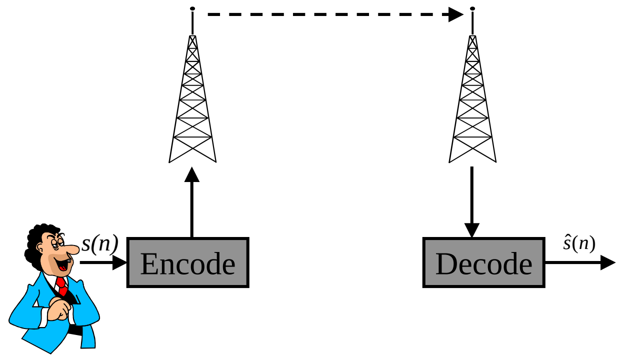
Lecture 8

Speech Coding

– Objectives of Speech Coding
– Quality versus bit rate
– Uniform Quantization
  • Quantisation Noise
– Non-uniform Quantisation
– Adaptive Quantization



Speech Coding

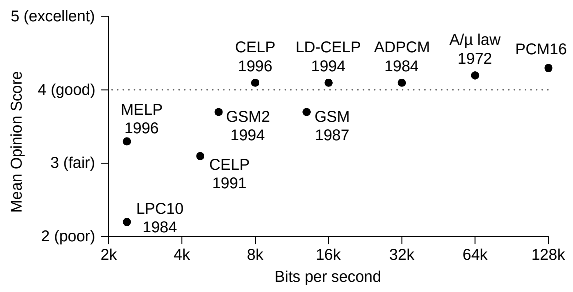
<u>Speech Coding Objectives</u>:
– High perceived quality
– High measured intelligibility
– Low bit rate (bits per second of speech)
– Low computational requirement (MIPS)
– Robustness to successive encode/decode cycles
– Robustness to transmission errors

Objectives for real-time only:
– Low coding/decoding delay (ms)
– Work with non-speech signals (e.g. touch tone)

Subjective Assessment

- MOS (Mean Opinion Score): a panel of test listeners rates the quality 1 to 5.
  1=bad, 2=poor, 3=fair, 4=good, 5=excellent.
- DRT (Diagnostic Rhyme Test): listeners choose between pairs of rhyming words (e.g. veal/feel)
- DAM (Diagnostic Acceptability Measure): Trained listeners judge various factors e.g. muffledness, buzziness, intelligibility

Quality versus data rate (8kHz sample rate)



music:

Objective Assessment

Not terribly closely related to subjective quality.

- **Segmental SNR**: average value of $10\log_{10}(E_{\text{speech}}/E_{\text{error}})$ evaluated in 20 ms frames.
  No good for coding schemes that can introduce delays as a one-sample time shift causes a huge decrease in SNR.
- **Spectral distances**
  Based on the power spectrum of the original and coded-decoded speech signals, $P(\omega)$ and $Q(\omega)$. $<\ldots>$ denotes the average over the frequency range $0 \ldots 2\pi$.

Itakura Distance

$$\log\left(\left\langle \frac{P(\omega)}{Q(\omega)} \right\rangle\right) - \left\langle \log\left(\frac{P(\omega)}{Q(\omega)}\right)\right\rangle$$

Itakura-Saito Distance

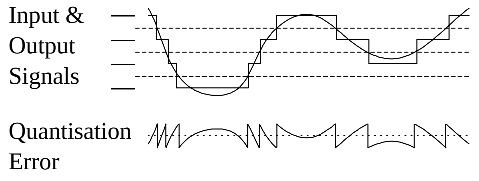
$$\left\langle \frac{P(\omega)}{Q(\omega)} - \log\left(\frac{P(\omega)}{Q(\omega)}\right) - 1 \right\rangle$$

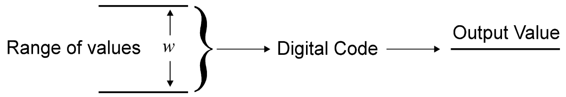
Both distances are evaluated over 20 ms frames and averaged.

