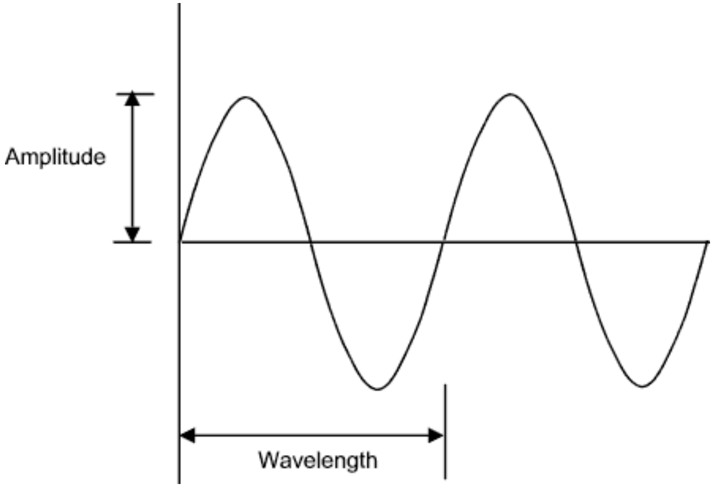
Recap of Basic Digital Audio Signal Processing

Recall some basic aspects of Digital Audio Signal Processing from CM0268:

- Some basic definitions and principles
- Filtering

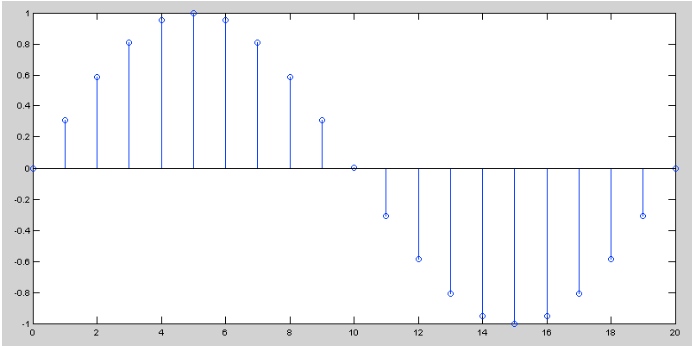
Simple Waveforms



- **Frequency** is the number of cycles per second and is measured in Hertz (Hz)
- **Wavelength** is *inversely proportional* to frequency

  i.e. Wavelength varies as $\frac{1}{frequency}$

The Sine Wave and Sound



The general form of the sine wave we shall use (quite a lot of) is as follows:

$$y = A.sin(2\pi.n.F_w/F_s)$$

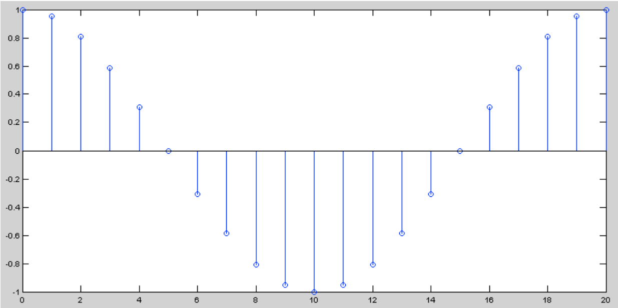
where:

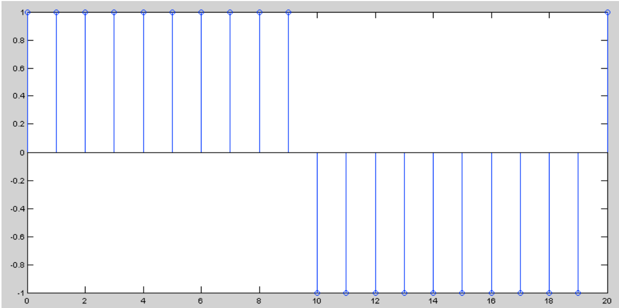
A is the amplitude of the wave,
F_w is the frequency of the wave,
F_s is the sample frequency,
n is the sample index.

