CM0340: Multimedia
Tutorial 1
Digital Signal Processing Recap
Basic DSP and Filters

Prof. David Marshall

School of Computer Science & Informatics

January 30, 2013

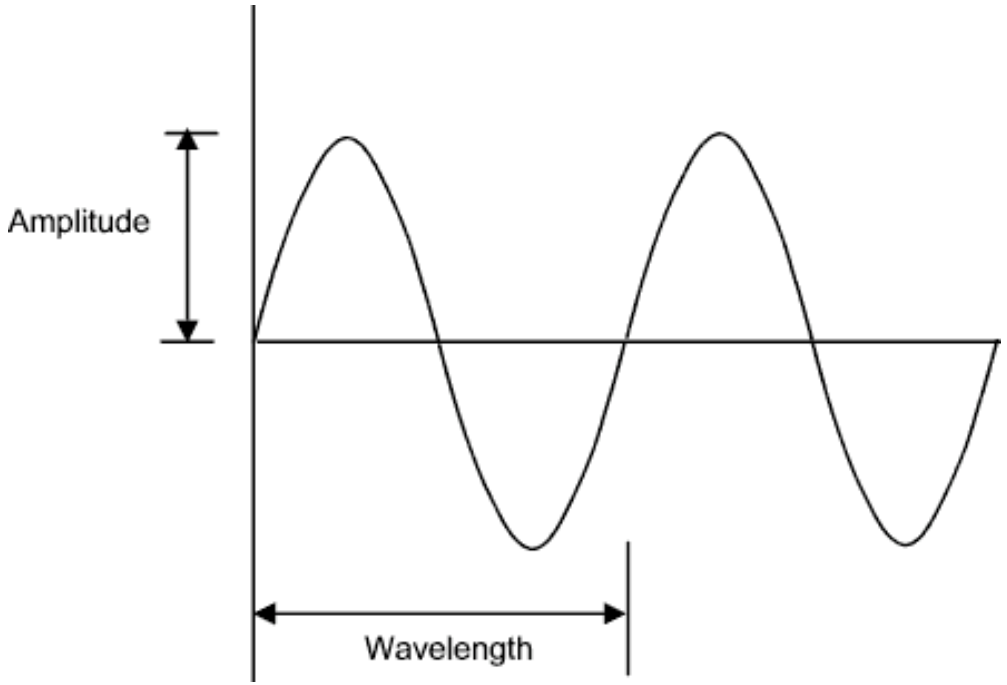
Digital Signal Processing and Digital Audio Recap from CM2202

Issues to be Recapped:

- Basic Digital Signal Processing and Digital Audio
  - Waveforms and Sampling Theorem
  - Digital Audio Signal Processing
  - Filters

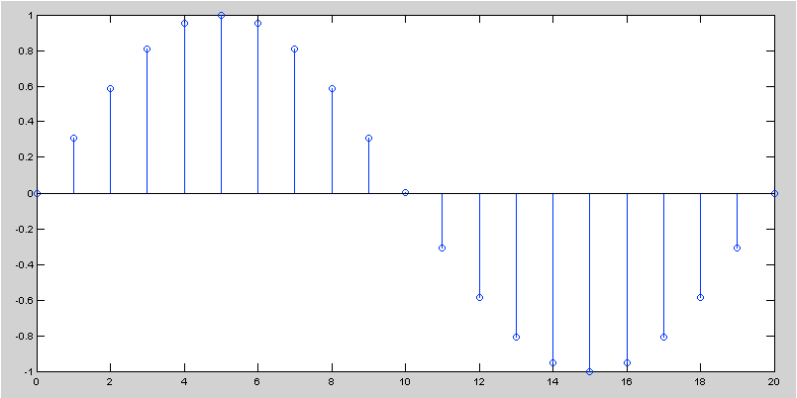
For full details please refer to last Year's **CM2202 Course Material** — **Especially detailed underpinning maths**.

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:

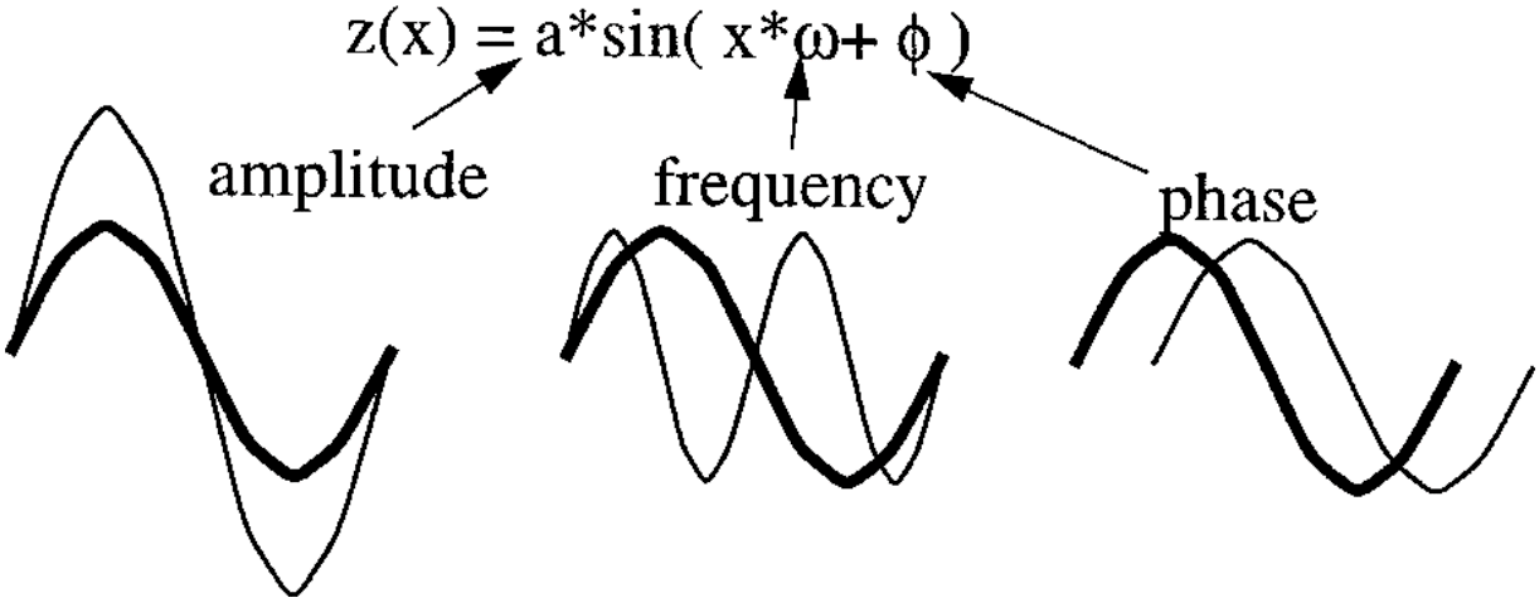
$$y = A.sin(2\pi.n.F_w/F_s)$$

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

Relationship Between Amplitude, Frequency and Phase

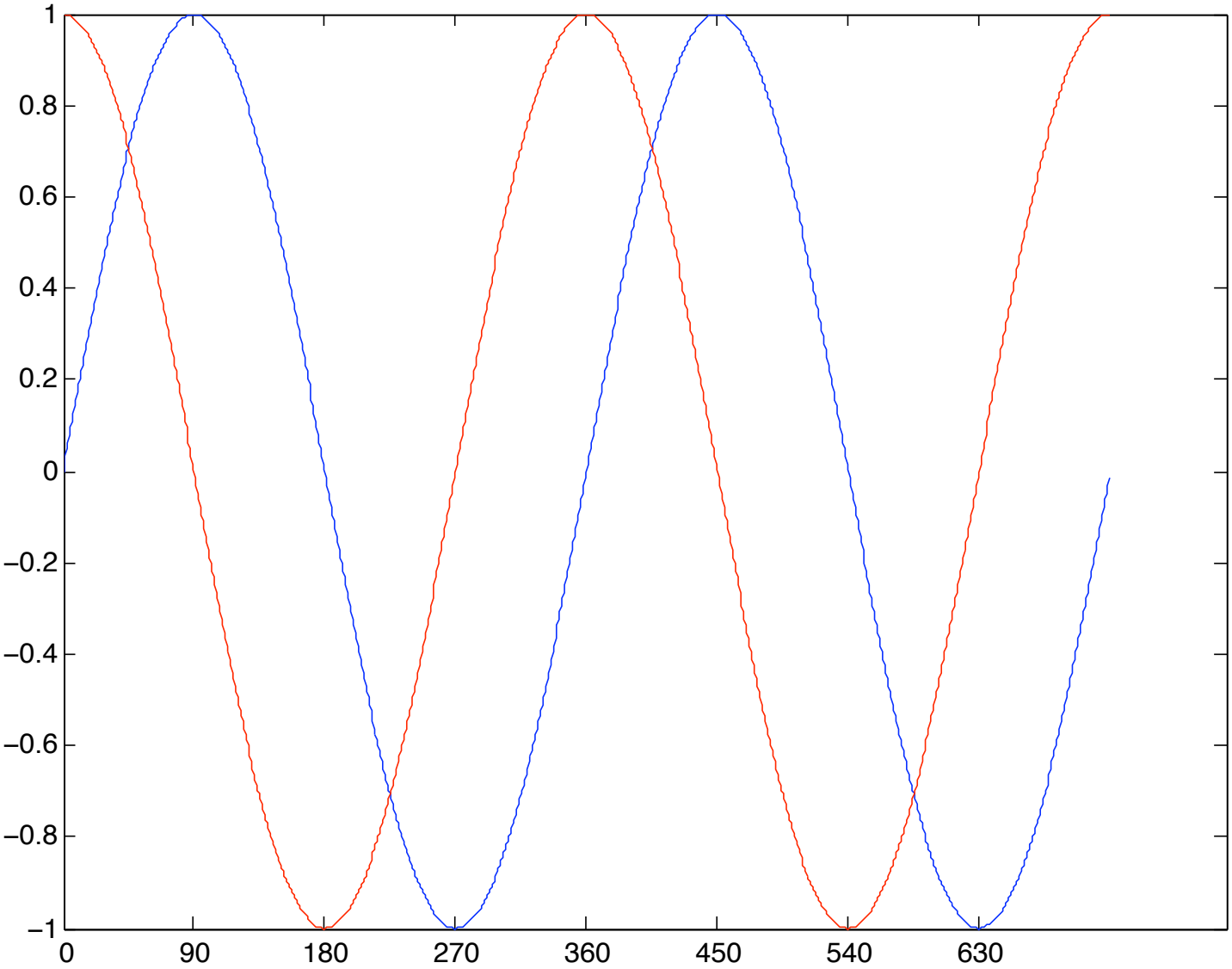


Phase of a Sine Wave

sinphasedemo.m

```
% Simple Sin Phase Demo

samp_freq = 400;
dur = 800; % 2 seconds
amp = 1; phase = 0; freq = 1;
s1 = mysin(amp,freq,phase,dur,samp_freq);

axisx = (1:dur)*360/samp_freq; % x axis in degrees
plot(axisx,s1);
set(gca,'XTick',[0:90:axisx(end)]);

fprintf('Initial Wave: \t Amplitude = ...\n', amp, freq, phase,...

% change amplitude
phase = input('\nEnter Phase:\n\n');

s2 = mysin(amp,freq,phase,dur,samp_freq);
hold on;
plot(axisx, s2,'r');
set(gca,'XTick',[0:90:axisx(end)]);
```