If you do LPC analysis on the original and coded speech, both these distances may also be calculated directly from the LPC coefficients.

Neither is a true distance metric since they are asymmetrical: $d(P,Q) \neq d(Q,P)$

Quantization Errors





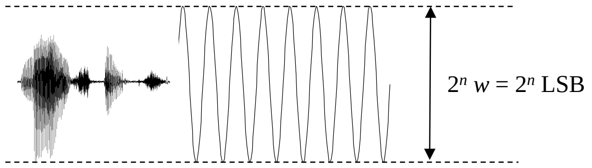
Coding/Decoding introduces a quantisation error of $\pm\frac{1}{2}w$

If input values are uniformly distributed within the bin, the mean square quantisation error is:

$$\int_{-\frac{1}{2}w}^{+\frac{1}{2}w} x^2 \frac{dx}{w} = \left[\frac{x^3}{3w}\right]_{-\frac{1}{2}w}^{+\frac{1}{2}w} = \frac{w^2}{12} \quad \Rightarrow \text{rms error} = 0.289w$$

If the quantisation levels are uniformly spaced, w is the separation between adjacent output values and is called a Least Significant Bit (LSB)

Linear PCM – Pulse Code Modulation



RMS Quantization Noise is $0.289w = 0.289$ LSB

Sine Wave SNR

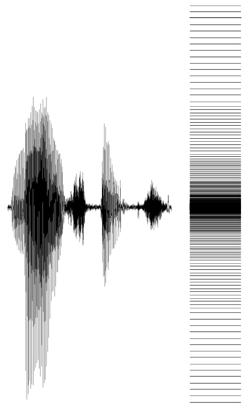
Full scale sine wave has peak of $0.5\, w \times 2^n$ and an rms value of $0.354\, w \times 2^n$

$$\text{SNR} = 20\log_{10}\left(\frac{\text{rms of maximal sine wave}}{\text{rms of quantisation noise}}\right)$$
$$= 20\log_{10}\left(\frac{0.354}{0.289} \times 2^n\right) = 20\log_{10}(1.22) + 20n\log_{10}(2)$$
$$= 1.76 + 6.02n \text{ dB}$$

Speech SNR

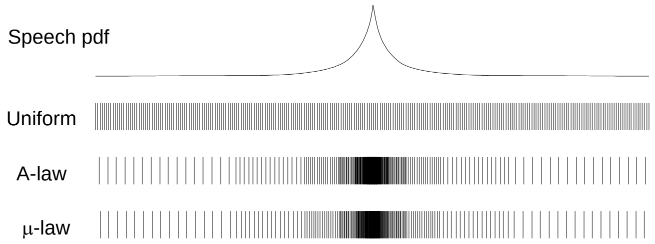
RMS value of speech or music signal is 10 to 20 dB less than a sine wave with the same peak amplitude. SNR is 10 to 20 dB worse.

Non-Uniform Quantization



– RMS quantization error = 0.29 ×Quantization step
– Low level signals are commoner than high level ones
– Conclusion:
  Make quantization steps closer together at the commoner signal levels.

Non-uniform PCM @ 64 kb/s (f_s = 8kHz)



Instead of linear coding, use floating point with smaller intervals for more frequently occuring signal levels:

s = sign bit (1 = +ve)
e = 3-bit exponent
m = 4-bit mantissa

s	e	e	e	m	m	m	m

Range of values ≈ ±4095
0dB ≈ 2002(μ) or 2017(A) rms

μ-law (US standard):

Bin centres at $\pm\{(m+16\tfrac{1}{2})\,2^e - 16\tfrac{1}{2}\}$

Bin widths: 2^e

A-law (European standard):

Bin centres at $\pm(m+16\tfrac{1}{2})\,2^e$ for $e>0$ and $\pm(2m+1)$ for $e=0$

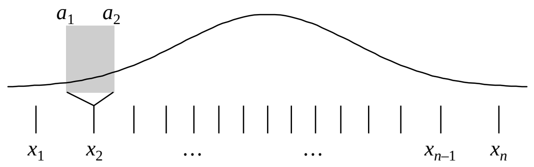
Bin widths: 2^e for $e>0$ and 2 for $e=0$

