Chapter 6
Basics of Digital Audio

6.1 Digitization of Sound
6.2 MIDI: Musical Instrument Digital Interface
6.3 Quantization and Transmission of Audio
6.4 Further Exploration

6.1 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) For example, a speaker in an audio system vibrates 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 values, as opposed to digitized ones.

(c) Even though such pressure waves are longitudinal, they still have ordinary wave properties and behaviors, such as reflection (bouncing), refraction (change of angle when 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 audio information.
⟶ Link to physical description of sound waves.

Digitization

- **Digitization** means conversion to a stream of numbers, and preferably these numbers should be integers for efficiency.
- Fig. 6.1 shows the 1-dimensional nature of sound: **amplitude** values depend on a 1D variable, time. (And note that images depend instead on a 2D set of variables, x and y).

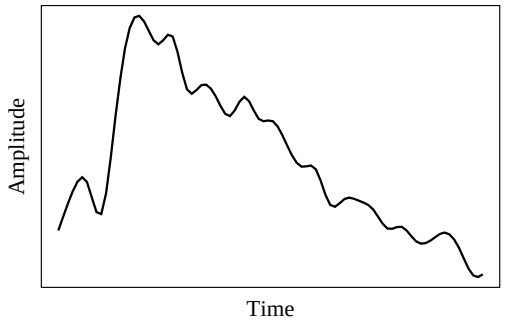


Fig. 6.1: An analog signal: continuous measurement of pressure wave.

- The graph in Fig. 6.1 has to be made digital in both time and amplitude. To digitize, the signal must be **sampled** in each dimension: in time, and in amplitude.
  (a) Sampling means measuring the quantity we are interested in, usually at evenly-spaced intervals.
  (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 frequency* (see Fig. 6.2(a)).
  (c) For audio, typical sampling rates are from 8 kHz (8,000 samples per second) to 48 kHz. This range is determined by Nyquist theorem discussed later.
  (d) Sampling in the amplitude or voltage dimension is called **quantization**. Fig. 6.2(b) shows this kind of sampling.

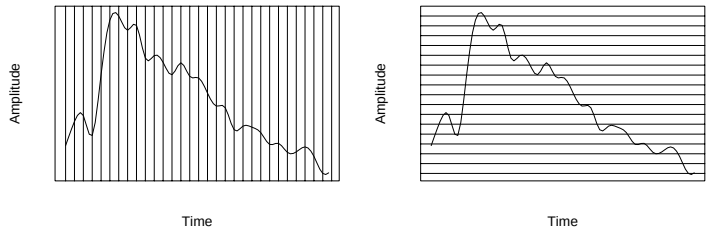


Fig. 6.2: Sampling and Quantization. (a): Sampling the analog signal in the time dimension. (b): Quantization is sampling the analog signal in the amplitude dimension.

- 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. Fig. 6.3 shows how weighted sinusoids can build up quite a complex signal.

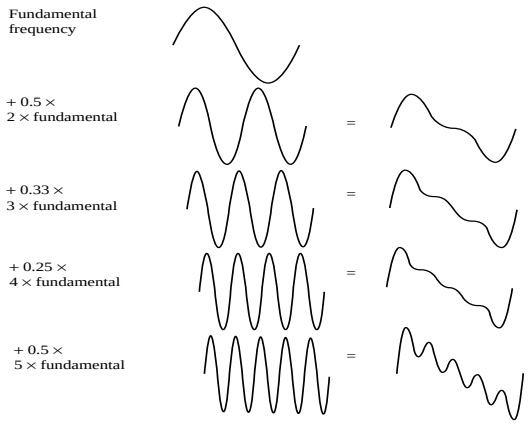


Fig. 6.3: Building up a complex signal by superposing sinusoids

- 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: Fig. 6.

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

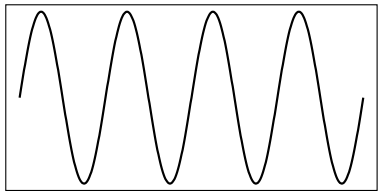
- The Nyquist theorem states how frequently we must sample in time to be able to recover the original sound.

(a) Fig. 6.4(a) shows a single sinusoid: it is a single, pure, frequency (only electronic instruments can create such sounds).

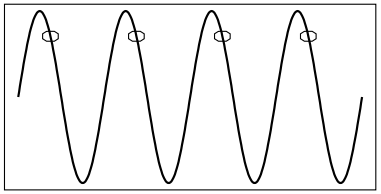
(b) If sampling rate just equals the actual frequency, Fig. 6.4(b) shows that a false signal is detected: it is simply a constant, with zero frequency.