Phase of a Sine Wave: `sinphasedemo` output



Sample Rates and Bit Size

Bit Size — Quantisation

How do we store each sample value (**Quantisation**)?

8 Bit Value (0-255)

16 Bit Value (Integer) (0-65535)

Sample Rate

How many Samples to take?

11.025 KHz — Speech (Telephone 8 KHz)

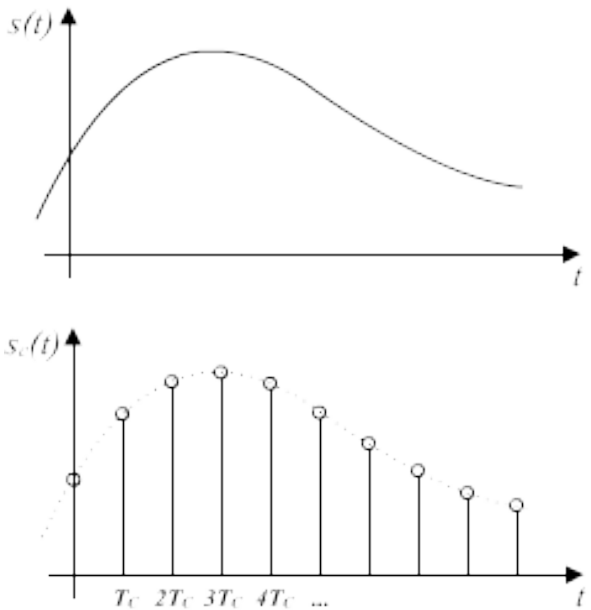
22.05 KHz — Low Grade Audio (WWW Audio, AM Radio)

44.1 KHz — CD Quality

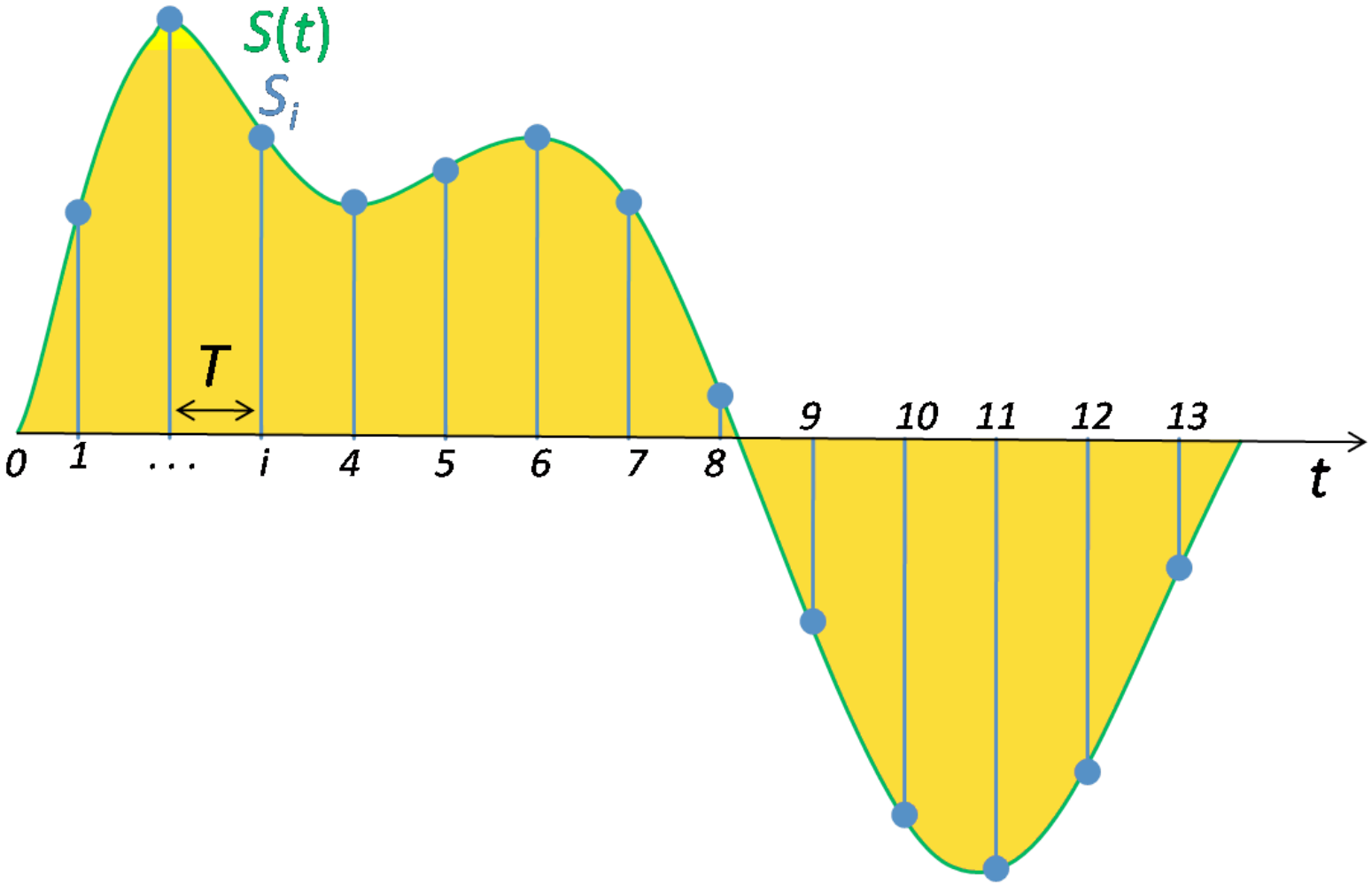
Digital Sampling (1)

Sampling process basically involves:

- **Measuring** the **analog signal** at **regular discrete intervals**
- **Recording** the **value** at **these points**



Digital Sampling (2)



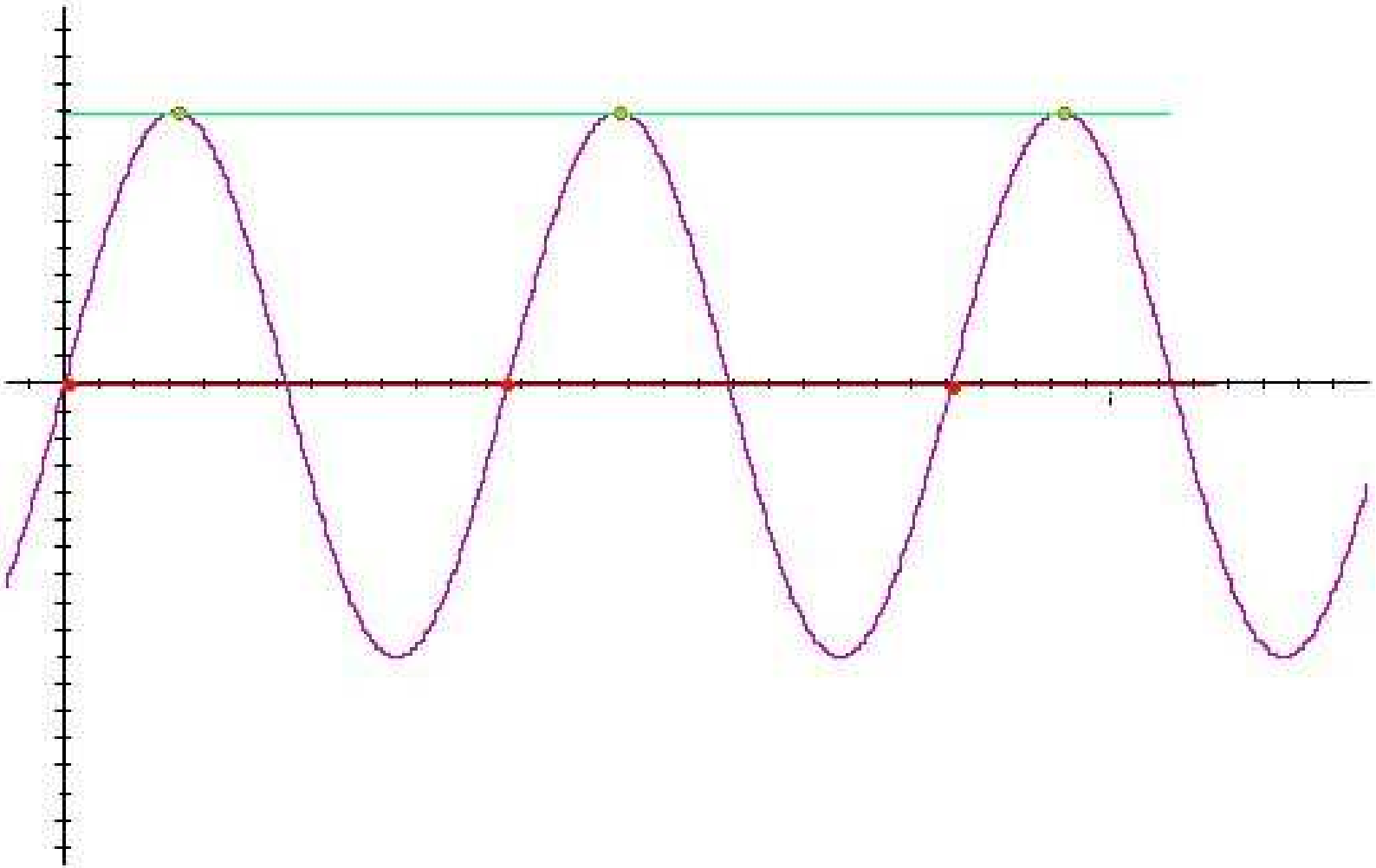
Nyquist's Sampling Theorem

The **Sampling Frequency** is **critical** to the **accurate reproduction** of a **digital version** of an analog waveform

