Lecture 2

Discrete systems

One-dimensional discrete signals

As opposed to an analog signal $x(t)$, a *discrete signal* (or discrete-time signal) x_n is a sequence of discrete values. Such signals are often produced by sampling a continuous signal at the instants of time $t_n = nT_s$ where T_s is the *sampling period* in seconds. The resulting sequence is

$$x_n = x(t_n) = x(nT_s)$$

The amplitude *x* can be continuous or discrete. In applications the amplitude is almost always discrete (*quantized* to a finite number of distinct values) since this is the only type of signal that can be physically represented in a digital system. We call a discrete-time, discrete-amplitude signal a *digital signal* and reserve the term "discrete signal" for a discrete-time, continuous-amplitude signal. By default we will assume any signal x_n is of this type, unless otherwise stated. If a signal is digital but has enough amplitude levels (bits of resolution), the effects of discretization can be negligible, and we can safely treat it as a continuous-amplitude discrete signal which usually simplifies analysis.

Written representation

Formally, a discrete signal is a function that associates a value x_n with each integer index $-\infty < n < \infty$. We call *n* the *time index*, or simply the *index*. We could describe a signal using a formula such as

$$x_n = \begin{cases} \dfrac{1}{2^n} & n \geq 0 \\ 0 & n < 0 \end{cases} \tag{1}$$

Or, we could list its values

$$x_n = \left[\ldots, 0, 0, \mathring{1}, \frac{1}{2^1}, \frac{1}{2^2}, \frac{1}{2^3}, \ldots \right] \tag{2}$$

where the ellipsis (…) indicates that the obvious pattern continues without end. Since the index isn't explicitly listed in this representation, we need a way to allow the reader to determine it. Our convention is to put a small circle over the $n = 0$ value so that the corresponding indices are apparent.

The listing-of-values approach is particularly practical when x_n has a finite number of non-zero values. For example

$$x_n = [1, \mathring{3}, -2, 5, 6, 7, 9]$$

tells us that

$$x_{-1} = 1, x_0 = 3, x_2 = -2, x_3 = 5, x_4 = 6, x_5 = 7, x_6 = 9$$

For all other values of *n* we assume $x_n = 0$. Alternately we could show the x_n values on a plot (Fig. 1).

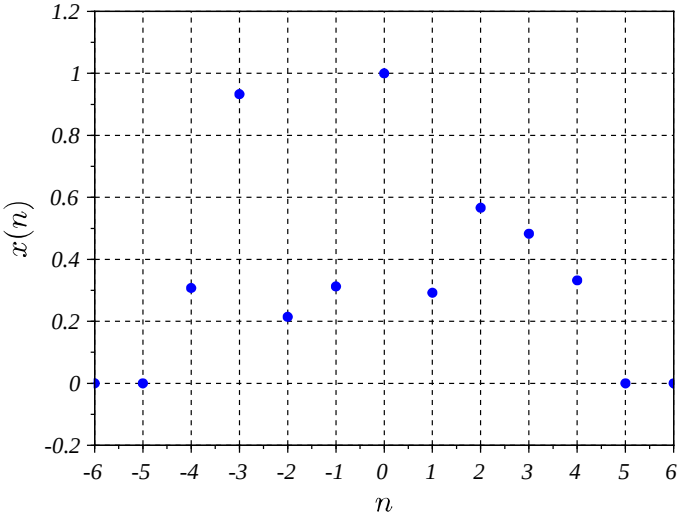

Fig. 1: Representing a discrete signal with a graph.

We will often use the functional notation

$$x(n) \equiv x_n$$

interchangeably with the array index notation. This is especially useful since this is how Scilab represents discrete signals (arrays). Although we will not, some authors use square brackets instead of parenthesis, $x[n]$ vs. $x(n)$, to distinguish a discrete signal from an analog signal. The notation $x[n]$ corresponds to how arrays are represented in languages such as C and Java.

However represented, keep in mind that *n* is a discrete variable that takes on only integer values. It cannot be treated like a continuous variable. For a discrete signal $x(1.37)$ is meaningless, as is an expression such as $\frac{d}{dn}x(n)$.

Computer representation

In Scilab a discrete signal might be an array we create, such as in the following example.

