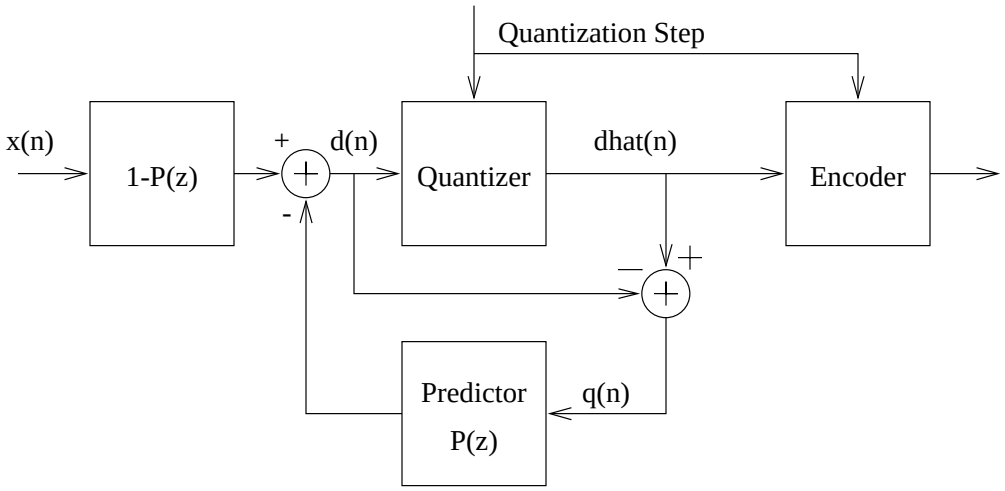

Figure 7.13: An alternative implementation of DPCM.

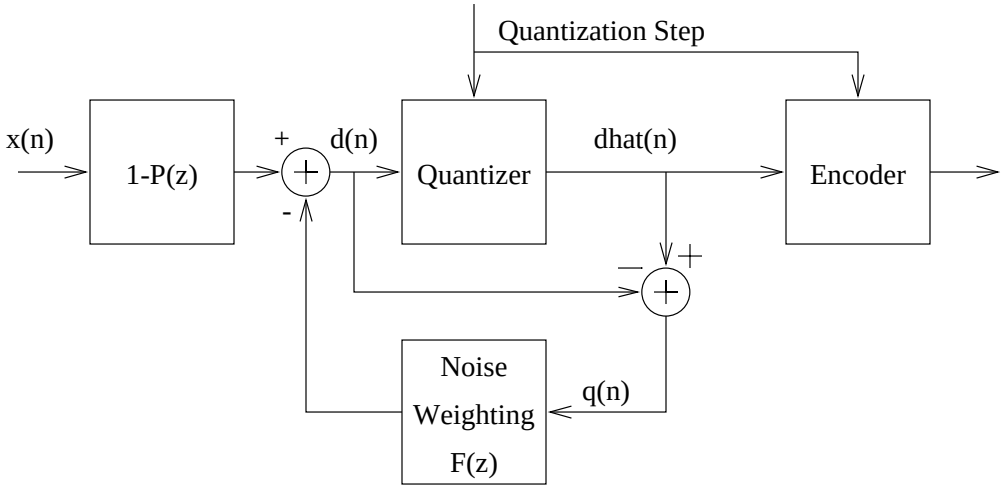

Figure 7.14: DPCM with noise filtering.

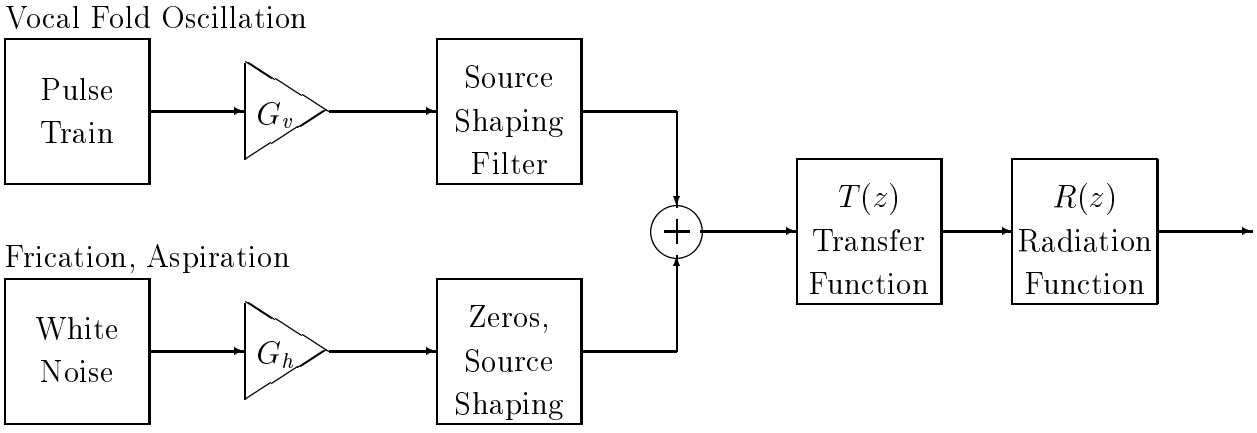

Figure 7.15: A model of speech production which might be used in a speech synthesis program.

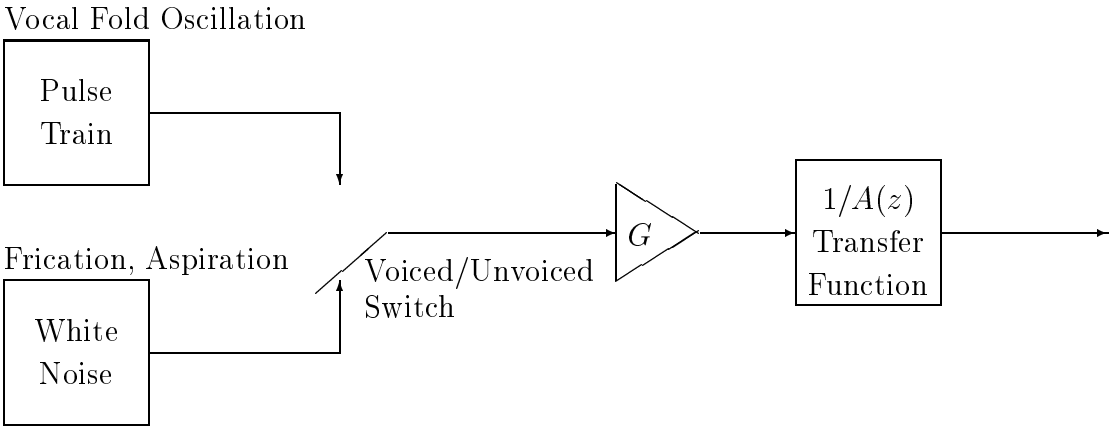

Figure 7.16: A simplified model of speech production, whose parameters can be transmitted efficiently across a digital channel.