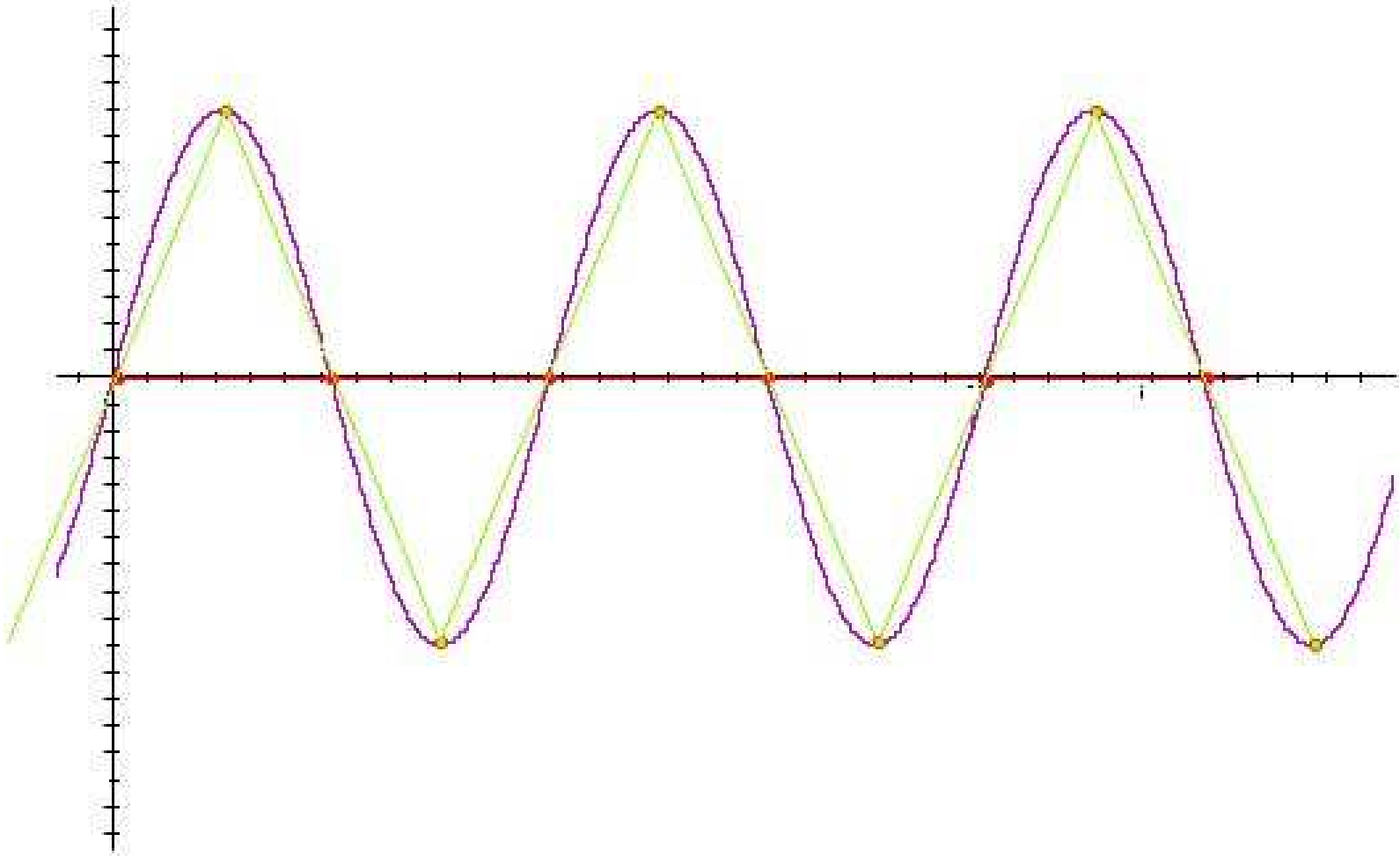
Nyquist's Sampling Theorem

The Sampling frequency for a signal must be **at least twice** the **highest frequency component** in the signal.

