Chapter 2

Discrete-Time Signals and Systems

Contents

2.1 Signal Basics

- A signal is a function of one or more independent variables
- 1-D signals are typically functions of time, although other independent variables are equally as valid
- 2-D signals, such as images, are functions of two spatial variables
- The independent variables themselves may be either continuous or discrete
  - A *continuous-time signal*, often termed an *analog signal*, is defined over a continuum of times
  - A *discrete-time signal* has independent variable defined only at discrete times
- A discrete-time signal is further specified to be a *Digital Signal* if the signal values or amplitudes take on discrete values (i.e. finite word-length representations)
- The above signal terminology can be applied to signal processing systems to obtain the following classifications:
  - For a *continuous-time system* both the input and output are continuous-time signals
  - For a *discrete-time system* both the input and output are discrete-time signals
  - For a *digital system* both the input and output are digital signals

- In this course we will deal for the most part with 1-D discrete-time signals, except when finite-precision numerical effects are considered

2.2 Discrete-time signals as sequences

- A discrete-time signal can be represented as a sequence of numbers

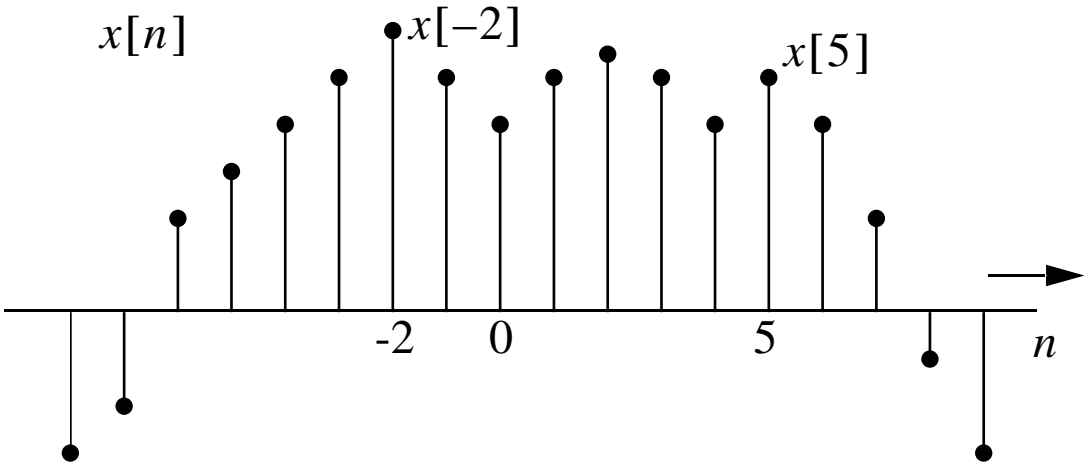
$$x_n = x\{n\} = x[n],\ -\infty < n < \infty$$

- If $x[n]$ is obtained by periodic or uniform sampling of some analog signal $x_a(t)$, then we can write

$$x[n] = x_a(nT),\ -\infty < n < \infty$$

  where T is the *sample spacing* and $f_s = 1/T$ is the *sampling frequency (rate)*

- We will show later that $x_a(t)$ can be reconstructed from $x[n]$ provided the *lowpass sampling theorem* is satisfied



Discrete-time signal plot.

2.3 Basic Sequences and Sequence Operations

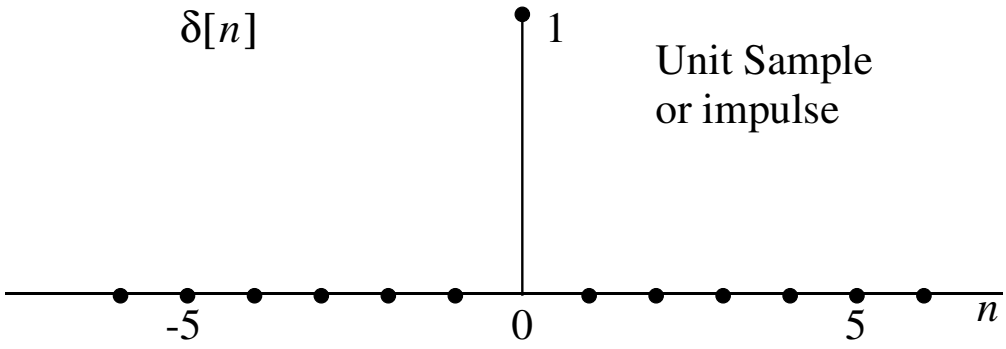
- *Sequence shift or delay* is obtained by letting $n \rightarrow n - n_o$ which defines a new signal

$$y[n] = x[n - n_o]$$

  If $n_o > 0$ then $y[n]$ is delayed and if $n_o < 0$ then $y[n]$ is advanced, with respect to $x[n]$

- *Unit sample sequence or impulse*, $\delta[n]$, is the discrete-time equivalent to $\delta(t)$

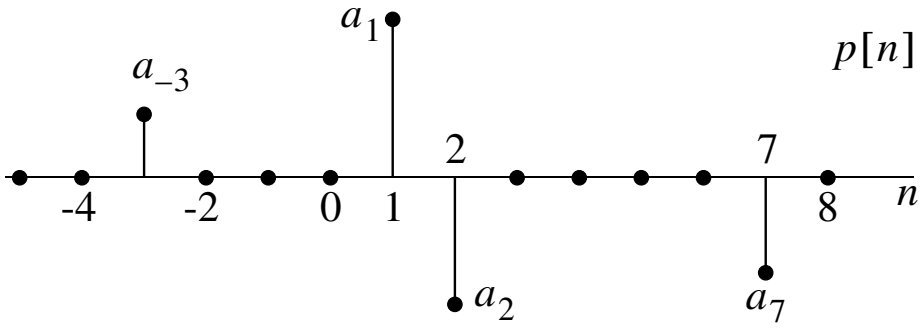
$$\delta[n] = \begin{cases} 0, & n \neq 0 \\ 1, & n = 0 \end{cases}$$



- Any sequence can be written as a linear combination of $\delta[n]$, e.g.

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$$

Example 2.1:



$$p[n] = a_{-3}\delta[n+3] + a_1\delta[n-1] + a_2\delta[n-2] + a_7\delta[n-7]$$

- *Unit step sequence*, $u[n]$

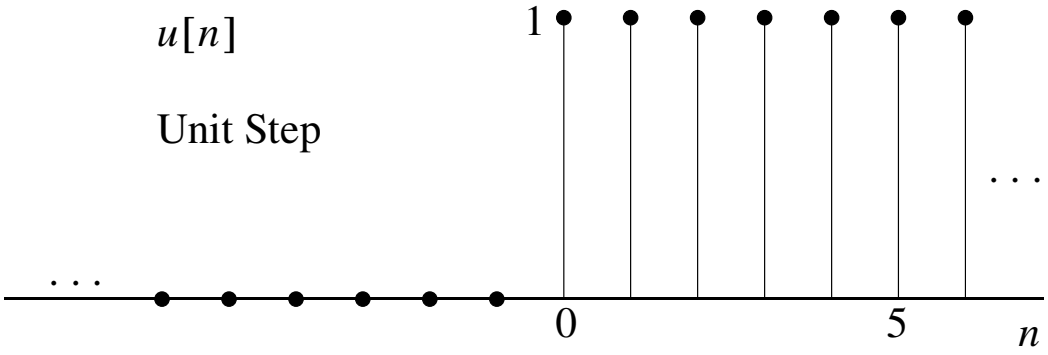
$$u[n] = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

or in terms of $\delta[n]$

$$\begin{aligned} u[n] &= \sum_{k=-\infty}^{n} \delta[k] \\ &= \sum_{k=0}^{\infty} \delta[n-k] \end{aligned}$$

which is analogous to the continuous-time unit step being written as

$$\begin{aligned} u(t) &= \int_{-\infty}^{t} \delta(\lambda)\, d\lambda \\ &= \int_{0}^{\infty} \delta(t-\lambda)\, d\lambda \end{aligned}$$



- Now turn the above relationships around and solve for the impulse function in terms of $u[n]$. We first recall that for continuous-time signals

$$\delta(t) = \frac{d}{dt}u(t)$$

Similarly $\delta[n]$ can be written as the *first backward difference* of $u[n]$

$$\delta[n] = u[n] - u[n-1]$$

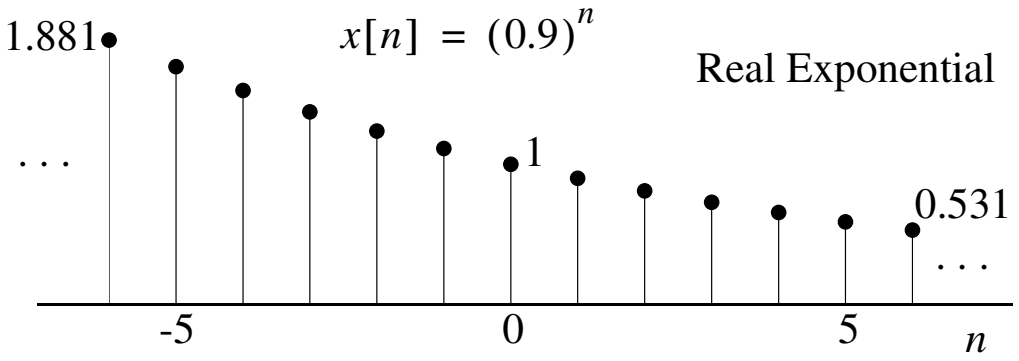
- *Exponential sequence*

$$x[n] = A\alpha^n$$

where A and α may in general be complex constants

1. Suppose A and α are both real and $0 < |\alpha| < 1$, then $|x[n]|$ is a decreasing sequence. If $\alpha < 0$ then the sequence is alternating.
2. Suppose $A = |A|e^{j\phi}$ and $\alpha = |\alpha|e^{j\omega_o}$, then $x[n]$ is a complex sinusoid with growing, constant, or decaying envelope if $|\alpha| > 1$, $|\alpha| = 1$, or $|\alpha| < 1$, respectively

$$\begin{aligned} x[n] &= |A||\alpha|^n e^{j(\omega_o n+\phi)} \\ &= |A||\alpha|^n[\cos(\omega_o n + \phi) + j\ \sin(\omega_o n + \phi)] \end{aligned}$$



- *Sinusoid sequence*

$$x[n] = A\cos(\omega_o n + \phi)$$

where ω_o has units of radians, but may be interpreted as radians/sample if n has units of samples

- The complex sinusoid, for the case $\alpha = 1$, and the real sinusoid both have the property that frequency translates of the form $\omega_o + 2\pi r$, r an integer, are indistinguishable from one another

- Defintion: A sequence $x[n]$ is periodic with period N if

$$x[n] = x[n + N], \text{ for all } n$$

Note that N must be an integer

Example 2.2: **Periodicity of the sinusoid sequence**

- We must establish the conditions under which

$$A\cos(\omega_o n + \phi) = A\cos(\omega_o n + \omega_o N + \phi)$$

- Clearly we must have

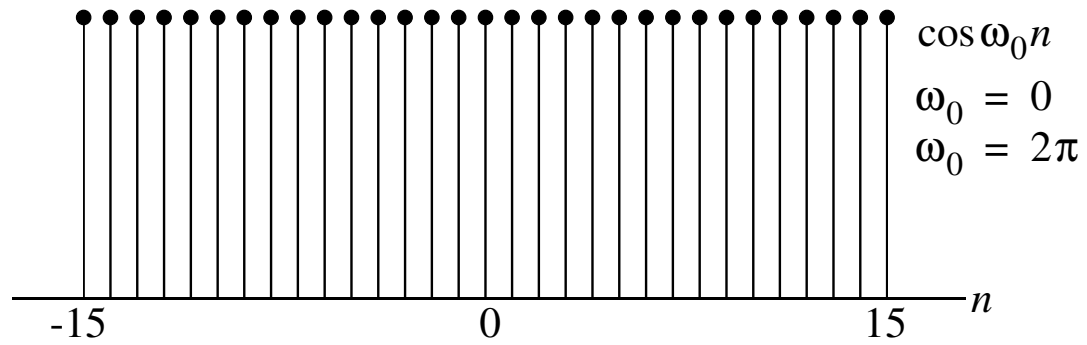
$$\omega_o N = 2\pi k \text{ or } \omega_o = \frac{2\pi k}{N},\ k \text{ an integer}$$

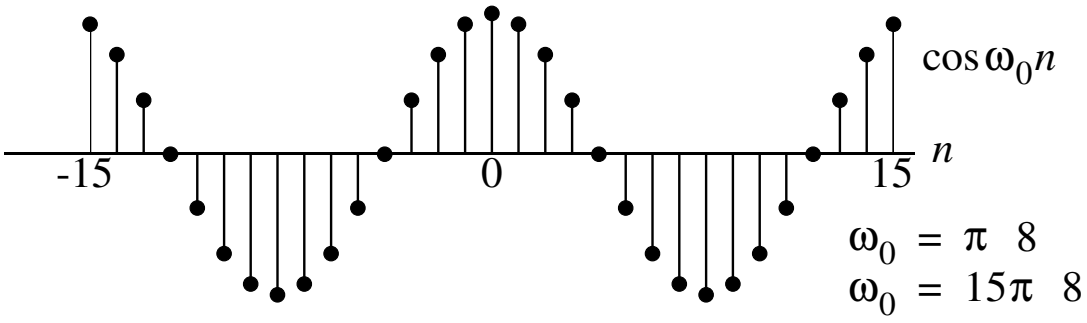
- Note that the period is not simply $2\pi/\omega_o$ as in the continuous-time case, $\omega_o/(2\pi)$ must be rational
- Suppose $\omega_o = 5\pi/8$, then the smallest N that satisfies the above equation is $N = 16$ ($k = 5$)
- Further observe that since ω_o and $\omega_o + 2\pi r$ are indistinguishable frequencies, then a sinusoidal sequence with period N has the N distinguishable frequencies

$$\omega_k = \frac{2\pi k}{N},\ k = 0, 1, \ldots, N-1$$

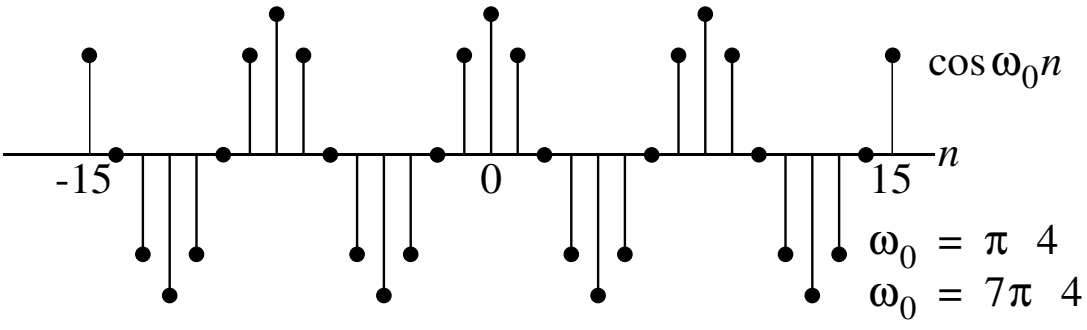
  on the interval $[0, 2\pi)$
- For $x(t) = A\cos(\Omega_o t + \phi)$, increasing Ω_o increases the oscillation rate of $x(t)$
- For $x[n] = A\cos(\omega_o n + \phi)$, increasing ω_o increases the oscillation rate of $x[n]$ only for $\omega_o \in [0, \pi]$
- For $\omega_o \in [\pi, 2\pi]$ the oscillation rate decreases back to zero