2. All of the filters cascade into a single approximately all-pole filter:

$$\frac{1}{A(z)} = V(z)T(z)R(z) \quad or \quad \frac{1}{A(z)} = N(z)T(z)R(z) \tag{7.124}$$

7.13.2 Vocoder Parameter Calculations

First, window the speech using a Hamming window of 20-40ms length, in order to produce the windowed speech frame $x(n)$. Then calculate the following parameters:

LPC Parameters: A(z), G

Calculate $A(z) = 1 - \sum_{k=1}^{p} \alpha_k z^{-k}$ using standard autocorrelation LPC methods. Recall that autocorrelation LPC chooses α_k to minimize

$$E_n = \sum_{n=0}^{N+p-1} e^2(n) \tag{7.125}$$

$$e(n) \equiv x(n) - \sum_{k=1}^{p} \alpha_k x(n-k) \tag{7.126}$$

Once the α_k have been calculated, the gain is chosen to be

$$G = \sqrt{E_n} \tag{7.127}$$

Excitation Parameters: Pitch Period, V/UV Decision

The LPC "excitation" signal is

$$d(n) = \frac{e(n)}{G} \tag{7.128}$$

Recall that the autocorrelation of $d(n)$, for a windowed $d(n)$, can been defined to be

$$R_d(m) = \frac{1}{N} \sum_{n=|m|}^{N-1} d(n)d(n-|m|) \tag{7.129}$$

If $d(n)$ is a stochastic signal (for example, if it is a white noise signal), it can be shown that $R_d(n)$ is a biased estimator of the stochastic autocorrelation:

$$E[R_d(m)] = \frac{N-|m|}{N} E[d(n)d(n-m)] \quad \text{if } d(n) \text{ stochastic} \tag{7.130}$$

It turns out that estimates of the pitch period and the degree of voicing of a signal are much more accurate if they are based on an "unbiased" autocorrelation function, $r_d(m)$:

$$r_d(m) = \frac{1}{N-|m|} \sum_{n=|m|}^{N-1} d(n)d(n-|m|) \tag{7.131}$$

$$E[r_d(m)] = E[d(n)d(n-m)] \quad \text{if } d(n) \text{ stochastic} \tag{7.132}$$

If $d(n)$ is a deterministic signal (for example, an impulse train), the expectation operator $E[d(n)d(n-m)]$ is not defined, but the "unbiased" autocorrelation $r_d(m)$ is still empirically useful.

Notice that

$$r_d(0) = R_d(0) = \frac{1}{N} \sum_{n=0}^{N-1} \frac{e(n)^2}{G^2} = 1.0 \tag{7.133}$$

1. **Voiced Excitation**

   Represent $d(n)$ using an impulse train, $\hat{d}(n)$:

$$\hat{d}(n) = \sqrt{N_0} \sum_{r=-\infty}^{\infty} \delta(n - rN_0) \tag{7.134}$$

$$r_{\hat{d}}(m) = \sum_{r=-\infty}^{\infty} \delta(m - rN_0) \tag{7.135}$$

$$|D(e^{j\omega})|^2 = \frac{2\pi}{N_0} \sum_{r=-\infty}^{\infty} \delta(\omega - \frac{2\pi r}{N_0}) \tag{7.136}$$

$$|\hat{X}(e^{j\omega})|^2 = \frac{2\pi}{N_0} \left| \frac{G}{A(e^{j\omega})} \right|^2 \sum_{r=-\infty}^{\infty} \delta(\omega - \frac{2\pi r}{N_0}) \tag{7.137}$$

$$\tag{7.138}$$

2. **Unvoiced Excitation**

   Represent $d(n)$ using uncorrelated Gaussian random noise, $\hat{d}(n)$:

$$\hat{d}(n) \sim \mathcal{N}(0,1) \tag{7.139}$$

$$\tag{7.140}$$

$$E[r_{\hat{d}}(m)] = E[d(n)d(n-m)] = \delta(m) \tag{7.141}$$

$$\tag{7.142}$$

$$E\left[|D(e^{j\omega})|^2\right] = 1 \tag{7.143}$$

$$E\left[|\hat{X}(e^{j\omega})|^2\right] = \left| \frac{G}{A(e^{j\omega})} \right|^2 \tag{7.144}$$

$$\tag{7.145}$$

3. **The Voiced/Unvoiced Decision**

   We have two possible excitation signals:

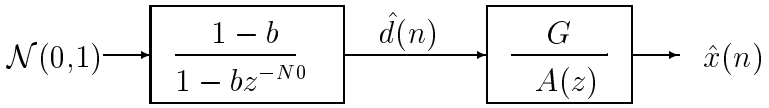

Figure 7.17: A pitch-predictive vocoder.

- If $\hat{d}(n)$ is white noise, $E\left[r_{\hat{d}}(m)\right] = \delta(m)$.
- If $\hat{d}(n)$ is an impulse train, then so is $r_{\hat{d}}(m) = \sum_{r=-\infty}^{\infty} \delta(m - rN_0)$.

Which one is a better representation of the unquantized excitation $d(n)$?

$$\text{IF} \left\{ \begin{array}{ll} \max_{m>0}(r_d(m)) < \theta & \ldots \text{then } r_d(m) \text{ is like a single impulse, so the frame is UNVOICED.} \\ \max_{m>0}(r_d(m)) \geq \theta & \ldots \text{then } r_d(m) \text{ is like an impulse train, so the frame is VOICED.} \end{array} \right\}, \quad \theta \approx 0.3 \tag{7.146}$$

If the frame is voiced, then the pitch period is set to be

$$N_0 = \arg\max_{m>0} r_d(m) \tag{7.147}$$

7.14 Pitch Prediction Vocoder

7.14.1 Purpose

To allow $\hat{d}(n)$ to be a mixture of voiced and unvoiced excitation. In other words, to get rid of vocoder assumption number 1.