(c) Now if sample at 1.5 times the actual frequency, Fig. 6.4(c) shows that we obtain an incorrect (**alias**) frequency that is lower than the correct one — it is half the correct one (the wavelength, from peak to peak, is double that of the actual signal).

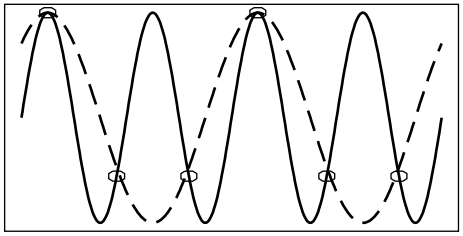
(d) Thus for correct sampling we must use a sampling rate equal to at least *twice the maximum frequency* content in the signal. This rate is called the **Nyquist rate**.



(a)



(b)



(c)

Fig. 6.4: Aliasing. (a): A single frequency. (b): Sampling at exactly the frequency produces a constant. (c): Sampling at 1.5 times per cycle produces an *alias* perceived frequency.

- **Nyquist Theorem**: If a signal is **band-limited**, i.e., there is a lower limit f_1 and an upper limit f_2 of frequency components in the signal, then the sampling rate should be at least $2(f_2 - f_1)$.

- **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 as follows:

$$f_{alias} = f_{sampling} - f_{true}, \quad \text{for} \quad f_{true} < f_{sampling} < 2 \times f_{true} \quad (6.1)$$

- In general, the apparent frequency of a sinusoid is the lowest frequency of a sinusoid that has exactly the same samples as the input sinusoid. Fig. 6.5 shows the relationship of the apparent frequency to the input frequency.

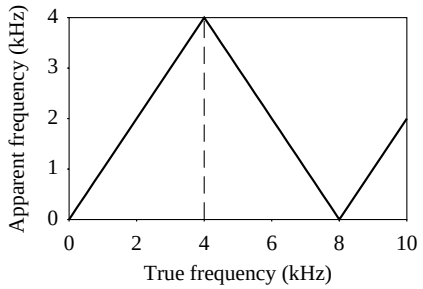


Fig. 6.5: Folding of sinusoid frequency which is sampled at 8,000 Hz. The folding frequency, shown dashed, is 4,000 Hz.

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, as follows:

$$SNR = 10 \log_{10} \frac{V_{signal}^2}{V_{noise}^2} = 20 \log_{10} \frac{V_{signal}}{V_{noise}} \qquad (6.2)$$

a) The power in a signal is proportional to the square of the voltage. For example, if the signal voltage V_{signal} is 10 times the noise, then the SNR is $20 * \log_{10}(10) = 20$dB.

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.

c) *To know:* Power — 10; Signal Voltage — 20.

- 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. Table 6.1 shows approximate levels for these sounds.

Table 6.1: Magnitude levels of common sounds, in decibels

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 that may have been present in the original analog signal, there is also an additional error that results from quantization.

(a) If voltages are actually in 0 to 1 but we have only 8 bits in which to store values, then effectively we force all continuous values of voltage 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.

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

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

$$= 20 \times N \times \log 2 = 6.02\,N(\text{dB}) \qquad (6.3)$$

- Notes:

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

(b) Eq. (6.3) is the *Peak* signal-to-noise ratio, PSQNR: peak signal and peak noise.

(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 ΔV=$(2V_{max})/2^N$, since there are 2^N intervals. The whole range V_{max} down to $(V_{max} - \Delta V/2)$ is mapped to $2^{N-1}-1$.

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

- $6.02N$ **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:

$$SQNR = 6.02N+1.76(dB) \qquad (6.4)$$

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:

$$\Delta\text{Response} \propto \Delta\text{Stimulus/Stimulus} \tag{6.5}$$

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

$$dr = k\,(1/s)\,ds \tag{6.6}$$

  with response r and stimulus s.