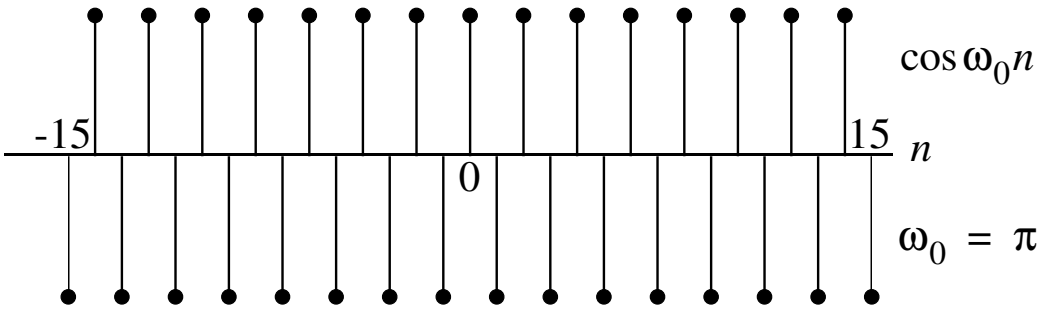
Plots of $\cos(\omega_o n)$





Plots of $\cos(\omega_o n)$

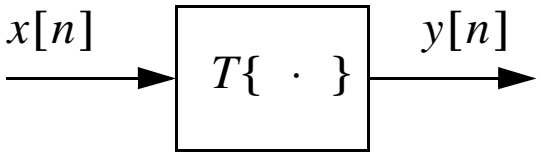




2.4 Discrete-Time Systems

- Definition: A discrete-time system is an operator that maps an input sequence into an output sequence

$$y[n] = T\{x[n]\}$$

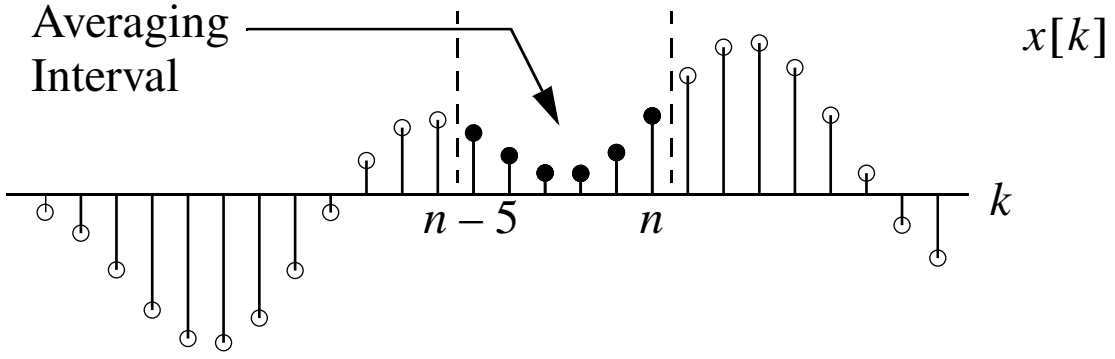


Example 2.3: Moving Average (MA) Operator

- Define $T\{\cdot\}$ such that

$$y[n] = \frac{1}{M_1 + M_2 + 1} \sum_{k=-M_1}^{M_2} x[n-k]$$

- The averaging is *causal* if we set $M_1 = 0$ so that no future values of the input are used to obtain the present output



A causal MA system with $M_2 = 5$, $M_1 = 0$

- A system is *memoryless* if each output $y[n]$ depends only on the present input $x[n]$

- An example of a memoryless system is a *power series nonlinearity*

$$y[n] = \sum_{k=0}^{N} a_k (x[n])^k$$

- A system is *linear* if superposition holds, that is if for $y_1[n] = T\{x_1[n]\}$, $y_2[n] = T\{x_2[n]\}$, and a and b constants, we can write

$$\begin{aligned} T\{ax_1[n] + bx_2[n]\} &= aT\{x_1[n]\} + bT\{x_2[n]\} \\ &= ay_1[n] + by_2[n] \end{aligned}$$

- Generalizing for $y_k[n] = T\{x_k[n]\}$ we can write

$$y[n] = T\left\{\sum_k a_k x_k[n]\right\} = \sum_k a_k y_k[n]$$

- A system is *time-invariant* or *shift-invariant* if for $y[n] = T\{x[n]\}$ and all n_o, the sequence $x_1[n] = x[n - n_o]$ produces system output $y_1[n] = y[n - n_o]$

- A system is *causal* or *nonanticipative* if all output values, $y[n_o]$, depend **only** on input values $x[n]$, $n \leq n_o$. Thus the present output value depends only on past and present input values.

- A system is *bounded-input bounded-output (BIBO) stable* if and only if every bounded input produces a bounded output

- $x[n]$ bounded implies there exists a constant $B_x > 0$ such that

$$|x[n]| \leq B_x < \infty \text{ for all } n$$

- Similarly $y[n]$ bounded implies there exists a constant $B_y > 0$ such that

$$|y[n]| \leq B_y < \infty \text{ for all } n$$

- It may be easier to show that a system is unstable since we need only find **one** bounded $x[n]$ which produces an unbounded $y[n]$

2.4.1 Linear Time-Invariant (LTI) Systems

The class of systems which we are most interested in are those which are both linear and time(shift)-invariant, the so-called LTI or LSI systems. A fundamental result for LTI systems is that the output sequence can be written as the convolution sum of the input sequence with the system *impulse response*.

- Definition: The impulse response, $h[n]$, is simply the system output when $x[n] = \delta[n]$, for an at rest system

- Recall that for any sequence $x[n]$ we can write

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$$

so inserting this into $y[n] = T\{x[n]\}$ we obtain

$$\begin{aligned} y[n] &= T\left\{\sum_{k=-\infty}^{\infty} x[k]\delta[n-k]\right\} \\ &\overset{\text{linearity}}{=} \sum_{k=-\infty}^{\infty} x[k]T\{\delta[n-k]\} \\ &\overset{\text{time-inv.}}{=} \sum_{k=-\infty}^{\infty} x[k]h[n-k] \end{aligned}$$

- The sum

$$\begin{aligned} y[n] &= \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n] \\ &= \sum_{k=-\infty}^{\infty} x[n-k]h[k] = h[n] * x[n] \end{aligned}$$

  is called the *convolution sum*

- In the above note that we use $*$ to denote convolution

Example 2.4: **Convolve pulse with exponential**

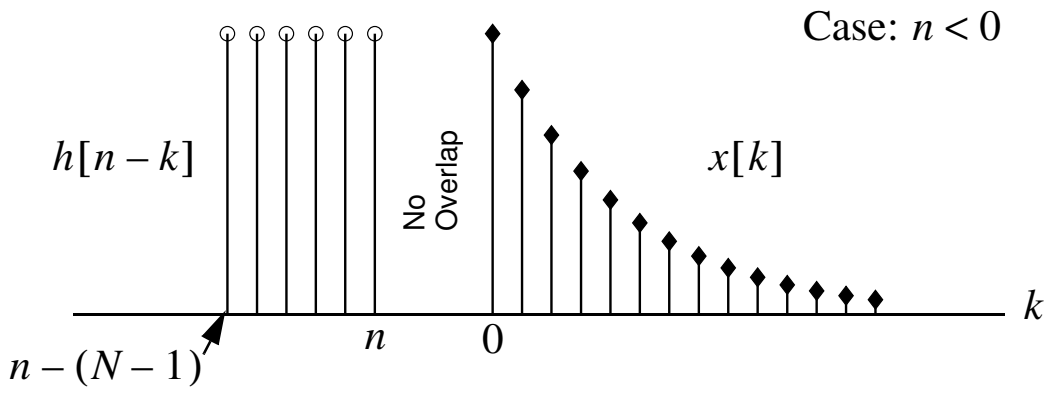
Consider a system with

$$h[n] = u[n] - u[n-N],\ N > 0$$

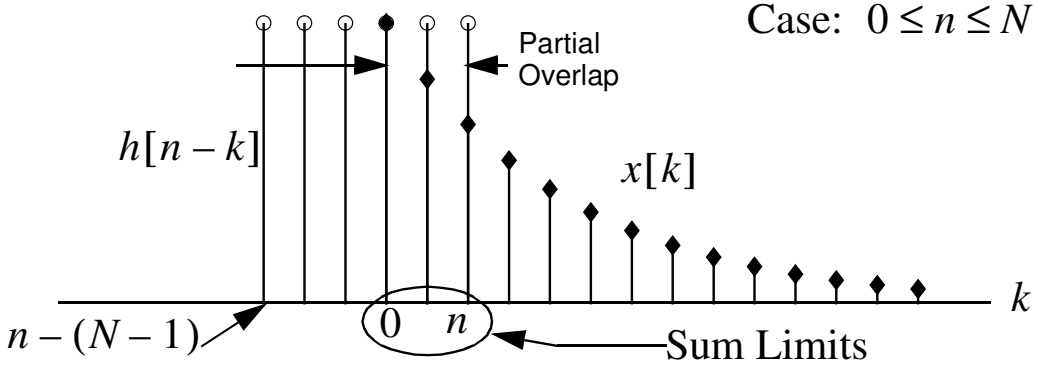
and

$$x[n] = a^n u[n],\ a \text{ real}$$

- To begin with sketch both $x[k]$ and $h[n-k]$



- Since both $x[k]$ and $h[n-k]$ do not overlap for $n < 0$ we conclude that $y[n] = 0$ for $n < 0$
- To obtain $y[n]$ for $n \geq 0$ we consider the case $0 \leq n \leq N-1$ first and then $n > N-1$, as shown in the following figures



- For $0 \leq n \leq N-1$ we can write

$$y[n] = \sum_{k=0}^{n} a^k$$

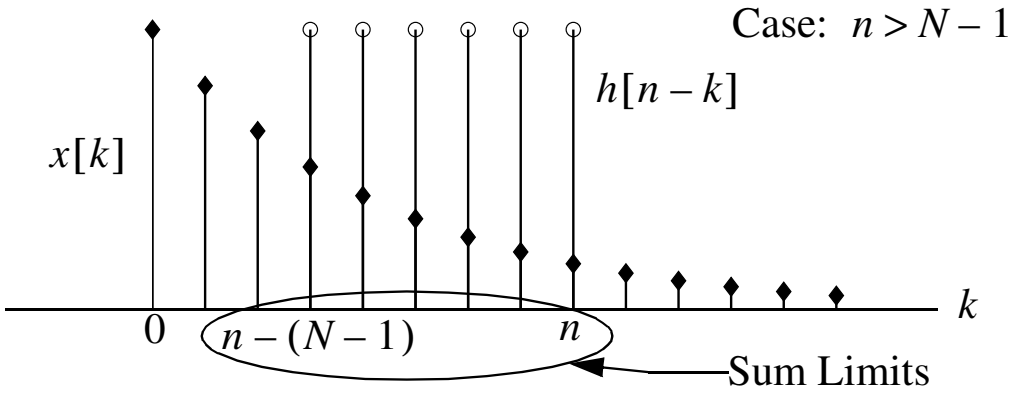
- Recall the *finite geometric series* sum formula

$$\sum_{k=N_1}^{N_2} \alpha^k = \frac{\alpha^{N_1} - \alpha^{N_2+1}}{1-\alpha}, \quad N_2 \geq N_1$$

- For the problem at hand

$$y[n] = \frac{1 - a^{n+1}}{1 - a}, \;\; 0 \leq n \leq N - 1$$

- For $n > N - 1$ we observe



so,

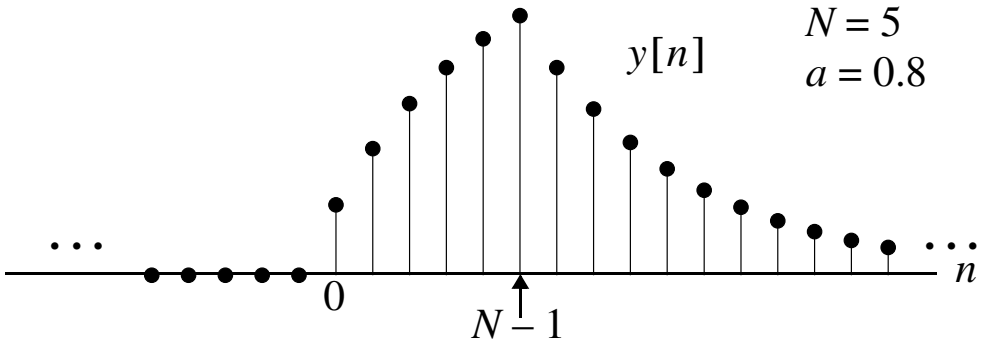
$$y[n] = \sum_{k=n-(N-1)}^{n} a^k$$

- If we reindex the sum by letting $k \rightarrow k - [n - (N - 1)]$ we can write

$$\begin{aligned} y[n] &= \sum_{k=0}^{n-[n-(N-1)]} a^{k+[n-(N-1)]} \\ &= a^{n-N+1} \sum_{k=0}^{N-1} a^k \\ &= a^{n-N+1} \left(\frac{1 - a^N}{1 - a} \right) \end{aligned}$$