For transmission, A-law is XORed with \$55 and μ-law with \$7F

Performance: MOS=4.3, DRT=95, DAM=73, SNR ≤38dB

For sine waves > –30dB, SNR is almost constant at 38dB
At very low signal levels (–40dB) μ-law≈13 bit, A-law ≈12 bit linear

Quantising a Gaussian Signal



Input values from a_{i-1} to a_i are converted to x_i.
For minimum error we must have $a_i = ½(x_i + x_{i+1})$

For this range of x, the quantisation error is $q(x) = x_i - x$
The mean square quantisation error is

$$E = \int_{-\infty}^{+\infty} p(x)q^2(x)dx = \sum_{i=1}^{n} \int_{a_{i-1}}^{a_i} p(x)\left(x_i - x\right)^2 dx$$

Differentiating w.r.t. x_i : $\frac{\partial E}{\partial x_i} = \int_{a_{i-1}}^{a_i} -2\,p(x)\left(x - x_i\right)dx$

$$= 2x_i \int_{a_{i-1}}^{a_i} p(x)dx - 2\int_{a_{i-1}}^{a_i} xp(x)dx$$

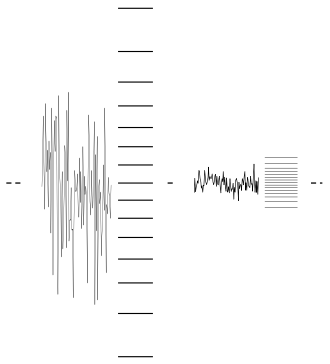
Find minimum error by setting the derivative to zero

$$x_i = \frac{\int_{a_{i-1}}^{a_i} xp(x)dx}{\int_{a_{i-1}}^{a_i} p(x)dx}$$ $\Leftrightarrow x_i$ is at the centroid of its bin.

To find optimal $\{x_i\}$, take an initial guess and iteratively use the above equation to improve their estimates.

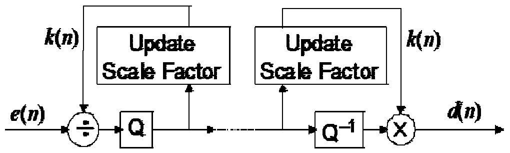
For 15 bins, SNR = 19.7dB

Adaptive Quantization



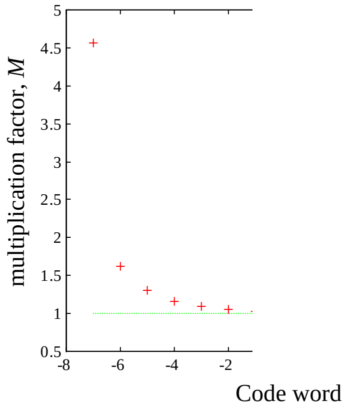
– We want to use smaller steps when the signal level is small
– We can adjust the step sizes automatically according to the signal level:
  - If almost all samples correspond to the central few quantisation levels, we must reduce the step size
  - If many samples correspond to high quantisation levels, we must increase the step sizes

Adaptive Quantization



- Transmitter:
  - Divide input signal by *k*
  - Q quantises to 15 levels
  - Transmit one of 15 code words
  - Update *k* to new value
- Receiver:
  - Inverse quantiser, Q^{-1} converts received code word to a number
  - Multiply by *k* to give output value
  - Update *k* to new value
- Update Algorithm
  - Decrease *k* whenever a small code word is transmitted/received
  - Increase *k* whenever a large code word is transmitted/received
  - Transmitter and Receiver will always have the same value of *k*

