ECE160 / CMPS182

Multimedia

Lecture 6: Spring 2008

Basics of Digital Audio

Digitization of Sound

What is Sound?

- Sound is a wave phenomenon like light, but is macroscopic and involves molecules of air being compressed and expanded under the action of some physical device.

  (a) A speaker in an audio system moves back and forth and produces a *longitudinal* pressure wave that we perceive as sound.

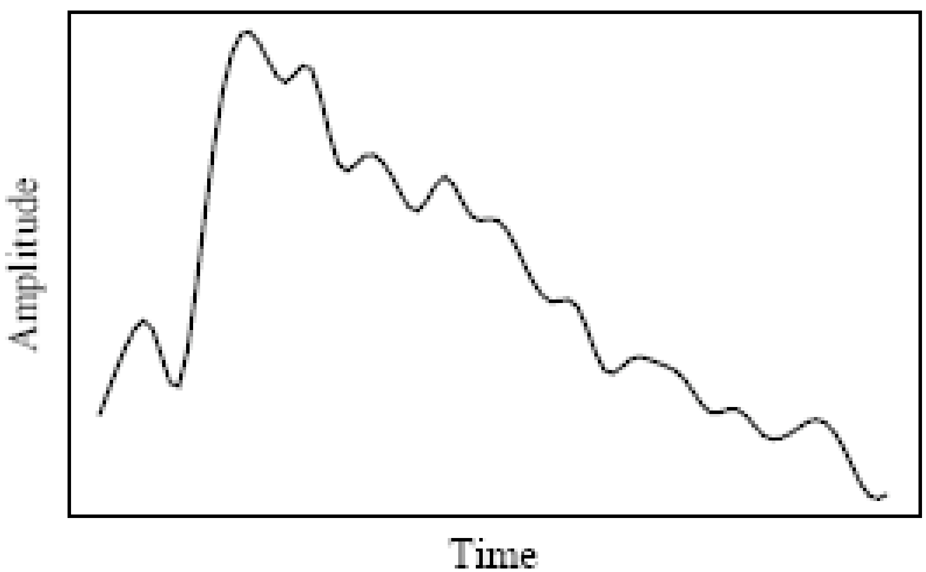
  (b) Since sound is a pressure wave, it takes on continuous analog values, as opposed to digitized ones.

- (c) Even though pressure waves are longitudinal, they still have ordinary wave properties and behaviors, i.e. reflection (bouncing), refraction (change of direction on entering a medium with a different density) and diffraction (bending around an obstacle).

  (d) If we wish to use a digital version of sound waves we must form digitized representations of analog audio information.

Digitization of Sound

- **Digitization** means conversion to a stream of numbers, and preferably these numbers should be integers for efficiency.
- The figure shows the 1-dimensional nature of sound: **amplitude** values depend on a 1D variable, time. (Images depend instead on a 2D set of variables, *x* and *y*).



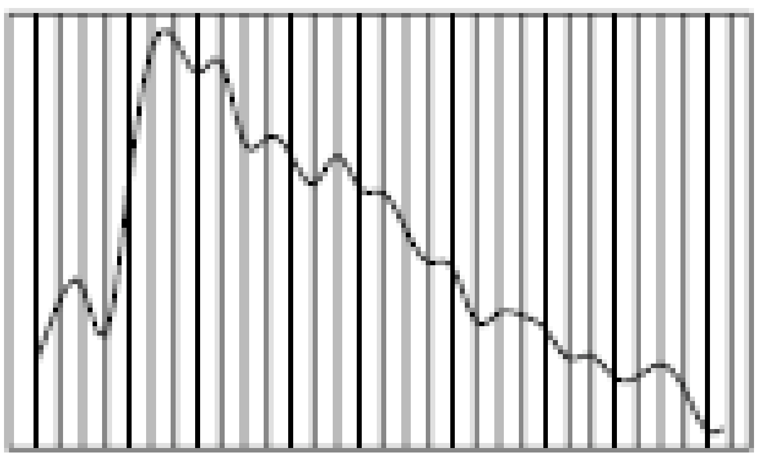
Digitization of Sound

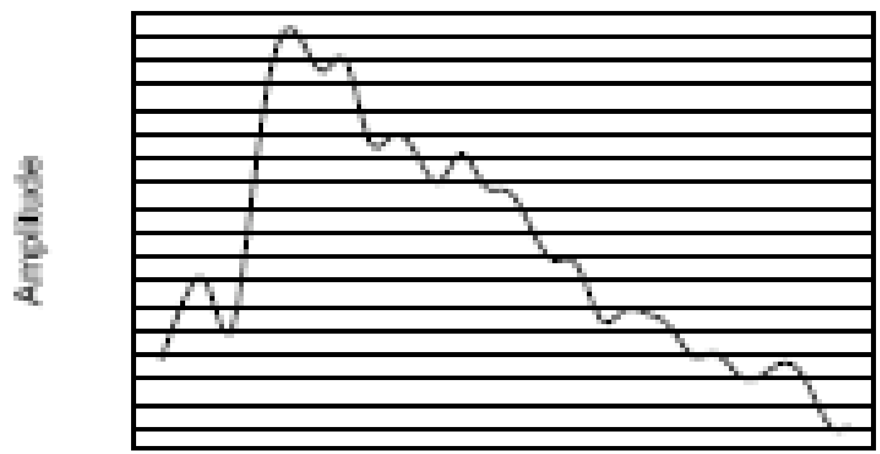
- The sound be made digital in both time and amplitude. To digitize, the signal is **sampled** in each dimension: in time, and in amplitude.

  (a) Sampling means measuring the quantity in one dimension, usually at evenly-spaced intervals in the other dimension.

  (b) The first kind of sampling, using measurements only at evenly spaced time intervals, is simply called, *sampling*. The rate at which it is performed is called the *sampling rate or frequency.*

  (c) For audio, typical sampling rates are from 8 kHz (8,000 samples per second) to 48 kHz. This rate is determined by Nyquist theorem.
- (d) Sampling in the amplitude dimension is called **quantization**.





Digitization of Sound

- Thus to decide how to digitize audio data we need to answer the following questions:

  1. What is the sampling rate?

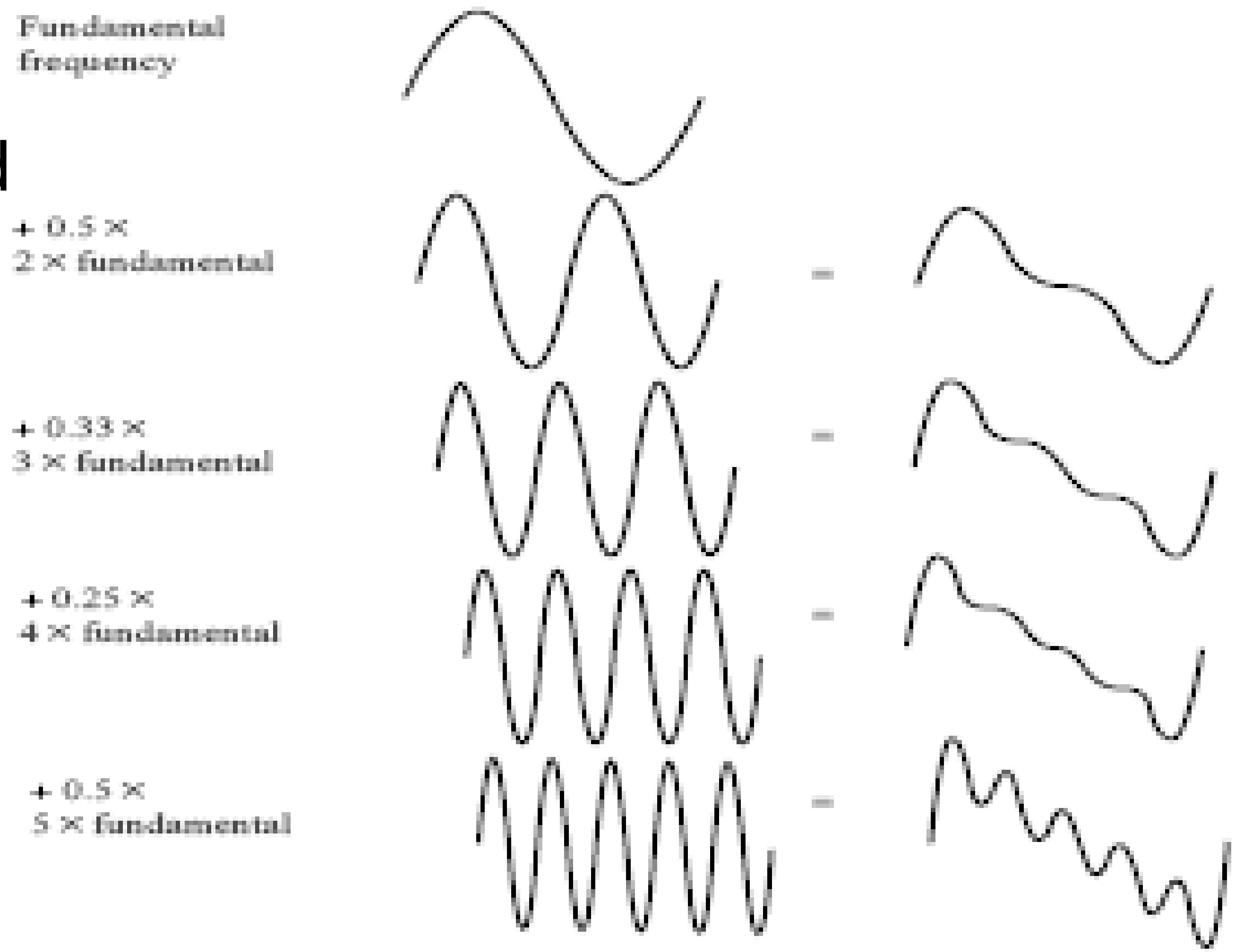
  2. How finely is the data to be quantized, and is quantization uniform?

  3. How is audio data formatted?

  (file format)

Nyquist Theorem

- Signals can be decomposed into a sum of sinusoids. The figure shows how weighted sinusoids can build up quite a complex signal.