- Combining the three solutions regions into one we can write

$$y[n] = \begin{cases} 0, & n < 0 \\ \frac{1-a^{n+1}}{1-a}, & 0 \le n \le N-1 \\ a^{n-N+1}\left(\frac{1-a^N}{1-a}\right), & N-1 < n \end{cases}$$



Example 2.5: **Piecewise Convolution Using Mathematica**

- Solving sequence convolution problems is tedious work
- Mathematica can help you check your work with minimal set-up work
- In this example I rework Example 2.4 by first defining the two signals $x[n]$ and $h[n]$, then use the `Sum[]` function to directly perform the convolutional sum:

```
h = UnitStep[n] - UnitStep[n - N]
```

```
UnitStep[n] - UnitStep[n - N]
```

```
x = a^n UnitStep[n]
```

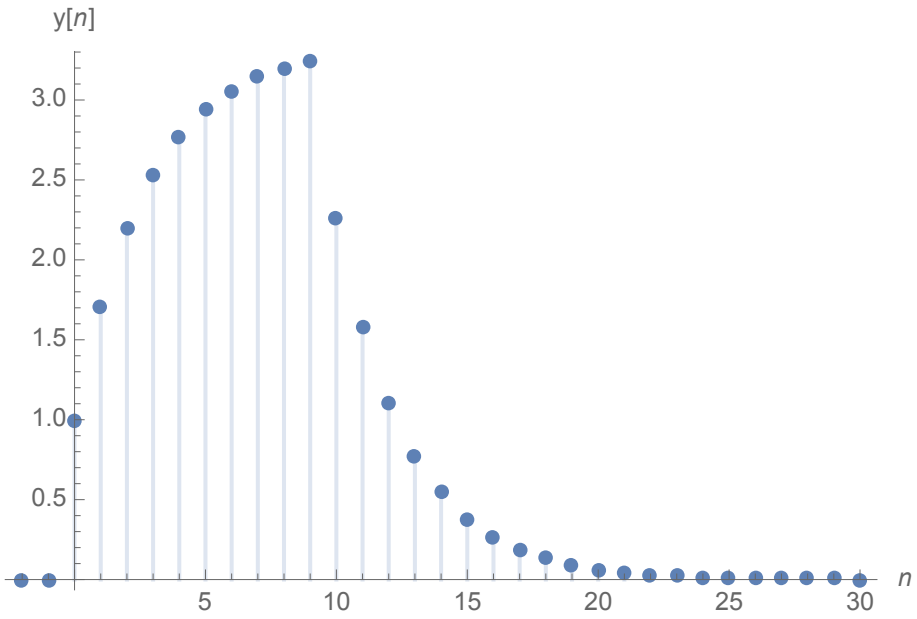
$$a^n \, \text{UnitStep}[n]$$

```
y = Sum[(x /. n → k) (h /. n → n - k), {k, -Infinity, Infinity}]
```

$$\begin{cases} -1 & n < 0 \;\&\&\; n == N \\ \dfrac{a\left(-1+a^{\text{Floor}[n]}\right)}{-1+a} & n \geq 0 \;\&\&\; n == N \\ \dfrac{-1+a^{1+\text{Floor}[n]}}{-1+a} & n \geq 0 \;\&\&\; n < N \\ \dfrac{a\left(a^{\text{Floor}[n]}-a^{\text{Floor}[n-N]}\right)}{-1+a} & n \geq 0 \;\&\&\; n > N \\ -\dfrac{-1+a^{1+\text{Floor}[n-N]}}{-1+a} & n < 0 \;\&\&\; n > N \\ 0 & \text{True} \end{cases}$$

- Plot using `DiscretePlot[]`, $N = 10$, and $a = 0.7$

```
DiscretePlot[y /. {N → 10, a → 0.7}, {n, -2, 30},
 AxesLabel -> {n, "y[n]"}]
```



Example 2.6: Python Sequence Convolution

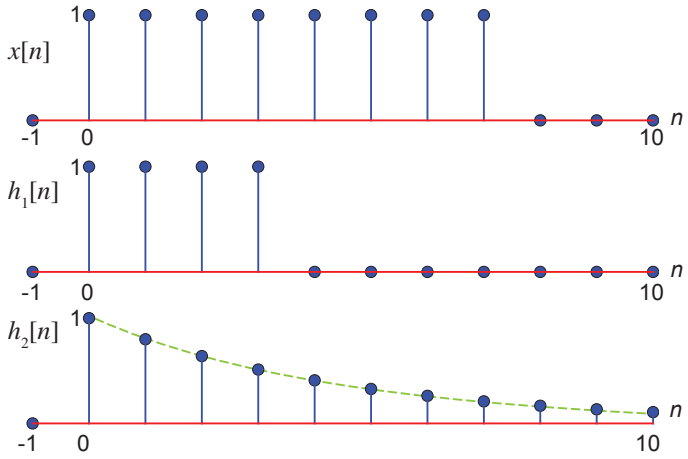
Signal and System Definitions

Convolve the following signals (sequences) using Python with Pylab

$$x[n] = \begin{cases} 1, & 0 \le n \le 7 \\ 0, & \text{otherwise} \end{cases}$$

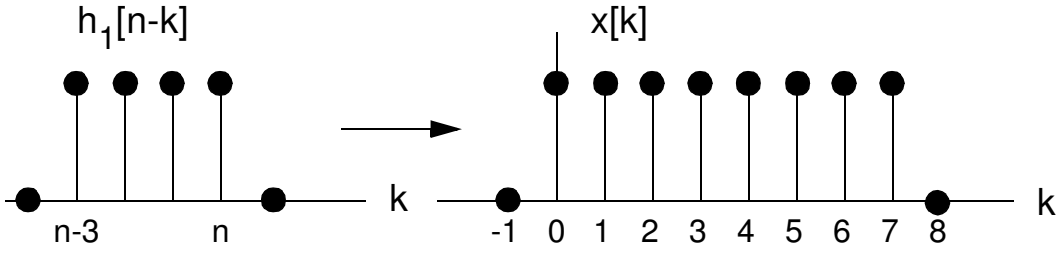
$$h_1[n] = \begin{cases} 1, & 0 \le n \le 3 \\ 0, & \text{otherwise} \end{cases}$$

$$h_2[n] = 0.8^n u[n]$$



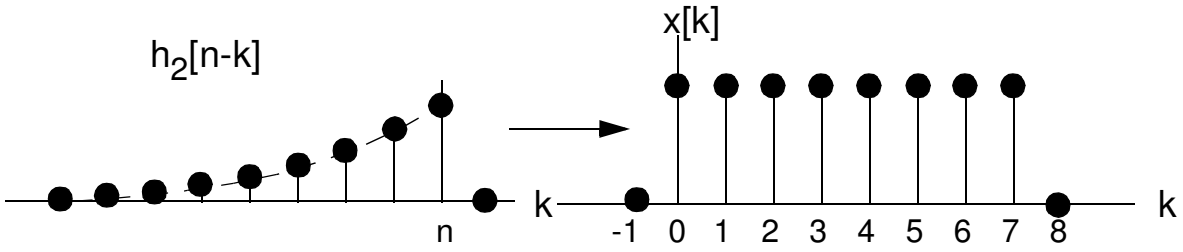
- First find

$$y_1[n] = x[n] * h_1[n] = \sum_{k=-\infty}^{\infty} x[k]h_1[n-k]$$



- Next find

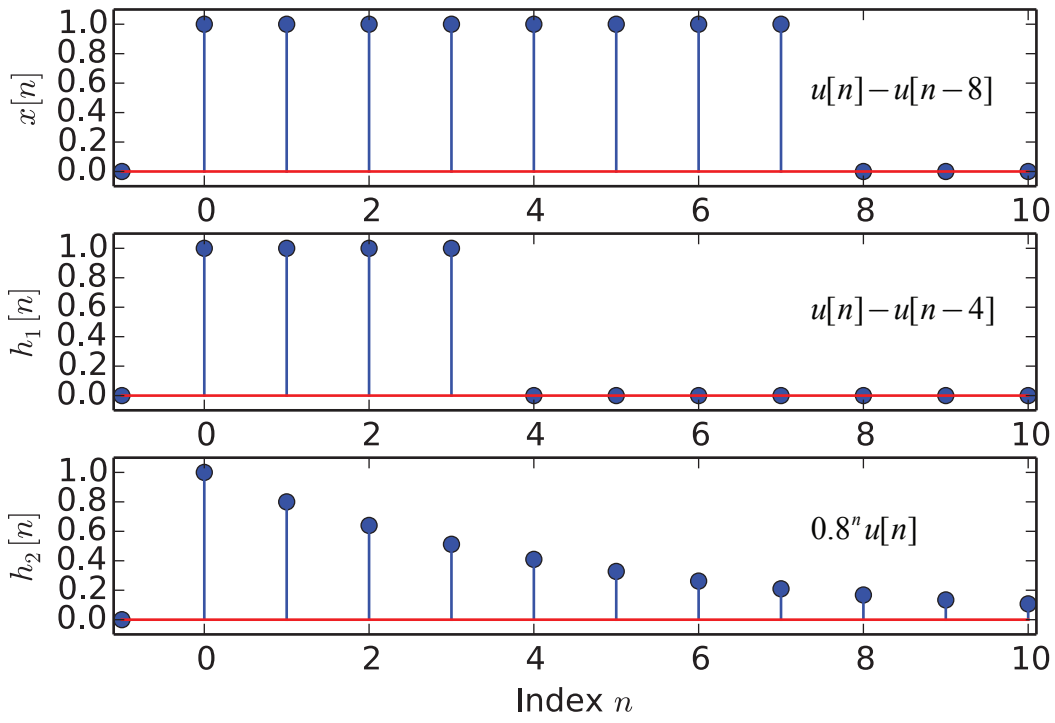
$$y_2[n] = x[n] * h_2[n] = \sum_{k=-\infty}^{\infty} x[k]h_2[n-k]$$



Signal Creation and Convolution Using Python

- Creating $x[n]$, $h_1[n]$, and $h_2[n]$ at the IPython qtconsole or in the Jupyter notebook with module `sigsys.py` imported with alias `ss`
- The function `ss.dstep(n)` is useful in this problem
- The starting point is to create an Python `ndarray` to hold the axes values of n over some support interval
- Here I choose $n \in [-1, 11)$

```
In [22]: import sk_dsp_comm.sigsys as ss
In [23]: n = arange(-1,11)
In [24]: x = ss.dstep(n) - ss.dstep(n-8)
In [25]: h1 = ss.dstep(n) - ss.dstep(n-4)
In [26]: h2 = 0.8**n * ss.dstep(n)
In [29]: subplot(311)
In [30]: stem(n,x)
In [31]: subplot(312)
In [32]: stem(n,h1)
In [33]: subplot(313)
In [34]: stem(n,h2)
In [35]: tight_layout()
```



The input signal and system impulse responses in Python.

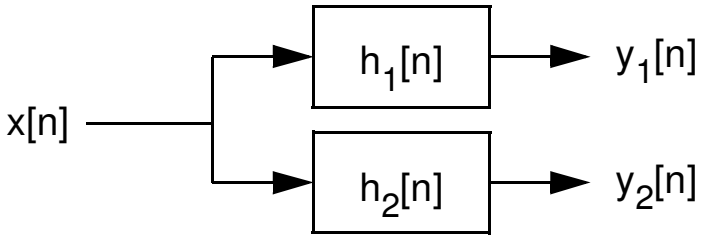
- Note I have made use of the helper function `dstep(n)` in creating the sequences
- The function description can be found to by typing `ss.dstep?`
- The Python code module `sigsys.py` (aliased as `ss`) contains many useful functions for doing signals and systems modeling and simulation; see scikit-dsp-comm
- The functions in `ss` include:

```
In [82]: dir(ss)
Out[82]: [BPSK_tx, CIC, NRZ_bits, NRZ_bits2, OA_filter,
OS_filter, PN_gen, am_rx,  am_rx_BPF, am_tx, biquad2, bit_errors,
cascade_filters, conv_integral,conv_sum, cpx_AWGN, cruise_control,
deci24, delta_eps, dimpulse, downsample, drect, dstep, env_det,
ex6_2, eye_plot, fft, fir_iir_notch, from_wav, fs_approx,
```

fs_coeff, interp24, line_spectra, lms_ic, lp_samp, lp_tri, m_seq, my_psd, peaking, plot_na, position_CD, prin_alias, rc_imp, rect, rect_conv, scatter, signal, simpleQuant, simple_SA, sinusoidAWGN, soi_snoi_gen, splane, sqrt_rc_imp, step, ten_band_eq_filt, ten_band_eq_resp, to_wav, tri, upsample, wavfile, zplane]

Obtaining System Outputs

In block diagram form we have the following:



At the IPython prompt (assuming `pylab` is loaded and you have imported `sk_dsp_comm.sigsys as ss`) simply enter

```
In [117]: n = arange(-1,11)
In [118]: x = ss.dstep(n)-ss.dstep(n-8)
In [119]: h1 = ss.dstep(n) - ss.dstep(n-4)
In [120]: y1,ny1 = ss.conv_sum(x,n,h1,n)
Output support: (-2, +20)
In [121]: subplot(211)
In [122]: stem(ny1,y1)

In [123]: n = arange(-1,31)
In [124]: x = ss.dstep(n)-ss.dstep(n-8)
In [125]: h2 = 0.8**n * ss.dstep(n)
In [126]: y2,ny2 = ss.conv_sum(x,n,h2,n,extent=('f','r'))
Output support: (-2, +29)
In [127]: subplot(212)
In [128]: stem(ny2,y2)
```