Scale Factor Updates

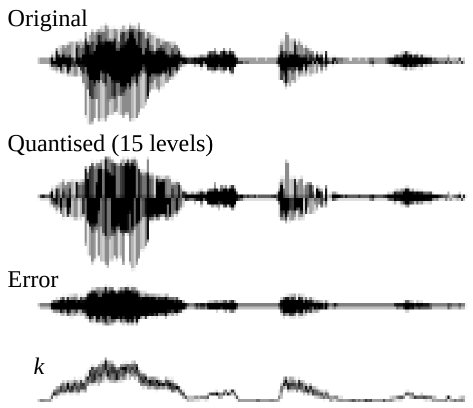


- k is decreased slightly for “zero” code word (× 0.98)
- k is increased rapidly for extreme code words (× 4.6)
- k is increased slightly for other code words
- Calculation done in log domain:

$$\log(k_n) = 0.97 \log(k_{n-1}) + \log(M)$$

“leakage factor” of 0.97 allows recovery from transmission errors.

Adaptive Quantization



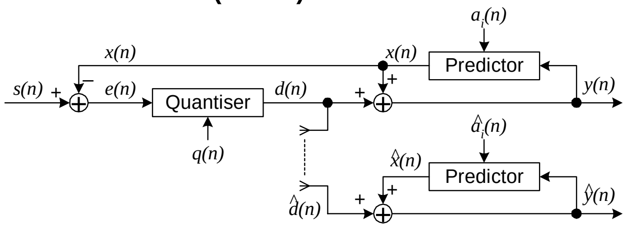
Small signal sections $\Rightarrow$ small $k \Rightarrow$ small errors

Lecture 9

Adaptive Differential PCM

– Differential Speech Coding
– ADPCM coding
  - Prediction Filter
  - Filter Adaptation
  - Stability Triangle

Differential PCM (DPCM)

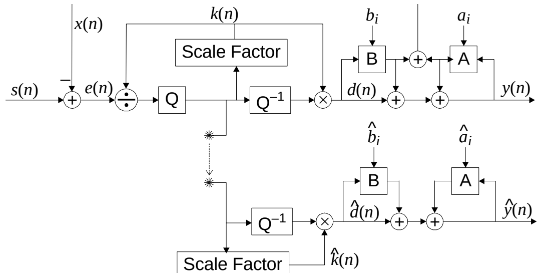


- The encoder contains a complete copy of the decoder
- If no transmission errors, $\hat{y}(n) \equiv y(n)$ etc
- Quantisation noise, $q(n) = d(n) - e(n)$
- Output $y(n) = x(n) + d(n) = s(n) - e(n) + d(n) = s(n) + q(n)$

$$SNR = \frac{E_s}{E_q} = \frac{E_s}{E_e} \times \frac{E_e}{E_q} = G_p \times \frac{E_e}{E_q}$$

- The predictor is chosen to maximize the *prediction gain*: G_p = signal energy ÷ prediction error energy.
- E_e/E_q is the quantiser SNR
- The a_i can be one of:
  - fixed constants
  - transmitted seperately
  - deduced from $d(n)$ and/or $y(n)$

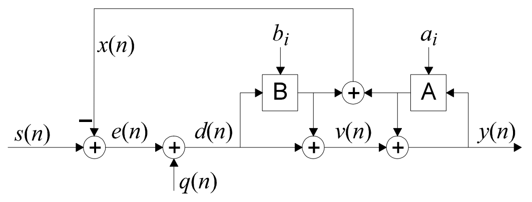
Adaptive Differential PCM (ADPCM) @ 32 kbit/s



- Quantiser has 15 levels ⇒ 4 bits × 8kHz = 32 kbit/s
- Code 0000 is never transmitted which makes transmission easier
- Quantisation levels assume $e(n) \div k(n)$ is Gaussian with unit variance. Scale factor $k(n)$ is adjusted to make this true
- The LPC filter, A, is only of order 2 so that it is easy to ensure stability.
- The FIR filter, B, is of order 6 and partially makes up for the short A.