Example 1: Generate the signal $x(n) = 2\cos(2\pi(0.05)n)$, $-2 \le n \le 2$ using Scilab.

```
--> n = -2:2
 n  =
  -2.  -1.   0.   1.   2.

--> x = cos(2*%pi*0.05*n)
 x  =
   0.809017   0.9510565   1.   0.9510565   0.809017
```

Try this for yourself. Note that we have created two arrays, one of the indices *n* and a second of the values x_n.

Exercise 1: Generate the signal $x(n)=2\sin(2\pi(0.125)n)$, $3\le n\le 8$ using Scilab.

Very often in this class a discrete signal will be an array we read from a sound file. For example

```
--> [x,Fs,Nbits] = wavread('jfk.wav');

--> disp(length(x));

   62379.
```

reads the 62,379 sound samples in the file `jfk.wav` into the array `x`. A discrete signal can also be defined using a function. For example

```
function v = x(n)
  if (n>=0)
    v = cos(2*%pi*0.1*n);
  else
    v = 0;
  end
endfunction
```

Two fundamental discrete signals

Here we introduce the discrete *delta* and *step* functions which are building blocks for many more complicated signals.

Delta function

The *delta function* is shown in Fig. 2 and defined as

$$\delta(n)=\begin{cases}1 & n=0\\ 0 & n\neq 0\end{cases} \tag{3}$$

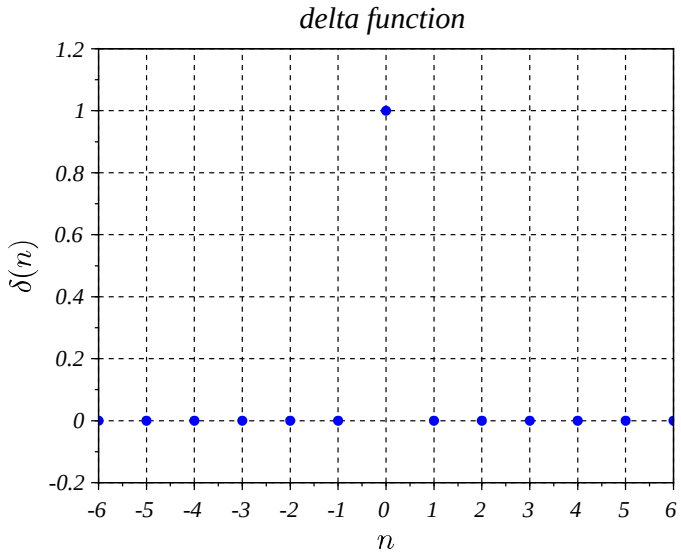

Fig. 2: Plot of delta function (impulse function).

This function is zero everywhere except for a *unit impulse* at time $n=0$. It is sometimes called the *impulse function*. This will play a central role in our analysis of discrete systems. In particular we will see that linear time-invariant discrete systems can be fully characterized by applying $\delta(n)$ as the input and observing the resulting *impulse response* $h(n)$.

Unit step

The *unit step* function can be thought of as representing a process "turning on" at discrete time $n=0$ (Fig. 3).

$$u_s(n)=\begin{cases}1 & n\geq 0\\ 0 & n<0\end{cases} \tag{4}$$

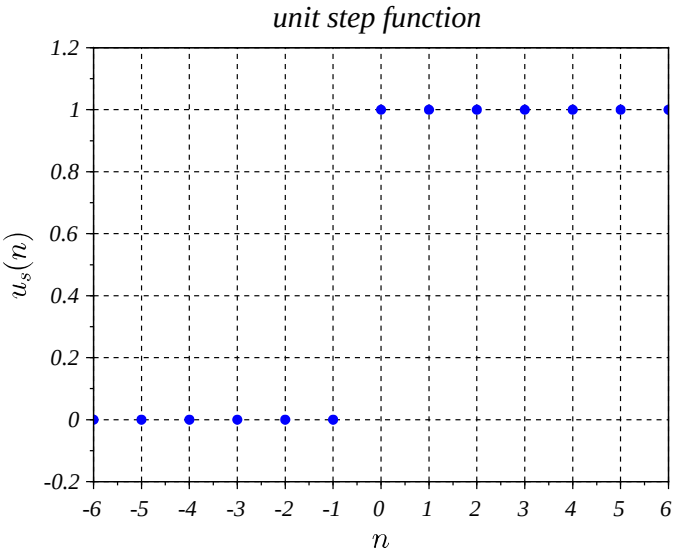

Fig. 3 Plot of unit-step function.