  - Integrating, we arrive at a solution

$$r = k \ln s + C \tag{6.7}$$

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

$$r = k \ln(s/s_0) \tag{6.8}$$

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

μ-law:

$$r = \frac{\text{sgn}(s)}{\ln(1+\mu)} \ln\left\{1+\mu\left|\frac{s}{s_p}\right|\right\}, \qquad \left|\frac{s}{s_p}\right| \le 1 \tag{6.9}$$

A-law:

$$r = \begin{cases} \frac{A}{1+\ln A}\left(\frac{s}{s_p}\right), & \left|\frac{s}{s_p}\right| \le \frac{1}{A} \\ \frac{\text{sgn}(s)}{1+\ln A}\left[1+\ln A\left|\frac{s}{s_p}\right|\right], & \frac{1}{A} \le \left|\frac{s}{s_p}\right| \le 1 \end{cases} \tag{6.10}$$

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

- Fig. 6.6 shows these curves. The parameter μ is set to $\mu = 100$ or $\mu = 255$; the parameter A for the A-law encoder is usually set to $A = 87.6$.

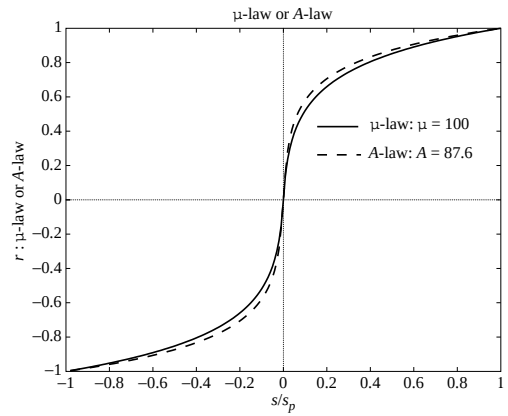


Fig. 6.6: Nonlinear transform for audio signals

- The μ-law in audio is used to develop a nonuniform quantization rule for sound: uniform quantization of r gives finer resolution in s at the quiet end.

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 the use of 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) So at the decoder side, a **lowpass** filter is used after the DA circuit.

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.

Table 6.2: Data rate and bandwidth in sample audio applications

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)

Synthetic Sounds

1. **FM** (Frequency Modulation): one approach to generating synthetic sound:

$$x(t) = A(t)\cos[\omega_c \pi t + I(t)\cos(\omega_m \pi t + \phi_m) + \phi_c] \qquad (6.11)$$

⟶ Link to details.

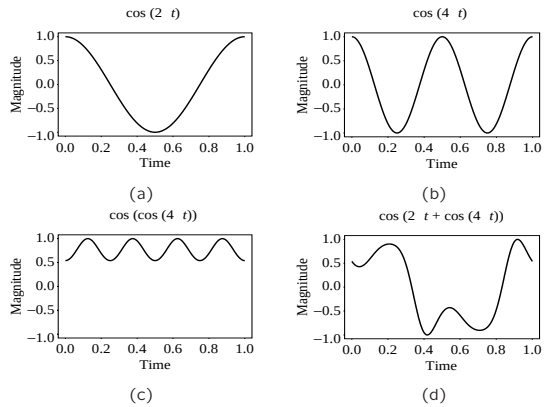


Fig. 6.7: Frequency Modulation. (a): A single frequency. (b): Twice the frequency. (c): Usually, FM is carried out using a sinusoid argument to a sinusoid. (d): A more complex form arises from a carrier frequency, $2\pi t$ and a modulating frequency $4\pi t$ cosine inside the sinusoid.

2. **Wave Table synthesis**: A more accurate way of generating sounds from digital signals. Also known, simply, as **sampling**.

   In this technique, the actual digital samples of sounds from real instruments are stored. Since wave tables are stored in memory on the sound card, they can be manipulated by software so that sounds can be combined, edited, and enhanced.

   ⟶ Link to details.

6.2 MIDI: Musical Instrument Digital Interface

- Use the sound card's defaults for sounds: ⇒ use a simple scripting language and hardware setup called **MIDI**.
- **MIDI Overview**

  (a) MIDI is a scripting language — it codes "events" that stand for the production of sounds. E.g., a MIDI event might include values for the pitch of a single note, its duration, and its volume.

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

(c) The 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 must have a special MIDI interface, but this is incorporated into most sound cards. The sound card must also have 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.

- **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.

- 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**.

- **General MIDI**: A standard mapping specifying what instruments (what patches) will be associated with what channels.

  (a) In General MIDI, channel 10 is reserved for percussion instruments, and there are 128 patches associated with standard instruments.

  (b) For most instruments, a typical message might be a Note On message (meaning, e.g., a keypress and release), consisting of what channel, what pitch, and what "velocity" (i.e., volume).

  (c) For percussion instruments, however, the pitch data means which kind of drum.

  (d) A Note On message consists of "status" byte — which channel, what pitch — followed by two data bytes. It is followed by a Note Off message, which also has a pitch (which note to turn off) and a velocity (often set to zero).