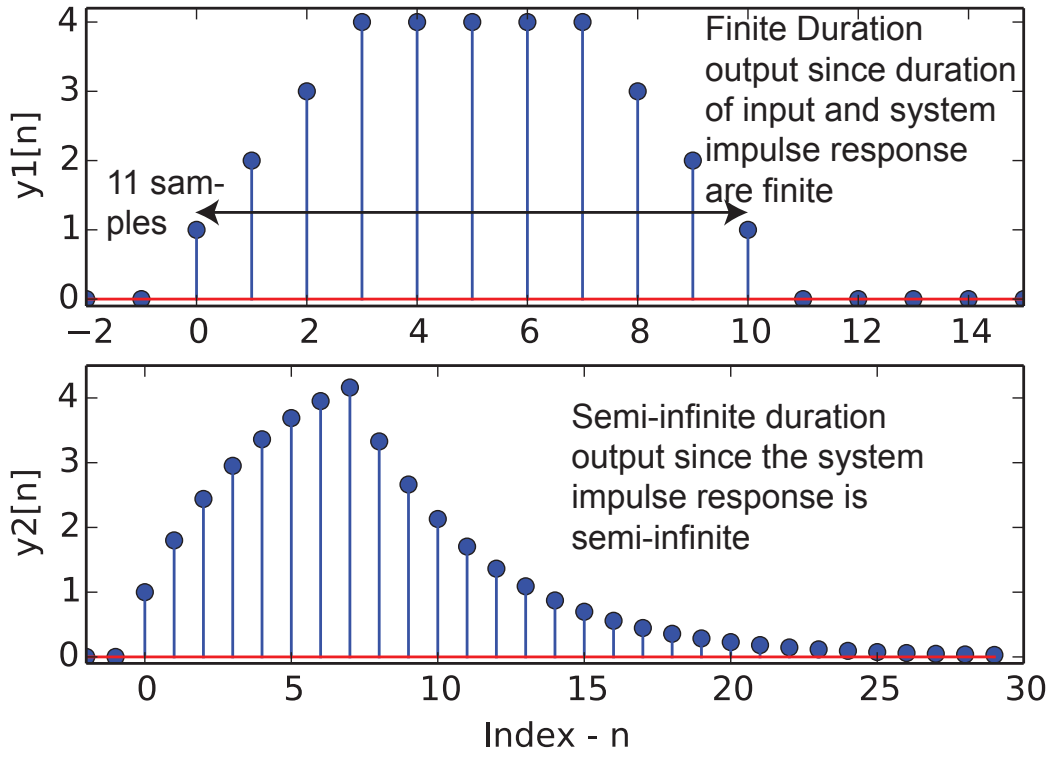
- The function `ss.conv_sum` performs discrete-time convolution using `y = signal.conv(x1,x2)` from the `scipy.signal` package

- The function `ss.conv_sum()` wraps `signal.conv()` and adds sequence index manipulation for accurate plotting and properly truncates the convolution if the sequences have infinite extent to the left or right

- In the above convolution operation it is assumed that both sequences start at $n = -1$, thus the output should start at $n = -2$ and end at $n = 11 + 31 = 42$

- For $y_1[n]$ the length of the output sequence will be the sum of the two input lengths minus one (why?), e.g., here we have $8 + 4 - 1 = 11$

- The theoretical length of $y_2[n]$ is infinite because $h_2[n]$ starts at $n = 0$ and extends to right out to $+\infty$



- The tabulated sequence values are:

```
In [129]: ny1
Out[129]:
array([-2, -1,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
       15, 16, 17, 18, 19, 20])

In [130]: y1
Out[130]:
array([ 0.,  0.,  1.,  2.,  3.,  4.,  4.,  4.,  4.,  4.,  3.,  2.,  1.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])

In [131]: ny2
Out[131]:
array([-2, -1,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
       15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29])

In [132]: y2
Out[132]:
array([ 0.        ,  0.        ,  1.        ,  1.8       ,  2.44      ,
        2.952     ,  3.3616    ,  3.68928   ,  3.951424  ,  4.1611392 ,
        3.32891136,  2.66312909,  2.13050327,  1.70440262,  1.36352209,
        1.09081767,  0.87265414,  0.69812331,  0.55849865,  0.44679892,
        0.35743914,  0.28595131,  0.22876105,  0.18300884,  0.14640707,
        0.11712566,  0.09370052,  0.07496042,  0.05996834,  0.04797467,
        0.03837973,  0.03070379])
```

- The fact that $y_1[n]$ is trapezoidal in shape should come as no surprise since convolving two rectangles in the continuous-time domain is known to produce a trapezoid

Convolving Other Types of Sequences

- The Python function `conv_sum`, found in the module `sigsys.py`, will actually properly convolve any two finite duration sequences, as well semi-infinite left- and right-sided sequences

- The help listing for the function, shown below, explains how this is done

```
def conv_sum(x1,nx1,x2,nx2,extent=('f','f')):
"""
Discrete convolution of x1 and x2 with proper tracking of the output time axis.

Convolve two discrete-time signals using the SciPy function signal.convolution.
The time (sequence axis) are managed from input to output. y[n] = x1[n]*x2[n].

Parameters
----------
x1 : ndarray of signal x1 corresponding to nx1
nx1 : ndarray time axis for x1
x2  : ndarray of signal x2 corresponding to nx2
nx2 : ndarray time axis for x2
extent : ('e1','e2') where 'e1', 'e2' may be 'f' finite, 'r' right-sided,
         or 'l' left-sided

Returns
-------
y : ndarray of output values y
ny : ndarray of the corresponding sequence index n

Notes
-----
The output time axis starts at the sum of the starting values in x1 and x2
and ends at the sum of the two ending values in x1 and x2. The default
extents of ('f','f') are used for signals that are active (have support)
on or within n1 and n2 respectively. A right-sided signal such as
a^n*u[n] is semi-infinite, so it has extent 'r' and the
convolution output will be truncated to display only the valid results.

Examples
--------
>>> nx = arange(-5,10)
>>> x = drect(nx,4)
>>> y,ny = conv_sum(x,nx,x,nx)
>>> stem(ny,y)
>>> # Consider a pulse convolved with an exponential ('r' type extent)
>>> h = 0.5**nx*dstep(nx)
>>> y,ny = conv_sum(x,nx,h,nx,('f','r')) # note extents set
>>> stem(ny,y) # expect a pulse charge and discharge sequence
"""
```

2.5 Properties of LTI Systems

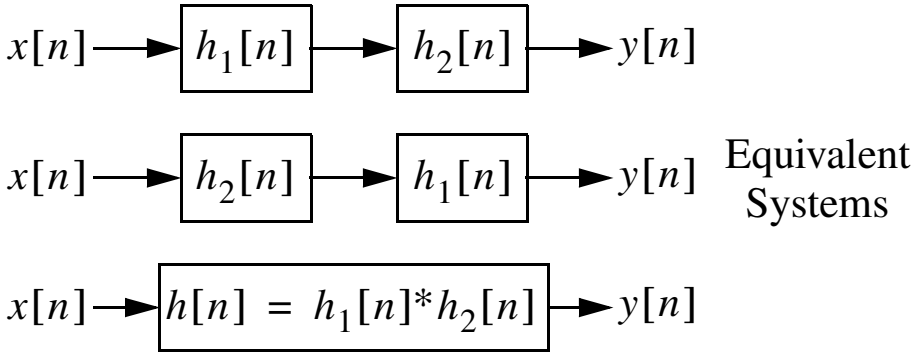
The LTI system properties considered here are for the most part based on relationships involving the convolution sum and the impulse response

- Consider a *cascade connection* of systems with impulse responses $h_1[n]$ and $h_2[n]$

$$y[n] = \{x[n] * h_1[n]\} * h_2[n]$$

- Since convolution is commutative, it follows that

$$y[n] = x[n] * h[n], \text{ where } h[n] = h_1[n] * h_2[n]$$

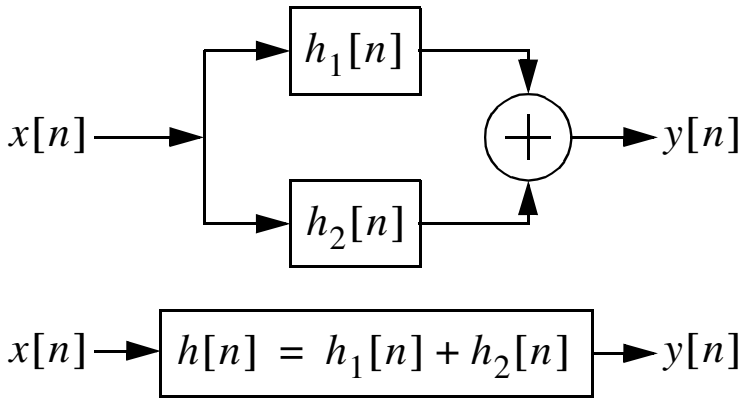


- Consider a *parallel connection* of systems with impulse responses $h_1[n]$ and $h_2[n]$

$$y[n] = x[n] * h_1[n] + x[n] * h_2[n]$$

- Since convolution is distributive, it follows that

$$y[n] = x[n] * h[n], \text{ where } h[n] = h_1[n] + h_2[n]$$



- Theorem: LTI systems are BIBO stable if and only if

$$S = \sum_{k=-\infty}^{\infty} |h[k]| < \infty$$

proof $\Leftarrow$

$$|y[n]| = \left| \sum_{k=-\infty}^{\infty} h[k]x[n-k] \right| \leq \sum_{k=-\infty}^{\infty} |h[k]||x[n-k]|$$

Now since $|x[n]| < B_x$ it follows that

$$|y[n]| \leq B_x \sum_{k=-\infty}^{\infty} |h[k]|$$

QED

proof $\Rightarrow$

First assume that $S = \infty$ and show that a (just one) bounded input can be found that will result in an unbounded output. One such sequence is

$$x[n] = \begin{cases} \frac{h^*[-n]}{|h[-n]|}, & h[n] \neq 0 \\ 0, & h[n] = 0 \end{cases}$$

Clearly $B_x = 1$, but $y[0]$ is

$$y[0] = \sum_{k=-\infty}^{\infty} x[-k]h[k] = \sum_{k=-\infty}^{\infty} \frac{|h[k]|^2}{|h[k]|} = S \to \infty$$

QED

- If an LTI system is causal it then follows that

$$h[n] = 0, \quad n < 0$$

- A *causal sequence* is one which is zero for $n < 0$
- An impulse response with only a finite number of nonzero samples is called a *finite-duration impulse response (FIR)*
- Clearly FIR systems are always stable so long as the impulse response values have finite magnitude
- An impulse response with infinite duration is said to have an *infinite-duration impulse response (IIR)*

Example 2.7: Impulse, Step, and Exponential

- An ideal delay has impulse response

$$h[n] = \delta[n - n_d], \quad n_d > 0$$

  Since $\sum_n |h[n]| = 1$ we know that the system is stable, and since $h[n] = 0$ for $n < 0$ provided $n_d > 0$, we also know that the system is causal.

- If $h[n] = u[n]$ then the system is an *accumulator*. Clearly an accumulator is IIR and causal, but since

$$\sum_{n=-\infty}^{\infty} |u[n]| \to \infty$$

  we see that the accumulator is also unstable.

- Consider a system with exponential impulse response $h[n] = a^n u[n]$, a real. The system is stable provided

$$S = \sum_{n=0}^{\infty} |a|^n < \infty$$

  - Recall the *infinite geometric series* formula

$$\sum_{n=0}^{\infty} r^n = \frac{1}{1-r}, \quad |r| < 1$$

  Thus

$$S = \frac{1}{1-|a|} < \infty$$

  and the system is stable provided

$$|a| < 1$$

2.6 Linear Constant-Coefficient Difference Equations

A subclass of LTI systems which is of considerable interest are those systems which have input output relationship satisfying an Nth-order constant coefficient difference equation (LCCDE) of the form

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]$$

Example 2.8: Simple LCCDE Examples

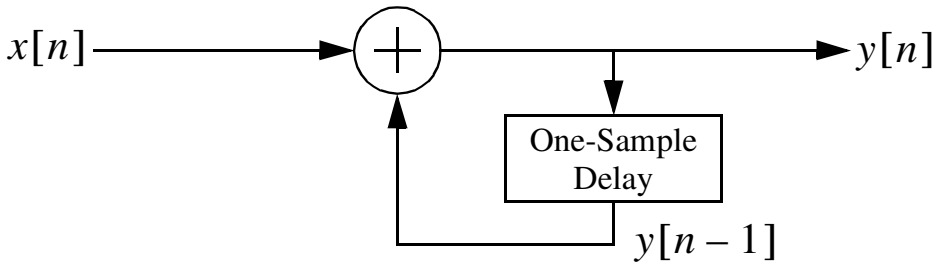
- As a specific example consider an accumulator

$$y[n] = \sum_{k=-\infty}^{n} x[k]$$

- Note that the first backward difference formed at the output returns the original input

$$y[n] - y[n-1] = x[n], \text{ or } y[n] = y[n-1] + x[n]$$

- The form $y[n] = y[n-1] + x[n]$ shows that the accumulator can be written as an LCCDE with $N = 1$, $a_0 = 1$, $a_1 = -1$, $M = 0$, and $b_0 = 1$



Accumulator LCCDE block diagram

- As a second example consider the moving average system with $M_1 = 0$ so that the system is causal

$$y[n] = \frac{1}{M_2 + 1} \sum_{k=0}^{M_2} x[n-k]$$

- This system fits the LCCDE form given above if we let $N = 0$, $M = M_2$, $a_0 = 1$, and $b_k = 1/(M_2 + 1)$, for $0 \leq k \leq M_2$