A difference of shifted unit step functions

$$p(n)=u_s(n-n_{\text{on}})-u_s(n-n_{\text{off}}) \tag{5}$$

can be used to represent a unit-amplitude pulse that turns on at $n=n_{\text{on}}$ and off at $n=n_{\text{off}}$

Discrete frequencies and sinusoids

Analog sinusoids of the form

$$x_a(t)=A\cos(2\pi F t+\phi) \tag{6}$$

play a central role in the analysis of continuous-time systems. Here F is the (analog) frequency in Hz (cycles per second). Sampling this with sampling period T_s we obtain the discrete signal

$$x(n)=x_a(nT_s)=A\cos(2\pi F T_s n+\phi) \tag{7}$$

The *sampling frequency* is $F_s=1/T_s$. For example, if $T_s=0.01\,\text{s}$ then the sampling frequency is $100\,\text{Hz}$, that is, we obtain 100 samples every second. Defining the *discrete frequency* f as

$$f = F\,T_s = \frac{F}{F_s} \rightarrow F = f\,F_s \tag{8}$$

we can express our discrete sinusoid as

$$x(n) = A\cos(2\pi\,f\,n + \phi) \tag{9}$$

Note that f is dimensionless. You can think of it as having "units" of "cycles per sample."

Aliasing

Unlike analog sinsusoids, discrete sinusoids are subject to *aliasing* due to "under sampling." Consider that for any integer k we have

$$\cos(2\pi[f + k]n + \phi) = \cos(2\pi\,f\,n + 2\pi\,k\,n + \phi) = \cos(2\pi\,f\,n + \phi) \tag{10}$$

since $k\,n$ is an integer and adding an integer multiple of 2π to the argument of a sinusoid does not change its value. This means that, for example, $\cos(2\pi\,0.1\,n)$ and $\cos(2\pi\,1.1\,n)$ are the same discrete signal. We say that the discrete frequency $f = 1.1$ is an *alias* of the frequency $f = 0.1$. This is illustrated in Fig. 4.

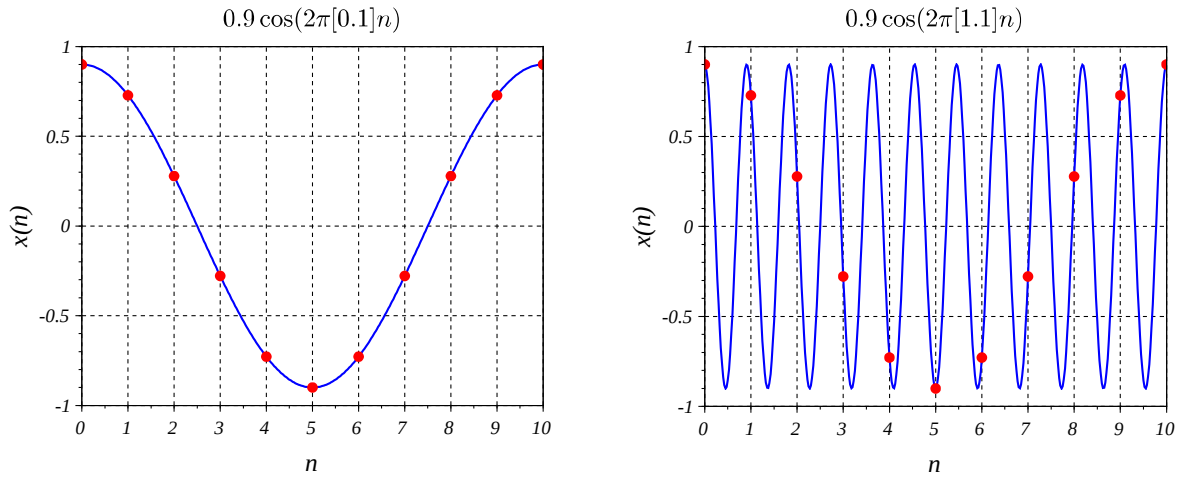

Fig. 4: Sinusoids with discrete frequencies $f = 0.1$, $f = 1.1$. The dots show the discrete signal values. The curves show corresponding continuous signals which the discrete values could be samples of. For $f = 1.1$ the sampling process suffers from aliasing.

