NUI Galway
OÉ Gaillimh

EE445 – Introduction to DSP

Introduction to DSP

What is Digital Signal Processing (DSP)?

- Familiar with *signals* – how to describe them mathematically, how to generate them, and how to process them.
  - In Electrical Circuits and Systems, you will have studied simple signals like sinusoids, the unit impulse and the unit step, as well as more complex signals.
  - Laplace Transform, Fourier Transform.
- *Systems* – analysis and design (e.g. EE357 Signals and Communications)
  - Again, Laplace Transform and Fourier Transform

What is Digital Signal Processing (DSP)?

- System Design – specify system, determine transfer function, then implement (e.g. using op-amps, resistors etc.)
- “Analogue” signals, i.e. continuous-time and also continuous in amplitude
- Digital Signal Processing (DSP) uses DIGITAL techniques to do essentially the same things – signals are manipulated by computer, implemented using software on a PC, or perhaps using digital logic implemented on an FPGA.
- The concepts that apply to analogue signal processing still apply here!

Benefits of using DSP

- **Reproducibility.** No problems with component tolerances.
- **Consistency.** No problems with component drift.
- **Flexibility.** Easy to change function.
- **Cost.** Takes advantage of developments in VLSI.
- **Superior performance.** Some things you just can't do without DSP (at least not easily …)
- **BUT** – still some areas where Analogue techniques are better (e.g. RF)

A Signal Processing System

- Typical DSP system

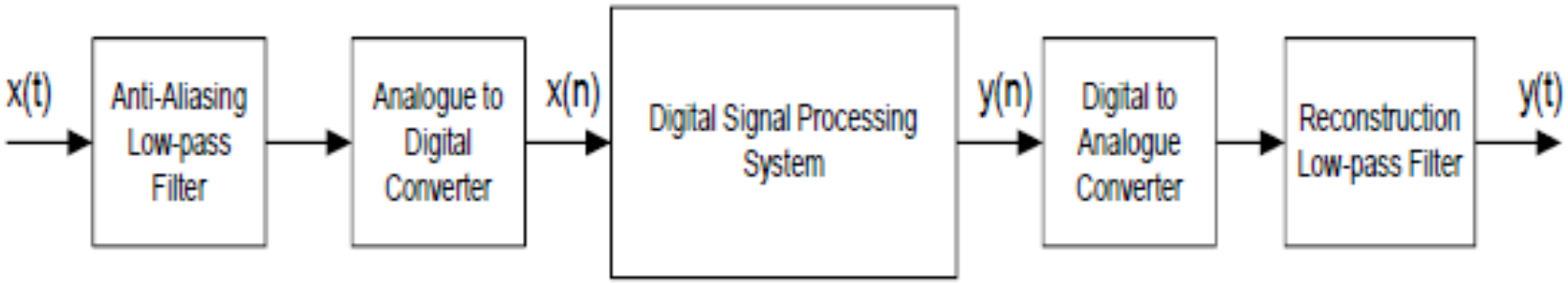


Figure 1.1. Block diagram of a typical DSP system.

- What does each block do?

Discrete-Time Signals: Sampling

- Discrete-time ("digital") signals are only strictly defined at specific instants of time, and are equal to zero at all other times.
- Normally, the instants of time at which discrete-time signals are specified are equal to integer multiples of the sampling period (which is the reciprocal of the sampling frequency).
- Analogue signal is sampled every T seconds, where T is the sampling period (125 µsec in Figure 1.2), to generate a sequence of numbers x(n), i.e.

$$x(n) = x(t)|_{t=nT}$$

Discrete-Time Signals: Sampling

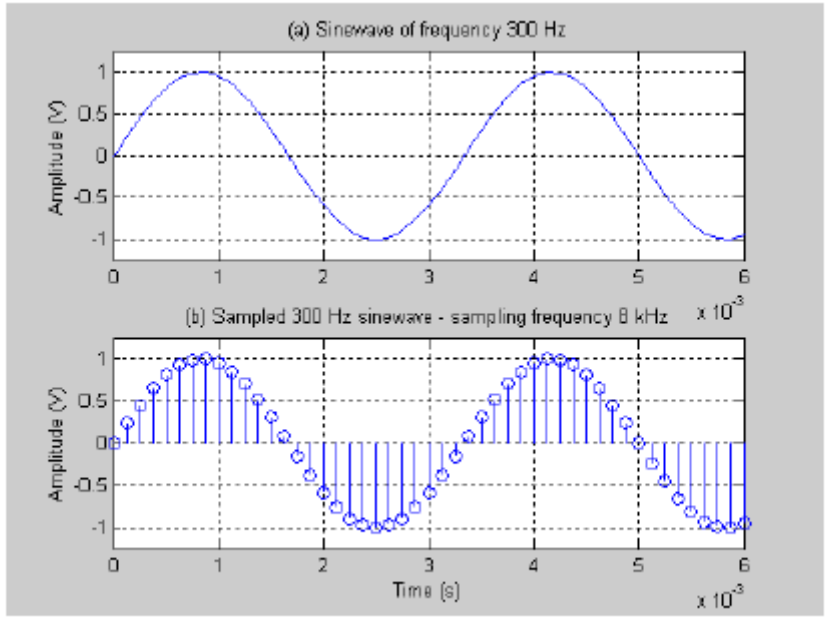


Figure 1.2. Example of a continuous-time signal, and its sampled version.

- Actually, there are two operations taking place:
  - Sampling
  - Quantisation (for the most part, we will not be overly concerned about quantisation)
- The terms “digital” and “discrete-time” will be used somewhat flexibly

Reconstruction

- How do we get the original signal back from the samples?

$$x(t) = \sum_{n=-\infty}^{\infty} x(n) sinc\left(\frac{t - nT}{T}\right)$$

- sinc(x) is the well-known sinc function, given by:
- $sinc(x) = \sin(\pi x) / (\pi x)$

Reconstruction

- What's actually happening here? "Analogue" signal is being reconstructed from a sum of weighted, delayed sinc functions.
- See Figure 1.3 for a sinc function (100 Hz sampling frequency, delay of 5 samples).
- Another example: reconstruction from a three-sample sequence {1, 3, 2}

Reconstruction

- Digital to Analogue Converter (DAC) produces “staircase” waveform, and Reconstruction Filter smooths this out (remember that “discontinuities” in the signal means high frequencies are present).