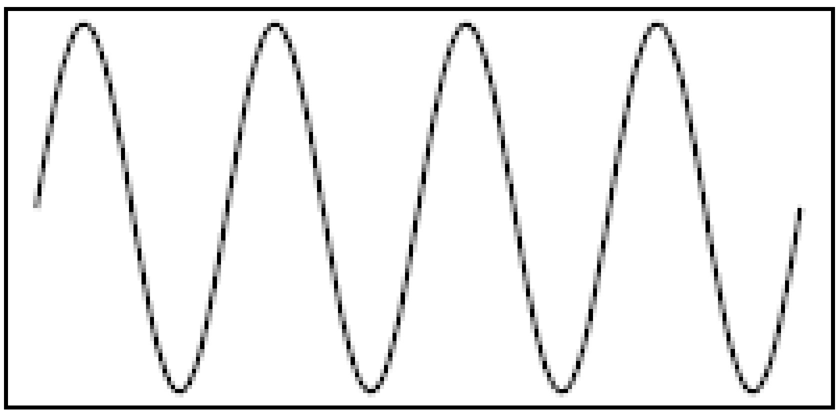


Nyquist Theorem

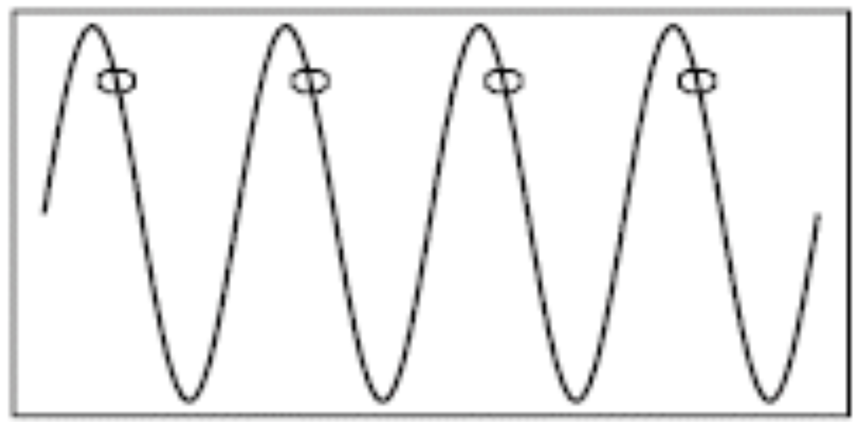
The Nyquist theorem states how frequently we must sample in time to be able to recover the original sound.

(a) The figure shows a single sinusoid: it is a single, pure, frequency (only electronic instruments can create such sounds).

(b) If sampling rate just equals the actual frequency, the figure shows that a false signal is detected: a constant, with zero frequency.



(a)

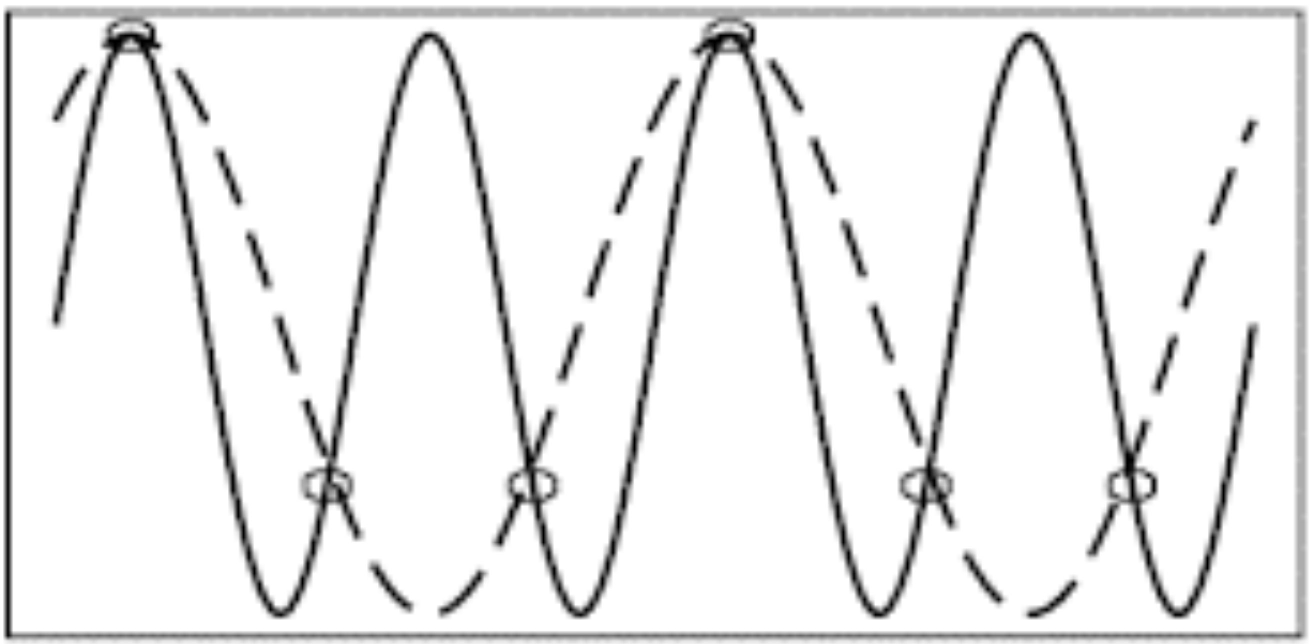


(b)

Nyquist Theorem

(c) If we sample at 1.5 times the actual frequency, the figure shows that we obtain an incorrect (**alias**) frequency that is lower than the correct frequency - it is half the correct frequency.

(d) For correct sampling we must use a sampling rate equal to at least *twice the maximum frequency* content in the signal. This rate is called the **Nyquist rate**.



Nyquist Theorem

- **Nyquist Theorem**: If a signal is **band-limited**, i.e., there is a lower limit $f1$ and an upper limit $f2$ of frequency components in the signal, then the sampling rate should be at least $2(f2 - f1)$.
- **Nyquist frequency**: half of the Nyquist rate.

  - Since it would be impossible to recover frequencies higher than Nyquist frequency in any event, most systems have an **antialiasing filter** that restricts the frequency content in the input to the sampler to a range at or below Nyquist frequency.
- The relationship among the Sampling Frequency, True Frequency, and the Alias Frequency is

$$f_{alias} = f_{sampling} - f_{true}, \quad \text{for} \quad f_{true} < f_{sampling} < 2 \times f_{true}$$

Pitch

- Whereas **frequency** is an absolute measure, **pitch** is generally relative - a perceptual subjective quality of sound.

  (a) Pitch and frequency are linked by setting the note A above middle C to exactly 440 Hz.

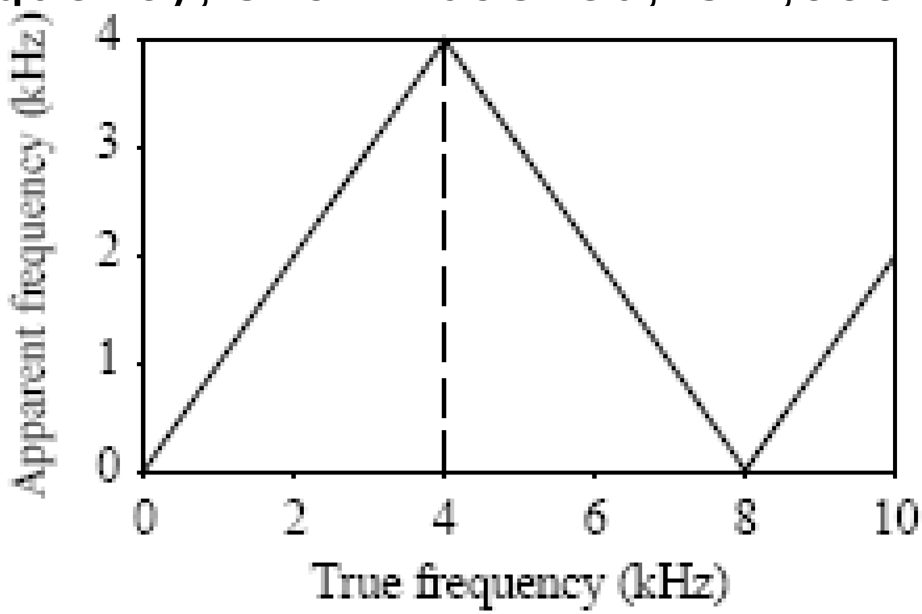
  (b) An **octave** above that note takes us to another A note. An octave corresponds to *doubling the frequency*. Thus with the middle “A" on a piano (“A4" or “A440") set to 440 Hz, the next “A" up is at 880 Hz, or one octave above.

  (c) **Harmonics**: any series of musical tones whose frequencies are integral multiples of the frequency of a fundamental tone.

  (d) If we allow non-integer multiples of the base frequency, we allow non-”A" notes and have a more complex resulting sound.

Pitch

- In general, the apparent pitch of a sinusoid is the lowest frequency of a sinusoid that has exactly the same samples as the input sinusoid. The figure shows the relationship of the apparent pitch (frequency) to the input frequency, which is sampled at 8,000 Hz. The folding frequency, shown dashed, is 4,000 Hz.



Signal to Noise Ratio (SNR)

- The ratio of the power of the correct signal and the noise is called the *signal to noise ratio* (**SNR**) - a measure of the quality of the signal.
- The SNR is usually measured in decibels (**dB**), where 1 dB is a tenth of a **bel**. The SNR value, in units of dB, is defined in terms of base-10 logarithms of squared voltages:

$$SNR = 10 \log_{10} \frac{V_{signal}^2}{V_{noise}^2} = 20 \log_{10} \frac{V_{signal}}{V_{noise}}$$

Signal to Noise Ratio (SNR)

a) The power in a signal is proportional to the square of the voltage. For example, if the signal voltage *Vsignal* is 10 times the noise, then the SNR is 20 log10(10)=20dB.

b) In terms of power, if the power from ten violins is ten times that from one violin playing, then the ratio of power is 10dB, or 1B.

Sound Levels

The usual levels of sound we hear around us are described in terms of decibels, as a ratio to the quietest sound we are capable of hearing. The table shows approximate levels for these sounds.

Threshold of hearing	0
Rustle of leaves	10
Very quiet room	20
Average room	40
Conversation	60
Busy street	70
Loud radio	80
Train through station	90
Riveter	100
Threshold of discomfort	120
Threshold of pain	140
Damage to ear drum	160

Signal to Quantization Noise Ratio (SQNR)

- Aside from any noise present in the original analog signal, there is also an additional error that results from quantization.

  (a) If voltages are 0 to 1 but we have only 8 bits to store values, then we force all continuous values into only 256 different values.

  (b) This introduces a roundoff error. It is not really “noise". Nevertheless it is called **quantization noise** (or quantization error).
- The quality of the quantization is characterized by the Signal to Quantization Noise Ratio (**SQNR**).

  (a) **Quantization noise**: the difference between the actual value of the analog signal, for the particular sampling time, and the nearest quantization interval value.

  (b) At most, this error can be as much as half of the interval.