Suppose an analog audio signal contains a tone at $F = 1{,}600\,\text{Hz}$ and a tone at $F = 8{,}800\,\text{Hz}$, and we sample it with frequency $F_s = 8{,}000\,\text{Hz}$. The corresponding discrete frequencies are $f = 0.2$ and $f = 1.1$. But samples with $f = 1.1$ are indistinguishable from samples with $f = 0.1$. Since $f = 0.1$ corresponds to $F = 800\,\text{Hz}$ we will obtain the same discrete signal we would have if the two analog tones were instead $F = 1{,}600\,\text{Hz}$ and $F = 800\,\text{Hz}$.

We can avoid aliasing ambiguities by limiting ourselves to discrete frequencies in the range

$$-\frac{1}{2} \le f \le \frac{1}{2} \tag{11}$$

The corresponding analog frequency range is

$$-\frac{F_s}{2} \le F \le \frac{F_s}{2} \tag{12}$$

Another way to express this is

$$F_s \geq 2|F| \tag{13}$$

for all frequencies F present in the signal. In words: *the sampling frequency must be at least twice the largest frequency magnitude in the analog signal.* This is know as the *Sampling Theorem*. As an example, humans can hear frequencies up to about 20 kHz. Unaliased sampling of audible frequencies therefore requires a sampling frequency of at least 40 kHz. The sampling rate for CD audio is $44.1\,\text{kHz} \geq 2(20\,\text{kHz})$. Therefore the discrete signal on a CD can represent any frequency components a human can hear.

Discrete systems

A discrete system transforms an input sequence x_n into an output sequence y_n as illustrated in the following figure.

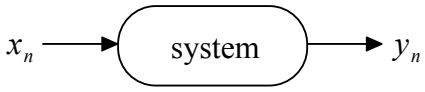

Fig. 5: Discrete system.

We write

$$y(n) = T\{x(n)\} \tag{14}$$

where $T\{\}$ represents the *transformation* by which the system produces output y from input x.

There is very little we can say about a general system. It is only when we constrain the system's properties that it begins to have identifiable characteristics. A system may have one or more of the following properties.

- *Linear*: It satisfies the *principle of superposition*.
- *Non-linear*: It does not satisfy the principle of superposition.
- *Time-invariant*: If a given input produces a given output, shifting the input in time produces the same output shifted in time.
- *Time-variant*: Shifted versions of a given input can produce different types of output signals.
- *Causal*: The present output depends only on the present and past inputs, not on future inputs.
- *Non-causal*: The present output depends on future inputs.
- *Stable*: For any finite input the output remains finite.
- *Unstable*: The output can "blow up" (grow without bound) for a finite input.
- *Deterministic*: The relation between input and output is precisely specified by a mathematical description of the system.
- *Stochastic*: The behavior of the system is unpredictable (random) at some level.

We will consider only deterministic systems in this course. For the most part we will be concerned with causal, linear, time-invariant systems, but not exclusively. Indeed, we will analyze some useful nonlinear systems later in the course.

Linear systems

A linear system satisfies the *superposition principle*. We state this as

$$\begin{aligned} &\text{if:} \quad T\{x_1(n)\} = y_1(n) \text{ and } T\{x_2(n)\} = y_2(n) \\ &\text{then: } T\{a_1 x_1(n) + a_2 x_2(n)\} = a_1 y_1(n) + a_2 y_2(n) \end{aligned} \tag{15}$$

for any constants a_1, a_2. The transformation produced by a linear system has the form of a linear combination of the input values. This can be expressed in the form

$$y(n) = \sum_{k=-\infty}^{\infty} h(n,k)x(k) \tag{16}$$

We call $h(n,k)$ the *impulse response* of the system. To see why, let the input be an impulse at, say, time step 2: $x(n) = \delta(n-2)$. Then

$$y(n) = \sum_{k=-\infty}^{\infty} h(n,k)\delta(k-2) = h(n,2) \tag{17}$$

The signal $h(n,2)$ is the *response* of the system to an *impulse* input at time sample 2. Likewise the signal $h(n,k)$ is the response of the system to an impulse input at time sample k (Fig. 6). For a general linear system the impulse response is a *two-dimensional* function.