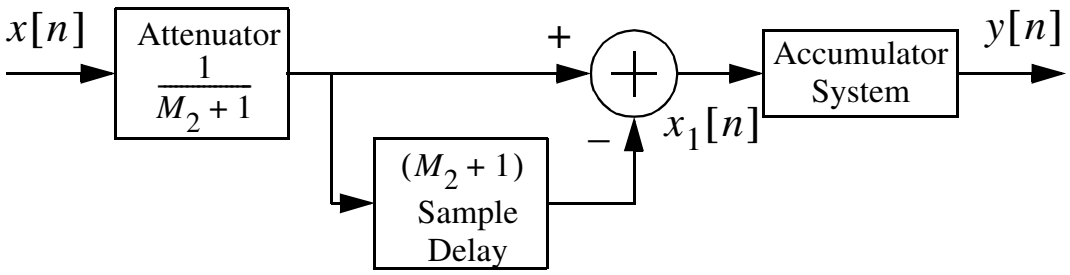
- The impulse response for the causal moving average system, as obtained by setting $x[n] = \delta[n]$, is given by

$$h[n] = \frac{1}{M_2 + 1}(u[n] - u[n - M_2 - 1])$$

- Note that since $u[n-M] = \delta[n-M] * u[n]$, we can also write

$$h[n] = \frac{1}{M_2 + 1}(\delta[n] - \delta[n - M_2 - 1]) * u[n]$$

- Recalling that $u[n]$ is the impulse response of an accumulator, we observe that the above form of the moving average system impulse response is in the form of a cascade system



Causal moving average, recursive LCCDE block diagram

- The first stage produces output

$$x_1[n] = \frac{1}{M_2 + 1}(x[n] - x[n - M_2 - 1])$$

- The second stage is the accumulator, thus the total input/output relationship is

$$y[n] = y[n-1] + \frac{1}{M_2 + 1}(x[n] - x[n - M_2 - 1])$$

  which corresponds to an LCCDE with coefficients $N = 1$, $a_0 = 1$, $a_1 = -1$, $b_0 = -b_{M_2+1} = 1/(M_2 + 1)$, and $b_k = 0$ otherwise

- From the above we conclude that the simple causal moving average system does not have a unique LCCDE representation

Example 2.9: A Nonlinear System Example

- An example of a nonlinear system with difference equation representation (not an LCCDE however) is the square root algorithm

$$y[n] = \frac{1}{2}\left[y[n-1] + \frac{x[n]}{y[n-1]}\right]$$

- If we let the input be an amplitude step (i.e. $x[n] = Au[n]$), then the initial condition $y[n-1]$ serves as the initial estimate of $\sqrt{A}$, and the response $y[n]$ tends to $\sqrt{A}$ as n increases
  - Let $A = 2$, and $y[-1] = 1$, then $y[0] = 3/2$, $y[1] = 1.416667$, and $y[2] = 1.4142157$
  - Let $A = 2$, and $y[-1] = 1.5$, then $y[0] = 1.41667$, and $y[1] = 1.4142157$
  - Note that $\sqrt{2} = 1.4142136$

2.6.1 Classical Solution of LCCDEs

- The solution of a LCCDE can be written as the sum

$$y[n] = y_h[n] + y_p[n]$$

where $y_h[n]$ is the *natural response* which is a solution to the *homogeneous* equation

$$\sum_{k=0}^{N} a_k y[n-k] = 0$$

and $y_p[n]$ is the *forced response* which is a *particular solution* of the *nonhomogeneous* equation

$$\sum_{k=0}^{N} a_k y[n-k] = x[n]$$

- To solve the homogeneous equation we assume a solution of the form

$$y_h[n] = z^n$$

If we substitute this solution into the homogeneous equation we obtain

$$\sum_{k=0}^{N} a_k y_h[n-k] = 0$$

or

$$z^{n-N} \underbrace{(a_0 z^N + a_1 z^{N-1} + \cdots + a_N)}_{\text{characteristic polynomial}} = 0$$

- The system *characteristic polynomial*, as indicated above, will have N roots, $z_i, i = 1, \ldots, N$, which may be real/complex and distinct/repeated. Assuming the coefficients $a_0, a_1, \ldots, a_N$ are real, then complex roots will always occur in complex-conjugate pairs.

- If the roots are distinct, then the general homogeneous solution is of the form

$$y_h[n] = A_1 z_1^n + A_2 z_2^n + \cdots + A_N z_N^n$$

where the coefficients $A_1, A_2, \ldots, A_N$ are determined from the system initial conditions.

- If a repeated root of multiplicity $m \leq N$ should occur, then the homogeneous solution is of the form

$$\begin{aligned} y_h[n] &= A_1 z_1^n + A_2 n z_1^n + \cdots + A_m n^{m-1} z_1^n \\ &\quad + A_{m+1} z_{m+1}^n + \cdots + A_N z_N^n \end{aligned}$$

- To solve the nonhomogeneous equation and obtain the forced response, we can use the *method of undetermined coefficents* which involves guessing a particular solution, typically having the same form as $x[n]$

- Particular solutions for several types of inputs

$x[n]$ ($u[n]$ implied)	$y_p[n]$ ($u[n]$ implied)
A (const.)	B
AM^n	BM^n
An^M	$B_0 n^M + K_1 n^{M-1} + \cdots + K_M$
$A^n n^M$	$A^n(B_0 n^M + K_1 n^{M-1} + \cdots + K_M)$
$A\cos\omega_o n$	$B_1 \cos\omega_o n + B_2 \sin\omega_o n$
$A\sin\omega_o n$	$B_1 \cos\omega_o n + B_2 \sin\omega_o n$

Example 2.10:

Classical solution: Consider the first order system

$$y[n] = ay[n-1] + x[n] \;\; |a| < 1$$

with auxiliary (initial) condition $y[-1] = c$. Let $x[n] = Ku[n]$.

- The homogeneous equation is $y[n] - ay[n-1] = 0$ and the assumed solution is $y_h[n] = z^n$, thus we must solve

$$z^n - az^{n-1} = 0 \;\Rightarrow\; z_1 = a$$

- The homogeneous solution is

$$y_h[n] = A_1 a^n$$

- The particular solution of the nonhomogeneous equation is obtained by assuming

$$y_p[n] = Bu[n]$$

where B is a scale factor

- Substituting the assumed solution into the nonhomogeneous equation we obtain

$$Bu[n] - aBu[n-1] = Ku[n]$$

- To solve for B we evaluate the above equation where none of the terms vanish, that is for $n \geq 1$

$$B - aB = K \text{ or } B = \frac{K}{1-a}$$

so

$$y_p[n] = \frac{K}{1-a}u[n]$$

- The total solution can now be written as

$$\begin{aligned} y[n] &= y_h[n] + y_p[n] \\ &= A_1 a^n + \frac{K}{1-a}, \; n \geq 0 \end{aligned}$$

- To solve for A_1 use the fact that $y[-1] = c$ and write

$$y[0] = A_1 + \frac{K}{1-a}$$

also

$$y[0] = ay[-1] + K$$

thus

$$A_1 = ay[-1] + K - \frac{K}{1-a} = ca - \frac{Ka}{1-a}$$

- Rewriting, the total solution becomes

$$y[n] = ca^{n+1} + K\frac{1-a^{n+1}}{1-a},\ n \geq 0$$

- Note that the zero state response (i.e. $c = 0$) is

$$y[n] = K\frac{1-a^{n+1}}{1-a}u[n]$$

Recursive solution: In general a recursive solution for $y[n]$ can be obtained using the recurrence formula

$$y[n] = -\sum_{k=1}^{N}\frac{a_k}{a_0}y[n-k] + \sum_{r=0}^{M}\frac{b_r}{a_0}x[n-r]$$

along with the auxiliary conditions $y[-1], \ldots, y[-N]$. We start by finding $y[0]$. A similar procedure can be used to find $y[n]$ for $n < 0$.

- For the example we are considering we can write

$$\begin{aligned} y[0] &= ac + K \\ y[1] &= a^2c + aK + K \\ y[2] &= a^3c + a^2K + aK + K \\ \vdots &= \vdots \\ y[n] &= a^{n+1}c + a^nK + \cdots + aK + K \\ &= ca^{n+1} + K\frac{1-a^{n+1}}{1-a} \end{aligned}$$

- For systems with input/output satisfying an LCCDE we can make the following observations:
  - The system will in general **not** be linear. Recall that in the previous example when $K = 0$ (i.e. $x[n] = 0$) we found that $y[n] = ca^{n+1} \neq 0$ unless $c = 0$
  - The system will in general **not** be time-invariant. Again with reference to the previous example we note that with $x_1[n] = x[n - n_o] = Ku[n - n_o]$ the output is

$$y_1[n] = ca^{n+1} + K\frac{1 - a^{n+1-n_o}}{(1 - a)}u[n - n_o]$$

  - If the system is initially at rest, then the system **will** be linear, time-invariant, and causal

---

Example 2.11:

Find the impulse response, $h[n]$, of the second order system

$$y[n] - 3y[n - 1] - 4y[n - 2] = x[n] + 2x[n - 1]$$

- Since $h[n] = y[n]$ when $x[n] = \delta[n]$ we must solve

$$y[n] - 3y[n - 1] - 4y[n - 2] = \delta[n] + 2\delta[n - 1]$$

  or

$$y[n] = 3y[n - 1] + 4y[n - 2] + \delta[n] + 2\delta[n - 1]$$

- For $n \geq 2$ we have the homogeneous equation

$$y[n] - 3y[n - 1] - 4y[n - 2] = 0$$

and characteristic equation $z^2 - 3z - 4 = 0$, which has solution

$$y[n] = A_1(-1)^n + A_2 4^n, \quad n \geq 2$$

- A_1 and A_2 are determined by satisfying the nonhomogeneous equation for $n = 0$ and 1
- Assuming zero initial conditions we must have

$$\begin{aligned} y[0] &= 1 = A_1 + A_2 \\ y[1] &= 3 + 2 = 5 = -A_1 + 4A_2 \end{aligned}$$

  which implies that $A_2 = 1 - A_1$ and $5 = -5A_1 + 4$, thus $A_1 = -1/5$ and $A_2 = 6/5$
- Substituting the constants we obtain

$$h[n] = \left[-\frac{1}{5}(-1)^n + \frac{6}{5}4^n \right] u[n]$$

- Note, if $M = N$ in the general case, then in order to satisfy the nonhomogeneous equation we would need to modify the solution by including the term $A_0\delta[n]$
  - For the more general case of $M > N$ we add to the solution a term of the form

$$\sum_{k=0}^{M-N} A_k \delta[n-k]$$

- A case in point being the LCCDE

$$y[n] - y[n-1] = x[n] + x[n-1]$$

- Since $M = 1 = N$ and $y[n] - y[n-1] = 0$ for $n \geq 2$ we have as general solution for $h[n]$

$$h[n] = A_1(1)^n u[n] + A_0\delta[n] = A_1 + A_0\delta[n], \;\; n \geq 0$$

- To find A_1 and A_0 we solve

$$\begin{aligned} y[0] &= 1 = A_1 + A_0 \\ y[1] &= 2 = A_1 \end{aligned}$$

- Thus $A_1 = 2$ and $A_0 = -1$ giving

$$h[n] = 2u[n] - \delta[n]$$

- As a check note that

$$\begin{aligned} y[n] &= y[n-1] + \delta[n] + \delta[n-1] \\ y[0] &= 0 + 1 + 0 = 1 \\ y[1] &= 1 + 0 + 1 = 2 \\ y[2] &= 2 + 0 + 0 = 2 \\ &\vdots \quad \vdots \\ y[n] &= 2, \;\; n \geq 1 \end{aligned}$$

Example 2.12: The Python Filter Function for LCCDEs

Given an LCCDE description of a system, the Python scipy.signal function `lfilter()` can be used to process arbitrary signals through the system.

- We begin with a description of the function `lfilter()`

```
Definition:  signal.lfilter(b, a, x, axis=-1, zi=None)
Docstring:
Filter data along one-dimension with an IIR or FIR filter.

Filter a data sequence, `x`, using a digital filter.  This works for many
fundamental data types (including Object type).  The filter is a direct
form II transposed implementation of the standard difference equation
(see Notes).

Parameters
----------
b : array_like
    The numerator coefficient vector in a 1-D sequence.
a : array_like
    The denominator coefficient vector in a 1-D sequence.  If ``a[0]``
    is not 1, then both `a` and `b` are normalized by ``a[0]``.
x : array_like
    An N-dimensional input array.
axis : int
    The axis of the input data array along which to apply the
    linear filter. The filter is applied to each subarray along
    this axis.  Default is -1.
zi : array_like, optional
    Initial conditions for the filter delays.  It is a vector
    (or array of vectors for an N-dimensional input) of length
    ``max(len(a),len(b))-1``.  If `zi` is None or is not given then
    initial rest is assumed.  See `lfiltic` for more information.

Returns
-------
y : array
    The output of the digital filter.
zf : array, optional
    If `zi` is None, this is not returned, otherwise, `zf` holds the
    final filter delay values.

Notes
-----
The filter function is implemented as a direct II transposed structure.
This means that the filter implements::

   a[0]*y[n] = b[0]*x[n] + b[1]*x[n-1] + ... + b[nb]*x[n-nb]
                         - a[1]*y[n-1] - ... - a[na]*y[n-na]

using the following difference equations::
```

```
    y[m] = b[0]*x[m] + z[0,m-1]
    z[0,m] = b[1]*x[m] + z[1,m-1] - a[1]*y[m]
    ...
    z[n-3,m] = b[n-2]*x[m] + z[n-2,m-1] - a[n-2]*y[m]
    z[n-2,m] = b[n-1]*x[m] - a[n-1]*y[m]

where m is the output sample number and n=max(len(a),len(b)) is the
model order.

The rational transfer function describing this filter in the
z-transform domain is::

                        -1                -nb
              b[0] + b[1]z  + ... + b[nb] z
    Y(z) = ---------------------------------- X(z)
                        -1                -na
              a[0] + a[1]z  + ... + a[na] z
```

- The system must first be put into a difference equation form
- The filter function can also use initial conditions
- As a first example consider

$$h_1[n] = 0.7^n u[n]$$

which is equivalent to the LCCDE (why?)