⟶ Link to General MIDI Instrument Patch Map

⟶ Link to General MIDI Percussion Key Map

- The data in a MIDI *status byte* is between 128 and 255; each of the *data bytes* is between 0 and 127. Actual MIDI bytes are 10-bit, including a 0 start and 0 stop bit.

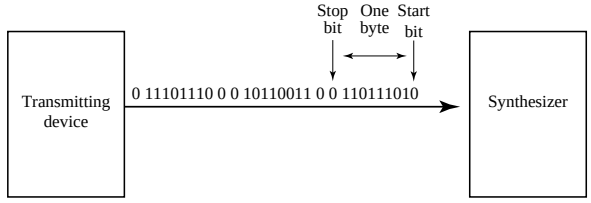


Fig. 6.8: Stream of 10-bit bytes; for typical MIDI messages, these consist of {Status byte, Data Byte, Data Byte} = {Note On, Note Number, Note Velocity}

- A MIDI device often is capable of **programmability**, and also can change the **envelope** describing how the amplitude of a sound changes over time.
- Fig. 6.9 shows a model of the response of a digital instrument to a Note On message:

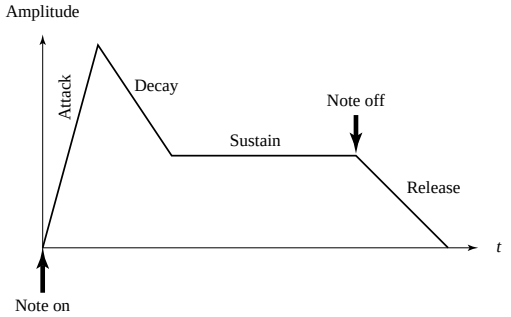


Fig. 6.9: Stages of amplitude versus time for a music note

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 in Fig. 6.10:

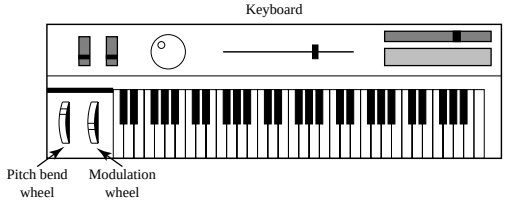


Fig. 6.10: A MIDI synthesizer

- 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 is shown in Fig. 6.11:

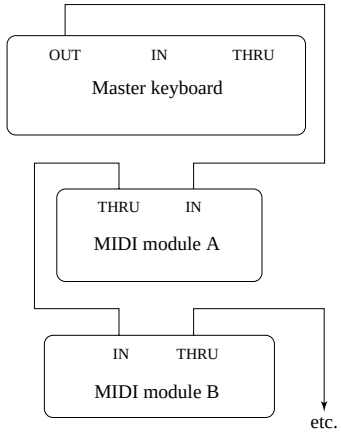


Fig. 6.11: A typical MIDI setup

Structure of MIDI Messages

- MIDI messages can be classified into two types: channel messages and system messages, as in Fig. 6.12:

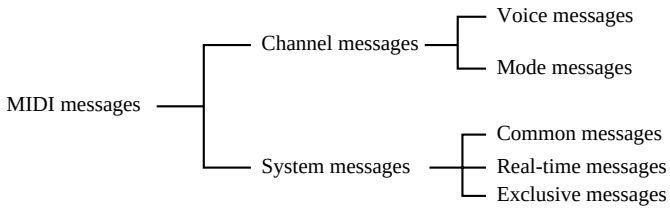


Fig. 6.12: MIDI message taxonomy

- **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 significant bit is set to **0**.

- **A.1. 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.

  c) Table 6.3 lists these operations.

Table 6.3: MIDI voice messages.

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

(** &H indicates hexadecimal, and 'n' in the status byte hex value stands for a channel number. All values are in 0..127 except Controller number, which is in 0..120)

- **A.2. Channel mode messages:**

  a) Channel mode messages: special case of the Control Change message $\longrightarrow$ opcode B (the message is &HBn, or 1011nnnn).

  b) However, a Channel Mode message has its first data byte in 121 through 127 (&H79–7F).

  c) 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.

  d) The data bytes have meanings as shown in Table 6.4.

Table 6.4: MIDI Mode Messages

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

- **B. 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:

B.1. **System common messages:** relate to timing or positioning.

Table 6.5: MIDI System Common messages.

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

B.2. **System real-time messages:** related to synchronization.

Table 6.6: MIDI System Real-Time messages.