7.14.2 Excitation

Excitation is white noise, filtered by a "pitch prediction filter" $P(z)$, as shown in figure 7.14.2.

$$P(z) = \frac{1-b}{1-bz^{-N_0}} \tag{7.148}$$

$$p(n) = (1-b)\sum_{r=0}^{\infty} b^r \delta(n - rN_0) \tag{7.149}$$

Since the input to $P(z)$ is uncorrelated noise, the autocorrelation of the output, $E[r_{\hat{d}}(m)]$, is a function only of the impulse response $p(n)$:

$$E[r_{\hat{d}}(m)] = E[\hat{d}(n)\hat{d}(n-m)] = \sum_{n=-\infty}^{\infty} p(n)p(n-m) \tag{7.150}$$

$$= \frac{1-b}{1+b}\sum_{q=-\infty}^{\infty} |b|^q \delta(m - qN_0) \tag{7.151}$$

So the quantized autocorrelation $r_{\hat{d}}(m)$ is a decaying impulse train with period N_0. The period and decay rate of $r_{\hat{d}}(m)$ can be chosen to match the unquantized $r_d(m)$ as closely as possible. For example, if we only care about matching the first peak in $r_d(m)$, we can choose

$$b = \max_{m>0} \frac{r_d(m)}{r_d(0)} \tag{7.152}$$

$$N_0 = \arg\max_{m>0} \frac{r_d(m)}{r_d(0)} \tag{7.153}$$

$$\tag{7.154}$$

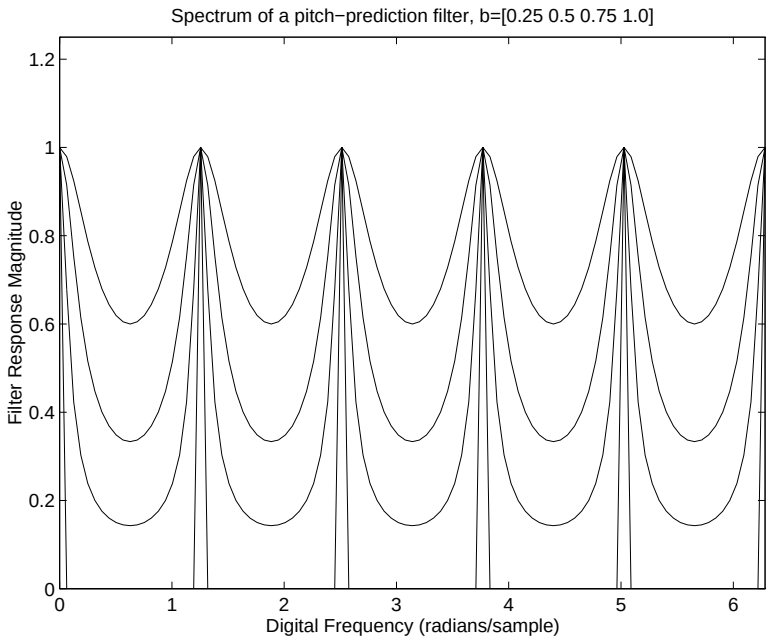

Figure 7.18: Transfer function of the pitch prediction filter for several values of the prediction coefficient.

The spectrum is:

$$E\left[|D(e^{j\omega})|^2\right] = \left|\frac{1-b}{1-be^{-j\omega N_0}}\right|^2 \tag{7.155}$$

$$\tag{7.156}$$

$$= \begin{cases} 1 & \omega = \frac{2\pi k}{N_0} \\ \left(\frac{1-b}{1+b}\right)^2 & \omega = \frac{(2k-1)\pi}{N_0} \end{cases} \tag{7.157}$$

$$\tag{7.158}$$

7.15 LPC-Based Analysis-by-Synthesis Coding

7.15.1 What is Analysis-by-Synthesis?

Coding Strategy

An "analysis-by-synthesis" speech coder, in the most general terms, consists of the following components:

- A model of speech production which depends on certain parameters θ:

$$\hat{s}(n) = f(\theta) \tag{7.159}$$

- A list of K possible parameter sets for the model,

$$\theta_1, \ldots, \theta_k, \ldots, \theta_K \tag{7.160}$$

- An error metric $|E_k|^2$ which compares the original speech signal $s(n)$ and the coded speech signal $\hat{s}(n)$:

$$|E_k|^2 = D(s(n), \hat{s}(n)) \tag{7.161}$$

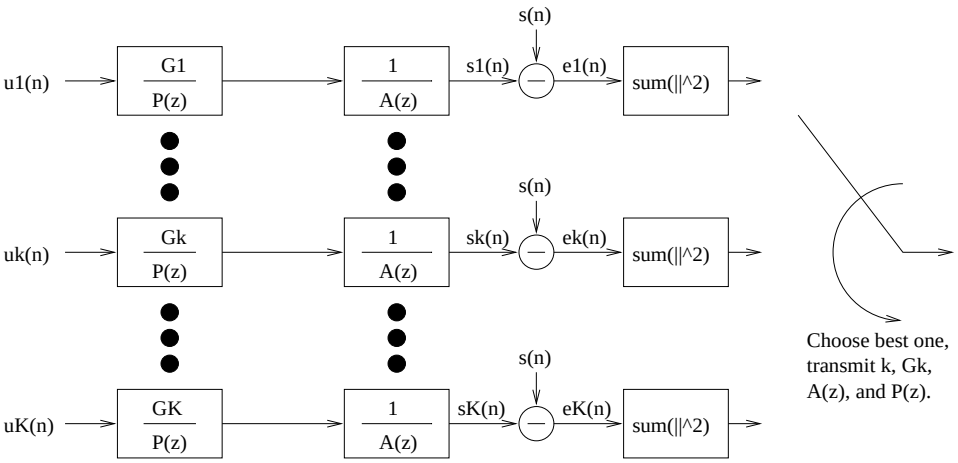

Figure 7.19: LPC analysis by synthesis coder.

A general AbS coder finds the optimum set of parameters by searching through all of the K possible parameter sets, and choosing the one which minimizes E_k. This kind of exhaustive optimization is called a "closed-loop" optimization procedure.

The Curse of Dimensionality

The problem with the most generalized AbS coding is that, as the model becomes relatively complex, K becomes quite large, so a closed-loop search takes a long time.

7.15.2 LPC-based Analysis-by-Synthesis

LPC-AS reduces the complexity of generalized AbS using the following strategy:

- LPC prediction filter $A(z)$: chosen "open-loop," often using the autocorrelation method:

$$\frac{1}{A(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{7.162}$$

- Pitch prediction filter $P(z)$: (sometimes) chosen "open-loop," using standard pitch-detection techniques ($I = 0$ implies an integer pitch period; $I > 0$ allows non-integer pitch periods):

$$\frac{1}{P(z)} = \frac{1}{1 - \sum_{i=-I}^{I} b_i z^{-(D+i)}} \tag{7.163}$$

- Prediction residual $u_k(n)$: chosen "closed-loop" from a set of K excitations, in order to minimize mean-squared-error:

$$k_{opt} = \arg\min_k \left(\sum_{n=0}^{L-1} (\hat{s}_k(n) - s(n))^2 \right) \tag{7.164}$$

7.15.3 Frame-Based Analysis

Frames and Sub-Frames

- The Problem:

  LPC works well with frames of 20-30ms, but exhaustive search for an excitation vector is only computationally feasible for frames of 5-10ms.