$$y[n] = 0.7y[n-1] + x[n]$$

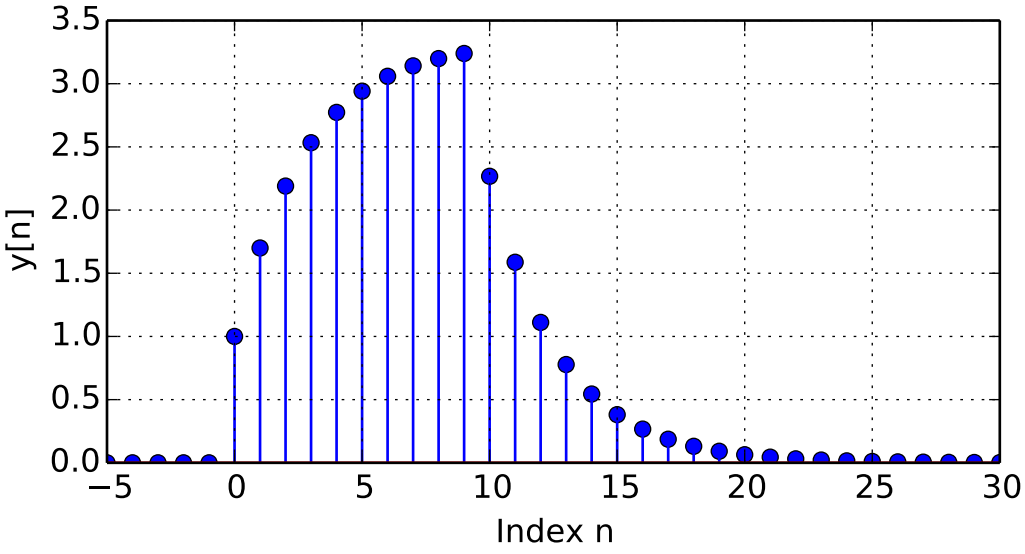
$$\text{or } \underbrace{1y[n] - 0.7y[n-1]}_{\texttt{a=[1 -0.7]}} = \underbrace{1x[n]}_{\texttt{b=[1]}},\ n \geq 0$$

- Let the input be a 10 sample pulse of the form

$$x[n] = \begin{cases} 1, & 0 \leq n \leq 9 \\ 0, & \text{otherwise} \end{cases}$$

- The Python code is shown below:

```
In [84]: n = arange(-5,30+1)
In [85]: x = ss.dstep(n) - ss.dstep(n-10)
In [86]: y = signal.lfilter([1],[1, -0.7],x)
In [87]: stem(n,y)
```



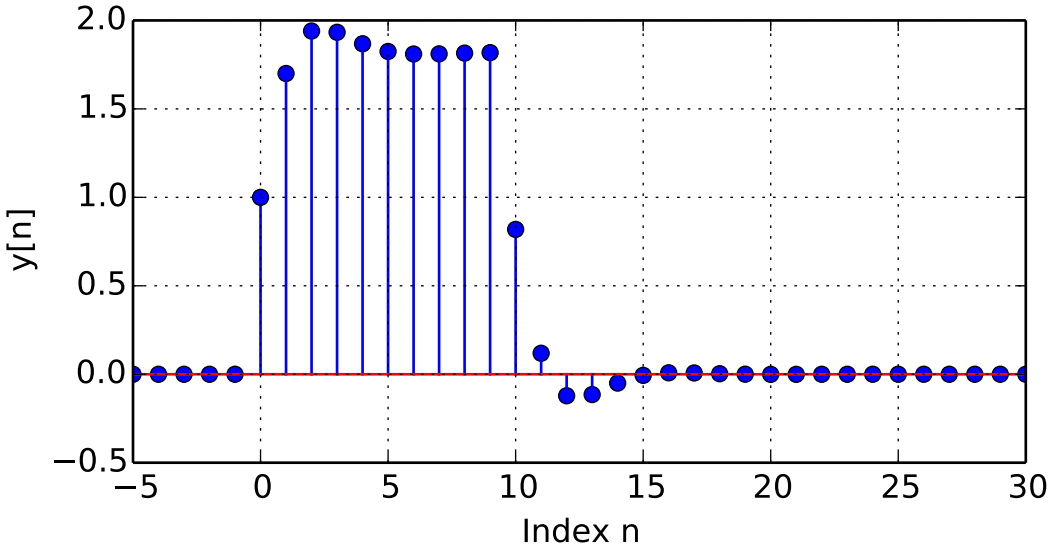
Pulse response of a first-order LCCDE.

- As a simple extension to the above consider the second-order difference equation

$$\underbrace{1y[n] - 0.7y[n-1] + 0.25y[n-2]}_{\texttt{a=[1 -0.7 0.25]}} = \underbrace{1x[n]}_{\texttt{b=[1]}}, \; n \geq 0$$

- The new Python code is:

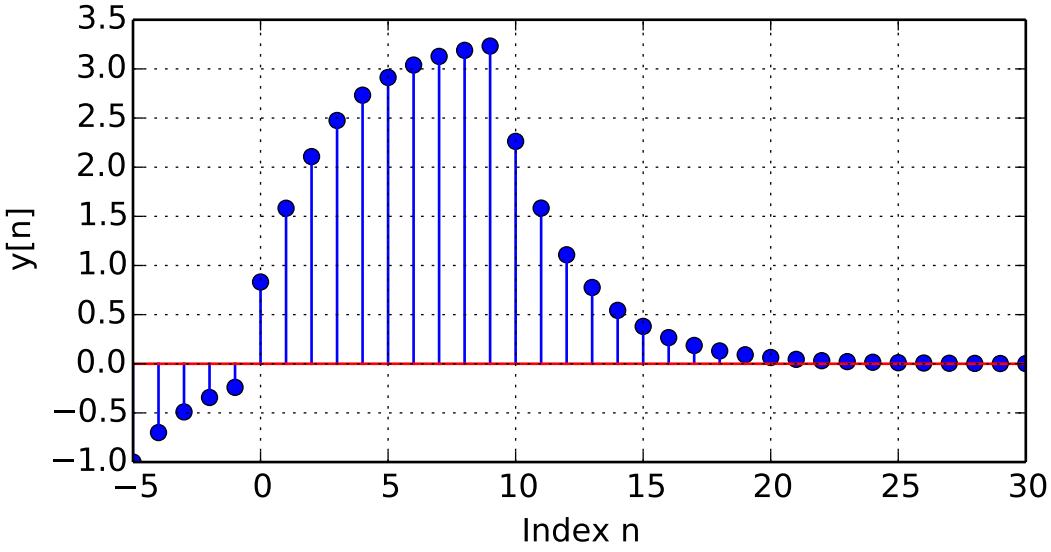
```
In [98]: y = signal.lfilter([1],[1, -0.7, 0.25],x)
In [99]: stem(n,y)
```



Pulse response of a second-order LCCDE.

- As a final variation let us apply an initial condition to `filter()` and consider again the first-order LCCDE
- We will set $y[n-1] = -1$ at the start of the simulation

```
In [118]: y,zi = signal.lfilter([1],[1, -0.7],x,zi=[-1])
```



Pulse response with $y[n-1] = -1$.

2.7 Frequency-Domain Representations

The complex exponential sequence $x[n] = e^{j\omega n}$, $-\infty < n < \infty$, is an eigenfunction of an LTI system. An LTI system with sinusoidal input thus has sinusoidal output of the same frequency and an amplitude and phase shift as a result of passing through the system.

- To verify the eigenfunction property consider an LTI system with impulse response $h[n]$, and input $x[n] = e^{j\omega n}$

$$\begin{aligned} y[n] &= \sum_{k=-\infty}^{\infty} h[k] e^{j\omega(n-k)} \\ &= e^{j\omega n} \left(\sum_{k=-\infty}^{\infty} h[k] e^{-j\omega k} \right) \end{aligned}$$

- Define:

$$H(e^{j\omega}) \equiv \sum_{k=-\infty}^{\infty} h[k] e^{-j\omega k}$$

- Now we can write

$$y[n] = H(e^{j\omega}) e^{j\omega n}$$

  from which we see that $e^{j\omega n}$ is indeed an eigenfunction and the eigenvalue, $H(e^{j\omega})$, is formally known as the *frequency response* of the system

- Since $H(e^{j\omega})$ is in general complex it can be expressed as

$$\begin{aligned} H(e^{j\omega}) &= H_R(e^{j\omega}) + jH_I(e^{j\omega}) \text{ (real/imaginary)} \\ &= |H(e^{j\omega})| e^{j\angle H(e^{j\omega})} \text{ (magnitude/phase)} \end{aligned}$$

- We can extend the above results to a linear combination of complex exponentials

$$x[n] = \sum_k \alpha_k e^{j\omega_k n}$$

and write $y[n]$ for an LTI system as

$$y[n] = \sum_k \alpha_k H(e^{j\omega_k}) e^{j\omega_k n}$$

Example 2.13:

Consider the ideal delay system $y[n] = x[n - n_d]$, n_d an integer.

- If $x[n] = e^{j\omega n}$, then clearly

$$y[n] = e^{j\omega(n-n_d)} = \underbrace{e^{-j\omega n_d}}_{H(e^{j\omega})} e^{j\omega n}$$

- As an alternate approach to finding $H(e^{j\omega})$ first recall that

$$h[n] = \delta[n - n_d]$$

- By definition the ideal delay system has frequency response

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \delta[n - n_d] e^{-j\omega n} = e^{-j\omega n_d}$$

- For this simple system the magnitude and phase response are

$$\begin{aligned} |H(e^{j\omega})| &= 1, \\ \angle H(e^{j\omega}) &= -\omega n_d \end{aligned}$$

- Suppose now that $x[n] = A\cos(\omega_o n + \phi)$, what is $y[n]$?
- We begin by writing the cosine sequence in terms of two complex exponential sequences and then use linearity to obtain $y[n]$ as the sum of the responses

$$\begin{aligned} y[n] &= H(e^{j\omega_o})\frac{A}{2}e^{j(\omega_o n+\phi)} \\ &\quad + H(e^{-j\omega_o})\frac{A}{2}e^{-j(\omega_o n+\phi)} \end{aligned}$$

- To simplify we will assume that $h[n]$ is real and hence that

$$\begin{aligned} H(e^{-j\omega_o}) &= \sum_{n=-\infty}^{\infty} h[n]e^{j\omega_o n} = \left[\sum_{n=-\infty}^{\infty} h[n]e^{-j\omega_o n}\right]^* \\ &= H^*(e^{j\omega_o}) \end{aligned}$$

- Using the above frequency response property we can then write

$$y[n] = A|H(e^{j\omega_o})|\cos\left(\omega_o n + \phi + \angle H(e^{j\omega_o})\right)$$

- For the special case of the ideal delay system we have

$$\begin{aligned} y[n] &= A\cos(\omega_o n + \phi - \omega_o n_d) \\ &= A\cos(\omega_o(n - n_d) + \phi) \end{aligned}$$

  as expected

A fundamental difference between the discrete-time and continuous-time frequency response is that in the discrete-time case $H(e^{j\omega})$ is **always** periodic with period 2π.

- To verify the periodicity note that

$$H(e^{j(\omega+2\pi)}) = \sum_{n=-\infty}^{\infty} h[n] \underbrace{e^{-j(\omega+2\pi)n}}_{e^{-j2\pi n}e^{-j\omega n}} = H(e^{j\omega})$$

- In general

$$H(e^{j(\omega+r2\pi)}) = H(e^{j\omega}) \text{ for } r \text{ an integer}$$

Example 2.14:

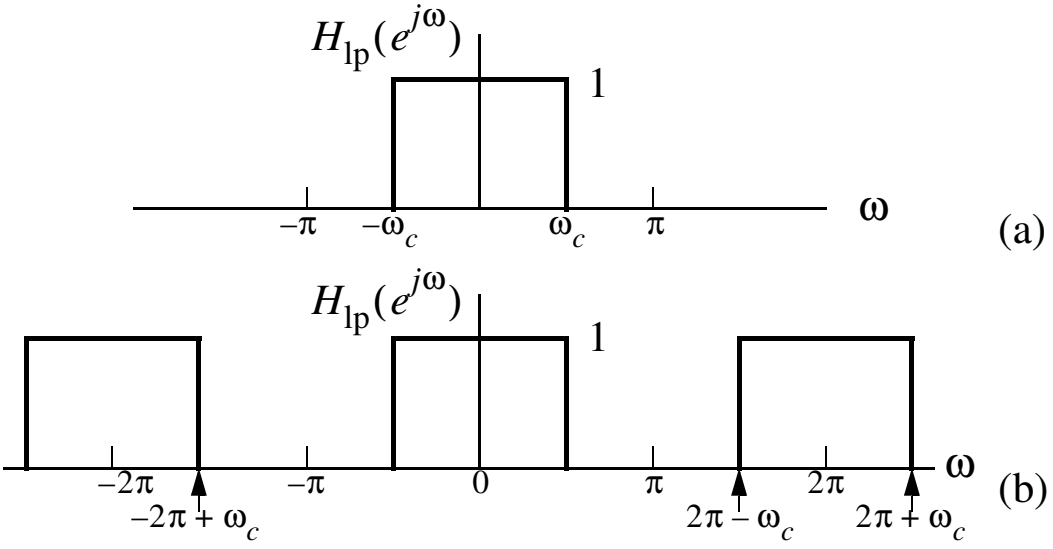
Consider a continuous-time ideal lowpass filter with cutoff frequency Ω_c.