Signal to Quantization Noise Ratio (SQNR)

(c) For a quantization accuracy of N bits per sample, the SQNR can be simply expressed:

$$SQNR = 20\log_{10}\frac{V_{signal}}{V_{quan\_noise}} = 20\log_{10}\frac{2^{N-1}}{\frac{1}{2}}$$

$$= 20 \times N \times \log 2 = 6.02N\,(\mathrm{dB})$$

Notes:

(a) We map the maximum signal to $2^{N-1} - 1$ ($\sim 2^{N-1}$) and the most negative signal to -2^{N-1} .

(b) The equation is the *Peak* signal-to-noise ratio, PSQNR: peak signal and peak noise.

Signal to Quantization Noise Ratio (SQNR)

(c) The *dynamic range* is the ratio of maximum to minimum absolute values of the signal: $V_{max/}V_{min}$. The max abs. value V_{max} gets mapped to $2^{N-1}-1$; the min abs. value V_{min} gets mapped to 1. V_{min} is the smallest positive voltage that is not masked by noise. The most negative signal, $-V_{max}$, is mapped to -2^{N-1} .

(d) The quantization interval is $V=(2V_{max})=2N$, since there are $2N$ intervals.
The whole range V_{max} down to $(V_{max} - V=2)$ is mapped to $2N-1 - 1$.

(e) The maximum noise, in terms of actual voltages, is half the quantization interval: $V=2=V_{max}=2N$.

(f) 6:02N **is the worst case**. If the input signal is sinusoidal, the quantization error is statistically independent, and its magnitude is uniformly distributed between 0 and half of the interval, then it can be shown that the expression for the SQNR becomes:

$$SQNR = 6.02N+1.76(dB)$$

Linear and Non-linear Quantization

- **Linear format**: samples are typically stored as uniformly quantized values.
- **Non-uniform quantization**: set up more finely-spaced levels where humans hear with the most acuity.

  - Weber's Law stated formally says that equally perceived differences have values proportional to absolute levels:

  $$\Delta\text{Response} \propto \Delta\text{stimulus}/\text{Stimulus}$$

  - Inserting a constant of proportionality *k*, we have a differential equation that states:

  $$dr = k(1/s)\, ds$$

  with response *r* and stimulus *s*.

Non-linear Quantization

- Integrating, we arrive at a solution

  $$r = k \ln s + C$$

  with constant of integration C.
  Stated differently, the solution is

  $$r = k \ln(s/s_0)$$

  s_0 = the lowest level of stimulus that causes a response (r = 0 when $s=s_0$).
- Nonlinear quantization works by first transforming an analog signal from the raw s space into the theoretical r space, and then uniformly quantizing the resulting values.
- Such a law for audio is called **μ-law** encoding, (or **u-law**). A very similar rule, called *A*-**law**, is used in telephony in Europe.

Non-linear Quantization

- The equations for these very similar encodings are as follows:

μ-law:

$$r = \frac{\text{sgn}(s)}{\ln(1+\mu)} \ln\left\{1 + \mu\left|\frac{s}{s_p}\right|\right\}, \qquad \left|\frac{s}{s_p}\right| \le 1$$

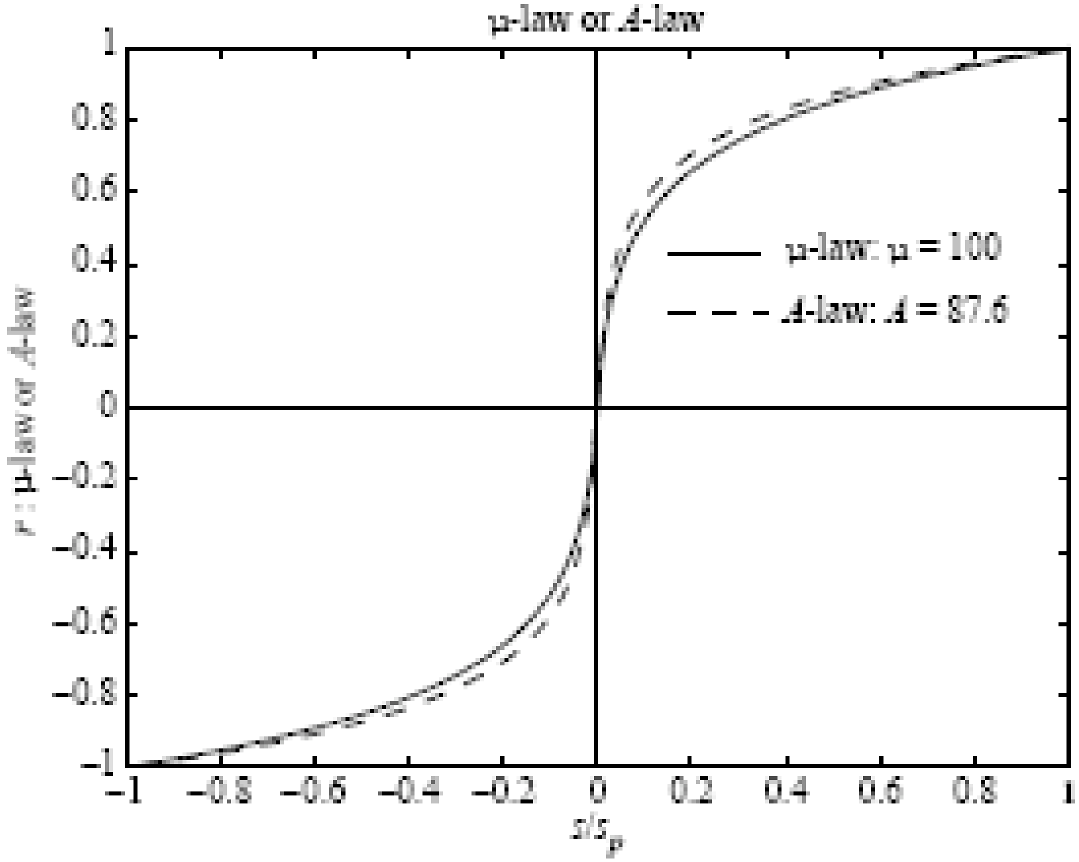
A-law:

$$r = \begin{cases} \frac{A}{1+\ln A}\left(\frac{s}{s_p}\right), & \left|\frac{s}{s_p}\right| \le \frac{1}{A} \\ \frac{\text{sgn}(s)}{1+\ln A}\left[1 + \ln A\left|\frac{s}{s_p}\right|\right], & \frac{1}{A} \le \left|\frac{s}{s_p}\right| \le 1 \end{cases}$$

$$\text{where } \text{sgn}(s) = \begin{cases} 1 & \text{if } s > 0, \\ -1 & \text{otherwise} \end{cases}$$

Non-linear Quantization

- The figure shows these curves. The parameter μ is set to μ = 100 or μ = 255; the parameter *A* for the *A*-law encoder is set to *A* = 87*:*6.



Audio Filtering

- Prior to sampling and AD conversion, the audio signal is also usually *filtered* to remove unwanted frequencies. The frequencies kept depend on the application:

  (a) For speech, typically from 50Hz to 10kHz is retained, and other frequencies are blocked by a **band-pass filter** that screens out lower and higher frequencies.

  (b) An audio music signal will typically contain from about 20Hz up to 20kHz.

  (c) At the DA converter end, high frequencies may reappear in the output - because of sampling and then quantization - smooth input signal is replaced by a series of step functions containing all possible frequencies.

  (d) At the decoder side, after the DA converter, a **lowpass filter** is used to remove those steps.

Audio Quality vs. Data Rate

- The uncompressed data rate increases as more bits are used for quantization. Stereo: double the bandwidth. to transmit a digital audio signal.

Quality	Sample Rate (KHz)	Bits per Sample	Mono/ Stereo	Data Rate (uncompressed) (kB/sec)	Frequency Band (KHz)
Telephone	8	8	Mono	8	0.200-3.4
AM Radio	11.025	8	Mono	11.0	0.1-5.5
FM Radio	22.05	16	Stereo	88.2	0.02-11
CD	44.1	16	Stereo	176.4	0.005-20
DAT	48	16	Stereo	192.0	0.005-20
DVD Audio	192 (max)	24 (max)	6 channels	1,200.0 (max)	0-96 (max)

MIDI: Musical Instrument Digital Interface

MIDI Overview

(a) MIDI is a scripting language with “events" for the production of sounds, including values for the pitch of a note, its duration, and its volume.

(b) MIDI is a standard adopted by electronic music for controlling devices, such as synthesizers and sound cards, that produce music.

(c) MIDI standard is supported by most synthesizers, so sounds created on one synthesizer can be played and manipulated on another synthesizer and sound reasonably close.

(d) Computers have a MIDI interface incorporated into most sound cards, with both D/A and A/D converters.

MIDI Concepts

- MIDI **channels** are used to separate messages.

  (a) There are 16 channels numbered from 0 to 15. The channel forms the last 4 bits (the least significant bits) of the message.

  (b) Usually a channel is associated with a particular instrument: e.g., channel 1 is the piano, channel 10 is the drums, etc.

  (c) Nevertheless, one can switch instruments midstream, if desired, and associate another instrument with any channel.

MIDI Concepts

- **System messages**

  (a) Several other types of messages, e.g. a general message for all instruments indicating a change in tuning or timing.

  (b) If the first 4 bits are all 1s, then the message is interpreted as a **system common** message.

- The way a synthetic musical instrument responds to a MIDI message is usually by simply ignoring any **play sound** message that is not for its channel.
  - If several messages are for its channel, then the instrument responds, provided it is **multi-voice**, i.e., can play more than a single note at once.

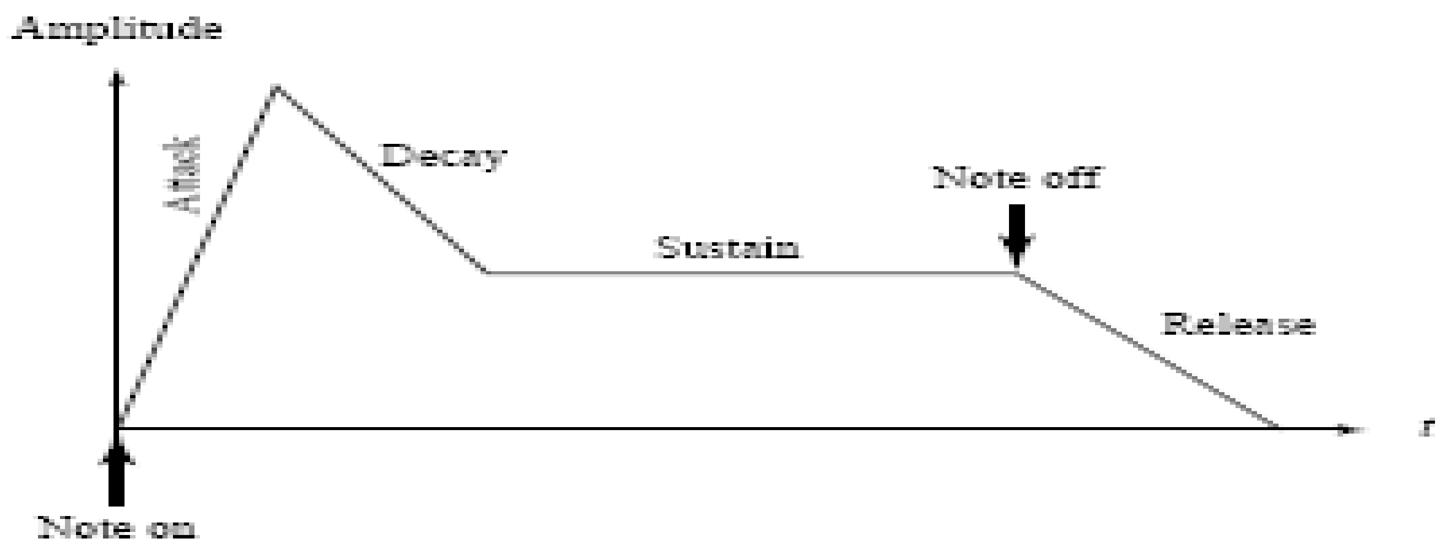
MIDI Concepts

- It is easy to confuse the term **voice** with the term **timbre -** the latter is MIDI terminology for just what instrument that is trying to be emulated, e.g. a piano as opposed to a violin: it is the quality of the sound.