Performance: MOS=4.1, DRT=94, DAM=68, 2 MIPS

Prediction Filter



Filters: $A(z) = a_1 z^{-1} + a_2 z^{-2}$

$$B(z) = b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} + b_5 z^{-5} + b_6 z^{-6}$$

Note that $\frac{\partial A}{\partial a_i} = \frac{\partial B}{\partial b_i} = z^{-i}$

Note that *A(z)* and *B(z)* filters involve only past values

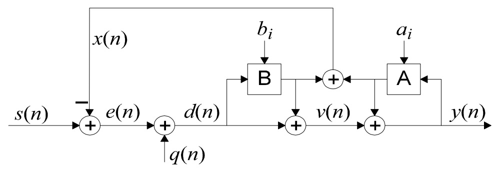
Signals: $Y = D + BD + AY \Rightarrow Y = \frac{1+B}{1-A} D$

$$X = BD + AY = Y - D = \frac{A+B}{1-A} D$$

$$D = S + Q - X = S + Q - Y + D \Rightarrow Y = S + Q$$

In the absence of transmission errors, the *y(n)* output at the receiver and transmitter will be identical and equal to the input speech plus the quantisation noise

Filter Adaptation



We adjust the b_i to reduce the signal $d^2(n)$ by moving them a little in the direction $-\frac{\partial}{\partial b_i}\left(d^2(n)\right) = -2d(n)\frac{\partial d(n)}{\partial b_i}$

We express D in terms of previous values of D:

$$D = S + Q - X = S + Q - \frac{A+B}{1-A} D$$

$$\frac{\partial D}{\partial b_i} = -z^{-i}\frac{1}{1-A} D \approx -z^{-i} D \Rightarrow \frac{\partial d(n)}{\partial b_i} \approx -d(n-i)$$

Hence we want to adjust b_i in the direction $d(n)d(n\text{-}i)$:

The function sgn() takes the sign of its argument. The 0.996 leakage factor assists in recovery from transmission errors and prevents b_i from exceeding ± 2.

$$b_i \leftarrow 0.996 b_i + 0.008 \times \mathrm{sgn}\left(d(n)d(n-i)\right)$$

Adjustment of a_i is similar but more ad-hoc:

$$a_1 \leftarrow 0.996 a_1 + 0.012 \times \mathrm{sgn}\left(v(n)v(n-1)\right)$$
$$a_2 \leftarrow 0.996 a_2 + 0.008 \times \mathrm{sgn}\left(v(n)v(n-2)\right) - 0.03 a_1 \times \mathrm{sgn}\left(v(n)v(n-1)\right)$$

Filter Stability

$\frac{1}{1-a_1z^{-1}-a_2z^{-2}}$ has poles at $½\left(a_1 \pm \sqrt{a_1^2+4a_2}\right)$

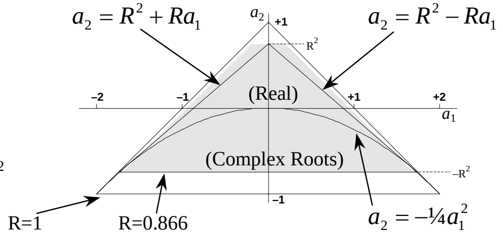
For stability, we make the poles have modulus less than R:

– Real: $a_1^2+4a_2>0$ and $\begin{cases} ½\left(a_1+\sqrt{a_1^2+4a_2}\right)<R \\ ½\left(a_1-\sqrt{a_1^2+4a_2}\right)>-R \end{cases}$

$$\Rightarrow \sqrt{a_1^2+4a_2}<2R\pm a_1$$
$$\Rightarrow a_1^2+4a_2<a_1^2\pm 4Ra_1+4R^2 \Rightarrow a_2<R^2\pm Ra_1$$

– Complex: $a_1^2+4a_2<0$ and $a_2>-R^2$

Shading:
$|a_2|<0.75$
$|a_1|<0.9375-a_2$



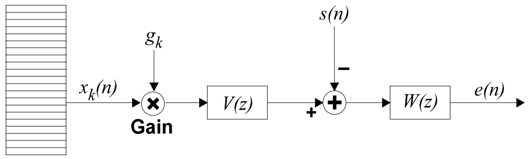
Poles more or less kept within a radius of 0.866. Prevents huge gains and coefficient sensitivity. Also means transients due to transmission errors die away quickly.