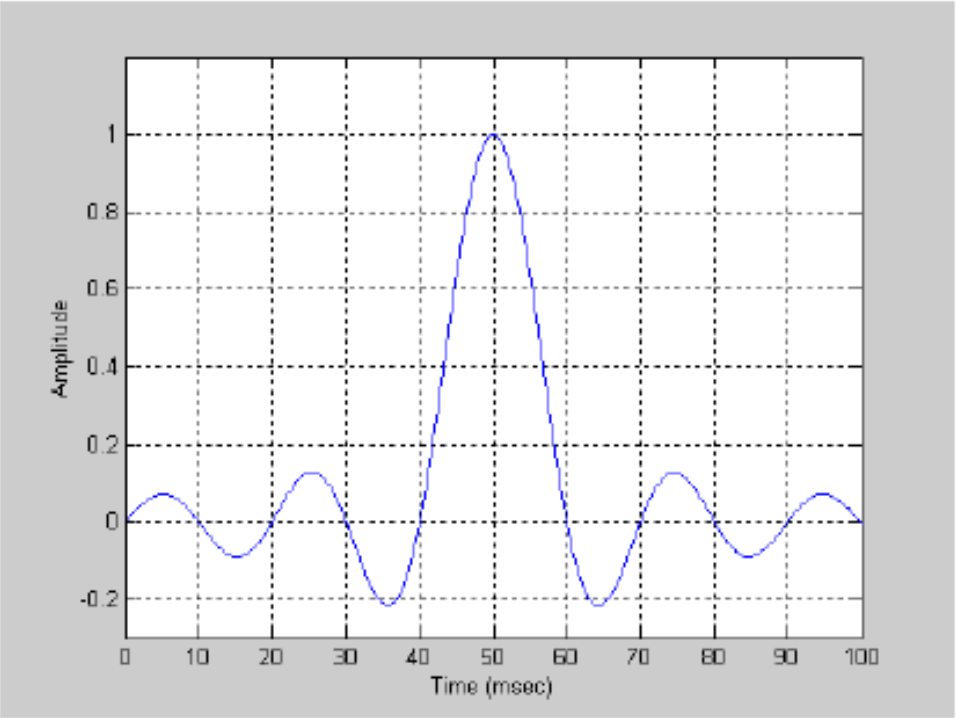


Figure 1.3. Plot of sinc function (sampling frequency 100 Hz)

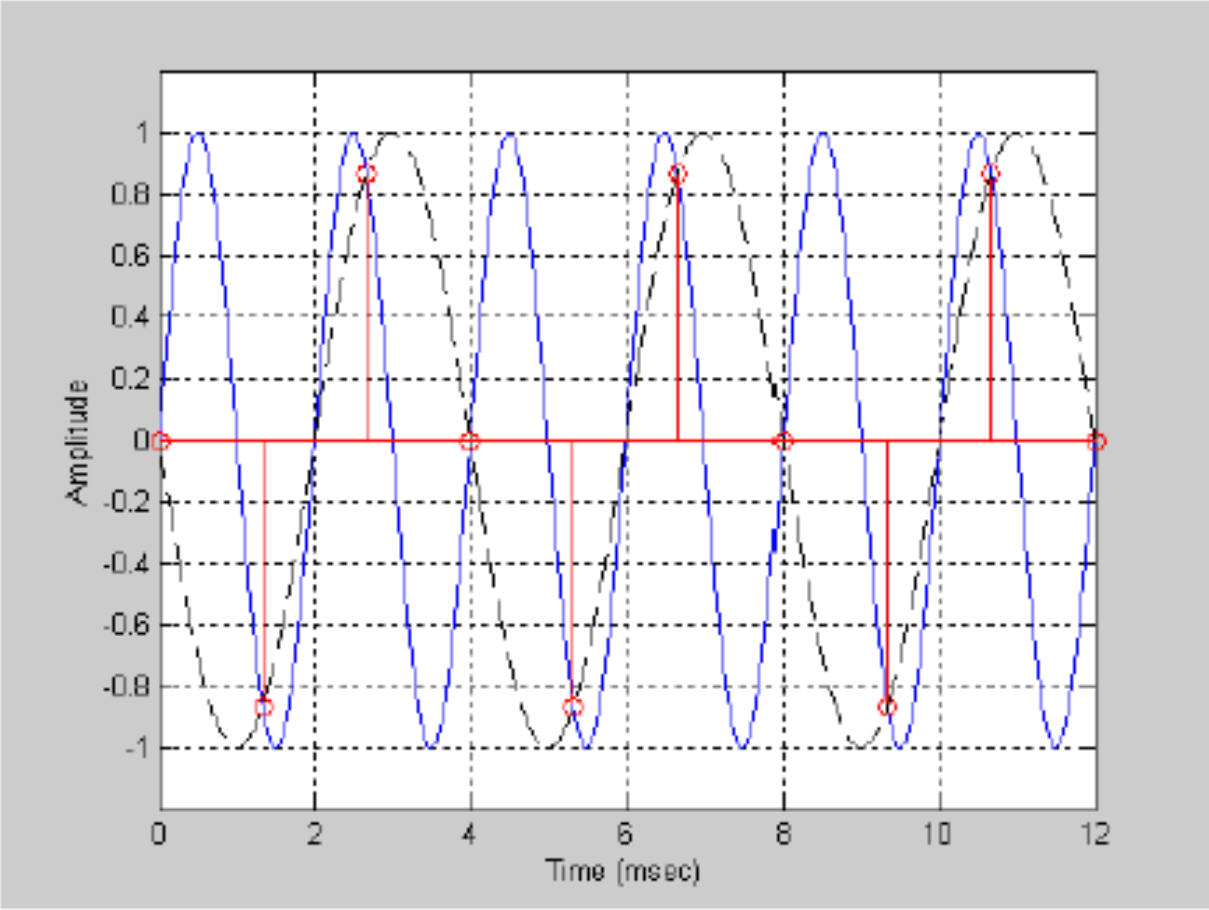
Choice of Sampling Frequency – Sampling Theorem