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

B.3. **System exclusive message:** included so that the MIDI standard can 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, as specified in Table 6.

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.

b) These programs essentially consist of large lookup files that try to substitute pre-defined or shifted WAV output for MIDI messages, with inconsistent success.

6.3 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.

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 (→ expanded discussion in Chap.7).

- 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. ⟶ Repeat of Fig. 6.2:

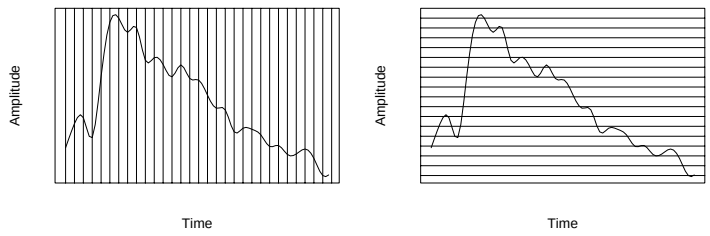


Fig. 6.2: Sampling and Quantization.

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 (Chap. 7).

- 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 (Chap. 7).

- For audio signals, we first consider PCM for digitization. This leads to Lossless Predictive Coding as well as the DPCM scheme; both methods use *differential coding*. As well, we look at the adaptive version, ADPCM, which can provide better compression.

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. Hence 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.

- However, there are two small wrinkles we must also address:

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

     – Also, once we arrive at a pulse signal, such as that in Fig. 6.13(a) below, we must still perform DA conversion and then construct a final output analog signal. But, effectively, the signal we arrive at is the staircase shown in Fig. 6.13(b).

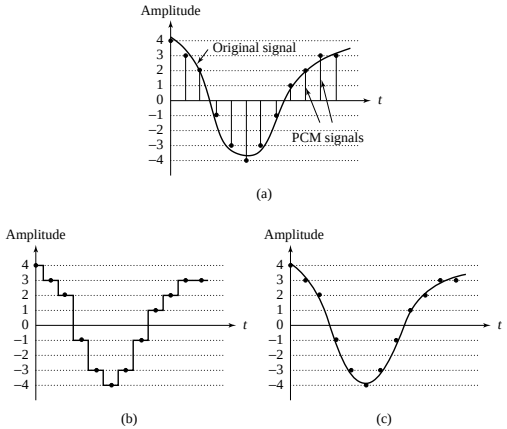


Fig. 6.13: Pulse Code Modulation (PCM). (a) Original analog signal and its corresponding PCM signals. (b) Decoded staircase signal. (c) Reconstructed signal after low-pass filtering.

2. A discontinuous signal contains not just frequency components due to the original signal, but also a theoretically infinite set of higher-frequency components:

   (a) This result is from the theory of **Fourier analysis**, in signal processing.

   (b) These higher frequencies are **extraneous**.

   (c) Therefore the output of the digital-to-analog converter goes to a **low-pass filter** that allows only frequencies up to the original maximum to be retained.

- The complete scheme for encoding and decoding telephony signals is shown as a schematic in Fig. 6.14. As a result of the low-pass filtering, the output becomes smoothed and Fig. 6.13(c) above showed this effect.

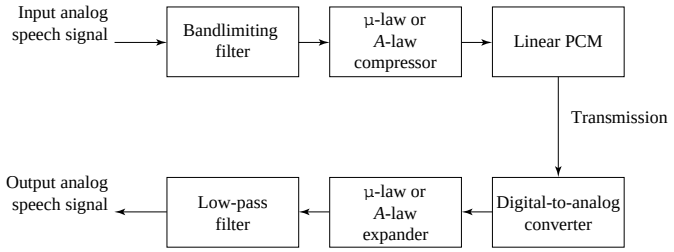


Fig. 6.14: PCM signal encoding and decoding.

Differential Coding of Audio

- Audio is often stored not in simple PCM but instead in a form that exploits differences — which are generally smaller numbers, so offer the possibility of using fewer bits to store.

  (a) If a time-dependent 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 around 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) So if we then go on to assign bit-string codewords to differences, we can assign short codes to prevalent values and long codewords to rarely occurring 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 $\widehat{f_n}$ as simply the previous value, and define the error e_n as the difference between the actual and the predicted signal:

$$\widehat{f_n} = f_{n-1}$$

$$e_n = f_n - \widehat{f_n} \tag{6.12}$$

(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:

$$\widehat{f_n} = \sum_{k=1}^{2 \text{ to } 4} a_{n-k} f_{n-k} \tag{6.13}$$

- The idea of forming differences is to make the histogram of sample values more peaked.