Lecture 10

Code-Excited Linear Prediction (CELP)

– Principles of CELP coding
– Adaptive and Stochastic Codebooks
– Perceptual Error Weighting
– Codebook searching

Code Excited Linear Prediction Coding (CELP)



Encoding

– LPC analysis $\rightarrow V(z)$

– Define perceptual weighting filter. This permits more noise at formant frequencies where it will be masked by the speech

– Synthesise speech using each codebook entry in turn as the input to $V(z)$

- Calculate optimum gain to minimise perceptually weighted error energy in speech frame
- Select codebook entry that gives lowest error

– Transmit LPC parameters and codebook index

Decoding

– Receive LPC parameters and codebook index

– Resynthesise speech using $V(z)$ and codebook entry

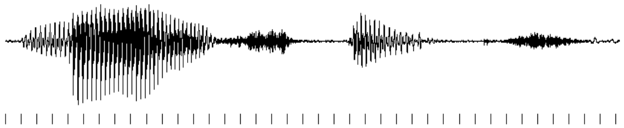
Performance:

– 16 kbit/s: MOS=4.2, Delay = 1.5 ms, 19 MIPS

– 8 kbit/s: MOS=4.1, Delay = 35 ms, 25 MIPS

– 2.4 kbit/s: MOS=3.3, Delay = 45 ms, 20 MIPS

LPC Analysis for CELP



– Divide speech up into 30 ms frames (240 samples)

– Multiply by Hamming window

– Perform 10th order autocorrelation LPC (no preemphasis)

– Bandwidth expansion: form $V(z/0.994)$ by multiplying a_i by 0.994^i

- Moves all poles in towards the origin of the z plane
- Pole on unit circle moves to a radius of 0.994 giving a 3dB bandwidth of $\approx -\ln(r)/\pi \approx (1-r)/\pi = 0.0019$; an unnormalised frequency of 15 Hz @ f_s = 8 kHz
- Improves LSP quantisation
- Reduces parameter sensitivity of spectrum
- Avoids chirps due to very sharp formant peaks

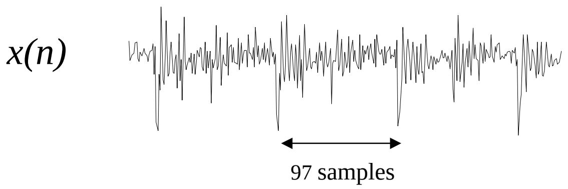
– Convert 10 LPC coefficients to 10 LSF frequencies

– Quantise using 4 bits for each of $f_2 - f_5$ and 3 bits for each of the others (34 bits in total) from empirically determined probability density functions.

– Smooth filter transitions by linearly interpolating a new set of LSP frequencies every ¼ frame.

LPC Residual

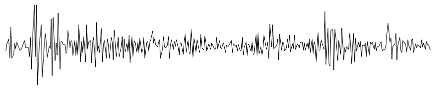
If we were to filter the speech signal by A(z), the inverse vocal-tract filter, we would obtain the LPC residual:



We can represent this signal as the sum of a periodic component and a noise component.

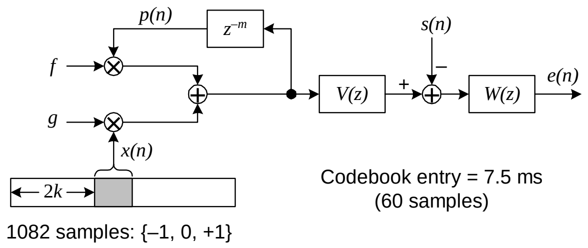
We can represent the periodic component by using a delayed version of *x(n)* to predict itself: this is called the *long-term predictor* or *LTP*. Subtracting this delayed version gives us just the noise component:

$$x(n) - 0.76x(n-97)$$



Having removed th periodic component, we search a codebook of noise signals (a *stochastic codebook*) for the best match to the residual excitation signal.