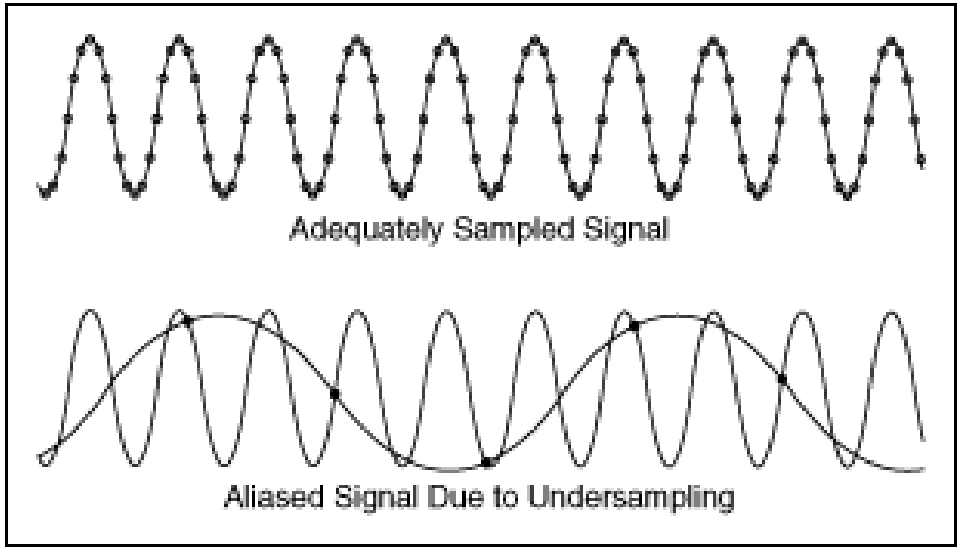
- Sampling Theorem:
  - *When sampling an analogue signal with a maximum frequency content of f_{max}, the sampling frequency f_{samp} must be greater than or equal to $2f_{max}$ in order to adequately preserve the information in the signal.*
- Or …
  - *If an analogue signal is sampled at a rate of f_{samp}, the analogue signal must not contain frequency components greater than $f_{samp}/2$, in order to adequately preserve the information in the signal.*

Sampling Theorem

- The maximum analogue frequency is often called the “Nyquist Frequency”, and the minimum acceptable sampling rate is often called the “Nyquist Rate”.
- What happens if we sample at too low a rate? Sample an analogue sinusoid of frequency $(f_{samp}/2)+A$ Hz, and you get the samples that you would get if analogue sinusoid actually had a frequency of $(f_{samp}/2)-A$ Hz.

Aliasing


Amplitude
Time (msec)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12


Adequately Sampled Signal
Aliased Signal Due to Undersampling

Examples of aliasing

Sampling Theorem

- Even if the sampling rate is high enough, we can still get some aliasing
- For example, broadband noise, or “unwanted” signal components
- Anti-Aliasing Low-pass Filter can never completely do the job … would require an ideal “brick wall” filter (unrealisable)
- So, the objective is to minimise the effect of aliasing (so that overall system performance is not affected).
- Notes
  - Sampling theorem as stated not the whole story
  - We generally want to keep the sampling rate as low as possible – but some important exceptions to this general rule

Important Digital Signals

- Unit Impulse

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

- The unit impulse also has a delayed version, defined by:

$$\delta(n-k) = \begin{cases} 1, & n = k \\ 0, & n \neq k \end{cases}$$

Unit Impulse

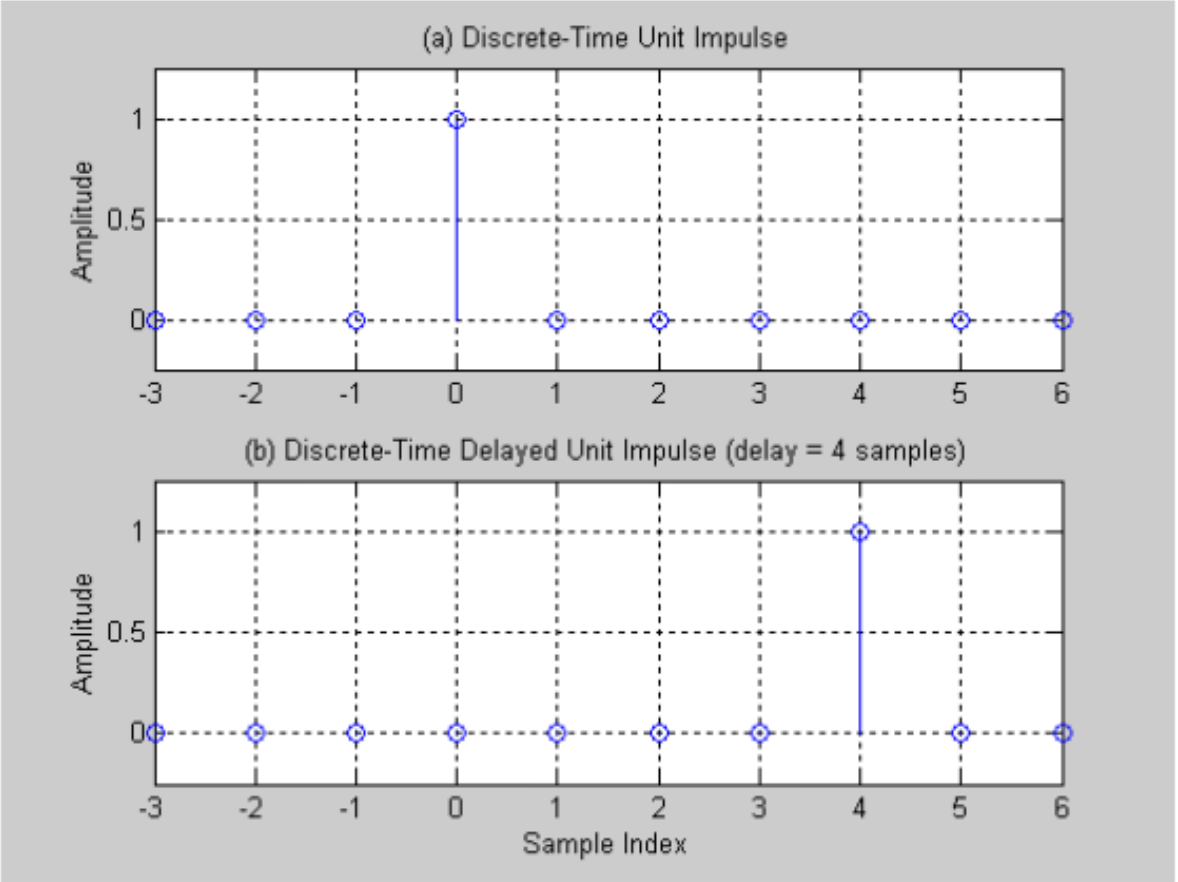


Figure 1.3. Plot of Unit Impulse and Delayed Unit Impulse.

Unit Step function

- Unit step function

$$u(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

- Delayed unit step is defined by:

$$u(n-k) = \begin{cases} 1, & n \geq k \\ 0, & n < k \end{cases}$$

- The unit step and delayed unit step (k=2) are plotted in the next slide…

Unit step

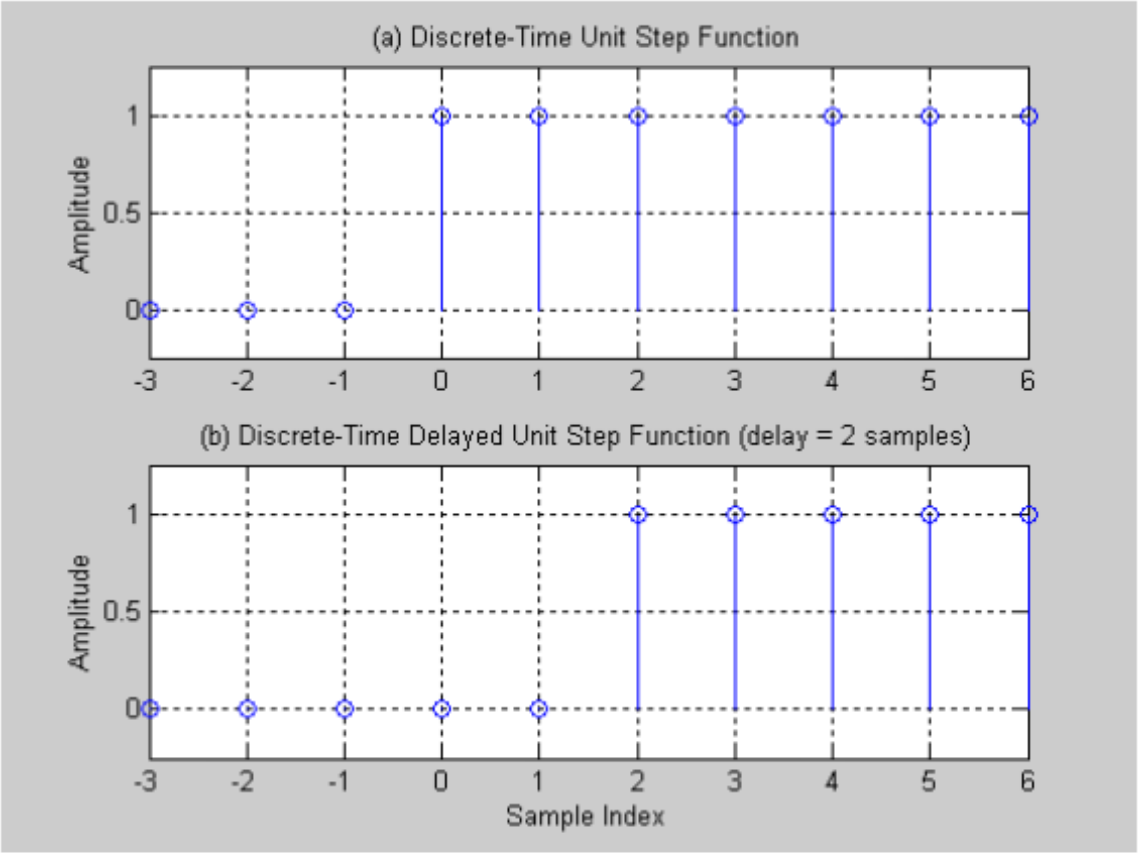


Figure 1.4. Discrete-time Unit Step and Delayed Unit Step.