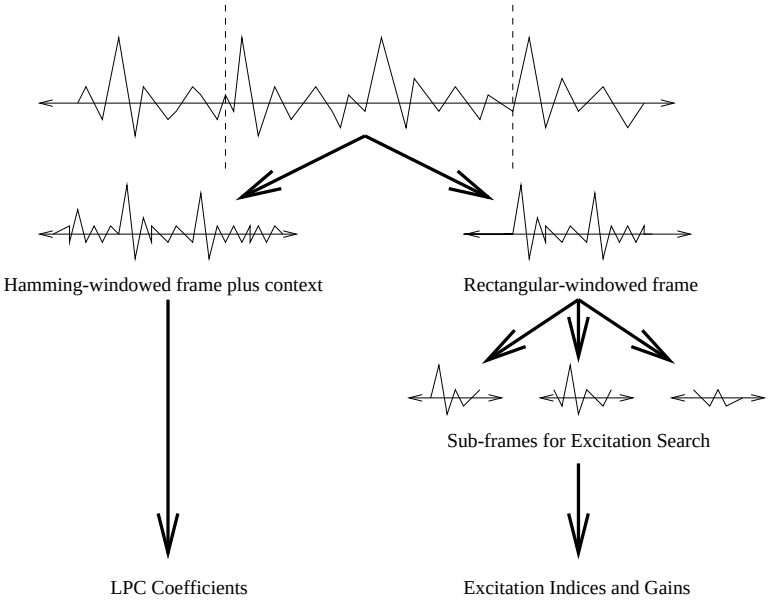

Figure 7.20: The frame/sub-frame structure of most LPC analysis by synthesis coders.

- The Solution:

  LPC coefficients are calculated using a longer "frame" (often a Hamming-windowed "frame plus context" equal to about twice the frame length). Excitation vectors are calculated using "sub-frames," with typically 3-4 sub-frames per frame, as shown in figure 7.15.3.

The original signal $s(n)$ and coded signal $\hat{s}(n)$ are often written as row vectors S and $\hat{S}$, where the dimension L is the length of a sub-frame:

$$S = [s(n), \ldots, s(n+L-1)], \qquad \hat{S} = [\hat{s}(n), \ldots, \hat{s}(n+L-1)] \tag{7.165}$$

ZSR and ZIR

$\hat{s}_k(n)$ can be written in terms of the infinite-length LPC impulse response, $h(n)$:

$$\hat{s}_k(n) = \sum_{i=0}^{\infty} h(i)\hat{d}_k(n-i) \tag{7.166}$$

Suppose that $\hat{s}(n)$ has already been computed for $n < 0$, and the coder is now in the process of choosing k_{opt} for the sub-frame $0 \le n \le L-1$. The sum above can be divided into two parts: a part which depends on k, and a part which does not:

$$\hat{s}_k(n) = \sum_{i=n+1}^{\infty} h(i)\hat{d}(n-i) + \sum_{i=0}^{n} h(i)\hat{d}_k(n-i) \tag{7.167}$$

or, in vector notation,

$$\hat{S} = \hat{S}_m + UH \tag{7.168}$$

where $\hat{S}_m$ is the "zero-input response" (ZIR) of the LPC filter:

$$\hat{S}_m = [\hat{s}_m(n), \ldots, \hat{s}_m(n+L-1)], \qquad \hat{s}_m(n) = \sum_{i=n+1}^{\infty} h(i)\hat{d}(n-i) = 0 - \sum_{k=1}^{p} a_k \hat{s}_m(n-k) \tag{7.169}$$

and UH is the "zero-state response" (ZSR) of $h(n)$ to the input $u(n)$:

$$H = \begin{bmatrix} h(0) & h(1) & \dots & h(L-1) \\ 0 & h(0) & \dots & h(L-2) \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & h(0) \end{bmatrix}, \quad U = [u(0), \dots, u(L-1)] \tag{7.170}$$

Finding the Gain

The coding error vector E can be defined as

$$E = \hat{S} - S = UH - \tilde{S} \tag{7.171}$$

where the "reference vector" $\tilde{S}$ is

$$\tilde{S} = S - \hat{S}_m \tag{7.172}$$

Suppose that the LPC excitation vector U is the sum of a set of "shape vectors" X_k, weighted by gain terms g_k:

$$U = GX, \quad G = [g_1, g_2, \dots], \quad X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \end{bmatrix} \tag{7.173}$$

The sum-squared error is written:

$$\sum_{n=0}^{L-1} e^2(n) = |E|^2 = |GXH - \tilde{S}|^2 = |GXH|^2 - 2GXH\tilde{S}' + |\tilde{S}|^2 \tag{7.174}$$

Suppose we define the following additional bits of notation:

$$\hat{S}_X \equiv XH, \quad R_X = \hat{S}_X \tilde{S}', \quad \Sigma = \hat{S}_X \hat{S}'_X \tag{7.175}$$

Then the mean squared error is

$$|E_X|^2 = G\Sigma G' - 2GR_X + \tilde{S}\tilde{S}' \tag{7.176}$$

For any given set of shape vectors X, G is chosen so that $|E_X|^2$ is minimized:

$$\frac{\partial |E_X|^2}{\partial G} = 2\Sigma G' - 2R_X = 0 \tag{7.177}$$

which yields

$$G = R'_X \Sigma^{-1} \tag{7.178}$$

Finding the Optimum Set of Shape Vectors

If we plug in the minimum-MSE value of G into equation 7.176, we get

$$|E_X|^2 = \tilde{S}\tilde{S}' - R'_X \Sigma^{-1} R_X \tag{7.179}$$

So, in order to minimize $|E_k|^2$, we choose the shape vectors X in order to maximize the covariance-weighted sum of correlations,

$$X_{opt} = \arg\max \left(R'_X \Sigma^{-1} R_X \right) \tag{7.180}$$

Example: One Shape Vector

For example, imagine a coder in which X is just a row vector:

$$X = [x(0), \ldots, \quad x(L-1)] \tag{7.181}$$

Therefore $\hat{S}_X$ is just a row vector:

$$\hat{S}_X = XH = [\hat{s}_x(0), \ldots, \quad \hat{s}_x(L-1)] \tag{7.182}$$

The "correlation" R_X is really just the correlation between the vectors $\hat{S}_X$ and $\tilde{S}$:

$$R_X = \hat{S}_X \tilde{S}' = \sum_{n=0}^{L-1} \hat{s}_x(n)\tilde{s}(n) \tag{7.183}$$