- The continuous-time frequency response is defined to be

$$H_{a,lp}(\Omega) = \begin{cases} 1, & |\Omega| < \Omega_c \\ 0, & \text{otherwise} \end{cases}$$

- The equivalent discrete-time frequency response in the fundamental interval $-\pi < \omega \leq \pi$ is

$$H_{lp}(e^{j\omega}) = \begin{cases} 1, & |\omega| < \omega_c \\ 0, & \omega_c < |\omega| \leq \pi \end{cases}$$



Discrete-time ideal lowpass frequency response, (a) fundamental interval (b) periodic response

Example 2.15:

Find $H(e^{j\omega})$ for the moving average system which has impulse response

$$h[n] = \begin{cases} \frac{1}{M_1+M_2+1}, & -M_1 \le n \le M_2 \\ 0, & \text{otherwise} \end{cases}$$

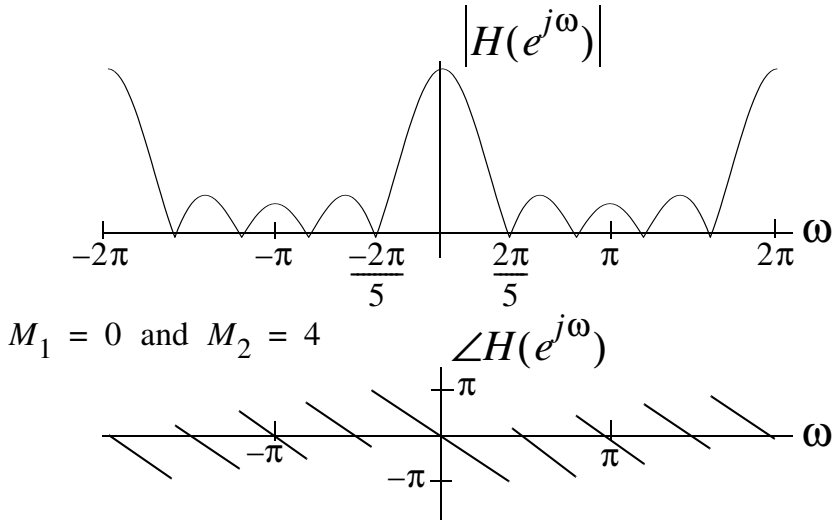
- To find $H(e^{j\omega})$ we simply use the definition

$$\begin{aligned} H(e^{j\omega}) &= \frac{1}{M_1+M_2+1} \sum_{n=-M_1}^{M_2} e^{-j\omega n} \\ &= \frac{1}{M_1+M_2+1} \frac{e^{j\omega M_1} - e^{-j\omega(M_2+1)}}{1-e^{-j\omega}} \end{aligned}$$

- After additional algebraic manipulation it can be shown that

$$H(e^{j\omega}) = \frac{1}{M_1 + M_2 + 1} e^{-j\omega(M_2 - M_1)/2} \cdot \frac{\sin[\omega(M_1 + M_2 + 1)/2]}{\sin(\omega/2)}$$

- Note that in discrete-time systems we often find the term $\sin(Nx)$ over $\sin(x)$ appearing where as in continuous-time systems we see $\sin(x)/x = \text{sinc}(x)$



(a) Magnitude and (b) phase response for MA system

2.7.1 Applying a Complex Exponential Now!

This section on frequency domain representations begins by applying $e^{j\omega n}$ on $n \in (-\infty, \infty)$. The interval $(-\infty, \infty)$ may seem impractical, but it is the basis for the definition of $H(e^{j\omega})$. The half-interval $[0, \infty)$ is perhaps more practical and at the present time may offer some insight into LTI system characteristics in both the time and frequency domains.

- Suppose we apply

$$x[n] = e^{j\omega n}u[n]$$

to a causal LTI system (i.e. $h[n] = 0, n < 0$)

- Applying the convolution sum formula yields

$$y[n] = \begin{cases} 0, & n < 0 \\ \left(\sum_{k=0}^{n} h[k]e^{-j\omega k}\right) e^{j\omega n}, & n \geq 0 \end{cases}$$

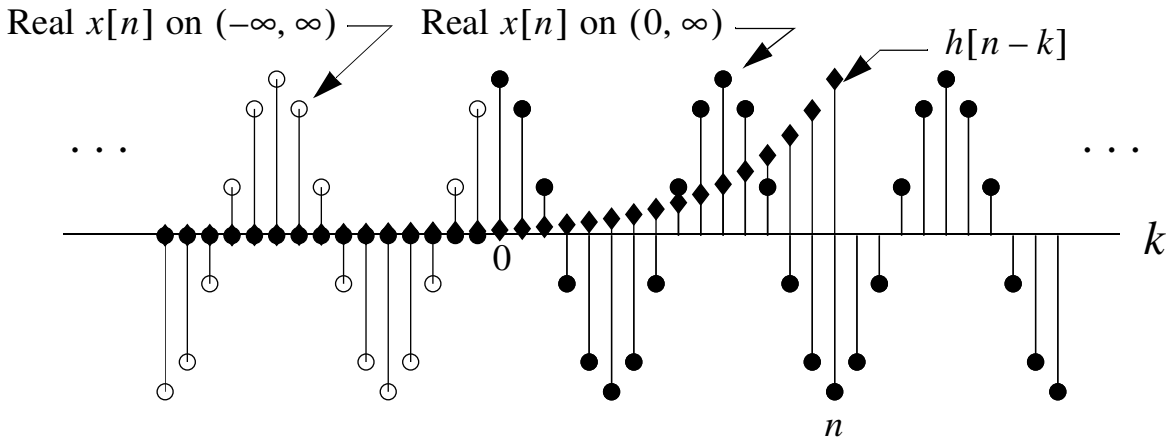
- The fact that $y[n] = 0$ for $n < 0$ is obvious
- What is interesting is that we can write $y[n]$ on the interval $n \geq 0$ as

$$\begin{aligned} y[n] &= \left(\sum_{k=0}^{\infty} h[k]e^{-j\omega k}\right) e^{j\omega n} - \left(\sum_{k=n+1}^{\infty} h[k]e^{-j\omega k}\right) e^{j\omega n} \\ &= H(e^{j\omega})e^{j\omega n} - \left(\sum_{k=n+1}^{\infty} h[k]e^{-j\omega k}\right) e^{j\omega n} \end{aligned}$$

- We can now view the output $y[n]$ as being composed of two terms:
  - The steady-state response $y_{ss}[n] = H(e^{j\omega})e^{j\omega n}$
  - The transient response $y_t[n] = -\left(\sum_{k=n+1}^{\infty} h[k]e^{-j\omega k}\right) e^{j\omega n}$
- For stable systems we expect that $\lim_{n\to\infty} y_t[n] = 0$, this is indeed the case
- Mathematically we can write that

$$|y_t[n]| = \left|\left(\sum_{k=n+1}^{\infty} h[k]e^{-j\omega k}\right) e^{j\omega n}\right| \leq \sum_{k=n+1}^{\infty} |h[k]| \leq \sum_{k=0}^{\infty} |h[k]|$$

- The term on the far right is bounded for stable LTI systems
- Suppose we consider just the real part of $x[n]$ with $\omega = 2\pi/10$ and $h[n] = 0.8^n u[n]$



- As n moves past 0 the tail of $h[n-k]$ that lies to the left of $n = 0$ quickly decays to zero, hence the transient term $y_t[n] \to 0$

2.8 Fourier Transform of Sequences

A valid representation for many sequences is

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega$$

where $X(e^{j\omega})$ given by,

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$$

is the Fourier transform (discrete-time) of $x[n]$. The synthesis formula for $x[n]$ from $X(e^{j\omega})$ is the inverse Fourier transform.

- Note that in computing $x[n]$ from $X(e^{j\omega})$ any 2π interval may be used

- Since $X(e^{j\omega})$ is periodic with period 2π we can also view the definition of $X(e^{j\omega})$ as a complex exponential Fourier series with coefficients $x[n]$. The inverse Fourier transform relation is equivalent to finding the Fourier coefficients of $X(e^{j\omega})$.

- The Fourier transform (F.T.) or spectrum of $x[n]$ can also be written as

$$\begin{aligned} X(e^{j\omega}) &= X_R(e^{j\omega}) + jX_I(e^{j\omega}) \text{ (real/imaginary)} \\ &= |X(e^{j\omega})|e^{j\angle X(e^{j\omega})} \text{ (magnitude/phase)} \end{aligned}$$

- $|X(e^{j\omega})|$ is referred to as the *magnitude* or *amplitude spectrum* of $x[n]$

- $\angle X(e^{j\omega})$ is referred to as the *phase spectrum* of $x[n]$

- Note that the functional values of the phase spectrum are unique to only within a factor of 2π

- We (the text) will denote the principle value of $\angle X(e^{j\omega})$ as ARG$[X(e^{j\omega})]$

- For LTI systems the frequency response is just the F.T. of $h[n]$, and $h[n]$ is the inverse F.T. of $H(e^{j\omega})$

Proof that $x[n]$ can be synthesized from $X(e^{j\omega})$

- Let

$$\hat{x}[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left(\sum_{m=-\infty}^{\infty} x[m]e^{-j\omega m}\right)e^{j\omega n}\,d\omega$$

and show that $\hat{x}[n] \rightarrow x[n]$

- By interchanging the order of integration and summation in the above expression (this will be justified shortly) we can write

$$\hat{x}[n] = \sum_{m=-\infty}^{\infty} x[m] \left(\frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega(n-m)} \, d\omega \right)$$

- Examining the integral we find that

$$\begin{aligned} \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega(n-m)} \, d\omega &= \frac{\sin \pi(n-m)}{\pi(n-m)} \\ &= \delta[n-m] \end{aligned}$$

- Inserting the above result back into the expression for $\hat{x}[n]$ we obtain

$$\hat{x}[n] = \sum_{m=-\infty}^{\infty} x[m]\delta[n-m] = x[n],$$

the desired result

- The interchange in order of integration and summation is possible if

$$X_M(e^{j\omega}) = \sum_{n=-M}^{M} x[n]e^{-j\omega n}$$

converges uniformly to $X(e^{j\omega})$, that is if for each $\epsilon > 0$ there exists $N(\epsilon)$ such that

$$|X(e^{j\omega}) - X_M(e^{j\omega})| \leq \epsilon \text{ for } M \geq N(\epsilon)$$

for all values of ω. Note that N does not depend on ω.

- A sufficient condition for convergence is that $x[n]$ be *absolutely summable* i.e.

$$\sum_{n=-\infty}^{\infty} |x[n]| < \infty$$

- Under these conditions the series also converges uniformly to a continuous function of ω

- Since a stable LTI system must also have $h[n]$ absolutely summable, we see that stable systems also have finite and continuous frequency responses

- For FIR systems this condition is easily satisfied

2.8.1 Mean-Square Convergence

Not all of the sequences we are interested in are absolutely summable. If we are willing to give up the condition of uniform convergence, then another sufficient condition for existence of the Fourier transform is *square summability* i.e.

$$\sum_{n=-\infty}^{\infty} |x[n]|^2 < \infty$$

- The square summability condition leads to mean-square convergence i.e.

$$\lim_{M\to\infty} \int_{-\pi}^{\pi} \underbrace{|X(e^{j\omega}) - X_M(e^{j\omega})|^2}_{\text{squared error}}\, d\omega \to 0$$

- The error energy approaches zero, but for some values of ω the error may not approach zero. Recall the Gibbs phenomenon in the study of Fourier series.

Example 2.16:

Consider an ideal lowpass filter with frequency response (fundamental interval)

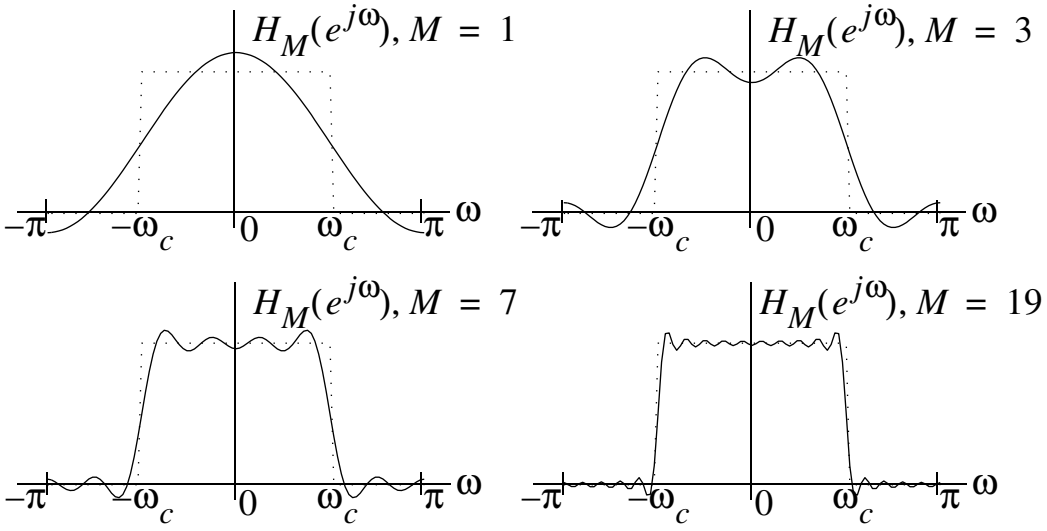
$$H_{lp}(e^{j\omega}) = \begin{cases} 1, & |\omega| < \omega_c \\ 0, & \omega_c < |\omega| \leq \pi \end{cases}$$

- To obtain $h_{lp}[n]$ we inverse transform $H_{lp}(e^{j\omega})$

$$\begin{aligned} h_{lp}[n] &= \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} e^{j\omega n}\, d\omega \\ &= \frac{\sin \omega_c n}{\pi n}, \quad -\infty < n < \infty \end{aligned}$$

- Note that $h_{lp}[n]$ is not absolutely summable (recall the harmonic series, $\sum_n (1/n)$), but is square summable

$$H_M(e^{j\omega}) = \sum_{n=-M}^{M} \frac{\sin \omega_c n}{\pi n} e^{-j\omega n}$$



Fourier transform convergence for an ideal lowpass filter

2.8.2 Special Cases

Certain sequences that we may wish to consider are neither absolutely or square summable. Sequences of this type often require impulses in the frequency domain.