  (a) An instrument (or sound card) that is **multi-timbral** is one that is capable of playing many different sounds at the same time, e.g., piano, brass, drums, etc.
- (b) On the other hand, the term **voice**, while sometimes used by musicians to mean the same thing as timbre, is used in MIDI to mean every different timbre and pitch that the tone module can produce at the same time.
- Different timbres are produced digitally by using a **patch -** the set of control settings that define a particular timbre.
- Patches are often organized into databases, called **banks**.

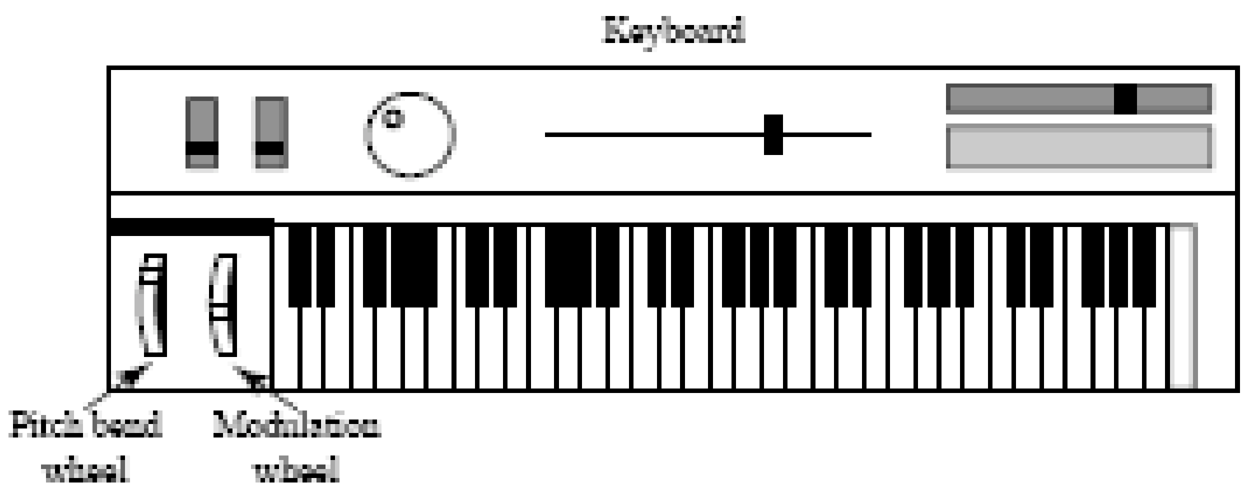
MIDI Concepts

- A MIDI device is capable of **programmability**, and can change the **envelope** describing how the amplitude of a sound changes over time.
- The figure shows a model of the response of a digital instrument to a Note On message:



Hardware Aspects of MIDI

- The MIDI hardware setup consists of a 31.25 kbps serial connection. Usually, MIDI-capable units are either Input devices or Output devices, not both.
- A traditional synthesizer is shown



Hardware Aspects of MIDI

- The physical MIDI ports consist of 5-pin connectors for IN and OUT, as well as a third connector called THRU.

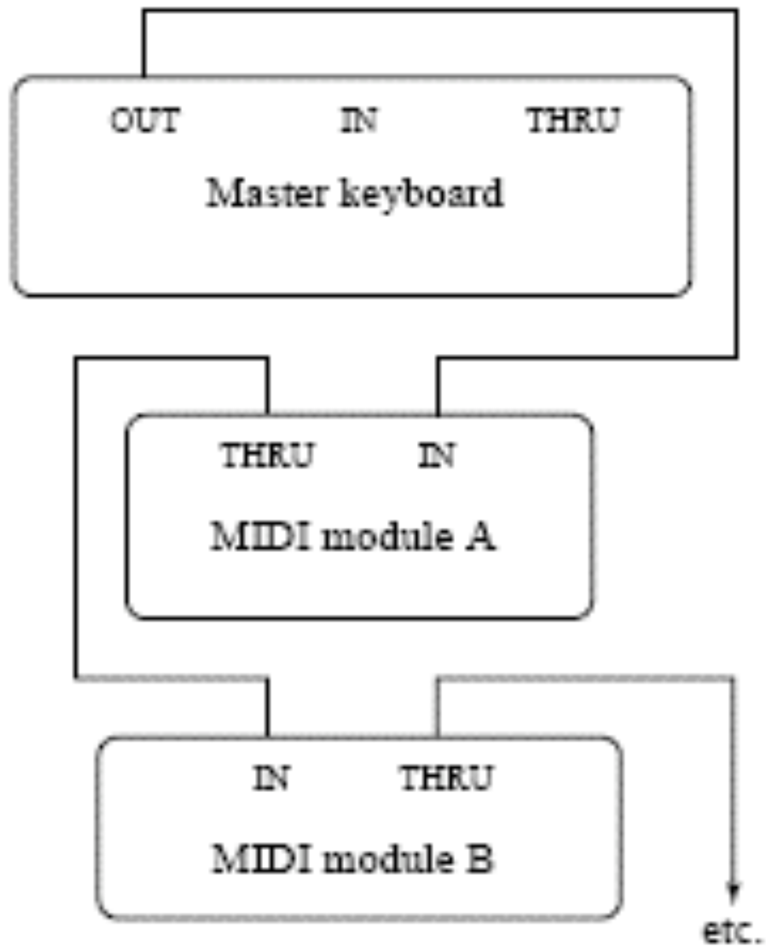
  (a) MIDI communication is half-duplex.

  (b) MIDI IN is the connector via which the device receives all MIDI data.

  (c) MIDI OUT is the connector through which the device transmits all the MIDI data it generates itself.

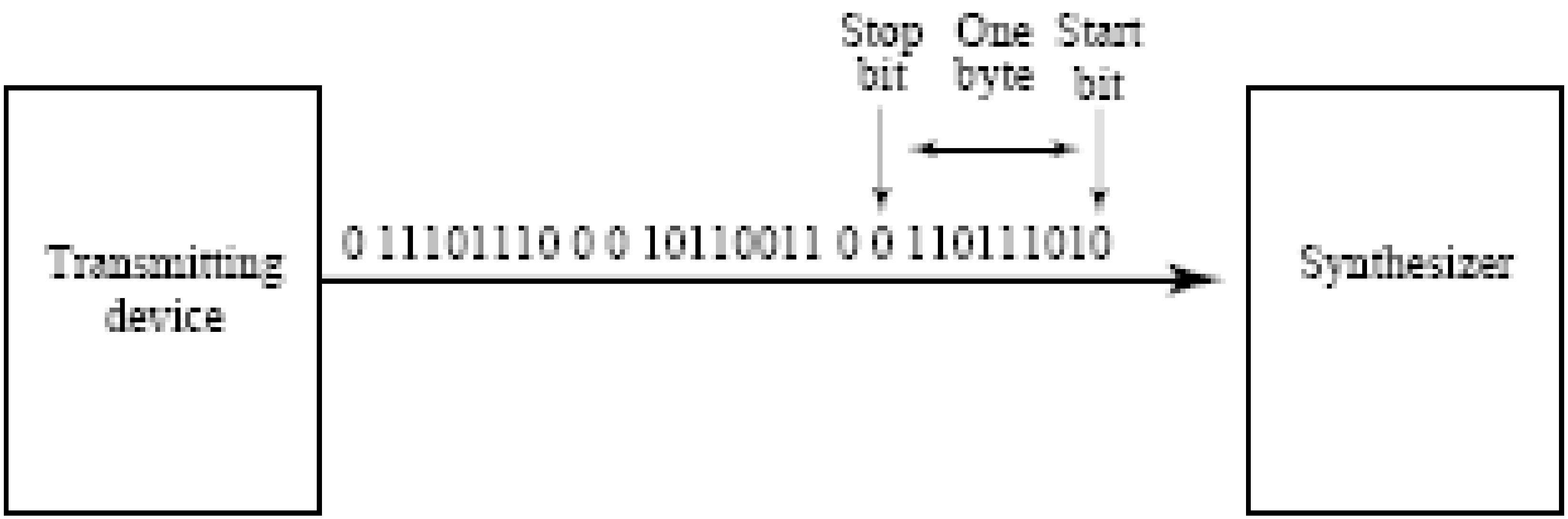
  (d) MIDI THRU is the connector by which the device echoes the data it receives from MIDI IN. Note that it is only the MIDI IN data that is echoed by MIDI THRU - all the data generated by the device itself is sent via MIDI OUT.