Linear time-invariant (LTI) systems

A system that has the property

$$\begin{aligned} &\text{if:} \quad T\{x(n)\} = y(n) \\ &\text{then: } T\{x(n-k)\} = y(n-k) \end{aligned}$$

is said to be *time invariant*. Suppose $x_1(n) = \delta(n)$; then $y_1(n) = h(n,0)$. If $x_2(n) = \delta(n-k)$ then $y_2(n) = h(n,k)$. But $x_2(n) = x_1(n-k)$ so we must have $y_2(n) = y_1(n-k)$. It follows that

$$h(n,k) = h(n-k,0) \equiv h(n-k) \tag{18}$$

and the impulse response of a LTI system is a *one-dimensional* function. (16) becomes

$$y(n) = \sum_{k=-\infty}^{\infty} h(n-k)x(k) \tag{19}$$

We say y is the *convolution* of h and x, and we use $*$ to denote it

$$y(n) = h(n) * x(n) \tag{20}$$

In (19) let $k = n-i$. Then $n-k = i$ and

$$y(n) = \sum_{k=-\infty}^{\infty} h(n-k)x(k) = \sum_{i=-\infty}^{\infty} x(n-i)h(i) \tag{21}$$

The convolution doesn't change if we change the order of the functions, that is,

$$h(n) * x(n) = x(n) * h(n) \tag{22}$$

Causal linear time-invariant systems

If the impulse response of an LTI system is

$$h(n) = T\{\delta(n)\} \tag{23}$$

and if

$$h(n)=0 \text{ for all } n<0 \tag{24}$$

then the system is *causal*. This means that the present is not affected by the future, only by the present and the past (Fig. 6).

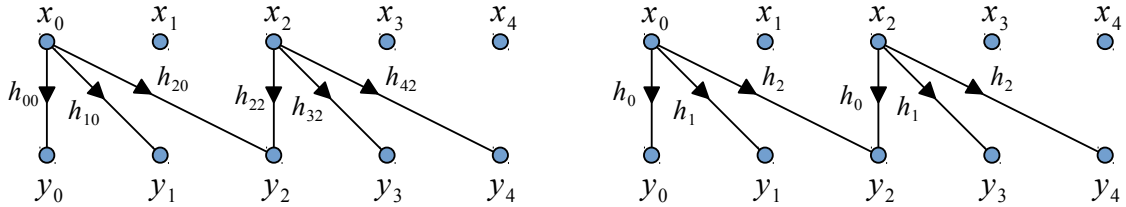

Fig. 6: (LEFT) For a general linear system the impulse response depends on both the output and input indices. (RIGHT) For a time-invariant system the impulse response depends only on the difference of the output and input indices. Both systems are causal because the arrows never connect a future input with a present output.

The input-output relationship can be written

$$y(n)=\sum_{k=0}^{\infty} h(k)\,x(n-k) \tag{25}$$

If $x(n)=y(n)=0$ for $n<0$ (x and y are also "causal") then

$$y(n)=\sum_{k=0}^{n} h(k)\,x(n-k) \tag{26}$$

This is a finite sum, so it can be calculated on a computer. We have

$$\begin{aligned} y(0)&=h(0)\,x(0)\\ y(1)&=h(0)\,x(1)+h(1)\,x(0)\\ y(2)&=h(0)\,x(2)+h(1)\,x(1)+h(2)\,x(0) \end{aligned} \tag{27}$$

and so on.

All *real-time* systems must be causal because we live in a causal world (the laws of physics are causal). But not all systems operate in real time. An important example is a filter applied to a prerecorded audio signal. The entire audio signal has "already happened." So, at any point in the signal the future is known; it's just the subsequent samples in the audio file. In this case a non-causal filter is possible, and often desirable, to implement.