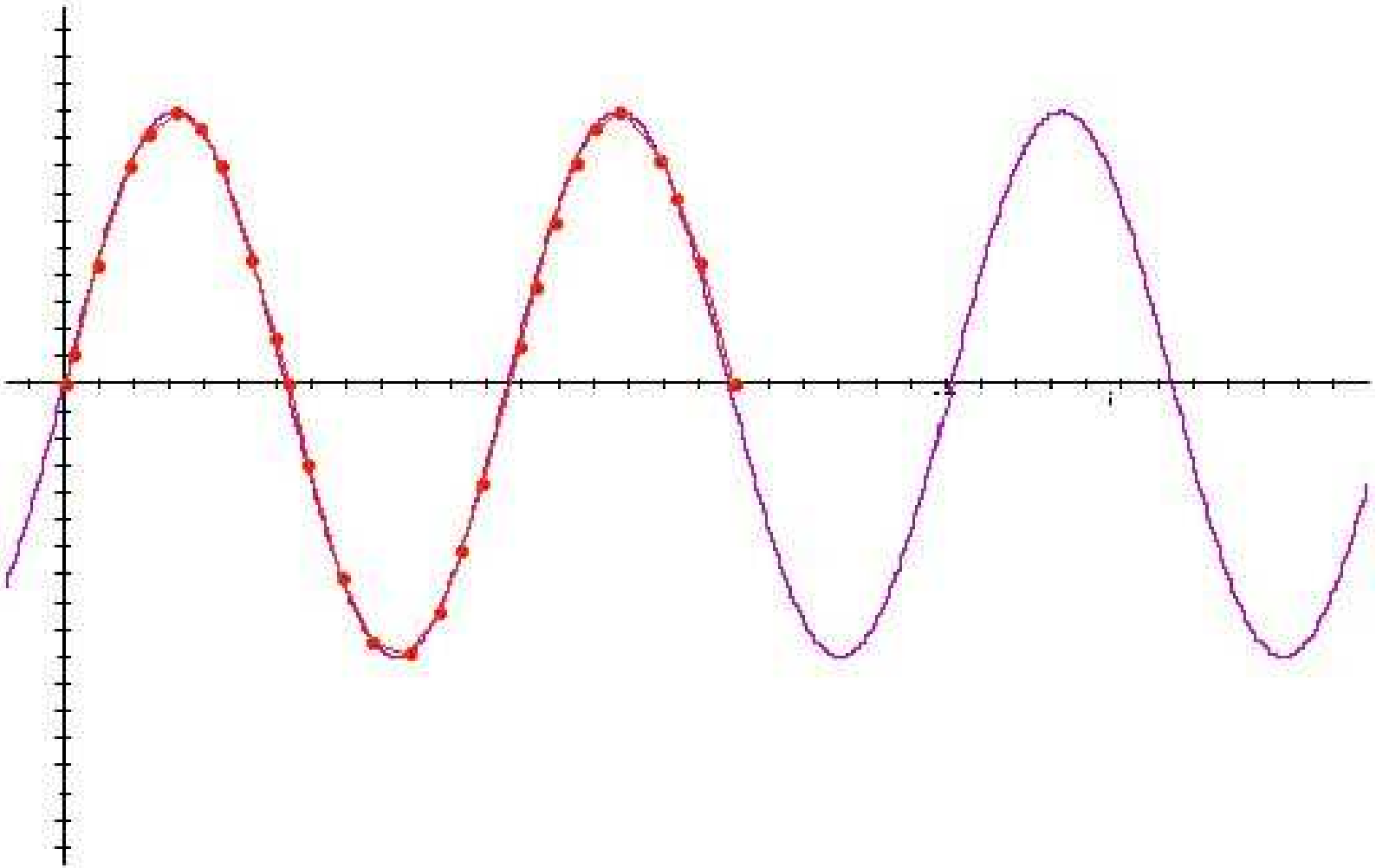
Sampling at Signal Frequency



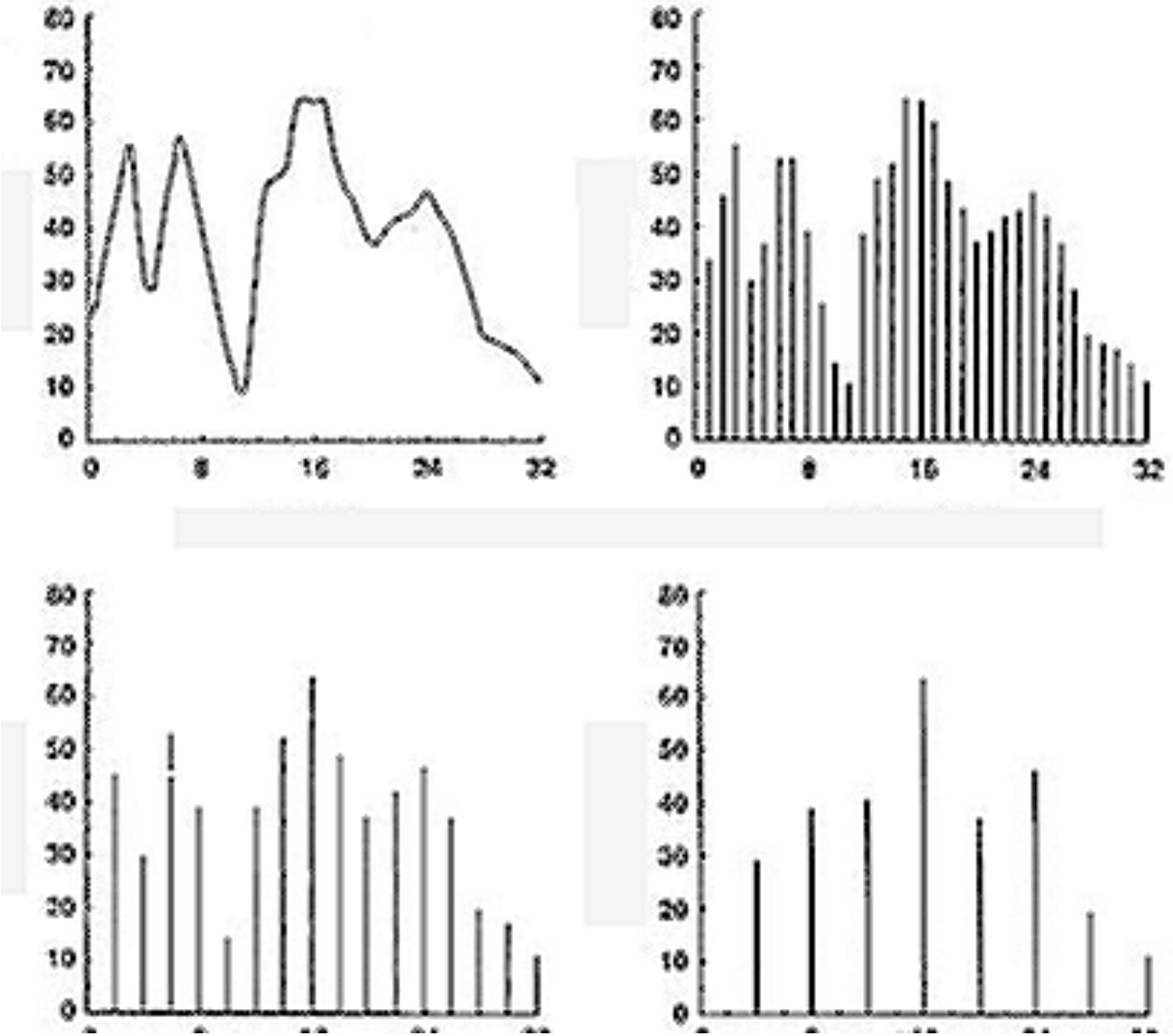
Sampling at Twice Nyquist Frequency



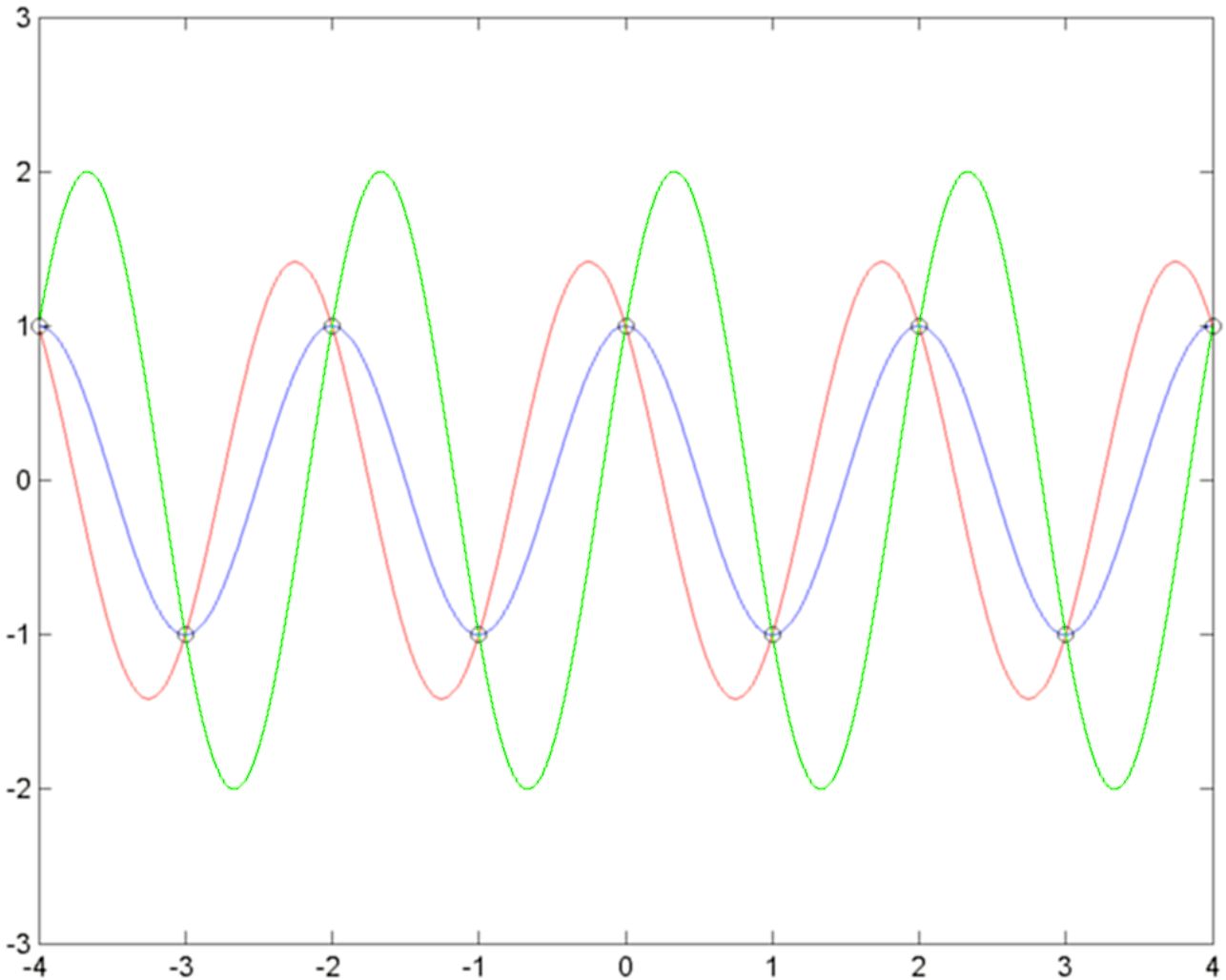
Sampling at above Nyquist Frequency



If you get Nyquist Sampling Wrong? (1)



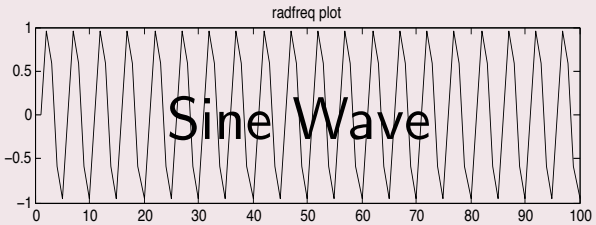
If you get Nyquist Sampling Wrong? (2)

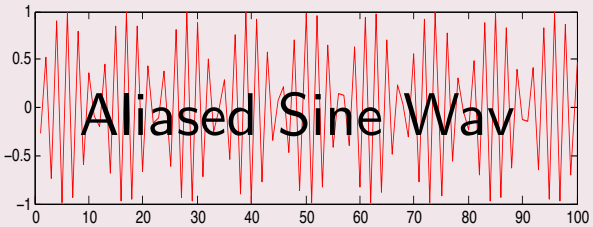


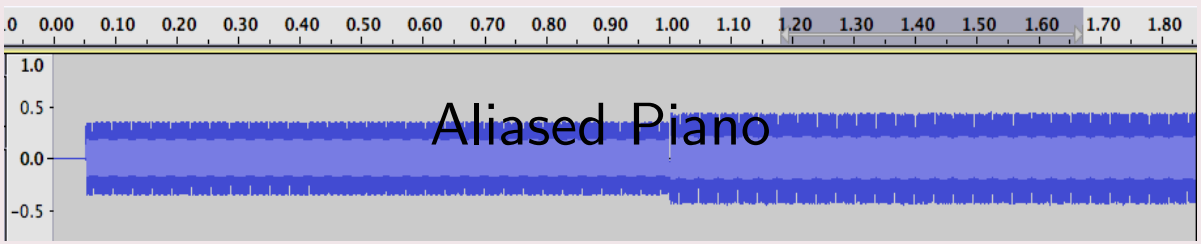
If you get Nyquist Sampling Wrong? (3)

What does aliasing sound like?

(**Click on Images to play sounds**)







MATLAB Code for Sine Demos above: Plot Version , Audio Version

Implications of Sample Rate and Bit Size (1)

Affects Quality of Audio

- Ears do not respond to sound in a linear fashion
- Decibel (**dB**) a logarithmic measurement of sound
- 16-Bit has a signal-to-noise ratio of 98 dB — virtually inaudible
- 8-bit has a signal-to-noise ratio of 50 dB
- Therefore, 8-bit is roughly 8 times as noisy
  - 6 dB increment is twice as loud

Implications of Sample Rate and Bit Size (2)

Audio Sample Rate and Bit Size Examples

File Type	Audio File (all mono)
44Hz 16 bit	
44KHz 8-bit	
22 KHz 16-bit	
22KHz 8-Bit	
11KHz 8-bit	

Web Link: Click Here to Hear Sound Examples

Implications of Sample Rate and Bit Size (2)

Affects Size of Data

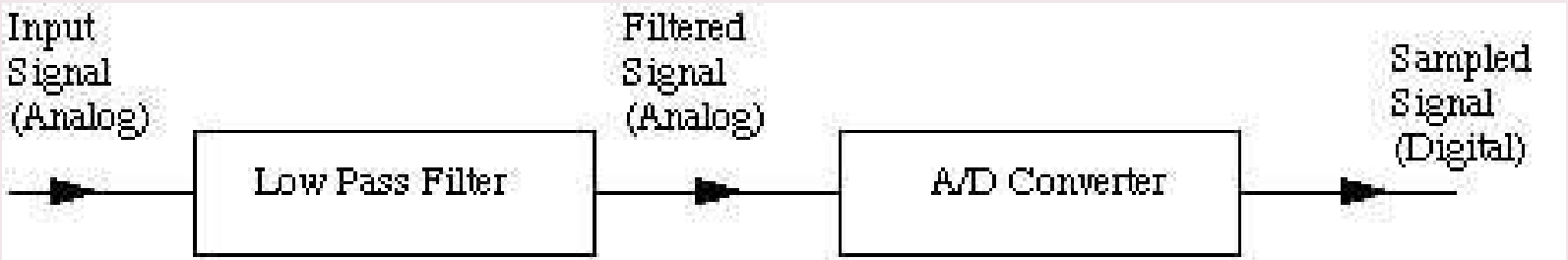
File Type	*44.1 KHz*	*22.05 KHz*	*11.025 KHz*
16 Bit Stereo	10.1 Mb	5.05 Mb	2.52 Mb
16 Bit Mono	5.05 Mb	2.52 Mb	1.26 Mb
8 Bit Mono	2.52 Mb	1.26 Mb	630 Kb

Memory Required for **1 Minute** of Digital Audio

Practical Implications of Nyquist Sampling Theory

Filtering of Signal

- Must (low pass) filter signal before sampling:



- Otherwise **strange artifacts** from high frequency (**above** Nyquist Limit)signals would appear in the sampled signal.