(a) For example, Fig.6.15(a) plots 1 second of sampled speech at 8 kHz, with magnitude resolution of 8 bits per sample.

(b) A histogram of these values is actually centered around zero, as in Fig. 6.15(b).

(c) Fig. 6.15(c) shows the histogram for corresponding speech signal **differences**: difference values are much more clustered around zero than are sample values themselves.

(d) As a result, a method that assigns short codewords to frequently occurring symbols will assign a short code to zero and do rather well: such a coding scheme will much more efficiently code sample differences than samples themselves.

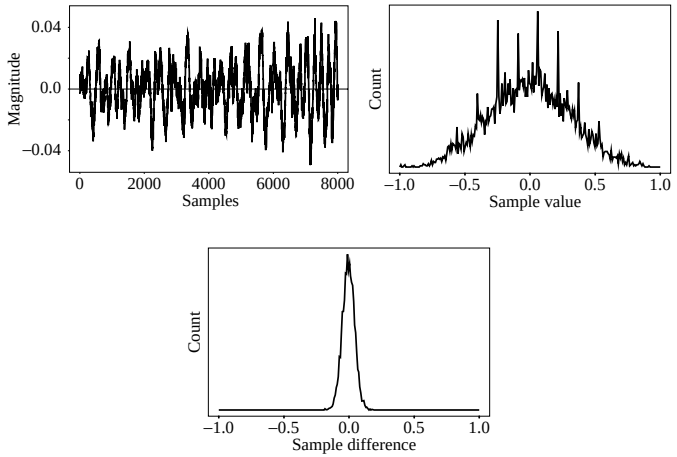


Fig. 6.15: Differencing concentrates the histogram. (a): Digital speech signal. (b): Histogram of digital speech signal values. (c): Histogram of digital speech signal differences.

- 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 ⟶ 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`. Some special code values will be reserved for these.

  (b) Then we can use codewords for only a limited set of signal differences, say only the range −15..16. Differences which lie in the limited range can be coded as is, but with the extra two values for `SU, SD`, a value outside the range −15..16 can be transmitted as a series of shifts, followed by a value that is indeed inside the range −15..16.

  (c) For example, 100 is transmitted as: `SU, SU, SU, 4`, where (the codes for) `SU` and for `4` are what are transmitted (or stored).

- *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:

$$\widehat{f_n} = \lfloor \tfrac{1}{2}(f_{n-1} + f_{n-2}) \rfloor$$
$$e_n = f_n - \widehat{f_n} \qquad (6.14)$$

- 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:

$$\widehat{f_2} = 21\,,\ e_2 = 22 - 21 = 1;$$
$$\widehat{f_3} = \lfloor \tfrac{1}{2}(f_2 + f_1) \rfloor = \lfloor \tfrac{1}{2}(22 + 21) \rfloor = 21,$$
$$e_3 = 27 - 21 = 6;$$
$$\widehat{f_4} = \lfloor \tfrac{1}{2}(f_3 + f_2) \rfloor = \lfloor \tfrac{1}{2}(27 + 22) \rfloor = 24,$$
$$e_4 = 25 - 24 = 1;$$
$$\widehat{f_5} = \lfloor \tfrac{1}{2}(f_4 + f_3) \rfloor = \lfloor \tfrac{1}{2}(25 + 27) \rfloor = 26,$$
$$e_5 = 22 - 26 = -4 \qquad (6.15)$$

- The error does center around zero, we see, and coding (assigning bit-string codewords) will be efficient. Fig. 6.16 shows a typical schematic diagram used to encapsulate this type of system:

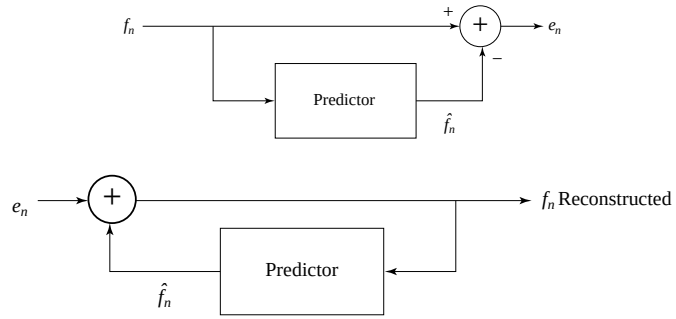


Fig. 6.16: Schematic diagram for Predictive Coding encoder and decoder.