The "covariance" Σ_X is likewise just the variance of the vector $\hat{S}_X$:

$$\Sigma = \hat{S}_X \hat{S}'_X = \sum_{n=0}^{L-1} \hat{s}_x^2(n) \tag{7.184}$$

The mean-squared error associated with any particular choice of X is

$$|E_x|^2 = \tilde{S}\tilde{S}' - R'_X \Sigma^{-1} R_X = \sum_{n=0}^{L-1} \tilde{s}^2(n) - \frac{R_X^2}{\Sigma} \tag{7.185}$$

Finally, the optimum gain $G = g_1$ is the ratio of two scalars:

$$G = g_1 = R'_X \Sigma^{-1} = \frac{R_x}{\Sigma} \tag{7.186}$$

7.15.4 Self-Excited LPC

Excitation from Past Samples of the Excitation

Suppose that the pitch prediction filter is run with no inputs, and that we call the pitch prediction output $u_D(n)$:

$$u_D(n) = \sum_{i=-I}^{I} b_i u(n - (D+i)) \tag{7.187}$$

This is like forming the excitation vector U_D as a weighted sum of past excitation samples:

$$U_D = BX_D \tag{7.188}$$

$$X_D = \begin{bmatrix} u(-(D-I)) & \ldots & u((L-1)-(D-I)) \\ \vdots & \vdots & \vdots \\ u(-D) & \ldots & u((L-1)-D) \\ \vdots & \vdots & \vdots \\ u(-(D+I)) & \ldots & u((L-1)-(D+I)) \end{bmatrix}, \qquad B = [b_{-I}, \ldots, \ b_0, \ldots, \ b_I] \tag{7.189}$$

- At the beginning of the sentence, X must be initialized somehow. Typically we just initialize it with random noise.
- SELP is often buzzy during unvoiced regions, because the excitation $u(n)$ is *always* periodic. The effect is alleviated somewhat, because the period D changes from sub-frame to sub-frame.
- The search for D is easiest if D is always larger than the sub-frame length L. For many speakers (e.g. females with high pitch), L will be longer than a single pitch period. Fortunately, pitch prediction works pretty well if DT is a small integer multiple of T_0, e.g.

$$DT = T_0 \quad \text{or} \quad 2T_0 \quad \text{or} \quad 3T_0 \tag{7.190}$$

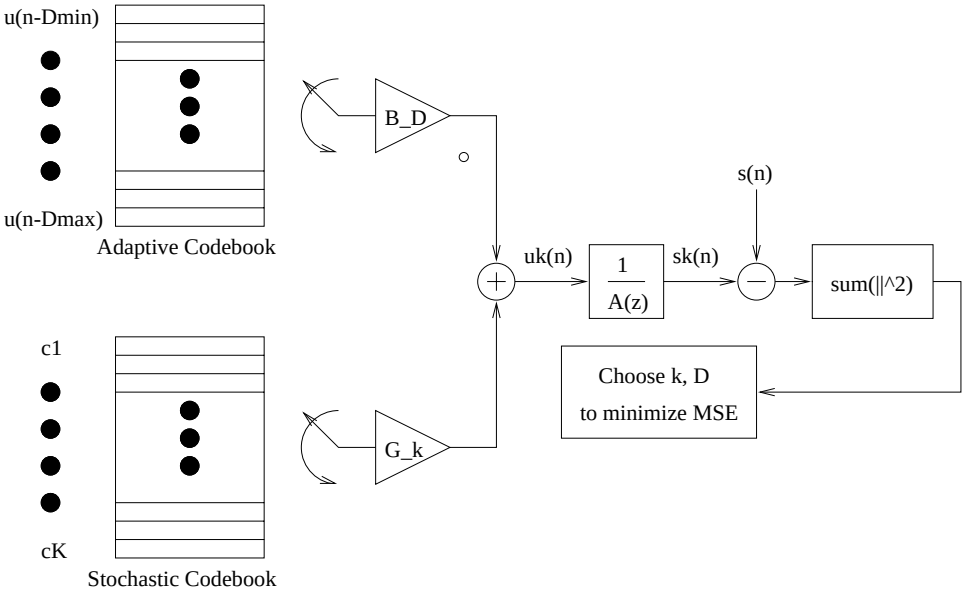

Figure 7.21: An LPC analysis by synthesis coder with two codebooks: an "adaptive" codebook, which represents the pitch periodicity, and a "stochastic" codebook, which represents the unpredictable innovations in each speech frame.

7.15.5 Multi-Vector LPC-AS

SELP sometimes produces speech with a kind of "buzzy" quality. This effect can be reduced by adding some new information in each sub-frame:

$$u_{M,k}(n) = \sum_{m=1}^{M} g_m c_{k_m}(n) + \sum_{i=-I}^{I} b_i u(n-(D+i)) \tag{7.191}$$

The signals $c_{k_m}(n)$ are called "innovation vectors," because they represent the new information in the current sub-frame which can not be deduced from the previous sub-frame. The numbers $k_1, \ldots, k_M$ are indices into a fixed codebook of innovation vectors which is known to both the sender and the receiver. In the original CELP (codebook-excited LPC) algorithm, the codebook consists of K Gaussian random vectors. In MPLPC (multipulse LPC), the codevectors are impulses:

$$\text{CELP:} \quad c_k(n) \sim \mathcal{N}(0,1), \quad 0 \le k \le K-1 \tag{7.192}$$

$$\text{MPLPC:} \quad c_k(n) = \delta(k), \quad 0 \le k \le L-1 \tag{7.193}$$

$$\tag{7.194}$$

In CELP systems, the fixed codebook is called the "stochastic codebook," because it is composed of Gaussian random noise vectors. The pitch prediction vectors $u(n-(D+i))$ are said to compose an "adaptive codebook," the contents of which change adaptively from sub-frame to sub-frame. The total structure looks like the coder in figure 7.15.5.

Gain Calculation: General Formulation

The gain in a mixed-codebook system can be calculated by extending the gain calculation of SELP, as follows:

$$U_{M,k} = G X_{M,k} \tag{7.195}$$

$$X_M = \begin{bmatrix} u(-(D-I)) & \dots & u((L-1)-(D-I)) \\ \vdots & \vdots & \vdots \\ u(-D) & \dots & u((L-1)-D) \\ \vdots & \vdots & \vdots \\ u(-(D+I)) & \dots & u((L-1)-(D+I)) \\ c_{k_1}(0) & \dots & c_{k_1}(L-1) \\ c_{k_2}(0) & \dots & c_{k_2}(L-1) \\ \vdots & \vdots & \vdots \\ c_{k_M}(0) & \dots & c_{k_M}(L-1) \end{bmatrix}, \qquad G = [b_{-I}, \dots, b_0, \dots, b_I, g_1, \dots, g_M] \tag{7.196}$$

With this structure, the optimum gain in a multi-vector excitation system such as CELP or MPLPC has the same form as the gain in SELP:

$$G = R'_X \Sigma^{-1} \tag{7.197}$$

$$R_X = \hat{S}_X \tilde{S}', \quad \Sigma = \hat{S}_X \hat{S}'_X, \quad \hat{S}_X = X_M H \tag{7.198}$$

Iterative Optimization

Usually, you don't want to simultaneously search the adaptive and stochastic codebooks. If there are N_D possible pitch vectors, then global optimization in the following systems would require:

- Normal CELP ($M = 1$): Calculate MSE KN_D times.
- Vector-sum LPC (VSELP: $M = 2$): Calculate MSE $K^2 N_D$ times.
- MPLPC ($M = 4 - 6$ or more): Calculate MSE $K^M N_D$ times.