Excitation Codebooks


$p(n)$
z^{-m}
$s(n)$
f
g
$V(z)$
+
–
$W(z)$
$e(n)$
$x(n)$
$2k$
Codebook entry = 7.5 ms (60 samples)
1082 samples: {–1, 0, +1}

Excitation signal is the sum of 60-sample-long segments (60 samples = 7.5 ms = ¼ frame) from two sources :

- *Long-term predictor* (also called *Adaptive codebook*) is a delayed version of previous excitation samples multiplied by a gain, *f*. The value of *m* is in the range $20 \le m \le 147 \Rightarrow$ 7 bits ($400\text{Hz} > f_0 > 54\text{Hz}$). Some coders use fractional m for improved resolution at high frequencies: this requires interpolation.
- *Stochastic codebook* contains 1082 independent random values from the set $\{-1, 0, +1\}$ with probabilities $\{0.1, 0.8, 0.1\}$. The values of k is in the range $0 \le k \le 511 \Rightarrow$ 9 bits are needed.

The values of m & f are determined first. Then each possible value of k is tried to see which gives the lowest weighted error.

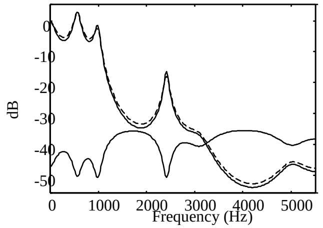
Perceptual Error Weighting

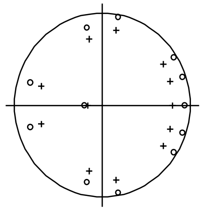
We want to choose the LTP delay and codebook entry that gives the best sounding resynthesised speech.

We exploit the phenomenon of *masking*: a listener will not notice noise at the formant frequencies because it will be overwhelmed by the speech energy. We therefore filter the error signal by:

$$W(z) = \frac{V(z/0.8)}{V(z)} = \frac{A(z)}{A(z/0.8)}$$

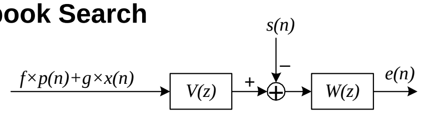
This emphasizes the error at noticeable frequencies



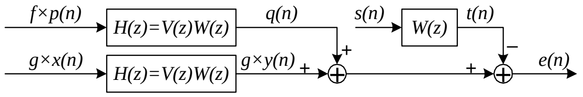


– Solid line is original $V(z)$
– Dashed line is bandwidth expanded $V(z/0.994)$ with slightly lower, broader peaks: more robust.
– Lower solid line is $W(z)$ shifted down a bit to make it more visible. Errors emphasized between formants.
– Pole-zero plot is $W(z)$. Poles from $1/A(z/0.8)$; Zeros from $A(z)$.

Codebook Search



We can interchange the subtraction and $W(z)$ and also separate the excitation signals:



Here $q(n)$ is the output from the filter $H(z)$ due to both the LTP signal *p(n)* <u>and</u> the excitation in previous sub-frames. $y(n)$ is the output due to the selected portion of the codebook. If $h(n)$ is the impulse response of $H(z)$ then:

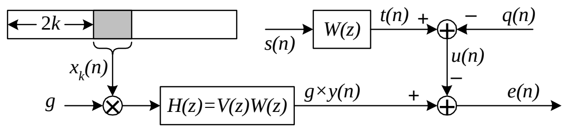
$$y(n) = \sum_{r=0}^{n} h(r)x(n-r) \quad \text{for } n = 0,1,\ldots,N-1$$

where the sub-frame length, N, is 60 samples (7.5 ms). Note that even though $h(r)$ is an infinite impulse response, the summation only goes to n since $x(n\text{–}r)$ is zero for $r>n$.