A typical MIDI sequencer setup



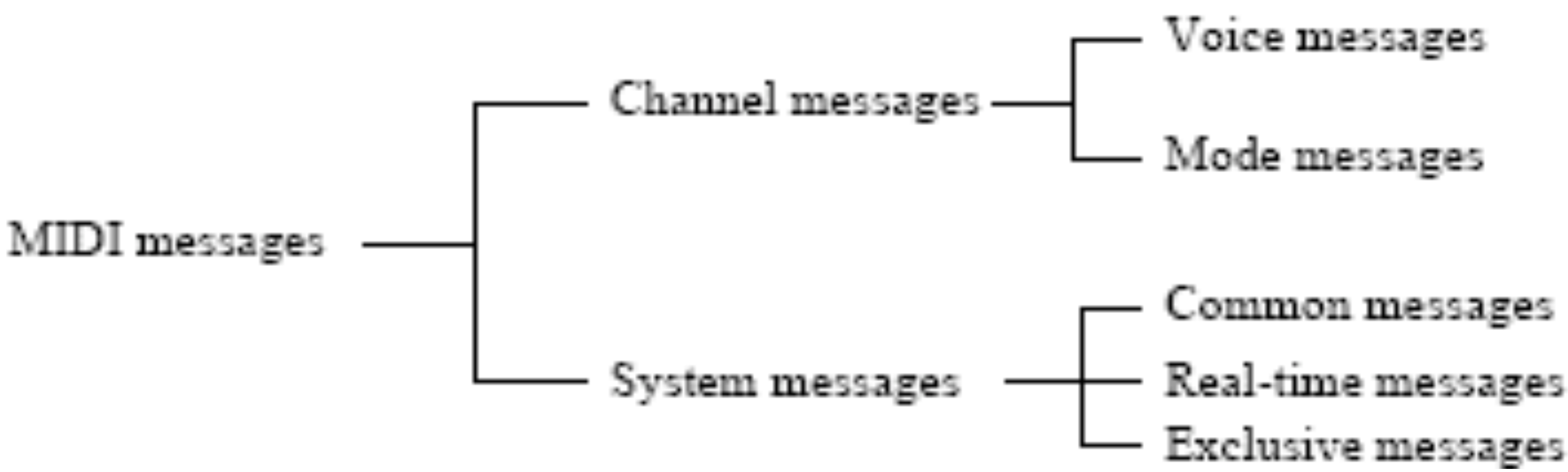
MIDI data stream

- A stream of 10-bit bytes for MIDI messages consist of {Status byte, Data Byte, Data Byte} = {Note On, Note Number, Note Velocity}.
  MIDI bytes are 10-bit, with a 0 start and 0 stop bit.



Structure of MIDI Messages

- MIDI messages can be classified into two types: channel messages and system messages



MIDI Messages

A. Channel messages: can have up to 3 bytes:

a) The first byte is the status byte (the opcode, as it were); has its most significant bit set to **1**.

b) The 4 low-order bits identify which channel this message belongs to (for 16 possible channels).

c) The 3 remaining bits hold the message. For a data byte, the most signicant bit is set to **0**.

MIDI Voice Messages

Voice messages:

a) This type of channel message controls a voice, i.e., sends information specifying which note to play or to turn off, and encodes key pressure.

b) Voice messages are also used to specify controller effects such as sustain, vibrato, tremolo, and the pitch wheel.

Voice Message	Status Byte	Data Byte1	Data Byte2
Note Off	&H8n	Key number	Note Off velocity
Note On	&H9n	Key number	Note On velocity
Poly. Key Pressure	&HAn	Key number	Amount
Control Change	&HBn	Controller num.	Controller value
Program Change	&HCn	Program number	None
Channel Pressure	&HDn	Pressure value	None
Pitch Bend	&HEn	MSB	LSB

MIDI Channel Mode Messages

Channel mode messages

a) Special case of the Control Change message −> opcode B (the message is &HBn, or 1011nnnn), with first data byte in 121 through 127 (&H79{7F).

b) Channel mode messages determine how an instrument processes MIDI voice messages: respond to all messages, respond just to the correct channel, don't respond at all, or go over to local control of the instrument.

1st Data Byte	Description	Meaning of 2nd Data Byte
&H79	Reset all controllers	None; set to 0
&H7A	Local control	0 = off; 127 = on
&H7B	All notes off	None; set to 0
&H7C	Omni mode off	None; set to 0
&H7D	Omni mode on	None; set to 0
&H7E	Mono mode on (Poly mode off)	Controller number
&H7F	Poly mode on (Mono mode off)	None; set to 0

MIDI System Messages

System Messages:

a) System messages have no channel number - commands that are not channel specific, such as timing signals for synchronization, positioning information in pre-recorded MIDI sequences, and detailed setup information for the destination device.

b) Opcodes for all system messages start with &HF.

c) System messages are divided into three classifications, according to their use:

MIDI System Messages

- **System common messages:** relate to timing or positioning.

System Common Message	Status Byte	Number of Data Bytes
MIDI Timing Code	&HF1	1
Song Position Pointer	&HF2	2
Song Select	&HF3	1
Tune Request	&HF6	None
EOX (terminator)	&HF7	None

MIDI System Messages

- **System real-time messages:** related to synchronization.

System Real-Time Message	Status Byte
Timing Clock	&HF8
Start Sequence	&HFA
Continue Sequence	&HFB
Stop Sequence	&HFC
Active Sensing	&HFE
System Reset	&HFF

MIDI System Messages

System exclusive message: allows the MIDI standard to be extended by manufacturers.

a) After the initial code, a stream of any specific messages can be inserted that apply to their own product.

b) A System Exclusive message is supposed to be terminated by a terminator byte &HF7.

c) The terminator is optional and the data stream may simply be ended by sending the status byte of the next message.

General MIDI

- General MIDI is a scheme for standardizing the assignment of instruments to patch numbers.

  a) A standard percussion map specifies 47 percussion sounds.

  b) Where a “note" appears on a musical score determines what percussion instrument is being struck: a bongo drum, a cymbal.

  c) Other requirements for General MIDI compatibility: MIDI device must support all 16 channels; a device must be multitimbral (i.e., each channel can play a different instrument/program); a device must be polyphonic (i.e., each channel is able to play many voices); and there must be a minimum of 24 dynamically allocated voices.
- **General MIDI Level2**: An extended general MIDI has recently been defined, with a standard .smf “Standard MIDI File" format defined - inclusion of extra character information, such as karaoke lyrics.

MIDI to WAV Conversion

- Some programs, such as early versions of Premiere, cannot include .mid files - instead, they insist on .wav format files.

  a) Various shareware programs exist for approximating a reasonable conversion between MIDI and WAV formats.

Csound

- Csound is a unit generator-based, user-programmable computer music system, written by Barry Vercoe at MIT in 1984. Since then Csound has received numerous contributions from researchers and musicians from around the world.
- Csound runs on many varieties of UNIX and Linux, Microsoft DOS and Windows, all versions of the Macintosh operating system including Mac OS X, and others.
- Csound can be considered one of the most powerful musical instruments ever created.
- To make music with Csound:

  1. Write an **orchestra** (.orc file) that creates instruments and signal processors by connecting unit generators (also called opcodes, in Csound-speak) using Csound's simple programming language.

  2. Write a **score** (.sco file) that specifies a list of notes and other events to be rendered by the orchestra.

  3. Run Csound to compile the orchestra and score, run the sorted and preprocessed score through the orchestra.

pluck -- Produces a naturally decaying plucked string or drum sound.

pluck kamp, kcps, icps, ifn, imeth [, iparm1] [, iparm2]

- *kamp* -- the output amplitude.
- *kcps* -- the resampling frequency in cycles-per-second. An audio buffer, filled at i-time according to *ifn*, is sampled with periodicity *kcps* and multiplied by *kamp*. The buffer is smoothed to simulate the effect of natural decay.
- *icps* -- intended pitch value in Hz, sets up a buffer of audio samples smoothed by a chosen decay.
- *ifn* -- number of a function used to initialize the cyclic decay buffer. If *ifn* = 0, a random sequence will be used.
- *imeth* -- method of natural decay. There are six, some of which use parameters values that follow.

  1. Simple averaging. A simple smoothing process, uninfluenced by parameter values.
  2. Stretched averaging. As above, with smoothing time stretched by a factor of iparm1 (=1).
  3. Simple drum. The range from pitch to noise is controlled by a 'roughness factor' in iparm1 (0 to 1). Zero gives the plucked string effect, while 1 reverses the polarity of every sample (octave down, odd harmonics). The setting .5 gives an optimum snare drum.
  4. Stretched drum. Combines both roughness and stretch factors. iparm1 is roughness (0 to 1), and iparm2 the stretch factor (=1).
  5. Weighted averaging. As method 1, with iparm1 weighting the current sample (the status quo) and iparm2 weighting the previous adjacent one. iparm1 + iparm2 must be <= 1.
  6. 1st order recursive filter, with coefs .5. Unaffected by parameter values.
- *iparm1, iparm2* (optional) -- parameter values for use by the smoothing algorithms (above).
- Plucked strings (1,2,5,6) are best realized by starting with a random noise source, which is rich in initial harmonics. Drum sounds (methods 3,4) work best with a flat source (wide pulse), which produces a deep noise attack and sharp decay.

streson -- A string resonator with variable fundamental frequency.

- **streson** asig, kfr, ifdbgain
- *asig* -- the input audio signal.
- *kfr* -- the fundamental frequency of the string.
- *streson* passes the input *asig* through a network composed of comb, low-pass and all-pass filters, similar to the one used in the Karplus-Strong algorithm, creating a string resonator effect. The fundamental frequency of the “string” is controlled by *kfr*.This opcode is used to simulate sympathetic resonances to an input signal.
- *ifdbgain* -- feedback gain, between 0 and 1, of the internal delay line. A value close to 1 creates a slower decay and a more pronounced resonance. Small values may leave the input signal unaffected. Depending on the filter frequency, typical values are > .9

Csound Score

Cscore is a program for generating and manipulating numeric score files. It comprises function subprograms, called by a user-written control program, invoked either as a stand alone score preprocessor, or as part of the Csound run-time system:

A **Score** (a collection of score statements) is divided into time-ordered sections by the *s statement*. Before being read by the orchestra, a score is preprocessed one section at a time. Each section is processed by 3 routines: *Carry*, *Tempo*, and *Sort*.

Carry

- Determines how long a note or sequence of notes should last.

Tempo

- Time warps a score section according to the information in a *t statement*. The tempo operation converts p2 (and, for *i statements*, p3) from original beats into real seconds, since those are the units required by the orchestra.

Sort

- This routine sorts all action-time statements into chronological order by p2 value.

Quantization and Transmission of Audio

- **Coding of Audio**: Quantization and transformation of data are collectively known as **coding** of the data.

  a) For audio, the μ-law technique for companding audio signals is usually combined with an algorithm that exploits the temporal redundancy present in audio signals.

  b) Differences in signals between the present and a past time can reduce the size of signal values and also concentrate the histogram of pixel values (differences, now) into a much smaller range.