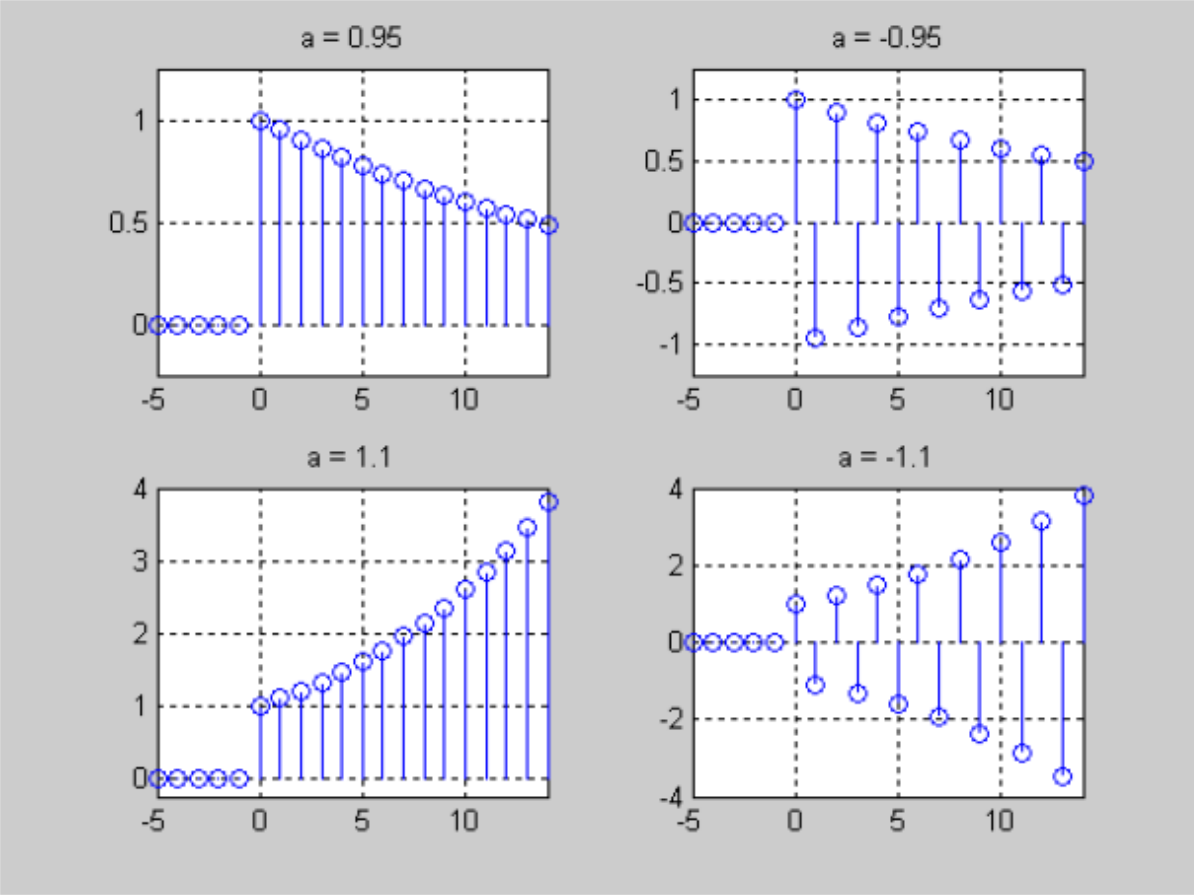
Sampled Exponential

- The sampled exponential is defined by:

$$x(n) = a^n u(n)$$

- Where *a* is some constant
- Note: multiplication by *u(n)* (the unit step function)
- Exact form of function depends on constant *a*

Sampled Exponential



Discrete-Time Sinusoidal Signal

- Derived from the underlying analogue sinusoidal signal:

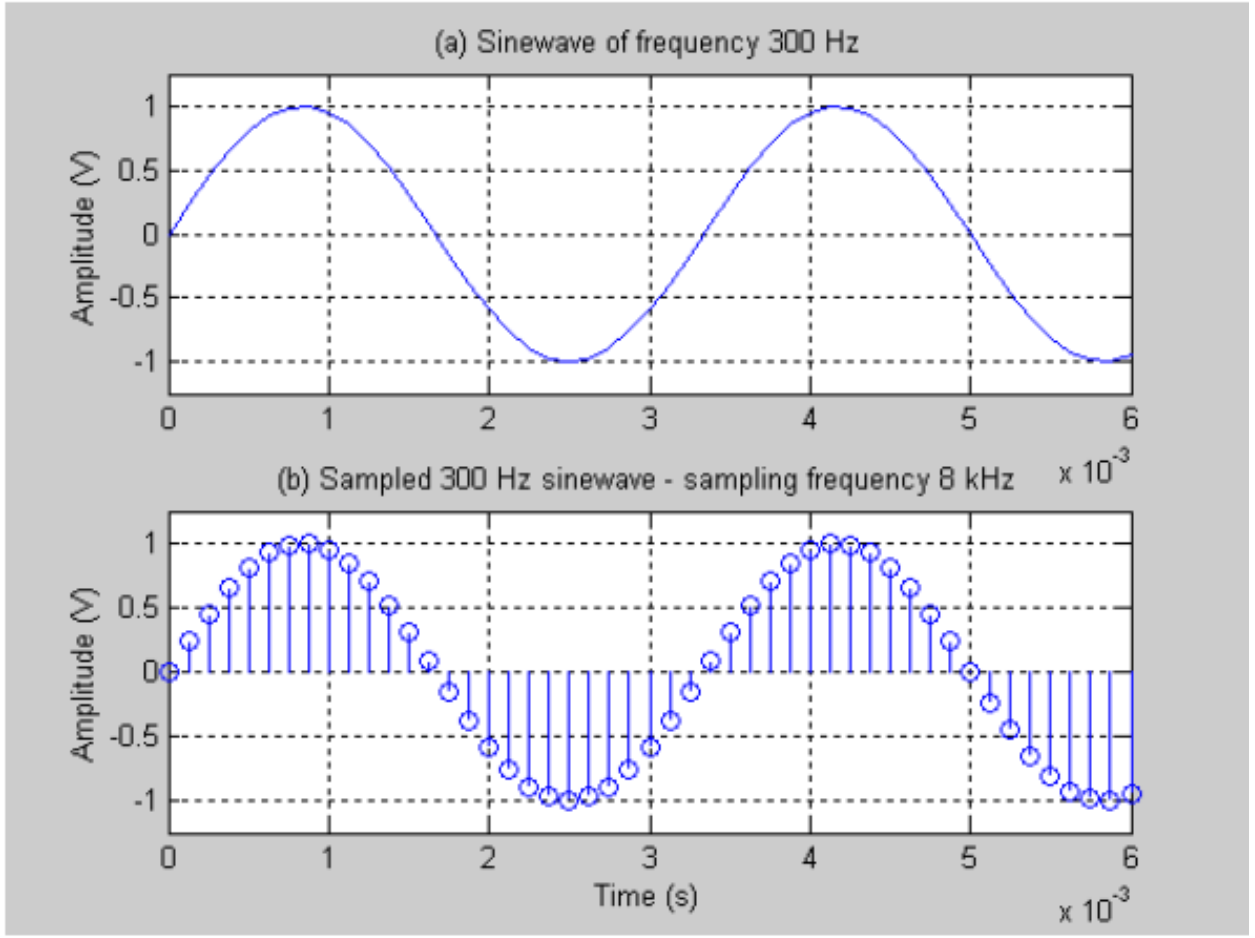
$$x(t) = Asin(\omega_a t)$$

- where A is the amplitude of the sinusoid, and ω_a is the frequency in radians per second.
- By sampling this signal every T seconds, we obtain:

$$x(t) = Asin(\omega_a nT) = Asin(2\pi f_a / f_s)$$

- where f_a is the frequency of the sinusoid (in Hz), f_s is the sampling frequency (in Hz), and n is the sample index as before (see Figure 1.2)

Discrete-Time Sinusoidal Signal



Discrete-Time Sinusoidal Signal

- $2\pi f_a/f_s$ is often referred to as the digital frequency or relative frequency, with symbol θ (other symbols may also be used in textbooks etc.). Therefore, the sampled sinusoid can be re-written as:

$$x(n) = Asin(n\theta)$$

- The units of θ are radians.
- As the underlying analogue frequency fa increases from 0 Hz (DC) up to the sampling frequency fs, the value of θ increases from 0 to 2π
- A digital sequence is said to be periodic, with period N, if N is the smallest integer for which:

$$x(n+N) = x(n)$$

Discrete-Time Sinusoidal Signal

- For example, the sampled sinusoid is periodic with period N samples, if
$$Asin([n+N]\theta) = Asin(n\theta)$$
- Can only be satisfied if $N\theta$ is an integer multiple of 2π, i.e. $N\theta = 2\pi k$, where k is an integer.
- The ratio of the sampling frequency to the analogue frequency must be an integer.
$$N = \frac{2\pi k}{\theta} = \frac{f_s}{f_a}k$$
- In the case of a sinusoid, there will be an exact integer number of digital samples in every period of the underlying analogue waveform.

Matlab examples

- **NB: Be careful when talking about "periodicity"!!**
- Matlab Example 1.1
  - Write Matlab code to generate 100 samples of a discrete-time sinewave of amplitude 3 volts, with sampling frequency of 8 kHz and underlying "analogue" frequency 1 kHz.
  - Exercise 1: Repeat this exercise for an analogue frequency of 1100 Hz. Examine the sequence of samples, and determine the period of the *sampled sequence*.
  - Exercise 2: In the above example, the first sample index is 1, which implies that the sample sequence starts at time 125 µsec. Modify the code (both methods) to generate 100 samples starting at time 0.

Matlab examples

- An Amplitude Modulated (AM) waveform is given by the following equation:

$$x(t) = A[1 + mcos(\omega_m t)]\cos(\omega_c t)$$

Where ω_c is the carrier frequency (in radians/s), ω_m is the modulating frequency, *A* is the carrier amplitude, and *m* is the modulation index.

- The digital equivalent is:

$$x(t) = A[1 + mcos(\omega_m t)]\cos(\omega_c t) = \mathrm{A}[1 + mcos(n\theta_m)]\cos(n\theta_c)$$

where T is the sampling period

- Write Matlab code to generate the digital AM waveform, given the carrier, modulating and sampling frequencies in Hz.

An alternative view of Digital Signals

- Recall sifting property of the Dirac delta function:

$$\int_{-\infty}^{\infty} x(t)\delta(t-\tau)dt = x(\tau)$$

- Discrete-time notation version is:

$$x(n)\delta(n-k) = x(k)\delta(t-k)$$

- We can represent any arbitrary sequence by means of an infinite sum of delayed unit impulses, each one weighted by the corresponding sample value of the arbitrary sequence. For example, Figure 1.6 …