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, $\widehat{f}_n$ – the predicted signal, and $\widetilde{f}_n$ 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, $\widetilde{e}_n$.
The set of equations that describe DPCM are as follows:

$$\begin{aligned} \widehat{f}_n &= \mathit{function\_of}(\widetilde{f}_{n-1},, \widetilde{f}_{n-2}, \widetilde{f}_{n-3}, ...) , \\ e_n &= f_n - \widehat{f}_n , \\ \widetilde{e}_n &= Q[e_n] , \\ &\text{transmit } codeword(\widetilde{e}_n) , \\ &\text{reconstruct: } \widetilde{f}_n = \widehat{f}_n + \widetilde{e}_n . \end{aligned} \tag{6.16}$$

Then *codewords* for quantized error values $\widetilde{e}_n$ are produced using entropy coding, e.g. Huffman coding (Chapter 7).

(d) 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 could 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. That is, starting at time i we could take a block of N values f_n and try to minimize the quantization error:

$$min \sum_{n=i}^{i+N-1} (f_n - Q[f_n])^2 \tag{6.17}$$

- Since signal **differences** are very peaked, we could 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 choosing steps to minimize

$$min \sum_{n=i}^{i+N-1} (d_n - Q[d_n])^2 l(d_n)\,. \tag{6.18}$$

- This is a least-squares problem, and can be solved iteratively == the Lloyd-Max quantizer.
- Schematic diagram for DPCM:

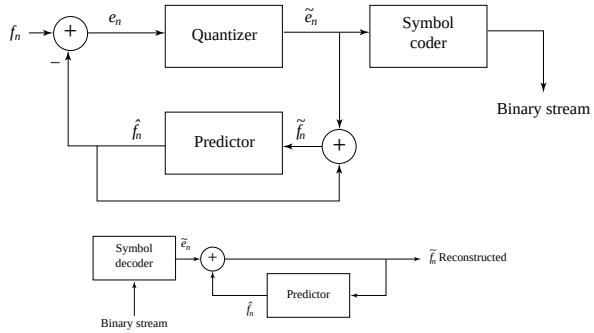


Fig. 6.17: Schematic diagram for DPCM encoder and decoder

- Notice that the quantization noise, $f_n - \tilde{f}_n$, is equal to the quantization effect on the error term, $e_n - \tilde{e}_n$.
- Let's look at actual numbers: Suppose we adopt the particular predictor below:

$$\hat{f}_n = \text{trunc}\left(\left(\tilde{f}_{n-1} + \tilde{f}_{n-2}\right)/2\right) \tag{6.19}$$

so that $e_n = f_n - \hat{f}_n$ is an integer.

- As well, use the quantization scheme:

$$\tilde{e}_n = Q[e_n] = 16 * \text{trunc}\left((255 + e_n)/16\right) - 256 + 8$$

$$\tilde{f}_n = \hat{f}_n + \tilde{e}_n \tag{6.20}$$

- First, we note that the error is in the range $-255..255$, i.e., there are 511 possible levels for the error term. The quantizer simply divides the error range into 32 patches of about 16 levels each. It also makes the representative reconstructed value for each patch equal to the midway point for each group of 16 levels.

- Table 6.7 gives output values for any of the input codes: 4-bit codes are mapped to 32 reconstruction levels in a staircase fashion.

Table 6.7 DPCM quantizer reconstruction levels.

e_n in range	Quantized to value
-255 .. -240	-248
-239 .. -224	-232
.	.
.	.
.	.
-31 .. -16	-24
-15 .. 0	-8
1 .. 16	8
17 .. 32	24
.	.
.	.
.	.
225 .. 240	232
241 .. 255	248

- As an example stream of signal values, consider the set of values:

f_1	f_2	f_3	f_4	f_5
130	150	140	200	230 .

- Prepend extra values $f = 130$ to replicate the first value, f_1. Initialize with quantized error $\tilde{e}_1 \equiv 0$, so that the first reconstructed value is exact: $\tilde{f}_1 = 130$. Then the rest of the values calculated are as follows (with prepended values in a box):

$$\begin{aligned}
\hat{f} &= \boxed{130}, \; 130, \; 142, \; 144, \; 167 \\
e &= \quad \boxed{0}, \; 20, \; -2, \; 56, \; 63 \\
\tilde{e} &= \quad \boxed{0}, \; 24, \; -8, \; 56, \; 56 \\
\tilde{f} &= \boxed{130}, \; 154, \; 134, \; 200, \; 223
\end{aligned}$$

- On the decoder side, we again assume extra values $\tilde{f}$ equal to the correct value $\tilde{f}_1$, so that the first reconstructed value $\tilde{f}_1$ is correct. What is received is $\tilde{e}_n$, and the reconstructed $\tilde{f}_n$ is identical to that on the encoder side, provided we use exactly the same prediction rule.