Basic DSP Concepts and Definitions: 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.

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**.
- **120 dB** is the quoted **Threshold of Pain** for Humans.

Examples of dB measurement in Sound (cont.)

Speech Sensitivity

Human ear is not equally sensitive to all the frequencies of sound within the entire spectrum:

- Maximum human sensitivity at noise levels at between 2 and 4 kHz (Speech)
  - These are factored more heavily into sound descriptions using a process called **frequency weighting**.
  - Filter (Partition) into frequency bands concentrated in this range.
  - Used for Speech Analysis
  - Mathematical Modelling of Human Hearing
  - Audio Compression (E.g. **MPEG Audio**)

Examples of dB measurement in Sound (cont.)

Digital Noise increases by 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:

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

System Representation: Algorithms and Signal Flow Graphs

It is common to represent digital system signal processing routines as a visual **signal flow graphs**.

We use a simple *equation* relation to describe the **algorithm**.

Three Basic Building Blocks

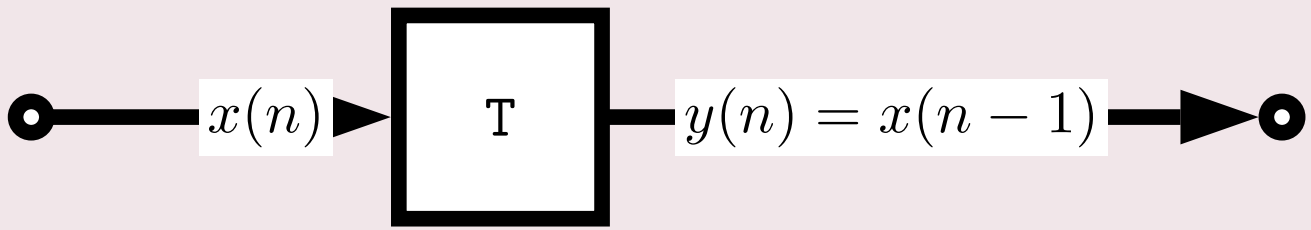
We will need to consider *three* processes:

- Delay
- Multiplication
- Summation

Signal Flow Graphs: Delay

Delay

- We represent a delay of **one sampling interval** by a block with a T **label**:



- We describe the algorithm via the equation: $y(n) = x(n-1)$

Signal Flow Graphs: Delay Example

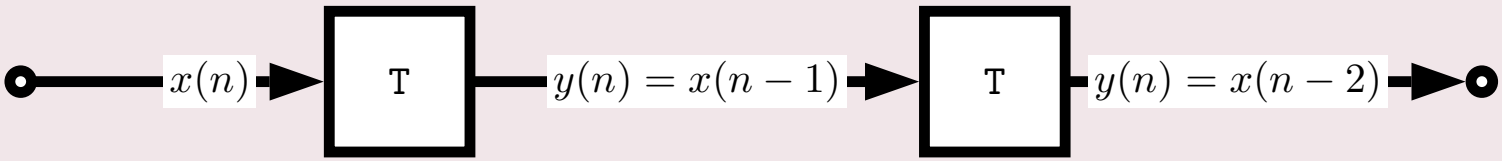
A Delay of 2 Samples

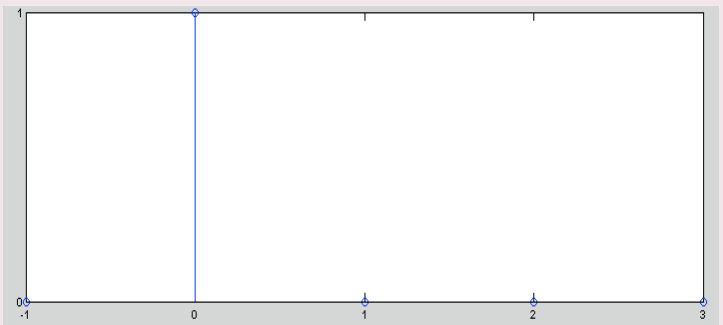
A delay of the input signal by **two** sampling intervals:

- We can describe the **algorithm** by:

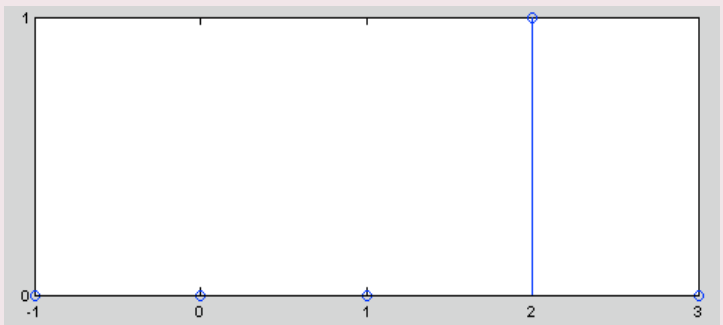
$$\mathbf{y(n) = x(n-2)}$$

- We can use the block diagram to represent the **signal flow graph** as:





$x(n)$

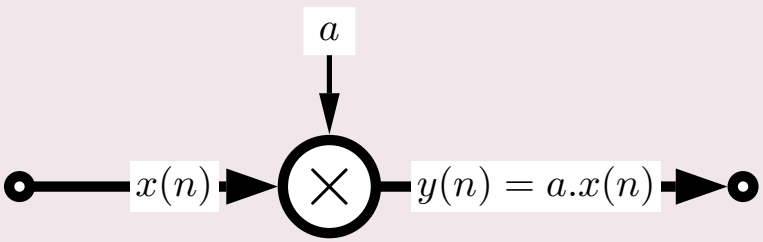


$y(n) = x(n-2)$

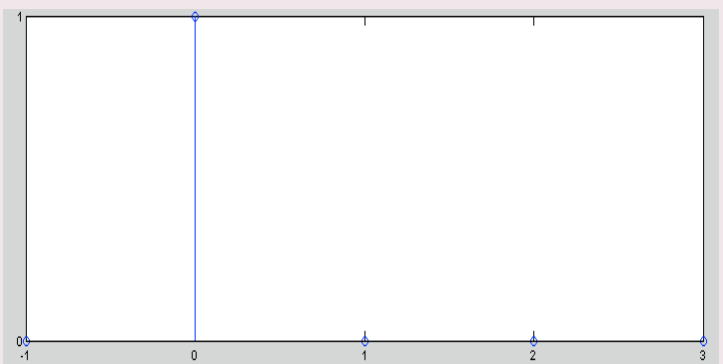
Signal Flow Graphs: Multiplication

Multiplication

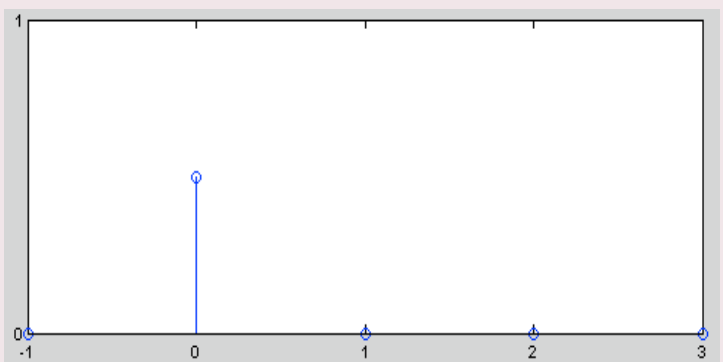
- We represent a multiplication or weighting of the input signal by **a circle with a $\times$ label** .
- We describe the algorithm via the equation: $\mathbf{y(n) = a.x(n)}$