Arbitrary finite-duration sample sequence

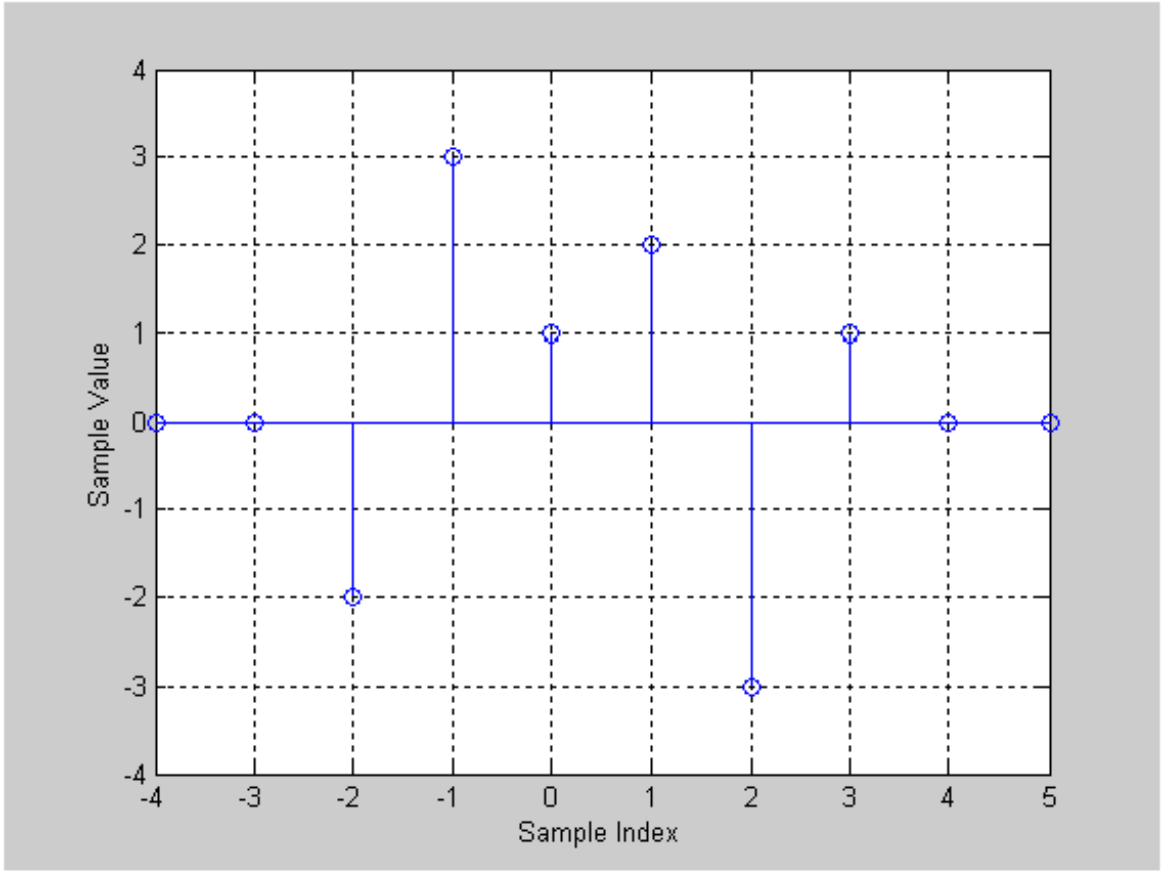


Figure 1.6. Arbitrary finite-duration sample sequence.

An alternative view of Digital Signals

$$x(n) = -2\delta(n+2) + 3\delta(n+1) + \delta(n) + 2\delta(n-1) - 3\delta(n-2) + \delta(n-3)$$

- More generally

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k)$$

More later…

Linear Time-Invariant (LTI) Discrete-Time Systems

- Introduction
- Linearity:
  - If the response of the system to an input $x_1(n)$ is $y_1(n)$, and its response to $x_2(n)$ is $y_2(n)$, then its response to $a_1x_1(n) + a_2x_2(n)$ will be $a_1y_1(n) + a_2y_2(n)$
- Time Invariance:
  - If the response of a system to an input $x(n)$ is $y(n)$, then its response to $x(n - N)$ will by $y(n - N)$

Linear Time-Invariant (LTI) Discrete-Time Systems

- “Digital filters”. Such filters can be constructed from three fundamental mathematical operations:
  - Addition (or subtraction)
  - Multiplication (usually of a signal by a constant)
  - Time delay, i.e. delaying a digital signal by one or more sample periods.
- Specification of Digital Filters:
  - Block diagram
  - Difference equation
  - Impulse response
  - Frequency response
  - Z-transform (more later)
  - Pole-zero diagram

Block Diagram

- Very commonly used graphical means of describing a digital filter
- For example, Figures 1.8(a) and (b):

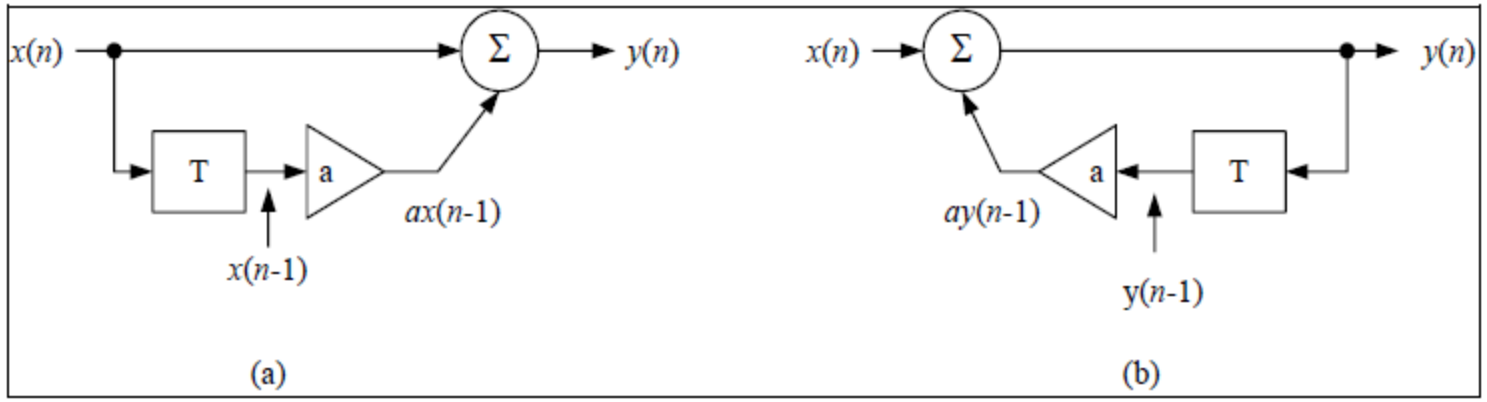


Figure 1.8. Block diagrams of simple digital filters.

Difference Equation

- A difference equation is a simple mathematical statement of how the output, *y(n)*, of a digital filter is calculated. It is based on some or all of the following quantities:
  - The current input, x(n)
  - Previous input samples *x(n-1)*, *x(n-2)* etc.
  - Previous output samples *y(n-1)*, *y(n-2)* etc.
  - Various constants that are used to multiply these samples – these constants in effect determine the behaviour of the filter, and are usually referred to as digital filter coefficients.
- Readily obtained from the block diagram, e.g. Figure 1.8(a):

$$y(n) = x(n) + ax(n-1)$$

Difference Equation

- Figure 1.8(b):

$$y(n) = x(n) + ay(n-1)$$

- A simple classification of digital filters is into two categories:
  - Non-recursive, where the output depends only on the current and previous inputs
  - Recursive, where the output depends not only on the current and (possibly) previous inputs, but also on previous outputs, i.e. there is feedback or recursion from the output to the input.