DM

- **DM** (Delta Modulation): simplified 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) $\Rightarrow$ a 1-bit coder. Produces coded output that follows the original signal in a staircase fashion. The set of equations is:

$$\begin{aligned} \hat{f}_n &= \tilde{f}_{n-1}\,, \\ e_n &= f_n - \hat{f}_n \;=\; f_n - \tilde{f}_{n-1}\,, \\ \tilde{e}_n &= \begin{cases} +k & \text{if } e_n > 0, \text{ where } k \text{ is a constant} \\ -k & otherwise \end{cases} \\ \tilde{f}_n &= \hat{f}_n + \tilde{e}_n\,. \end{aligned} \tag{6.21}$$

Note that the prediction simply involves a **delay**.

(b) Consider actual numbers: Suppose signal values are

f_1	f_2	f_3	f_4
10	11	13	15 .

As well, define an exact *reconstructed* value $\tilde{f}_1 = f_1 = 10$.

(c) E.g., use step value $k = 4$:

$$\begin{array}{llll} \hat{f}_2 = 10, & e_2 = 11 - 10 = 1, & \tilde{e}_2 = 4, & \tilde{f}_2 = 10 + 4 = 14 \\ \hat{f}_3 = 14, & e_3 = 13 - 14 = -1, & \tilde{e}_3 = -4, & \tilde{f}_3 = 14 - 4 = 10 \\ \hat{f}_4 = 10, & e_4 = 15 - 10 = 5, & \tilde{e}_4 = 4, & \tilde{f}_4 = 10 + 4 = 14\,. \end{array}$$

The reconstructed set of values 10, 14, 10, 14 is close to the correct set 10, 11, 13, 15.

(d) However, DM copes less well with rapidly changing signals. One approach to mitigating this problem is to simply increase the sampling, perhaps to many times the Nyquist rate.

2. **Adaptive DM**: If the slope of the actual signal curve is high, the staircase approximation cannot keep up. For a steep curve, should 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.

2. 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

$$\hat{f}_n = \sum_{i=1}^{M} a_i \tilde{f}_{n-i} \tag{6.22}$$

- 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:

$$min \sum_{n=1}^{N} (f_n - \hat{f}_n)^2 \tag{6.23}$$

  (b) Here we would sum over a large number of samples f_n, for the current patch of speech, say. But because $\hat{f}_n$ 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.

   (c) 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 a_i, we wish to solve:

$$min \sum_{n=1}^{N} (f_n - \sum_{i=1}^{M} a_i f_{n-i})^2 \tag{6.24}$$

   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.)

- Fig. 6.18 shows a schematic diagram for the ADPCM coder and decoder:

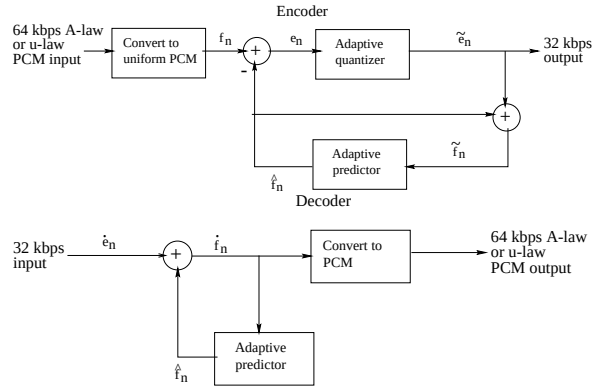


Fig. 6.18: Schematic diagram for ADPCM encoder and decoder

6.4 Further Exploration

⟶ Link to Further Exploration for Chapter 6.

The Useful links included are:

- An excellent discussion of the use of FM to create synthetic sound.
- An extensive list of audio file formats.
- CD audio file formats are somewhat different. The main music format is called "red book audio." A good description of various CD formats is on the website.
- A General MIDI Instrument Patch Map, along with a General MIDI Percussion Key Map.
- A link to good tutorial on MIDI and wave table music synthesis.
- A link to a java program for decoding MIDI streams.
- A good multimedia/sound page, including a source for locating Internet sound/music materials.