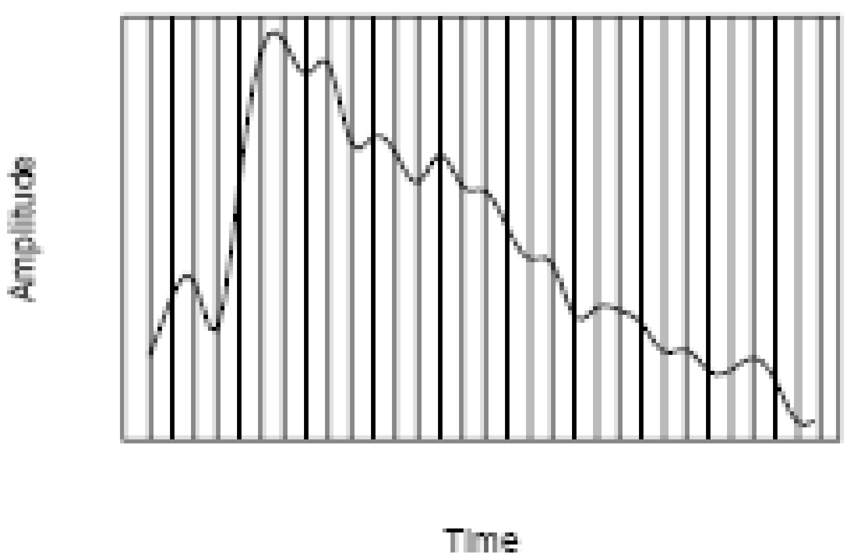
Quantization and Transmission of Audio

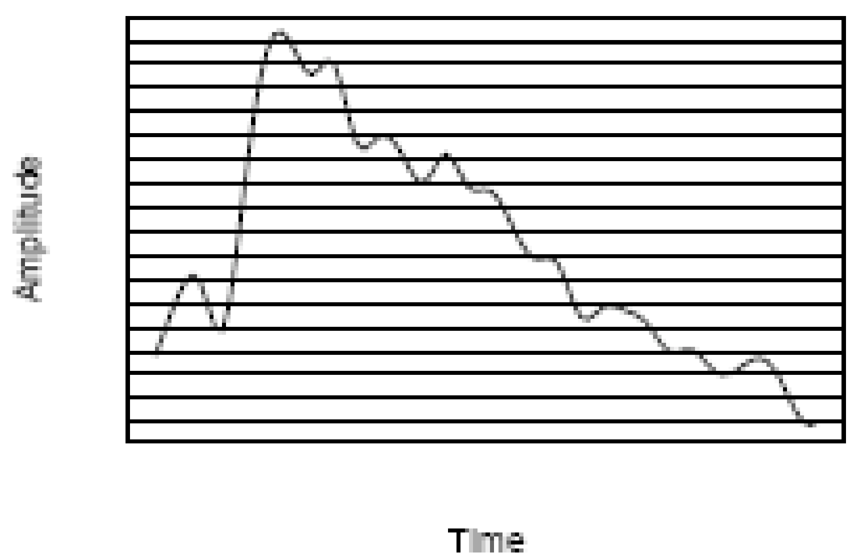
c) The result of reducing the variance of values is that lossless compression methods produce a bitstream with shorter bit lengths for more likely values

- In general, producing quantized sampled output for audio is called **PCM** (Pulse Code Modulation). The differences version is called **DPCM** (and a crude but efficient variant is called **DM**). The adaptive version is called **ADPCM**.

Pulse Code Modulation

- The basic techniques for creating digital signals from analog signals are **sampling** and **quantization**.
- Quantization consists of selecting breakpoints in magnitude, and then re-mapping any value within an interval to one of the representative output levels.





Pulse Code Modulation

a) The set of interval boundaries are called **decision boundaries**, and the representative values are called **reconstruction levels**.

b) The boundaries for quantizer input intervals that will all be mapped into the same output level form a **coder mapping**.

c) The representative values that are the output values from a quantizer are a **decoder mapping**.

- d) Finally, we may wish to **compress** the data, by assigning a bit stream that uses fewer bits for the most prevalent signal values

Pulse Code Modulation

Every compression scheme has three stages:

A. The input data is **transformed** to a new representation that is easier or more efficient to compress.

B. We may introduce **loss** of information. *Quantization* is the main lossy step => we use a limited number of reconstruction levels, fewer than in the original signal.

C. **Coding**. Assign a codeword (thus forming a binary bitstream) to each output level or symbol. This could be a fixed-length code, or a variable length code such as Huffman coding

PCM in Speech Compression

Assuming a bandwidth for speech from about 50 Hz to about 10 kHz, the Nyquist rate would dictate a sampling rate of 20 kHz.

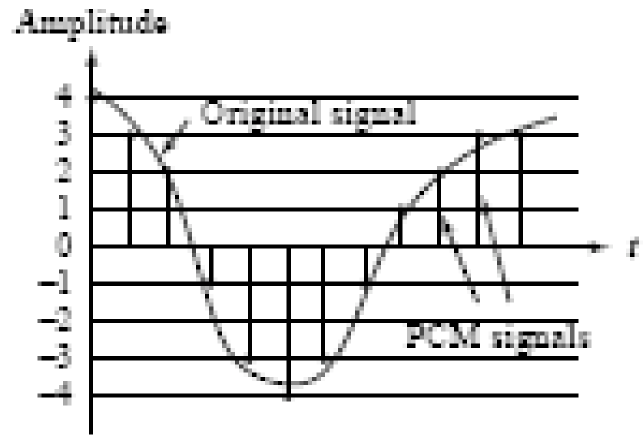
(a) Using uniform quantization without companding, the minimum sample size we could get away with would likely be about 12 bits. For mono speech transmission the bit-rate would be 240 kbps.

(b) With companding, we can reduce the sample size down to about 8 bits with the same perceived level of quality, and thus reduce the bit-rate to 160 kbps.

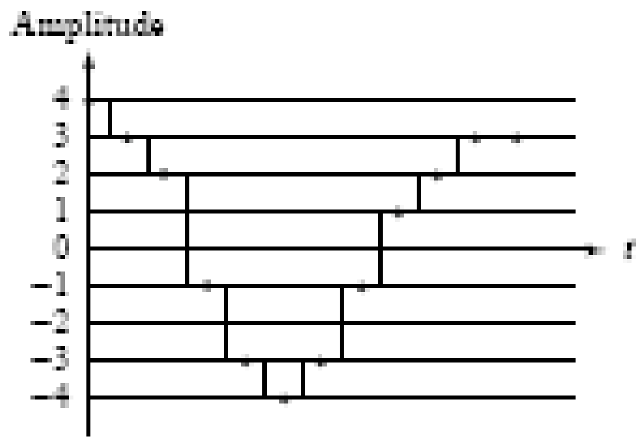
(c) However, the standard approach to telephony in fact assumes that the highest-frequency audio signal we want to reproduce is only about 4 kHz. Therefore the sampling rate is only 8 kHz, and the companded bit-rate thus reduces this to 64 kbps.

(d) Since only sounds up to 4 kHz are to be considered, all other frequency content must be noise. Therefore, we remove this high-frequency content from the analog input signal using a band-limiting filter that blocks out high, as well as very low, frequencies.

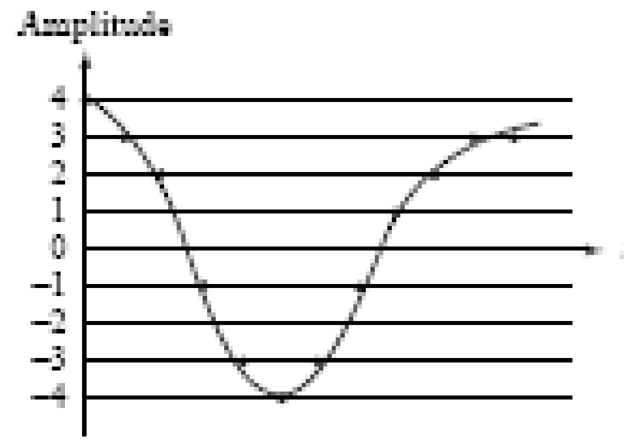
PCM in Speech Compression



(a)



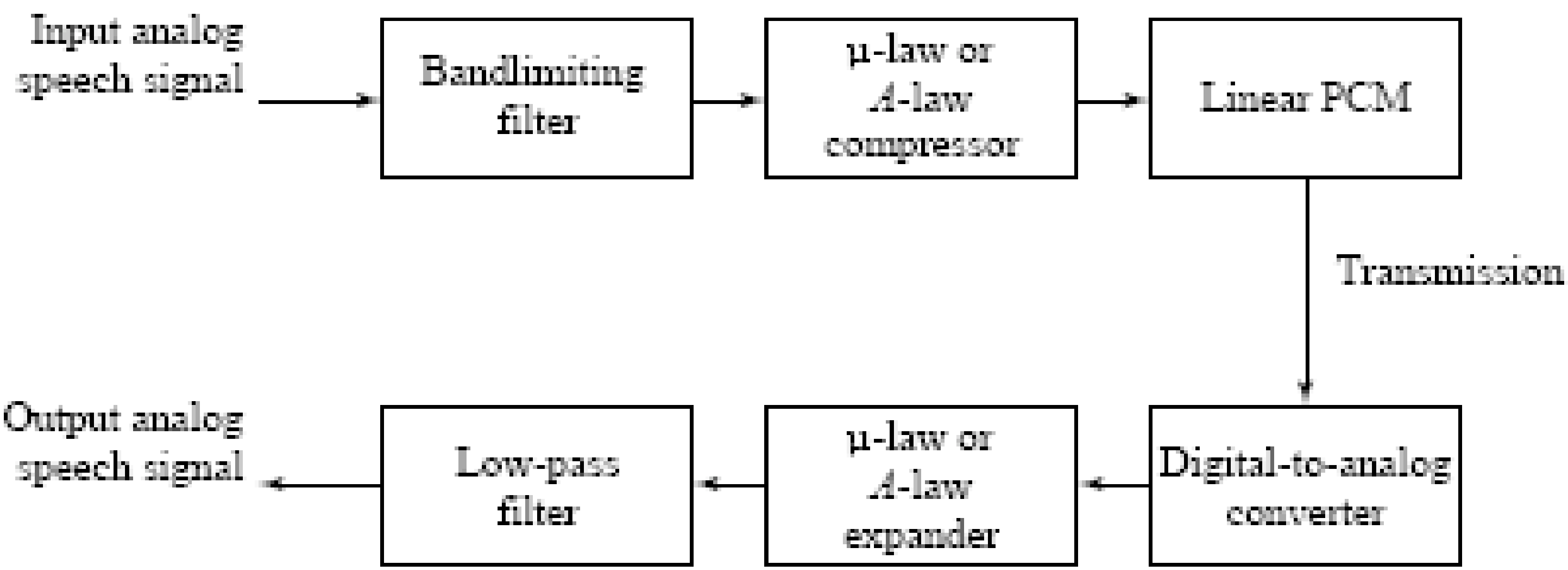
(b)



(c)

PCM in Speech Compression

- The complete scheme for encoding and decoding telephony signals is shown below. As a result of the low-pass filtering, the output becomes smoothed.



Differential Coding of Audio

- Audio is often stored in a form that exploits differences - which are generally smaller numbers, needing fewer bits to store them.