Most practical systems use some type of iterative optimization procedure, in which the pitch delay D is optimized first, and then the first codebook index k_1, and then the second codebook index. There are many possible iterative optimization procedures, each with its own tradeoffs; one possible algorithm looks like this:

$$\text{For } i = 1, \dots, M \text{ do:} \tag{7.199}$$

1. Find the optimum vector C_{k_i} which minimizes

$$|E_{k_i}|^2 = \tilde{S}_i \tilde{S}'_i - \frac{r_k^2}{\sigma_k^2} \tag{7.200}$$

$$r_k = C_{k_i} H \tilde{S}', \quad \sigma_k^2 = C_{k_i} H H' C'_{k_i} \tag{7.201}$$

2. Append C_{k_i} to the excitation matrix:

$$X_i = \begin{bmatrix} X_{i-1} \\ C_{k_i} \end{bmatrix} \tag{7.202}$$

3. Recalculate the vector gain:

$$G_i = R_X \Sigma^{-1}, \quad R_X, \ \Sigma \text{ based on } \hat{S}_{X_i} = X_i H \tag{7.203}$$

4. Create a reference vector $\tilde{S}_{i+1}$ for the next iteration:

$$\tilde{S}_{i+1} = \tilde{S} - G_i \hat{S}_{X_i} \tag{7.204}$$

Kondoz suggests an iterative algorithm which differs from this one in that G_i is not re-calculated after each iteration; instead, it is assumed (during the iterative procedure) that

$$G_i \approx [G_{i-1}, \quad g_{k_i}] \tag{7.205}$$

$$\tilde{S}_{i+1} \approx \tilde{S}_i - g_{k_i} C_k H \tag{7.206}$$

$$\tag{7.207}$$

7.15.6 Perceptual Error Weighting

At the low bit rates of typical LPC-AS coders, perceptual error weighting is very important. Fortunately, it is also easy. Remember that the synthesized speech signal has the following spectrum:

$$\hat{S}(z) = \frac{U(z)}{A(z)} \tag{7.208}$$

If quantization error is weighted using a filter of the form

$$W(z) = \frac{A(z)}{A(z/\alpha)} \tag{7.209}$$

then the "perceptually weighted error" $E_w(z)$ has the form

$$E_w(z) = W(z)(\hat{S}(z) - S(z)) = \hat{S}_w(z) - S_w(z) \tag{7.210}$$

$$S_w(z) = W(z)S(z), \quad \hat{S}_w(z) \equiv \frac{U(z)}{A(z/\alpha)} \tag{7.211}$$

In the time domain, $\hat{s}_w(n)$ is the sum of a ZIR and a ZSR:

$$\hat{S}_w = \hat{S}_{w,m} + U_k H_w \tag{7.212}$$

where H_w and $S_{w,m}$ are analogous to H and S_m in unweighted LPC-AS:

$$H_w(z) = \frac{1}{A(z/\alpha)}, \qquad \hat{s}_m(n) = \sum_{k=1}^{p} a_k \alpha^k \hat{s}_{w,m}(n-k) \tag{7.213}$$

Thus, perceptual weighting in an LPC-AS system consists of the following steps:

1. Filter the input signal $s(n)$ using $W(z)$ to create a weighted input signal S_w.
2. Use the filter $H_w(z)$ instead of $H(z)$ to create $\hat{S}_{w,m}$ and H_w.
3. Calculate the weighted reference vector:

$$\tilde{S}_w = S_w - \hat{S}_{w,m} \tag{7.214}$$

4. For each possible set of shape vectors X, compute the following quantities:

$$\hat{S}_{x,w} = X H_w, \qquad R_{x,w} = \hat{S}_{x,w}\tilde{S}'_w, \qquad \Sigma = \hat{S}_{x,w}\hat{S}'_{x,w} \tag{7.215}$$

5. Choose the set of shape vectors which maximizes the weighted correlation:

$$X_{opt} = \arg\max_X \left(R'_{x,w}\Sigma^{-1}R_{x,w}\right) \tag{7.216}$$

7.16 Exercises

1. **Scalar Quantization**

   (a) **Implement a uniform quantizer**

   Write a matlab function, **XHAT = linpcm(X, XMAX, B)**, which quantizes **X** using a **B**-bit linear PCM quantizer, with maximum output values of +/- **XMAX**.

   (b) **Adjusting the bit rate in linear PCM**

   Quantize the male sentence using linear PCM with 3, 4, 5, 6, and 7 bit quantization. In each case, set **XMAX=max(abs(male_sent))**. Plot SNR in decibels as a function of the number of bits.

   Rabiner & Schafer comment, in section 5.3.1, that the error signal $e(n) = \hat{x}(n) - x(n)$ at low bit rates is correlated with the speech signal. Is this true? Listen to the error signals $e(n)$ produced at each bit rate. At low bit rates, $e(n)$ may be so highly correlated with $x(n)$ that it is actually intelligible. Are any of your error signals intelligible? Which ones?

   (c) **Implement companded PCM**

   Write a function, **T = mulaw(X, MU, XMAX)**, which compresses **X** using μ-law compression (R&S 5.3.2). Write another function, **X = invmulaw(T, MU, XMAX)**, which expands **t** to get **x** again. Check your work by finding

   XHAT=invmulaw(mulaw(sent_male, 255, XMAX), 255, XMAX)

   Confirm that **XHAT** is identical to **sent_male**.[1]

   (d) **Adjusting the bit rate in companded PCM**

   Quantize **T** using the function **linpcm**, and expand the quantized version to obtain a μ-law quantized version of the input sentence. In this way, create μ-law quantized sentences using 3, 4, 5, 6, and 7 bit quantizers, with a value of $\mu = 255$. Plot SNR as a function of the number of bits, and compare to the values obtained with linear PCM.

   (e) **Calculating SEGSNR**

   Complete the skeleton matlab file **segsnr(X, E, N)** found in **~ee214a/coding**. This function should divide the signal **X** and error **E** into frames of **N** samples each, compute the SNR (in decibels) within each frame, and return the average segmental SNR.