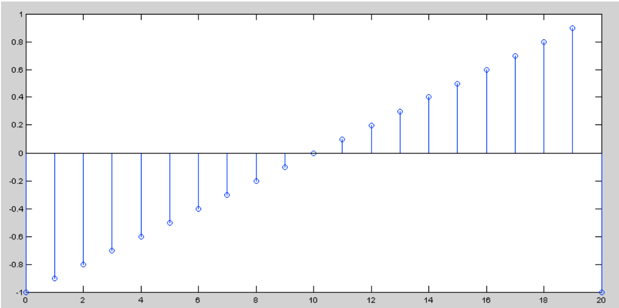
MATLAB function: `sin()` used — works in radians

Cosine, Square and Sawtooth Waveforms

MATLAB functions `cos()` (cosine), `square()` and `sawtooth()` similar.







The Decibel (dB)

When referring to measurements of power or intensity, we express these in decibels (dB):

$$X_{dB} = 10\log_{10}\left(\frac{X}{X_0}\right)$$

where:

- X is the actual value of the quantity being measured,
- X_0 is a specified or implied reference level,
- X_{dB} is the quantity expressed in units of decibels, relative to X_0.
- X and X_0 **must** have the same dimensions — they must measure the same type of quantity in the the same units.
- The reference level itself is **always at 0 dB** — as shown by setting $X = X_0$ (**note:** $\log_{10}(1) = 0$).

Why Use Decibel Scales?

- When there is a large range in frequency or magnitude, logarithm units often used.
- If X is greater than X_0 then X_{dB} is positive (Power Increase)
- If X is less than X_0 then X_{dB} is negative (Power decrease).
- Power Magnitude $= |X(i)|^2|$ so (with respect to reference level)

$$
\begin{aligned}
X_{dB} &= 10\log_{10}(|X(i)^2|) \\
&= 20\log_{10}(|X(i)|)
\end{aligned}
$$

which is an expression of dB we often come across.

Decibel and acoustics

- dB is commonly used to quantify sound levels relative to some 0 dB reference.
- The reference level is typically set at the *threshold of human perception*
- Human ear is capable of detecting a very large range of sound pressures (**see later**).

Examples of dB measurement in Sound

Threshold of Pain : The ratio of sound pressure that causes permanent damage from short exposure to the limit that (undamaged) ears can hear is above a million:

- The ratio of the maximum power to the minimum power is above one (short scale) trillion (10^{12}).
- The log of a trillion is 12, so this ratio represents a **difference of 120 dB**.

Speech Sensitivity : Human ear is not equally sensitive to all the frequencies of sound within the entire spectrum (**see later**):

- Noise levels at maximum human sensitivity — between 2 and 4 kHz (Speech) are factored more heavily into sound descriptions using a process called frequency weighting.

Examples of dB measurement in Sound (cont.)

6dB per bit : In digital audio sample representation (**linear pulse-code modulation (PCM)**),

- The first bit (least significant bit, or LSB) produces residual quantization noise (bearing little resemblance to the source signal)
- Each subsequent bit offered by the system **doubles** the resolution, corresponding to a 6 ($= 10 * \log_{10}(4)$) dB.
- So a 16-bit (linear) audio format offers 15 bits beyond the first, for a dynamic range (between quantization noise and clipping) of (15 x 6) = 90 dB, meaning that the maximum signal is 90 dB above the theoretical peak(s) of quantisation noise.
- 8-bit linear PCM similarly gives (7 x 6) = 42 dB.
- 48 dB difference between 8- and 16-bit which is (48/6 (dB)) 8 times as noisy.

Signal to Noise

Signal-to-noise ratio is a term for the power ratio between a signal (meaningful information) and the background noise:

$$SNR = \frac{P_{signal}}{P_{noise}} = \left(\frac{A_{signal}}{A_{noise}}\right)^2$$

where P is average power and A is RMS amplitude.

- Both signal and noise power (or amplitude) must be measured at the same or equivalent points in a system, and within the same system bandwidth.

Because many signals have a very wide dynamic range, SNRs are usually expressed in terms of the logarithmic decibel scale:

$$SNR_{dB} = 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) = 20\log_{10}\left(\frac{A_{signal}}{A_{noise}}\right)$$

Filtering

Filtering in a broad sense is selecting portion(s) of data for some processing.

In many multimedia contexts this involves the removal of data from a signal — This is essential in almost all aspects of lossy multimedia data representations.

We will look at filtering in the frequency space very soon, but first we consider Filtering via impulse responses.