$e.g.\ a = 0.5$



$x(n)$



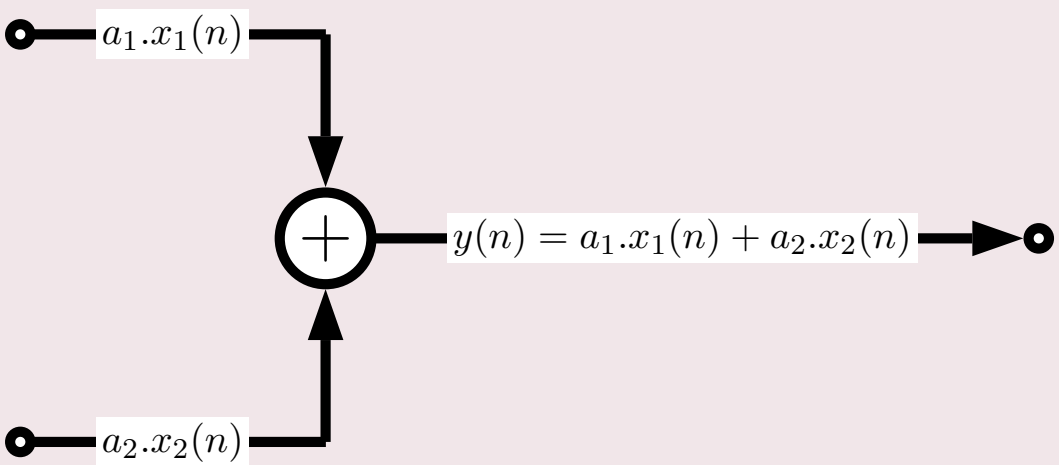
$y(n) = 0.5x(n)$

Signal Flow Graphs: Addition

Addition

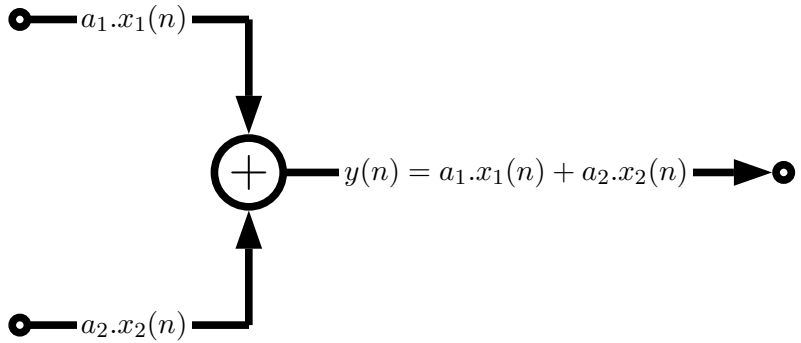
- We represent a addition of two input signal by **a circle with a + label** .
- We describe the algorithm via the equation:

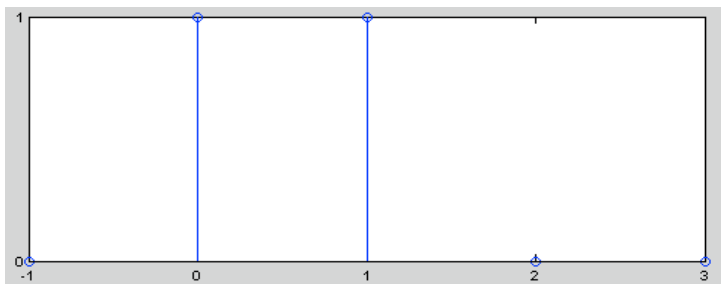
$$\mathbf{y(n) = a_1.x_1(n) + a_2.x_2(n)}$$



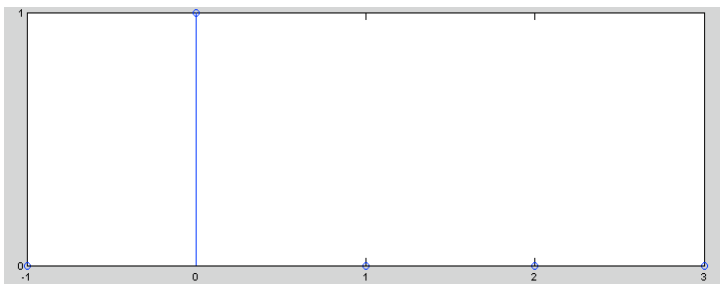
Signal Flow Graphs: Addition Example

In the example, set $a_1 = a_2 = 1$:

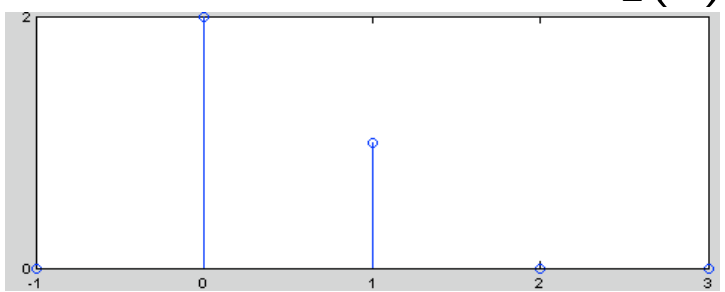




$x_1(n)$



$x_2(n)$



$$y(n) = x_1(n) + x_2(n)$$

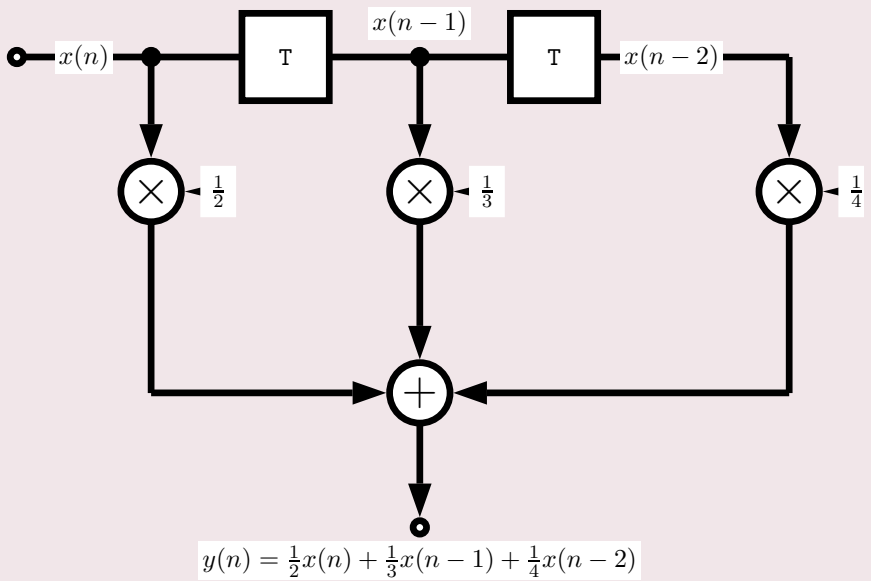
Signal Flow Graphs: Complete Example

All Three Processes Together

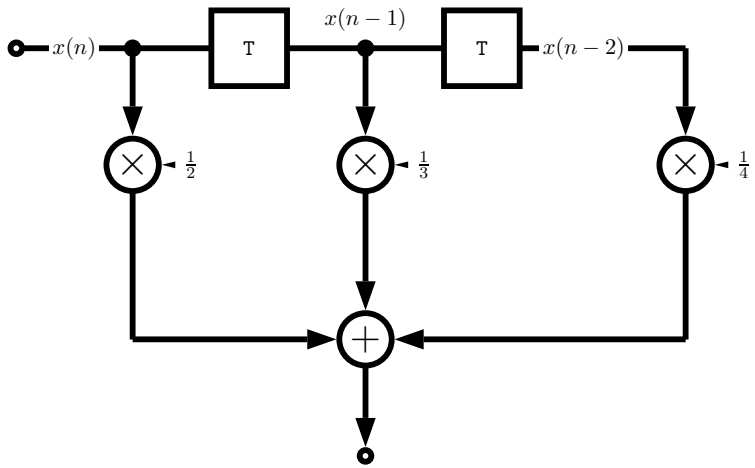
We can combine all above algorithms to build up more complex algorithms:

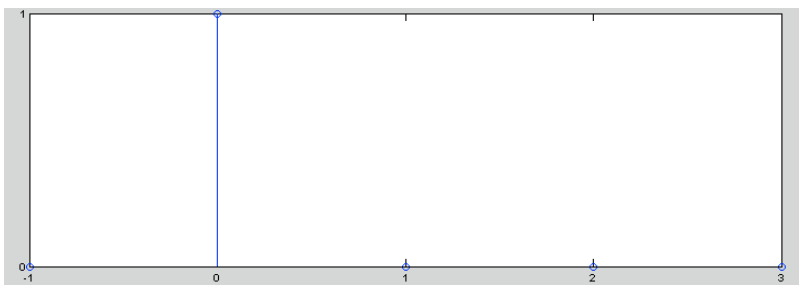
$$\mathbf{y(n) = \frac{1}{2}x(n) + \frac{1}{3}x(n-1) + \frac{1}{4}x(n-2)}$$

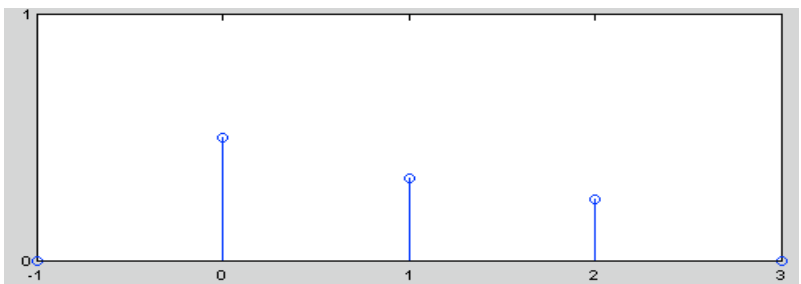
- This has the following signal flow graph:



Signal Flow Graphs: Complete Example Impulse Response







$x(n)$

$$y(n) = \frac{1}{2}x(n) + \frac{1}{3}x(n-1) + \frac{1}{4}x(n-2)$$

Filtering

Filtering

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

Filtering Examples:

- 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.
  - **JPEG Image** Compression
  - **MPEG Video** Compression
  - **MPEG Audio** Compression
- In **Digital Audio** we may wish to determine a range of frequencies we wish the enhance or diminish to equalise the signal, *e.g.*:
  - **Tone** — Treble and Bass — **Controls** (**Example coming soon**)
  - **Graphic Equaliser**

How can we filter a Digital Signal

Two Ways to Filter

- Temporal Domain — *E.g.* Sampled (PCM) Audio
- Frequency Domain — Analyse frequency components in signal. **Next Tutorial**

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

Temporal Domain Filters

We will look at:

IIR Systems : Infinite impulse response systems

FIR Systems : Finite impulse response systems

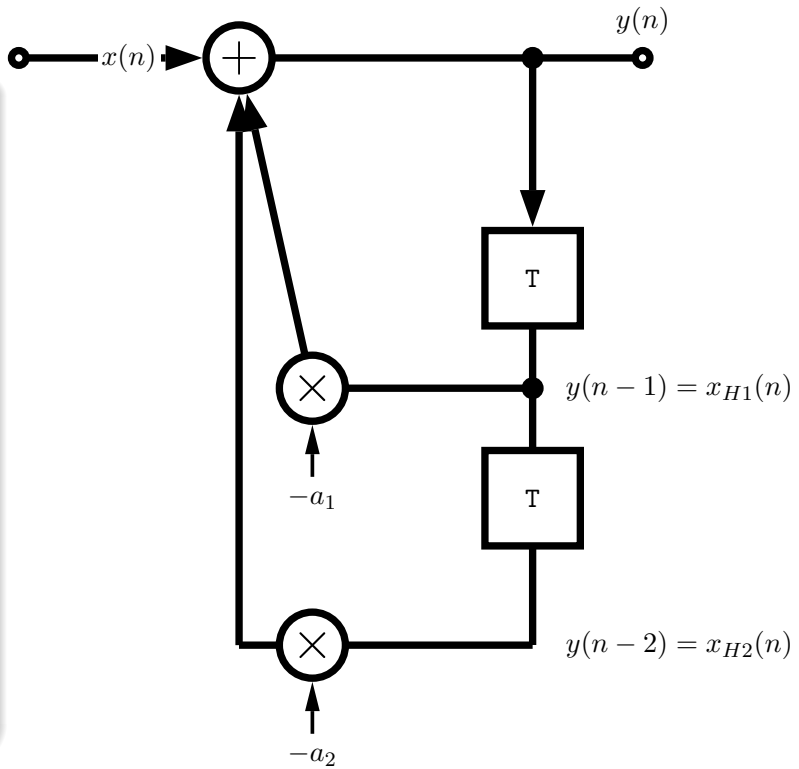
Infinite Impulse Response (IIR) Systems

Simple Example IIR Filter

- 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**

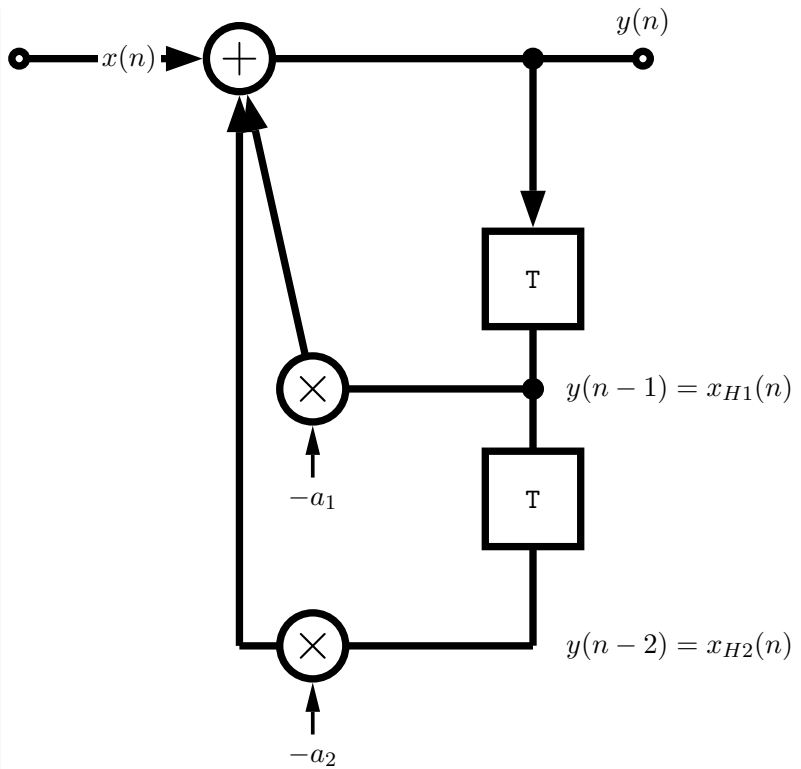


Infinite Impulse Response (IIR)Systems Explained

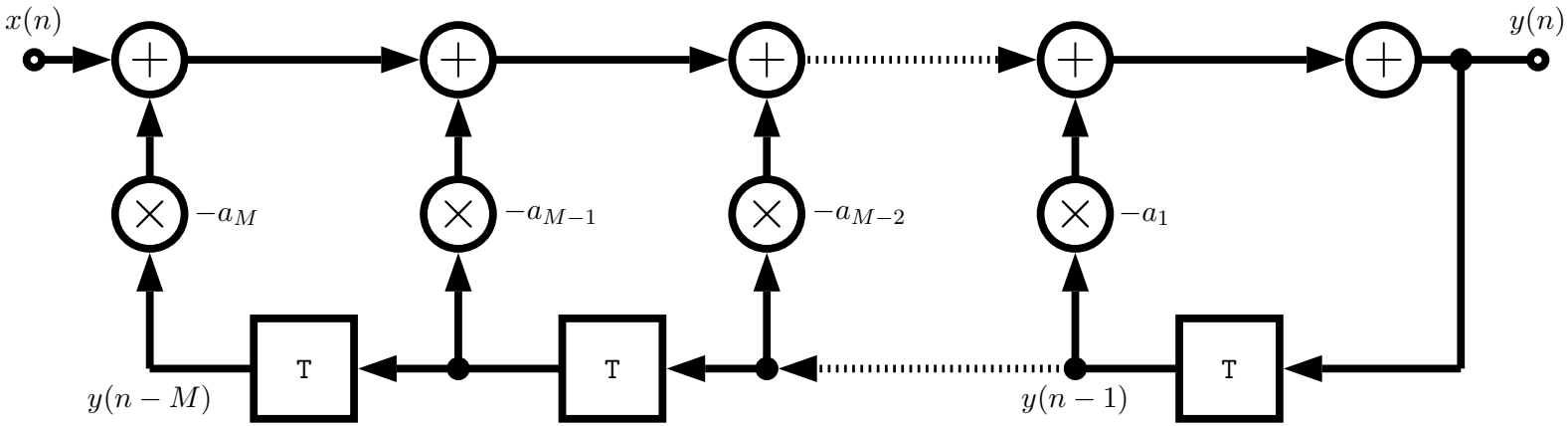
IIR Filter Explained

The following happens:

- The **output signal** $y(n)$ is **fed back** through a **series** of **delays**
- Each **delay** is **weighted**
- Each fed back **weighted delay** is **summed** and passed to **new output**.
- Such a **feedback** system is called a **recursive system**



A Complete IIR System



Complete IIR Algorithm

Here we extend:

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

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

We can represent the IIR system algorithm by the difference equation:

$$y(n) = x(n) - \sum_{k=1}^{M} a_k \, y(n-k)$$

Finite Impulse Response (FIR) Systems

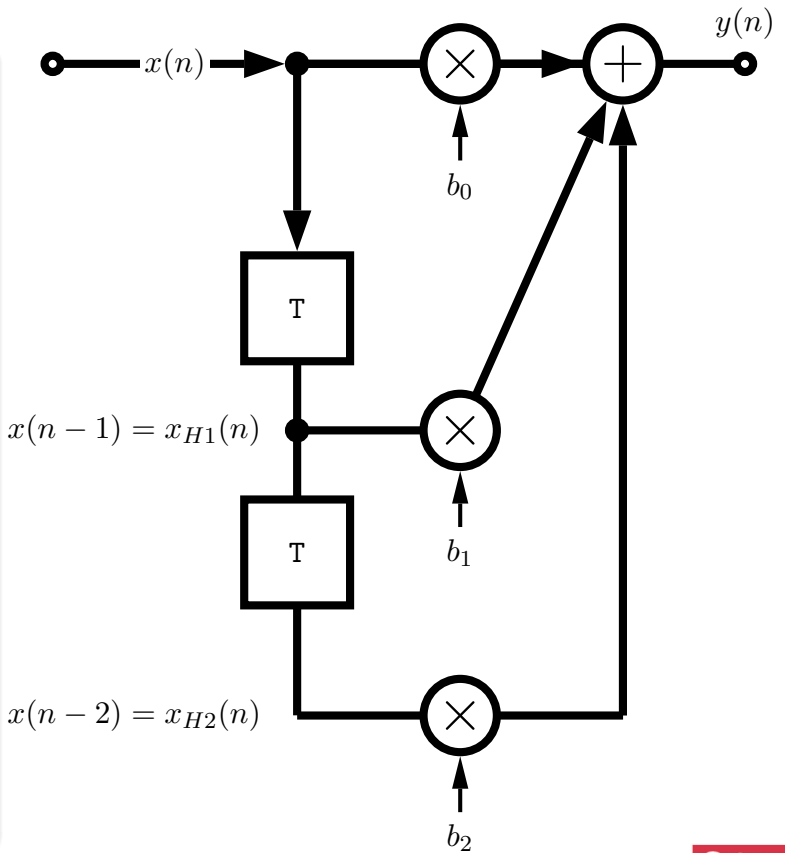
FIR system's are slightly simpler — there is **no feedback loop**.