   Compute the SEGSNR for each of the quantizers in sections 1.2 and 1.4 using 15ms frames, and plot them. How do SEGSNR and SNR differ?

   Sort the quantized utterances (including both linear and companded PCM) on a scale from lowest to highest SNR. Now sort them from lowest to highest SEGSNR. Finally, sort them on a scale from "worst sounding" to "best sounding." Which of the two objective measures (SNR or SEGSNR) is a better representation of the subjective speech quality?

2. **Minimum-MSE Quantization, Gaussian Distribution**

   Consider a signal $x(n)$ with a unit-variance Gaussian distribution:

   $$p_{x(n)}(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{7.217}$$

   (a) Design a 1-bit-per-sample scalar quantizer for $x(n)$. Find the decision boundary and reconstruction levels which minimize the expected mean squared error.

   (b) Generate a matlab data vector $x(n)$ consisting of 1000 Gaussian random numbers. Cluster your data into two regions using either the K-means algorithm or the binary-split algorithm.

   What are the centroids of your two regions? What is the decision boundary?

[1] ...with allowance for floating-point roundoff error.

(c) Comment on any differences between your answers in parts (a) and (b), and explain them if you can. You may or may not find it useful to calculate the mean and the histogram of your data vector $x(n)$.

3. **Vocoder Synthesis**

   Write a matlab function **DHAT = vocode(N0, B, THETA)** which creates a simulated LPC excitation signal **DHAT**.

   In frames for which **B(i)<THETA**, the excitation signal should be unit-variance uncorrelated Gaussian noise (use the **randn** function). In frames for which **B(i)>THETA**, the excitation signal should be an impulse train with a period of **N0** samples, starting at some sample number called **START**.

   When two neighboring frames are voiced, make sure to set the variable **START** so that the spacing between the last impulse in one frame and the first impulse in the next frame is a valid pitch period — otherwise, you will hear a click at every frame boundary! Also, be careful to set the amplitude of the impulse train so that the average energy of the excitation signal in each frame is always

$$\frac{1}{\text{STEP}} \sum_{n=1}^{\text{STEP}} \hat{d}^2(n) = 1 \tag{7.218}$$

   Create a signal **DHAT** using a value of **THETA** which you think will best divide voiced and unvoiced segments, and using values of **N0** and **B** calculated from a natural utterance. Pass the output of the **vocode** function through your **lpcsyn** function. Listen to the sentence. How does the coding quality compare to 3 and 4 bit linear PCM? How about companded PCM? Do certain parts of the sentence sound better than others?

   Calculate the SNR and SEGSNR of your vocoded utterance. Is the SEGSNR of an LPC vocoder better or worse than the SEGSNR of a PCM coder with similar perceptual quality? Why?

4. **Pitch-Prediction Vocoder**

   Write a matlab function **DHAT = ppv(N0, B)** which creates a synthesized LPC excitation signal by filtering white noise through a "pitch-prediction filter." Be careful to carry filter state forward from one frame to the next.

   Use **ppv** to create an excitation signal **DHAT**, and pass it to **lpcsyn** to synthesize a copy of your input sentence. How does this version of the sentence sound? How does it compare to the regular vocoder? Do some parts of the sentence sound better? Do some parts sound worse?

5. **Voice Modification**

   Synthesize a sentence, using either the regular vocoder or the pitch-prediction vocoder, but use one of the following modified parameters in place of the correct parameter. What does the sentence sound like? Why?

```
N0_modified = median(N0) * ones(size(N0));
N0_modified = round(0.5*N0);
B_modified = zeros(size(B));
```

6. **Self-Excited LPC**

   In Self-Excited LPC (SELP), the LPC excitation vector $u(n)$ is created by scaling and adding past samples of itself:

$$u(n) = \sum_{i=-I}^{I} b_i u(n - D - i) \quad \Leftrightarrow \quad U(z) = \frac{1}{P(z)} = \frac{1}{1 - \sum_{i=-I}^{I} b_i z^{-(D+i)}} \tag{7.219}$$

   In order to obtain useful LPC excitation values, the samples of $u(n)$ for $n < 0$ (before the beginning of the sentence) are initialized using samples of Gaussian white noise.

(a) **Pitch Prediction Coefficients for a Perfectly Periodic Signal**

Self-excited LPC implicitly assumes that the ideal continuous-time excitation signal $u_c(t)$ is perfectly periodic, but that the period may not be an exact multiple of the sampling period T:

$$U_c(s) = \frac{1}{P_c(s)} = \frac{1}{1 - e^{-sT_0}}, \quad T_0 = DT + \epsilon \tag{7.220}$$

> Quest. 1: Choose coefficients b_i such that $p(n) = p_a(nT)$, where $p_a(t)$ is obtained by lowpass-filtering $p_c(t)$ at the Nyquist rate. You may assume that $I = \infty$.

(b) **Pitch Prediction Coefficients for a Real-Life Signal**

Equation 7.219 can be written in vector form as follows:

$$U = BX \tag{7.221}$$

$$X = \begin{bmatrix} u(n-(D-1)) & \dots & u(n-(D-1)+L-1) \\ u(n-D) & \dots & u(n-D+L-1) \\ u(n-(D+1)) & \dots & u(n-(D+1)+L-1) \end{bmatrix}, \qquad B = [b_{-1},\ b_0,\ b_1] \tag{7.222}$$

> Quest. 2: Find the value of B which minimizes the squared quantization error $|E|^2 = |UH - \tilde{S}|^2$. Hint: The derivation follows Kondoz equations 6.21-6.28.

(c) **Sub-Frames**

The pitch prediction delay D is chosen by searching over some range of possible pitch periods, $D_{min} \le D \le D_{max}$, and choosing the value of D which minimizes the squared quantization error $|E|^2$. If there is no overlap between the samples of X and U, that is, if $D > L$ for all possible D, then it is possible to compute the squared error $|E|^2$ using pure matrix computation. Matrix computation reduces the computational complexity, and (in matlab) it reduces the programming time by a lot; therefore all practical systems set the minimum value of D to $D_{min} = L + 1$.

LPC is almost never computed using frames of size $L < D$, so analysis by synthesis systems often break up each LPC frame into M sub-frames, where M is some integer:

$$L = \frac{L_{LPC}}{M} \tag{7.223}$$

LPC coefficients are thus calculated using frames of $\frac{L_{LPC}}{F_s} \approx 20-30$ms in length, but the excitation parameters D and B are calculated using smaller frames of length L.

Even if $L = L_{LPC}/M$, very short pitch periods may be shorter than one sub-frame in length. Fortunately, pitch prediction, as shown in equation 7.219, works pretty well if D is any small integer multiple of the pitch period, for example $D = 2T_0/T$. The accuracy of pitch prediction drops slightly, however, so it is best for D to be no more than 2-3 times the pitch period.