  (a) If a signal has some consistency over time (“temporal redundancy"), the difference signal, subtracting the current sample from the previous one, will have a more peaked histogram, with a maximum near zero.

  (b) For example, as an extreme case the histogram for a linear ramp signal that has constant slope is flat, whereas the histogram for the derivative of the signal (i.e., the differences from sampling point to sampling point) consists of a spike at the slope value.

  (c) If we assign codewords to differences, we can assign short codes to prevalent values and long codewords to rare ones.

Lossless Predictive Coding

- **Predictive coding**: simply means transmitting differences - predict the next sample as being equal to the current sample; send not the sample itself but the difference between previous and next.

  (a) Predictive coding consists of finding differences, and transmitting these using a PCM system.

  (b) Note that differences of integers will be integers. Denote the integer input signal as the set of values f_n. Then we **predict** values $\hat{f}_n$ as simply the previous value, and define the error e_n as the difference between the actual and the predicted signal:

$$\hat{f}_n = f_{n-1}$$

$$e_n = f_n - \hat{f}_n$$

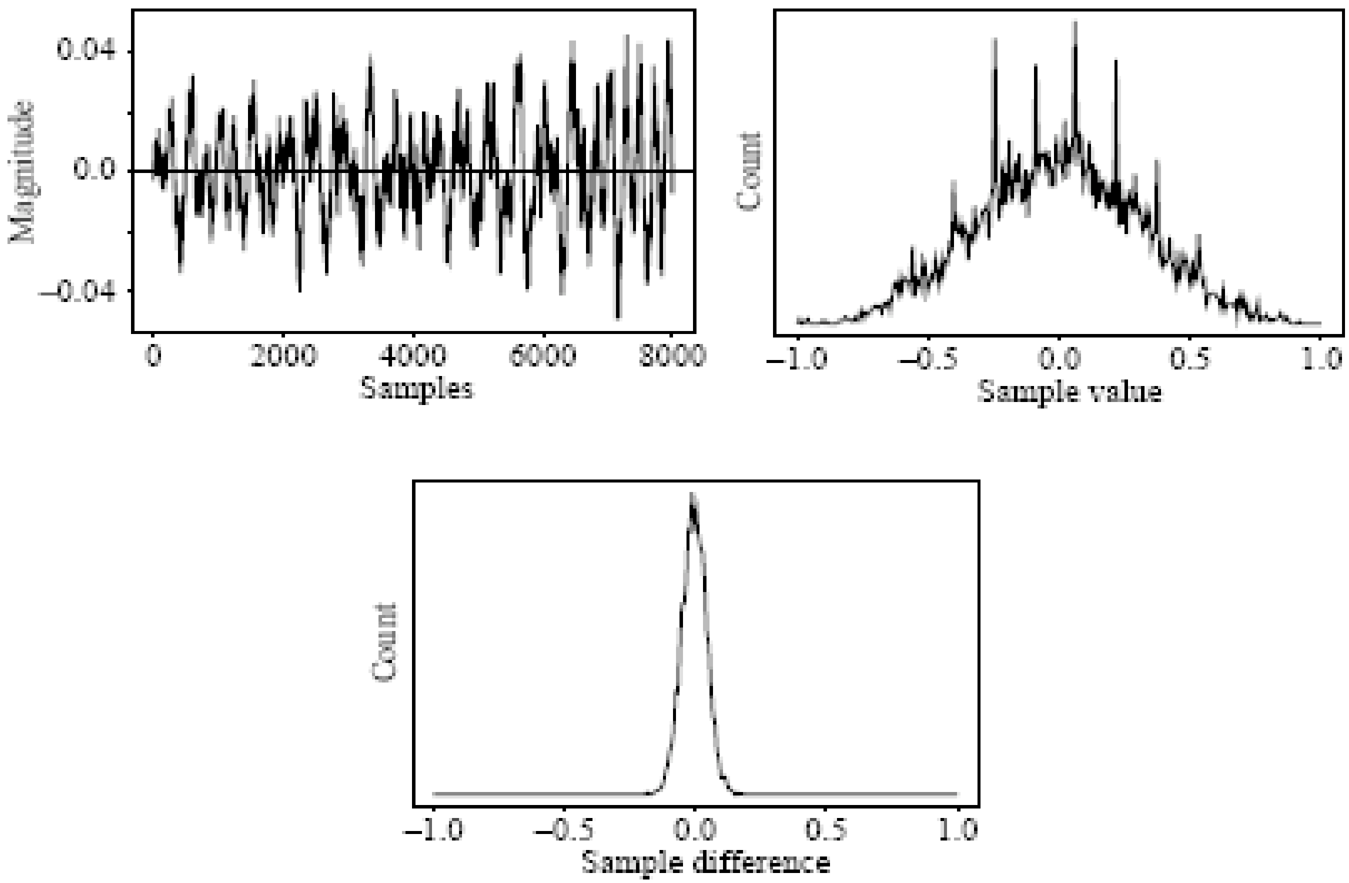
Lossless Predictive Coding

- (c) But it is often the case that some **function** of a few of the previous values, f_{n-1}, f_{n-2}, f_{n-3}, etc., provides a better prediction. Typically, a linear **predictor** function is used:

$$\widehat{f_n} = \sum_{k=1}^{2 \text{ to } 4} a_{n-k} f_{n-k}$$

Differential Coding of Audio

- Differencing concentrates the histogram.
  (a): Digital speech signal.
  (b): Histogram of digital speech signal values.
  (c): Histogram of digital speech signal differences.



Differential Coding of Audio

- One problem: suppose our integer sample values are in the range 0..255. Then differences could be as much as -255..255 - we've increased our **dynamic range** (ratio of maximum to minimum) by a factor of two and need more bits to transmit some differences.

  (a) A clever solution for this: define two new codes, denoted SU and SD, standing for Shift-Up and Shift-Down. Special code values are reserved for these.

  (b) We use codewords for only a limited set of signal differences, say only the range $-15::16$. Differences in the range are coded as is, a value outside the range $-15::16$ is transmitted as a series of shifts, followed by a value that is inside the range $-15::16$.

  (c) For example, 100 is transmitted as: SU, SU, SU, 4.

Lossless predictive coding

- *Lossless predictive coding* - the decoder produces the same signals as the original. As a simple example, suppose we devise a predictor for $\widehat{f_n}$ as follows:

$$\widehat{f_n} = \lfloor \frac{1}{2}(f_{n-1} + f_{n-2}) \rfloor$$

$$e_n = f_n - \widehat{f_n}$$

Lossless predictive coding

- Let's consider an explicit example. Suppose we wish to code the sequence $f_1; f_2; f_3; f_4; f_5$ = 21, 22, 27, 25, 22. For the purposes of the predictor, we'll invent an extra signal value f_0, equal to f_1 = 21, and first transmit this initial value. uncoded:

$$\widehat{f_2} = 21\ ,\ e_2 = 22 - 21 = 1;$$

$$\widehat{f_3} = \lfloor \tfrac{1}{2}(f_2 + f_1) \rfloor = \lfloor \tfrac{1}{2}(22 + 21) \rfloor = 21,$$
$$e_3 = 27 - 21 = 6;$$

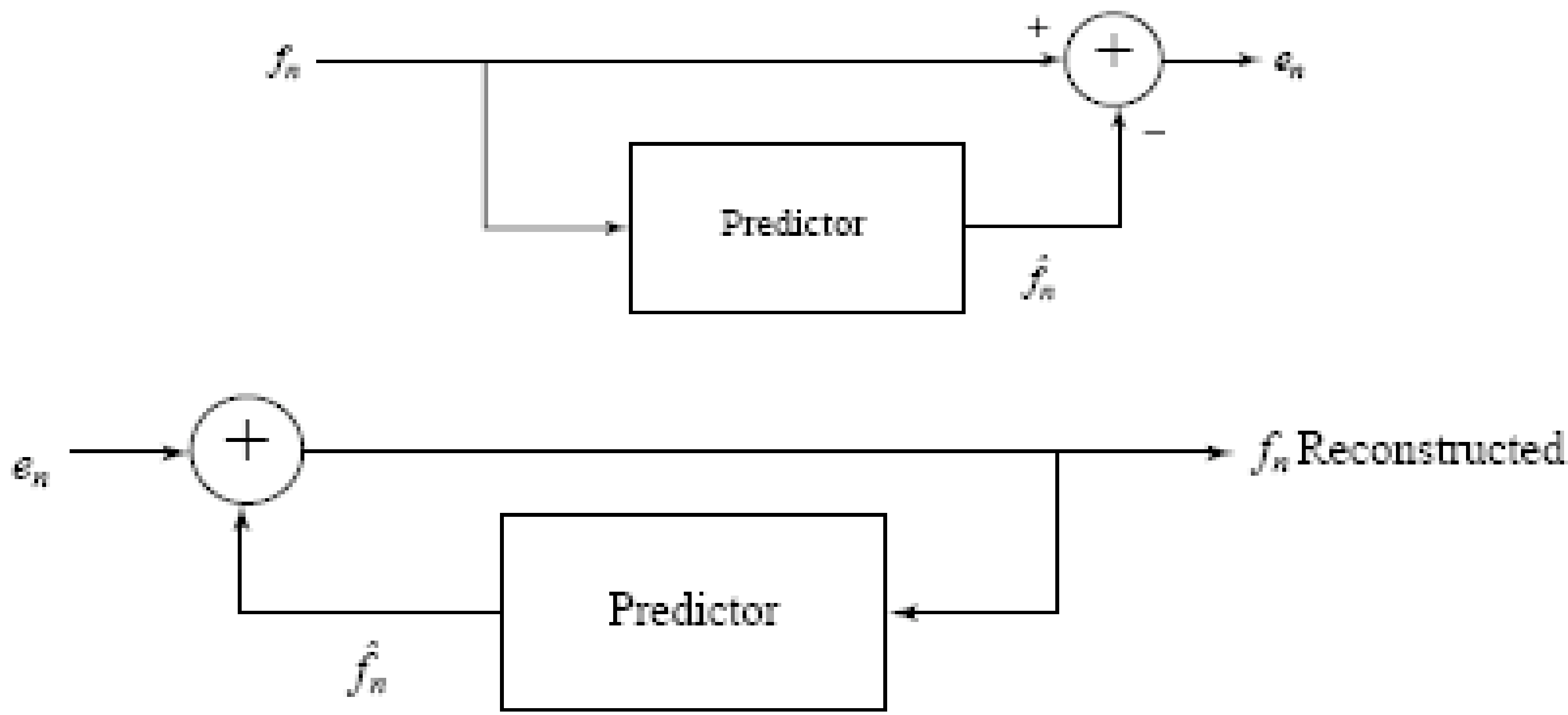
$$\widehat{f_4} = \lfloor \tfrac{1}{2}(f_3 + f_2) \rfloor = \lfloor \tfrac{1}{2}(27 + 22) \rfloor = 24,$$
$$e_4 = 25 - 24 = 1;$$

$$\widehat{f_5} = \lfloor \tfrac{1}{2}(f_4 + f_3) \rfloor = \lfloor \tfrac{1}{2}(25 + 27) \rfloor = 26,$$

$$e_5 = 22 - 26 = -4$$

Lossless predictive coding

- Schematic diagram for Predictive Coding encoder and decoder.