Simple Example FIR Filter

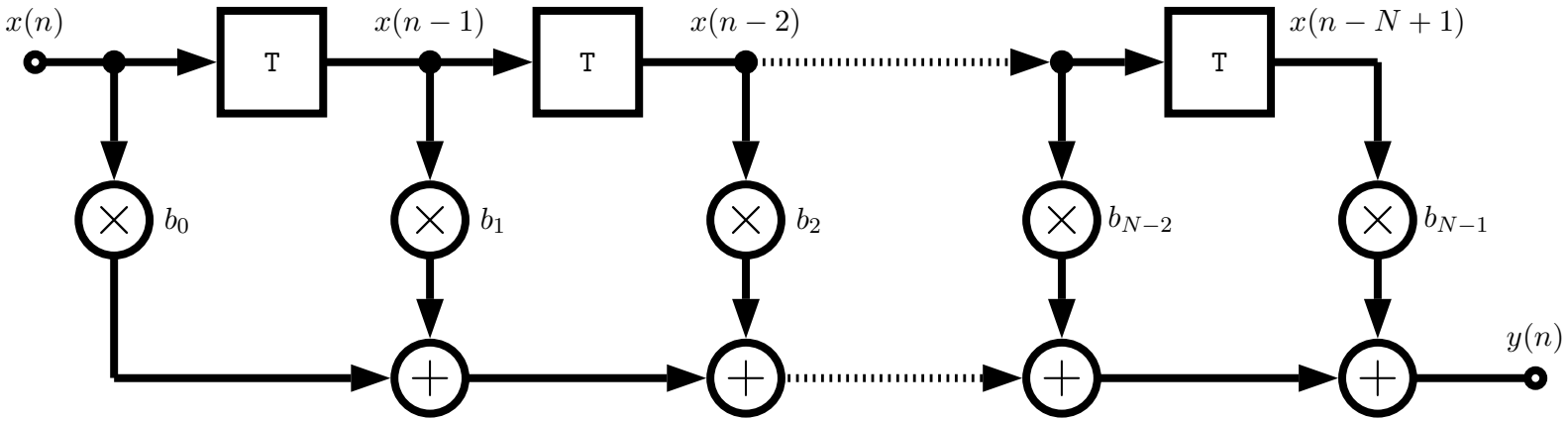
A simple FIR system can be described as follows:

$$y(n) = b_0x(n) + b_1x(n-1) + b_2x(n-2)$$

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



A Complete FIR System



FIR Algorithm

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

We can describe this with the algorithm:

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

Filtering with IIR/FIR

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:

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

- If a(1) is **not equal** to **1**, filter **normalizes** the filter coefficients by a(1). If **a(1) equals 0**, `filter()` **returns** an **error**

Creating Filters

How do I create Filter banks `A` and `B`

- Filter banks can be created manually — Hand Created: **See next slide** and **Equalisation** example later in slides
- MATLAB can provide some predefined filters — **a few slides on, see lab classes**
  - Many standard filters provided by MATLAB
- See also `help filter`, online MATLAB `docs` and lab classes.

Filtering with IIR/FIR: Simple Example

The MATLAB file IIRdemo.m sets up the filter banks as follows:

IIRdemo.m

```
fg=4000;
fa=48000;
k=tan(pi*fg/fa);

b(1)=1/(1+sqrt(2)*k+k^2);
b(2)=-2/(1+sqrt(2)*k+k^2);
b(3)=1/(1+sqrt(2)*k+k^2);
a(1)=1;
a(2)=2*(k^2-1)/(1+sqrt(2)*k+k^2);
a(3)=(1-sqrt(2)*k+k^2)/(1+sqrt(2)*k+k^2);
```

Apply this filter

How to apply the (previous) difference equation:

- By hand

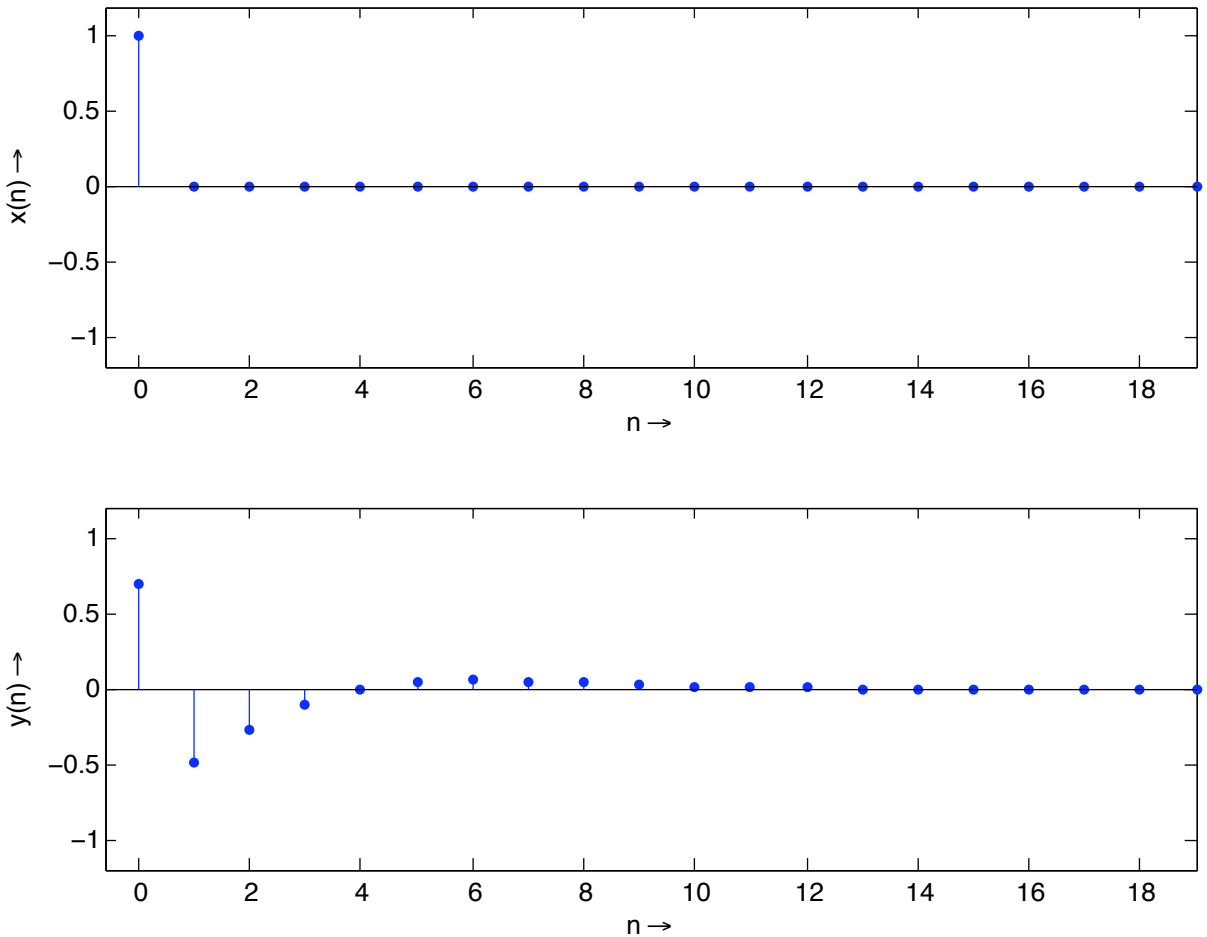
IIRdemo.m Cont.

```
for n=1:N
y(n)=b(1)*x(n) + b(2)*xh1 + b(3)*xh2 ...
            - a(2)*yh1 - a(3)*yh2;
xh2=xh1;xh1=x(n);
yh2=yh1;yh1=y(n);
end;
```

- Use MATLAB `filter()` function — **see next but one slide**
  - Far more **preferable**: general — **any length filter**

Filtering with IIR: Simple Example Output

This produces the following output:



MATLAB filters

Matlab `filter()` function implements an IIR (or an FIR no A components).

Type `help filter`:

```
FILTER One-dimensional digital filter.
    Y = FILTER(B,A,X) filters the data in vector X with the
    filter described by vectors A and B to create the filtered
    data Y.  The filter is a "Direct Form II Transposed"
    implementation of the standard difference equation:

    a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
                          - a(2)*y(n-1) - ... - a(na+1)*y(n-na)

    If a(1) is not equal to 1, FILTER normalizes the filter
    coefficients by a(1).

    FILTER always operates along the first non-singleton dimension,
    namely dimension 1 for column vectors and non-trivial matrices,
    and dimension 2 for row vectors.
```

Using MATLAB to make filters for `filter()` (1)

MATLAB provides a few built-in functions to create ready made filter parameter A and B:

Some common MATLAB Filter Bank Creation Functions

E.g: `butter`, `buttord`, `besself`, `cheby1`, `cheby2`, `ellip`.

See `help` or `doc` appropriate function.

Two Examples of Filtering

Application of Filtering

There are numerous examples of Filtering in DSP:

- Noise Removal
- Signal Analysis
- Audio Synthesis
- Audio Effects
- Many more ....

Two Examples

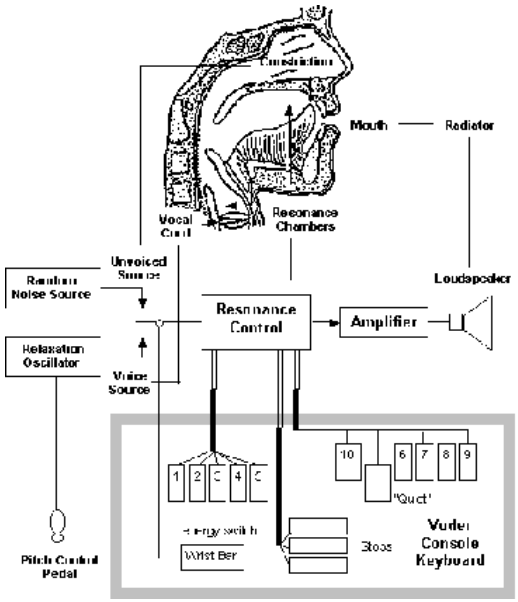
- **Subtractive Synthesis See Lecture Next Week**
- **Equalisation** — Tone control **See CM2202 Notes**

More Filtering examples throughout the Module

Subtractive Synthesis

Basic Idea: **Subtractive synthesis** is a method of **subtracting overtones** from a **sound** via by the application of a **filter**.

- First Example: Vocoder — talking robot (1939).
- Popularised with Moog Synthesisers 1960-1970s



Subtractive Synthesis: A Human Example

Human Filtering — cf. Wah-Wah Effect

We can model how humans make utterances as subtractive synthesis: (e.g. Vocoder)

Oscillator — the vocal cords act as the sound source and

Filter — the mouth and throat modify the sound.

- **A sweeping filter — vary (modulate) the filter frequency**

- Make a "ooh" and "aah" sound — same pitch
- By gradually changing from "ooh" to "aah" and back again – simulate the sweeping filter effect
- Effect widely used in electronic music/synthesis
- Basis of the **wah-wah guitar effect**, so named for obvious reasons.

Subtractive Synthesis: One More Human Example

Making Aeroplane, Wind and Ocean Wave Noises

Make a "ssh" sound — white noise

- Now "synthesise" a "jet plane landing" sound
- Vary mouth shape to filter the white noise into pink noise by removing the higher frequencies.
- The same technique (filtered white noise) can be used to electronically synthesise the sound of ocean waves and wind,
- Used in early drum machines to create snare drum and other percussion sounds.