> Quest. 3: What is the largest sub-frame size L which will allow you to represent pitches up to $F_0 = 250$Hz with a delay D of no more than twice the pitch period?

(d) **Implementation**

Implement self-excited LPC.

- For each sub-frame, calculate all of the possible values of U_D for $D_{min} \le D \le D_{max}$, where $D_{min} = L + 1$ and D_{max} is the pitch period of a low-pitched male speaker ($F_0 \approx 80Hz$).

- For each value of U_D, calculate the squared quantization error $|E|^2$.
- Choose the value of D which minimizes $|E|^2$.

Be sure to carry forward, from frame to frame, both the LPC filter states, and enough delayed samples of U to allow you to perform pitch prediction. The LPC filter states should be initialized to zero at the beginning of the sentence, but the pitch prediction samples should be initially set to unit-variance Gaussian white noise.

Quantize D and B. Examine the segmental SNR and sound quality of your coder using both quantized and unquantized D and B.

> Quest. 4: What segmental SNR do you obtain using unquantized D and B? With quantized D and B? How many bits per second do you need to transmit $A(z)$, D, and B?

7. Multi-Vector LPC-AS

By its nature, SELP tends to represent periodic signals well, but fails to represent noisy signals, or even voiced signals with a glottal wave-shape that changes over time.

(a) Stochastic Codevectors

We can represent aperiodic and partially periodic signals by extending the excitation matrix X as follows:

$$X_M = \begin{bmatrix} u(n-(D-1)) & \dots & u(n-(D-1)+L-1) \\ u(n-D) & \dots & u(n-D+L-1) \\ u(n-(D+1)) & \dots & u(n-(D+1)+L-1) \\ c_{k1}(0) & \dots & c_{k1}(L-1) \\ c_{k2}(0) & \dots & c_{k2}(L-1) \\ \vdots & \vdots & \vdots \\ c_{kM}(0) & \dots & c_{kM}(L-1) \end{bmatrix} \tag{7.224}$$

Where $c_{k1}(m)$ are the samples of a "code vector" C_{k1} which is chosen from a set of K possible code vectors in order to minimize the squared error,

$$|E|^2 = |\hat{S} - S|^2 = |BXH - \tilde{S}|^2 \tag{7.225}$$

In the original CELP algorithm, the codebook consists of K Gaussian random vectors. In MPLPC, the codevectors are impulses:

$$\text{CELP:} \quad c_k(m) \sim \mathcal{N}(0,1), \quad 0 \le k \le K-1 \tag{7.226}$$

$$\text{MPLPC:} \quad c_k(m) = \delta(k), \quad 0 \le k \le L-1 \tag{7.227}$$

$$\tag{7.228}$$

> Quest. 1: Suppose you are creating an MPLPC coder in which the X matrix you used in your SELP coder will be augmented by 5 impulse vectors, numbered C_{k1} through C_{k5}. If you want to find the globally optimum combination of D, $k1$, $k2$, $k3$, $k4$, and $k5$, how many times do you need to evaluate equation 7.225?

(b) Iterative Optimization

In order to avoid the impossible computational complexity of a global search, many CELP coders and all MPLPC coders perform an iterative search. In an iterative search, the best pitch delay D is calculated as in the SELP coder, resulting in a $3 \times L$ matrix X_0, and a 3-element gain vector B_0:

$$\tilde{S} \approx B_0 \hat{S}_0 = B_0 X_0 H \tag{7.229}$$

Given X_0 and B_0, the fourth excitation vector can be chosen by first creating a reference vector $\tilde{S}_1$,

$$\tilde{S}_1 = \tilde{S} - B_0 X_0 H \tag{7.230}$$

$\tilde{S}_1$ represents the part of $\tilde{S}$ which is not well represented by $B_0\hat{S}_0$; in fact, $\tilde{S}_1$ is the part of $\tilde{S}$ which is orthogonal to $B_0\hat{S}_0$. Therefore, the optimum fourth excitation vector is the one which minimizes

$$|E_1^k|^2 = |\tilde{S}_1 - g_1 C_{k1} H|^2 \tag{7.231}$$

Once the optimum value of $k1$ has been found, the gain vector B_1 must be recomputed, in order to minimize the total squared error

$$|E_1|^2 = |\tilde{S} - B_1 X_1 H|^2 \tag{7.232}$$

> Quest. 2: Find the optimum value of g_1 as a function of the reference vector $\tilde{S}_1$ and the codebook vector C_{k1}. Find the optimum value of B_1 as a function of $\tilde{S}$ and X_1. Show that, in general, $B_1 \neq [B_0 g_1]$.

Once B_1 and X_1 have been computed, the procedure outlined above can be repeated, as often as necessary. Typically, the number of pulse vectors required to achieve good quality using MPLPC is a fraction of L. Classical CELP coders only use one stochastic codevector, but the VSELP algorithm used in most cellular telephone standards uses two.

(c) **Implementation**

Add a stochastic codebook to your SELP coder, in order to create a generalized LPC-AS coder (be sure to save your SELP coder under a different name, so that you can go back to it if you need to!) Your generalized LPC-AS coder should accept as input a $K \times L$ codebook matrix C, each of whose rows contains one stochastic code vector C_k. You should also accept an input argument M which tells the function how many stochastic codevectors to find. The final command line might look something like this:

```
[YMAT(i,:), filter_states, ...] = myfunc(XMAT(i,:), filter_states, ..., C, M);
```

Create an MPLPC codebook (consisting of impulses) and a CELP codebook (consisting of about 1000 Gaussian random vectors). Test your coder using both CELP and MPLPC codebooks.

> Quest. 3: Plot the segmental SNR of your coded speech as a function of the number of stochastic code vectors, M, for both MPLPC and CELP, with the codebook gain vector B still unquantized. Comment on any differences between MPLPC and CELP.

Quantize the codebook gain vector B.

> Quest. 4: Quantize all of your gain terms, then choose coder configurations for both CELP and MPLPC which produce reasonable-sounding speech. For both of these two configurations, what is the total bit rate, in bits per second?

8. **Perceptual Weighting**

When humans listen to a coded speech signal, some components of the quantization noise are masked by peaks in the signal spectrum. Low bit rate LPC-AS systems therefore often weight the error, in order to reduce the importance of error components near a spectral peak. The most common error weighting filter is based on the LPC analysis filter $A(z)$:

$$E_w(z) = E(z)\frac{A(z)}{A(z/\alpha)} \tag{7.233}$$

Implement perceptual error weighting in one of your LPC-AS coders in the most efficient way possible.

Compare the segmental SNR achieved by a CELP or an MPLPC coder both with and without perceptual error weighting. Which gives better SEGSNR? Listen to the two reconstructed speech signals. Which sounds better?