Difference equation

n	$x(n)$	$x(n\text{-}1)$	$ax(n\text{-}1)$	$y(n)$
0	1	0	0	1
1	0.5	1	0.9	1.4
2	-1.3	0.5	0.45	-0.85
3	0.6	-1.3	-1.17	-0.57
4	1.1	0.6	0.54	1.64
		etc.		

Table 1.1. Evaluation of difference equation of filter in Figure 1.8(a), for a=0.9

Difference equation

n	$x(n)$	$x(n\text{-}1)$	$ax(n\text{-}1)$	$y(n)$
0	1	0	0	1
1	0.5	1	0.9	1.4
2	-1.3	0.5	0.45	-0.85
3	0.6	-1.3	-1.17	-0.57
4	1.1	0.6	0.54	1.64
		etc.		

Table 1.1. Evaluation of difference equation of filter in Figure 1.8(a), for a=0.9

- Causality
  - Primarily interested in signals that are non-zero only for n>=0

Difference Equation

- Filter output can only depends on current and previous values of the input, and previous values of the output, and not on future samples
- However … what about where a block of signal samples have been recorded and stored, for later processing?
- We assume that x(n-1), y(n-1) etc. are zero for n=0 (not always the case, though …)

Exercise

- Draw the block diagrams, and calculate the first 5 output samples, for the digital filters represented by the following difference equations:

  (a) $y(n) = x(n) + 0.5x(n - 1) - 0.2x(n - 2)$
  (b) $y(n) = 0.5x(n) - 0.6x(n - 1) + 0.1x(n - 3)$
  (c) $y(n) = x(n) + 0.6x(n - 1) - 0.1x(n - 2) + 0.3y(n - 1) - 0.3y(n - 2)$

- In all cases, the input signal is the unit step function.

Impulse Response

- Impulse response is the response of the filter to an input consisting of the unit impulse function, $\delta(n)$
- It plays a role in discrete-time systems that is exactly the same as its role in continuous-time systems.
- If the impulse response of a system is known, it is possible to calculate the system response to any input sequence, $x(n)$
- We have already seen that any arbitrary sequence can be represented by a sum of weighted, delayed unit impulses, i.e.

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k)$$

Impulse Response

- Response of an LTI system to a delayed unit impulse $\delta(n-k)$ will be a delayed version of the impulse response, i.e. $\mathrm{h}(n-k)$.
- From the linearity property, we know that the response of the system to a weighted sum of inputs will be a weighted sum of the responses of the system to each of the individual inputs.
- Therefore:

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

Or in the case of a causal input (normally the case)

$$y(n) = \sum_{k=0}^{\infty} x(k)h(n-k)$$

Convolution Sum

- This equation called the convolution sum of two sequences x(n) and h(n):

$$y(n) = x(n) * h(n) = \sum_{k=0}^{\infty} x(k)h(n-k)$$

- Causal impulse response : h(n) is zero for n<0.

Impulse response

- The process of convolving two sequences, x(n) and h(n), can be described by the following steps:

  1. Re-write x(n) and h(n) as x(k) and h(k); all we are doing here is changing the variable used for the sample index.

  2. “Fold” over (or reverse) h(k) in time, to get h(-k).

  3. Delay h(-k) by n samples to get h(n-k).

  4. Multiply x(k) by h(n-k), to obtain x(k)h(n-k).

  5. Sum this product over all k to obtain $y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$; in practice, x(k) and h(k) will often be of finite duration, thus limiting the range of summation

  6. Repeat for all values of n. Again, because the two sequences are of finite duration, this will limit the range of values of n for which y(n) has to be evaluated.

- For two finite-duration sequences of length N_1 and N_2 samples, the sequence resulting from their convolution will have $N_1 + N_{2-1}$ non-zero samples.

Exercises

- Exercise:
  - Calculate the convolution of the following two causal sequences, where each sequence starts at n=0):

    $x(n) = \{1, 1, 1, 1, 1\}$, $h(n) = \{2, 1\}$

- Matlab example
  - Calculate the convolution of the following two causal sequences, where each sequence starts at n=0):

    $x(n) = \{0.5, 0.5, 0.5\}$, $h(n) = \{3, 2, 1\}$
- Calculate the impulse responses of each of the filters described by the difference equations in Exercise above.

Classification of Digital Filters

- Alternative classification of digital filters:
  - Finite Impulse Response (FIR). As the name suggests, these are digital filters for which the impulse response is of finite duration, i.e. h(n) has a finite number of non-zero samples.
  - Infinite Impulse Response (IIR). These are filters for which the impulse response is of infinite duration.
- Generally-speaking non-recursive filters are FIR in nature, while recursive filters are IIR
- Exercise
  - Show that the samples of the impulse response of an FIR filter are the same as the digital filter coefficients.

Stability of Digital Filters

- Define a *bounded* signal

$$|x(n)| < M < \infty$$

- A digital filter is Bounded Input Bounded Output (BIBO) stable if a bounded input sequence x(n) produces an output sequence y(n) that is also bounded, i.e.

$$|y(n)| < K < \infty$$

- Impulse response must be absolutely summable, i.e.

$$\sum_{k=0}^{\infty} |h(k)| < \infty$$

Exercise

- Draw the following impulse responses, and state if the digital filters to which they correspond are (i) causal or non-causal, (ii) stable or unstable.

$$h_1(n) = \left(\frac{2}{3}\right)^n u(n)$$

$$h_2(n) = \left(\frac{3}{2}\right)^n u(n)$$

$$h_3(n) = \left(\frac{2}{3}\right)^{-n} u(-n)$$

$$h_4(n) = \left(\frac{3}{2}\right)^{-n} u(-n)$$