We want to choose the codebook entry that minimizes

$$E = \sum_{n=0}^{N-1} e^2(n) = \sum_{n=0}^{N-1}\left(gy(n)+q(n)-t(n)\right)^2$$
$$= \sum_{n=0}^{N-1}\left(gy(n)-u(n)\right)^2 \quad \text{where } u(n) = t(n)-q(n)$$

Gain Optimization



We can find the optimum g by setting $\frac{\partial E}{\partial g}$ to zero:

$$E = \sum_{n=0}^{N} e^2(n) = \sum_{n=0}^{N} \left(gy(n) - u(n)\right)^2$$

$$\Rightarrow\ 0 = ½\frac{\partial E}{\partial g} = \sum_n y(n)e(n) = \sum_n gy^2(n) - \sum_n y(n)u(n)$$

$$\Rightarrow\ g_{opt} = \frac{\sum_n y(n)u(n)}{\sum_n y^2(n)}$$

Substituting this in the expression for E gives:

$$E_{opt} = \sum_n \left(g_{opt}y(n) - u(n)\right)^2 = \sum_n \left(g_{opt}^2 y^2(n) - 2g_{opt}u(n)y(n) + u^2(n)\right)$$

$$= g_{opt}^2 \sum_n y^2(n) - 2g_{opt}\sum_n u(n)y(n) + \sum_n u^2(n)$$

$$= \sum_n u^2(n) - \frac{\left(\sum_n u(n)y(n)\right)^2}{\sum_n y^2(n)}$$

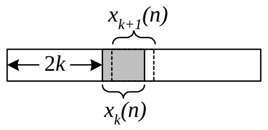
Energy Calculation

We need to find the k that minimizes:

$$E_{opt} = \sum_n u^2(n) - \frac{\left(\sum_n u(n)y_k(n)\right)^2}{\sum_n y_k^2(n)} \quad \text{where } y_k(n) = \sum_{j=0}^{n} h(j)x_k(n-j)$$

Note that for $n\geq 2$

$$x_k(n) = x_{k+1}(n-2)$$



Hence $y_k(n) = \sum_{r=0}^{n} h(r)x_k(n-r)$

$$= h(n)x_k(0) + h(n-1)x_k(1) + \sum_{r=0}^{n-2} h(r)x_{k+1}(n-2-r)$$

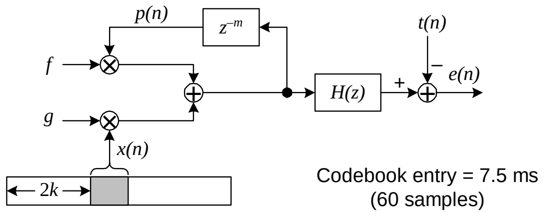
$$= h(n)x_k(0) + h(n-1)x_k(1) + y_{k+1}(n-2)$$

Thus we can derive $y_k()$ from $y_{k+1}()$ with very little extra computation. We only need to perform the full calculation for the last codebook entry. Note too that the multiplication $h(n)x_k(0)$ is particularly simple since $x_k(0) \in \{-1,0,+1\}$.

No divisions are needed for comparing energies since:

$$\left(C - \frac{A_2^2}{B_2^2}\right) < \left(C - \frac{A_1^2}{B_1^2}\right) \Leftrightarrow A_1^2B_2^2 < A_2^2B_1^2$$

4.8 kbit/s CELP Encoding



– Do LPC analysis for 30 ms frame (34 bits transmitted)
– For each ¼-frame segment (7.5 ms) do the following:
  - Calculate $q(n)$ as the output of $H(z)$ due to the excitation from previous sub-frames.
  - Search for the optimal LTP delay, m (4 × 7 bits transmitted per frame) and determine the optimal LTP gain, f (4 × 5 bits transmitted)
  - Recalculate $q(n)$ to equal the output of $H(z)$ when driven by the sum of the selected LTP signal as well as the inputs from previous frames
  - Search for the optimal stochastic codebook entry, k (4 × 9 bits transmitted) and gain, g (4 × 5 bits transmitted)
– Calculate a Hamming error correction code to protect high-order bits of m (5 bits transmitted)
– Allow one bit for future expansion of the standard

Total = 144 bits per 30 ms frame

Decoding is the same as encoding but without $W(z)$