We will look at:

IIR Systems : Infinite impulse response systems

FIR Systems : Finite impulse response systems

Infinite Impulse Response (IIR) Systems

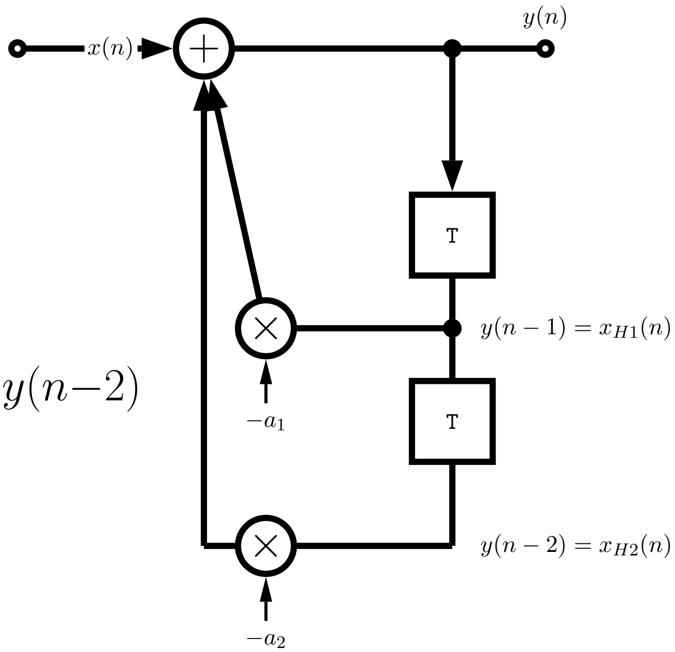
If $h(n)$ is an infinite impulse response function then the digital system is called and IIR system.

Example:

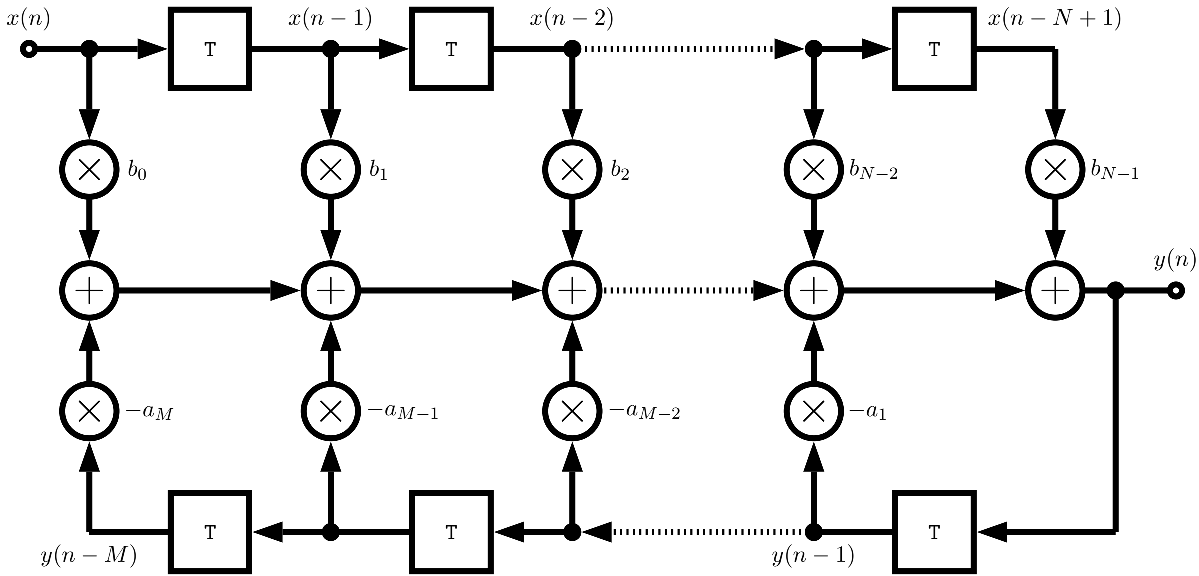
- The algorithm is represented by the difference equation:

$$y(n) = x(n) - a_1 y(n-1) - a_2 y(n-2)$$

- This produces the opposite signal flow graph



A Complete IIR System



Here we extend:

The **input** delay line up to $N-1$ elements and

The **output** delay line by M elements.