DPCM

- Differential PCM is exactly the same as Predictive Coding, except that it incorporates a *quantizer* step.

  (a) One scheme for analytically determining the best set of quantizer steps, for a non-uniform quantizer, is the **Lloyd-Max** quantizer, which is based on a least-squares minimization of the error term.

  (b) Our nomenclature: signal values:
  f_n - the original signal, $\hat{f}_n$ – the predicted signal, and
  $\tilde{f}_n$ the quantized, reconstructed signal.

  (c) **DPCM**: form the prediction; form an error e_n by subtracting the prediction from the actual signal; then quantize the error to a quantized version, $\tilde{e}_n$.

DPCM

- The set of equations that describe DPCM are

$$\hat{f}_n = function\_of(\tilde{f}_{n-1},, \tilde{f}_{n-2}, \tilde{f}_{n-3}, \ldots),$$

$$e_n = f_n - \hat{f}_n,$$

$$\tilde{e}_n = Q[e_n],$$

$$\text{transmit } codeword(\tilde{e}_n),$$

$$\text{reconstruct: } \tilde{f}_n = \hat{f}_n + \tilde{e}_n.$$

- Then *codewords* for quantized error values $\tilde{e}_n$ are produced using entropy coding,

DPCM

- The main effect of the coder-decoder process is to produce reconstructed, quantized signal values $\tilde{f}_n = \hat{f}_n + \tilde{e}_n$. The **distortion** is the average squared error $[\sum_{n=1}^{N} (\tilde{f}_n - f_n)^2]/N$; one often plots distortion versus the number of bit-levels used. A Lloyd-Max quantizer will do better (have less distortion) than a uniform quantizer.
- For speech, we modify quantization steps adaptively by estimating the mean and variance of a patch of signal values, and shifting quantization steps accordingly for every block of signal values. Starting at time i we take a block of N values f_n and minimize the quantization error:

$$min \sum_{n=i}^{i+N-1} (f_n - Q[f_n])^2$$

DPCM

- Since signal **differences** are very peaked, we can model them using a Laplacian probability distribution function, which is strongly peaked at zero: it looks like

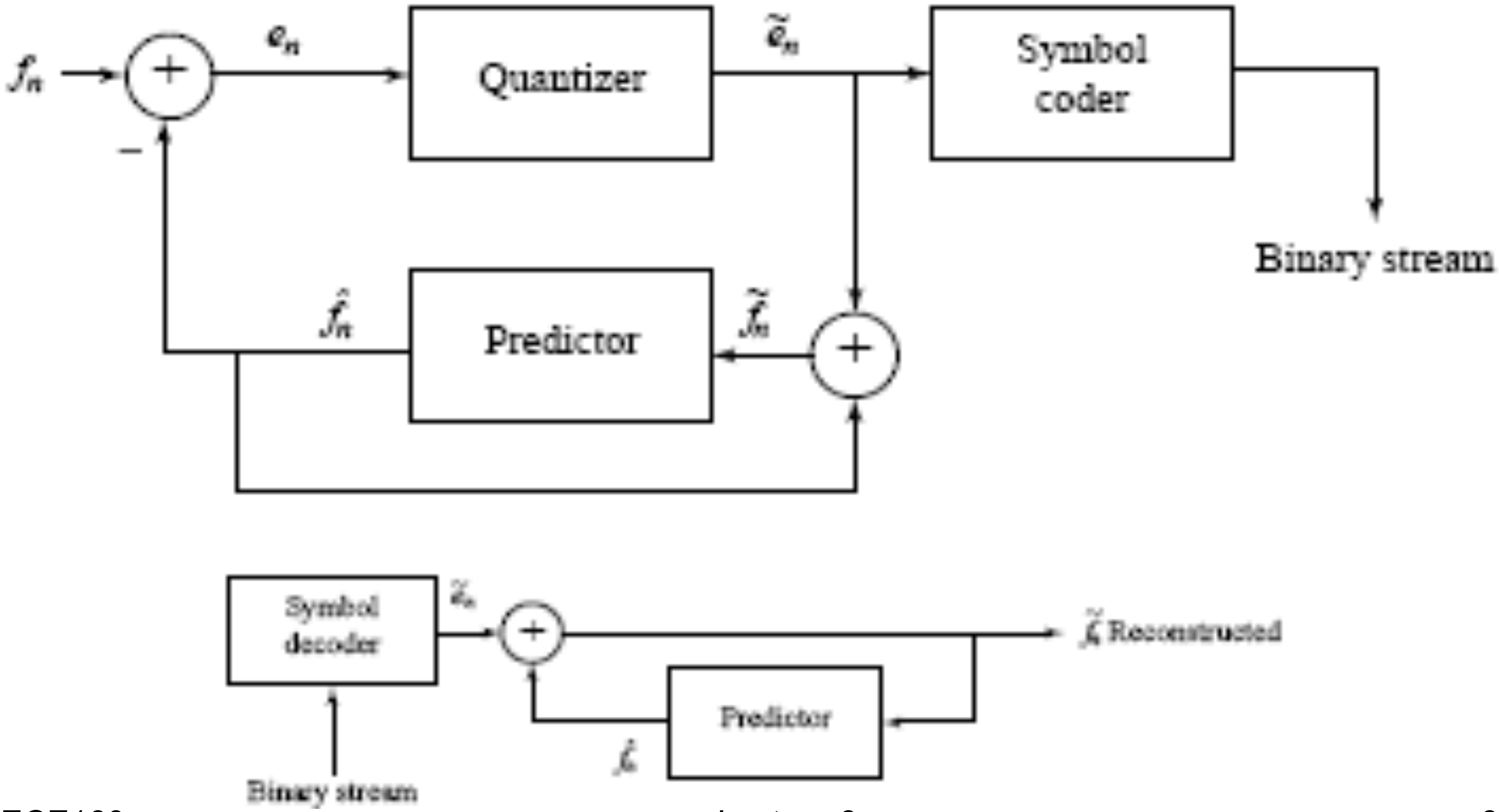
$$l(x) = (1/\sqrt{2\sigma^2})exp(-\sqrt{2}|x|/\sigma)$$

  for variance σ^2.
- So typically one assigns quantization steps for a quantizer with nonuniform steps by assuming signal differences, d_n are drawn from such a distribution and then choose steps to minimize

$$min \sum_{n=i}^{i+N-1} (d_n - Q[d_n])^2 \, l(d_n)$$

DPCM

- Schematic diagram for DPCM encoder and decoder



DM

- **DM** (Delta Modulation): simplied version of DPCM. Often used as a quick AD converter.

  1. **Uniform-Delta DM**: use only a single quantized error value, either positive or negative.

  (a) => a 1-bit coder. Produces coded output that follows the original signal in a staircase fashion. The set of equations is:

$$\hat{f}_n = \tilde{f}_{n-1} ,$$
$$e_n = f_n - \hat{f}_n = f_n - \tilde{f}_{n-1} ,$$

$$\tilde{e}_n = \begin{cases} +k & \text{if } e_n > 0, \text{ where } k \text{ is a constant} \\ -k & otherwise \end{cases}$$

$$\tilde{f}_n = \hat{f}_n + \tilde{e}_n .$$

DM

- DM does not cope well with rapidly changing signals. One approach to mitigating this problem is to simply increase the sampling, perhaps to many times the Nyquist rate.
- **Adaptive DM**: If the slope of the actual signal curve is high, the staircase approximation cannot keep up. For a steep curve, can change the step size *k* adaptively.

  - One scheme for analytically determining the best set of quantizer steps, for a non-uniform quantizer, is *Lloyd-Max*.

ADPCM

- **ADPCM** (Adaptive DPCM) takes the idea of adapting the coder to suit the input much farther. The two pieces that make up a DPCM coder: the quantizer and the predictor.

  1. In Adaptive DM, adapt the quantizer step size to suit the input. In DPCM, we can change the step size as well as decision boundaries, using a non-uniform quantizer.

  We can carry this out in two ways:

  (a) **Forward adaptive quantization**: use the properties of the input signal.

  (b) **Backward adaptive quantizationor**: use the properties of the quantized output. If quantized errors become too large, we should change the non-uniform quantizer.

ADPCM

- We can also **adapt the predictor**, again using forward or backward adaptation. Making the predictor coefficients adaptive is called *Adaptive Predictive Coding* (APC):

  (a) Recall that the predictor is usually taken to be a linear function of previous reconstructed quantized values, $\tilde{f}_n$.

  (b) The number of previous values used is called the “order" of the predictor. For example, if we use M previous values, we need M coefficients a_i , i = 1..M in a predictor

$$\hat{f}_n = \sum_{i=1}^{M} a_i \tilde{f}_{n-i}$$

ADPCM

- However we can get into a difficult situation if we try to change the prediction coefficients, that multiply previous quantized values, because that makes a complicated set of equations to solve for these coefficients:

  (a) Suppose we decide to use a least-squares approach to solving a minimization trying to find the best values of the a_i:

$$min \sum_{n=1}^{N} (f_n - \hat{f}_n)^2$$

- Here we would sum over a large number of samples *fn*, for the current patch of speech, say. But because $\tilde{f}_n$ depends on the quantization we have a difficult problem to solve. As well, we should really be changing the fineness of the quantization at the same time, to suit the signal's changing nature; this makes things problematical.

ADPCM

- Instead, one usually resorts to solving the simpler problem that results from using not $\tilde{f}_n$ in the prediction, but instead simply the signal f_n itself. Explicitly writing in terms of the coefficients *ai*, we wish to solve:

$$min \sum_{n=1}^{N} (f_n - \sum_{i=1}^{M} a_i f_{n-i})^2$$

- Differentiation with respect to each of the a_i, and setting to zero, produces a linear system of M equations that is easy to solve. (The set of equations is called the Wiener-Hopf equations.)

Dolby

- Dolby started by analog coding to hide hiss on cassette tapes. He amplified small sounds before recording and then reduced them (and the hiss) on playback.
- Dolby moved on to multichannel sound for movie widescreens (but no multichannels on the movie film).
- You will encounter Dolby AC-3, (5.1 channels), right front, center, left front, right rear, left rear and subwoofer. Dolby EX (6.1) has a center rear Dolby TrueHD has 8 channels and high quality