Filtering with IIR

We have **two** **filter banks** defined by vectors: $A = \{a_k\}$, $B = \{b_k\}$.

These can be applied in a *sample-by-sample* algorithm:

- MATLAB provides a generic `filter(B,A,X)` function which filters the data in vector `X` with the filter described by vectors `A` and `B` to create the filtered data `Y`.

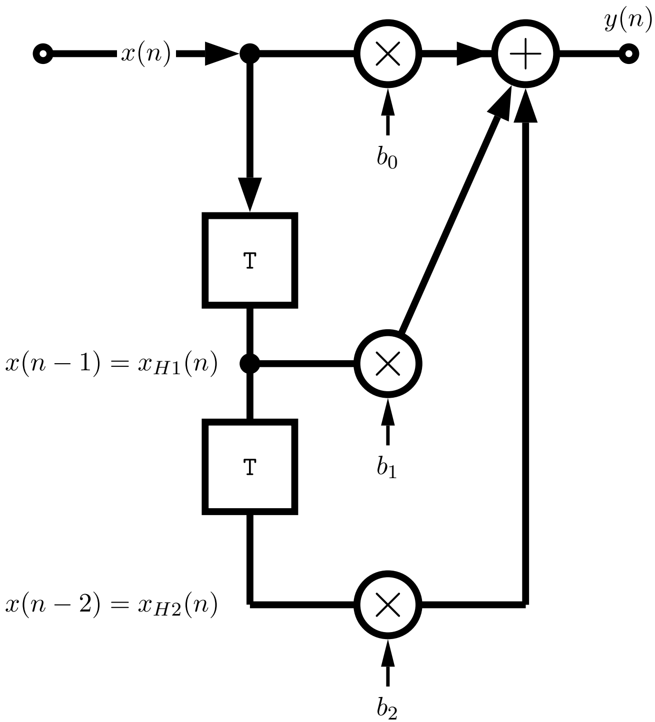
  The filter is of the standard difference equation form:

$$\begin{aligned} a(1) * y(n) &= b(1) * x(n) + b(2) * x(n-1) + \ldots + b(nb+1) * x(n-nb) \\ &\quad - a(2) * y(n-1) - \ldots - a(na+1) * y(n-na) \end{aligned}$$

- Filter banks can be created manually or MATLAB can provide some predefined filters — **see CM0268**
- See also `help filter`, online MATLAB docs and tutorials on filters.

Finite Impulse Response (FIR) Systems

FIR system's are slightly simpler — there is no feedback loop.



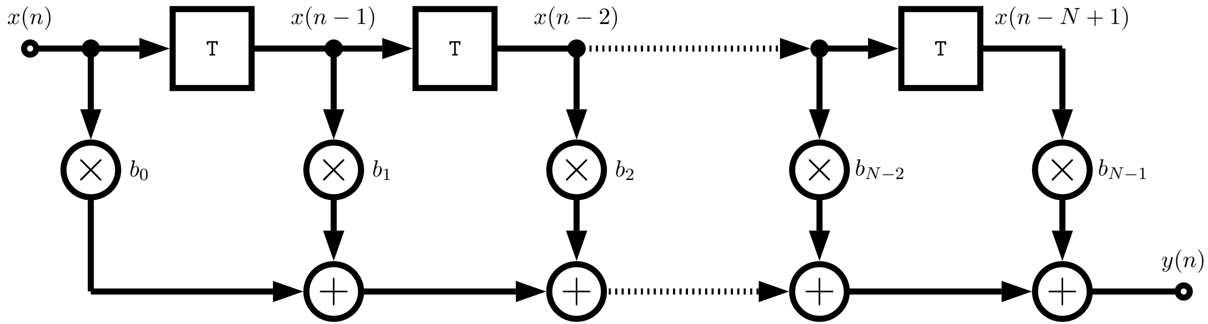
A simple FIR system can be described as follows:

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2)$$

- The input is fed through delay elements
- Weighted sum of delays give $y(n)$

A Complete FIR System

To develop a more complete FIR system we need to add $N-1$ feed forward delays:



We can describe this with the algorithm:

$$y(n) = \sum_{k=0}^{N-1} b_k \, x(n-k)$$