Example 2.17:

- Suppose we have a sequence $x[n]$ which has a Fourier transform which is a periodic impulse train

$$X(e^{j\omega}) = \sum_{r=-\infty}^{\infty} 2\pi\delta(\omega - \omega_o + 2\pi r)$$

- To find $x[n]$ we simply evaluate

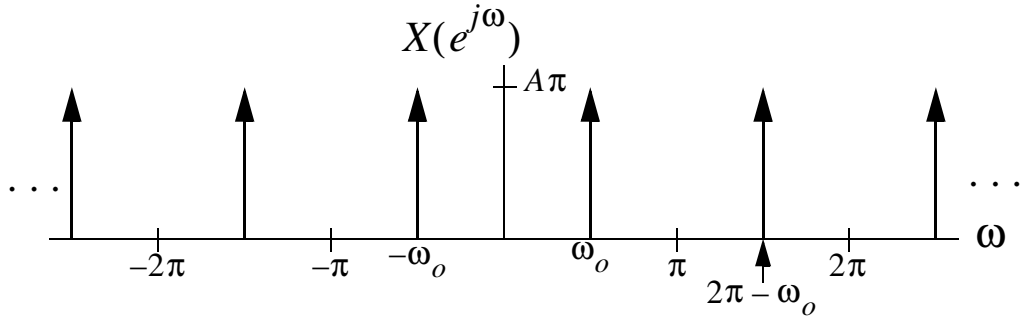
$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} 2\pi\delta(\omega - \omega_o)e^{j\omega n}\,d\omega = e^{j\omega_o n}$$

- Note that using the above result we see that if $x[n]$ is a constant sequence, $x[n] = A \ \forall n$, then

$$X(e^{j\omega}) = \sum_{r=-\infty}^{\infty} A2\pi\delta(\omega + 2\pi r)$$

- Suppose $x[n] = A\cos\omega_o n$ then,

$$X(e^{j\omega}) = \sum_{r=-\infty}^{\infty} A\pi[\delta(\omega - \omega_o + 2\pi r) + \delta(\omega + \omega_o + 2\pi r)]$$



- For a linear combination of complex exponential sequences

$$x[n] = \sum_k a_k e^{j\omega_k n}$$

we can write

$$X(e^{j\omega}) = \sum_{r=-\infty}^{\infty} \sum_k 2\pi a_k \delta(\omega - \omega_k + 2\pi r)$$

Example 2.18: Python and Fourier Transforms

- Find the Fourier transform of the following sequences using the Python function `signal.freqz()` found in the Scipy package `signal`

$$x[n] = \begin{cases} 1, & 0 \leq n \leq 6 \\ 0, & \text{otherwise} \end{cases}$$

$$h_1[n] = \begin{cases} 1, & 0 \leq n \leq 3 \\ 0, & \text{otherwise} \end{cases}$$

and

$$y[n] = x[n] * h_1[n]$$

- Direct analytical evaluation yields

$$\begin{aligned} X(e^{j\omega}) = \mathcal{F}\{x[n]\} &= \sum_{n=0}^{6} e^{-j\omega n} \\ &= \frac{1 - e^{-j\omega 7}}{1 - e^{-j\omega}} = \frac{\sin(7\omega/2)}{\sin(\omega/2)} e^{-j\,6\omega/2} \\ H_1(e^{j\omega}) &= \frac{\sin(4\omega/2)}{\sin(\omega/2)} e^{-j\,3\omega/2} \end{aligned}$$

- To obtain $Y(e^{j\omega})$ use the convolution theorem

$$\begin{aligned} Y(e^{j\omega}) = \mathcal{F}\{x[n] * h[n]\} &= X(e^{j\omega}) H_1(e^{j\omega}) \\ &= \frac{\sin(7\omega/2)\sin(4\omega/2)}{\sin^2(\omega/2)} e^{-j\,9\omega/2} \end{aligned}$$

The `signal.freqz()` Function

```
Definition:  signal.freqz(b, a=1, worN=None, whole=0, plot=None)
Docstring:
Compute the frequency response of a digital filter.

Given the numerator ‘b‘ and denominator ‘a‘ of a digital filter,
compute its frequency response::

            jw                -jw              -jmw
    jw  B(e)    b[0] + b[1]e + .... + b[m]e
 H(e) = ---- = ------------------------------------
            jw                -jw              -jnw
        A(e)    a[0] + a[1]e + .... + a[n]e

Parameters
----------
b : ndarray
    numerator of a linear filter
a : ndarray
    denominator of a linear filter
worN : None, int, array_like, optional
    If None (default), then compute at 512 frequencies equally spaced
    around the unit circle.
    If a single integer, then compute at that many frequencies.
    If an array_like, compute the response at the frequencies given (in
    radians/sample).
whole : bool, optional
    Normally, frequencies are computed from 0 to the Nyquist frequency,
    pi radians/sample (upper-half of unit-circle).  If ‘whole‘ is True,
    compute frequencies from 0 to 2*pi radians/sample.
plot : callable
    A callable that takes two arguments. If given, the return parameters
    ‘w‘ and ‘h‘ are passed to plot. Useful for plotting the frequency
    response inside ‘freqz‘.

Returns
-------
w : ndarray
    The normalized frequencies at which h was computed, in radians/sample.
h : ndarray
    The frequency response.

Notes
-----
Using Matplotlib’s "plot" function as the callable for ‘plot‘ produces
```

```
unexpected results,  this plots the real part of the complex transfer
function, not the magnitude.  Try ‘‘lambda w, h: plot(w, abs(h))‘‘.

Examples
--------
>>> from scipy import signal
>>> b = signal.firwin(80, 0.5, window=('kaiser', 8))
>>> w, h = signal.freqz(b)

>>> import matplotlib.pyplot as plt
>>> fig = plt.figure()
>>> plt.title('Digital filter frequency response')
>>> ax1 = fig.add_subplot(111)

>>> plt.plot(w, 20 * np.log10(abs(h)), 'b')
>>> plt.ylabel('Amplitude [dB]', color='b')
>>> plt.xlabel('Frequency [rad/sample]')

>>> ax2 = ax1.twinx()
>>> angles = np.unwrap(np.angle(h))
>>> plt.plot(w, angles, 'g')
>>> plt.ylabel('Angle (radians)', color='g')
>>> plt.grid()
>>> plt.axis('tight')
>>> plt.show()
```

- For $x[n]$ enter the following command (some commands omitted to save space):

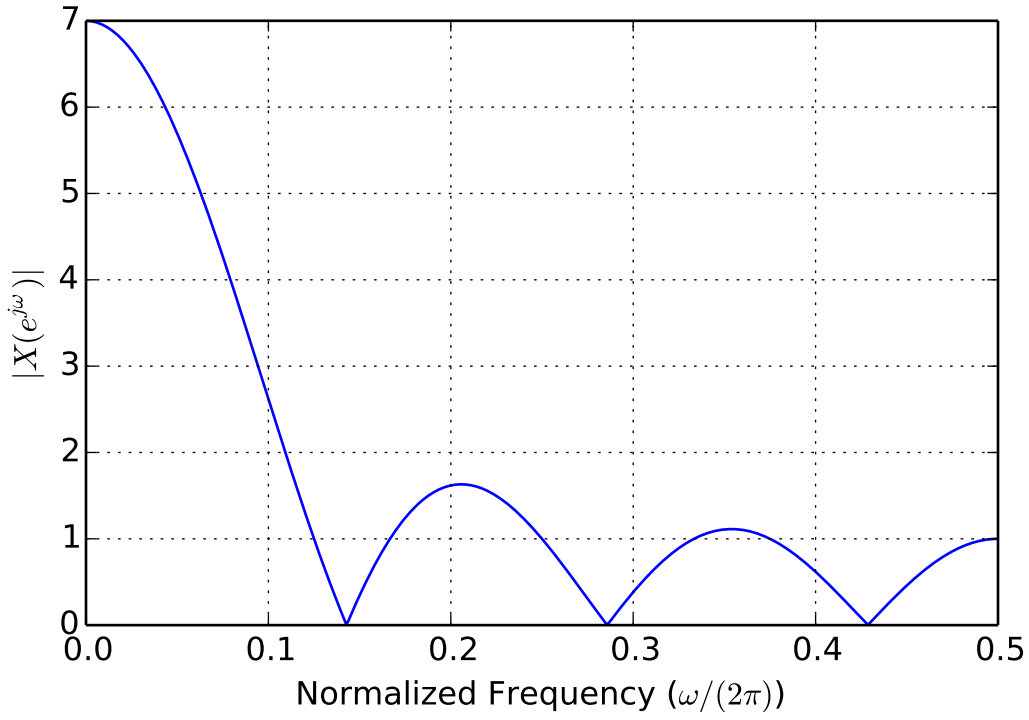
```
In [162]: # Plot 1024 points on [0,pi]
In [163]: f = arange(0,0.5,1./1024)
In [166]: w,X = signal.freqz([1,1,1,1,1,1,1],[1],2*pi*f)
In [168]: plot(f,abs(X))
In [169]: plot(f,angle(X))
In [188]: plot(f,X.real)
In [189]: plot(f,X.imag,'g--')

In [174]: w,H = signal.freqz([1,1,1,1],[1],2*pi*f)
In [176]: plot(f,abs(H))

In [197]: plot(f,abs(X*H))
```

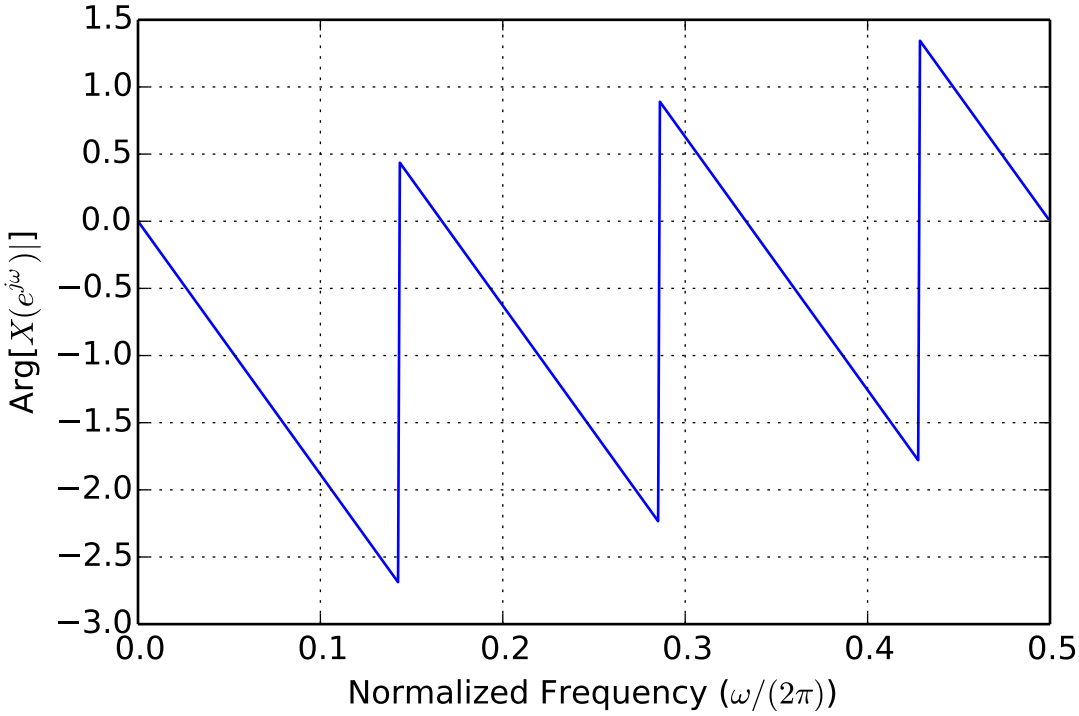
- The vector `X` holds the complex values of $X(e^{j\omega})$ evaluated at each point $\omega = 2\pi f$ held in the vector `w`
- This is accomplished by loading the input sequence $x[n]$ into `b` and setting `a=[1]` so there are no contributions from the denominator
  - The full use of `signal.freqz`, i.e., loading both the `a` and `b` vectors, will be explained shortly
- In summary to obtain $X(e^{j\omega})$ using `freqz` we have set

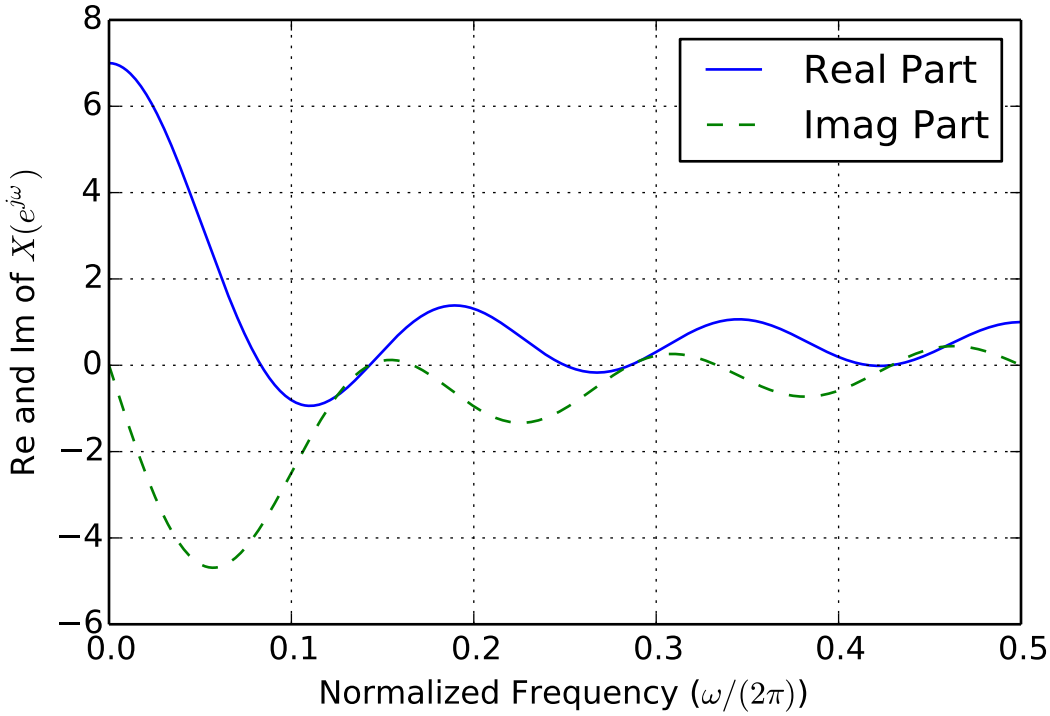
$$b = x[n], n = 0, 1, \ldots, M$$

where the interval $[0, M)$ is the support of $x[n]$

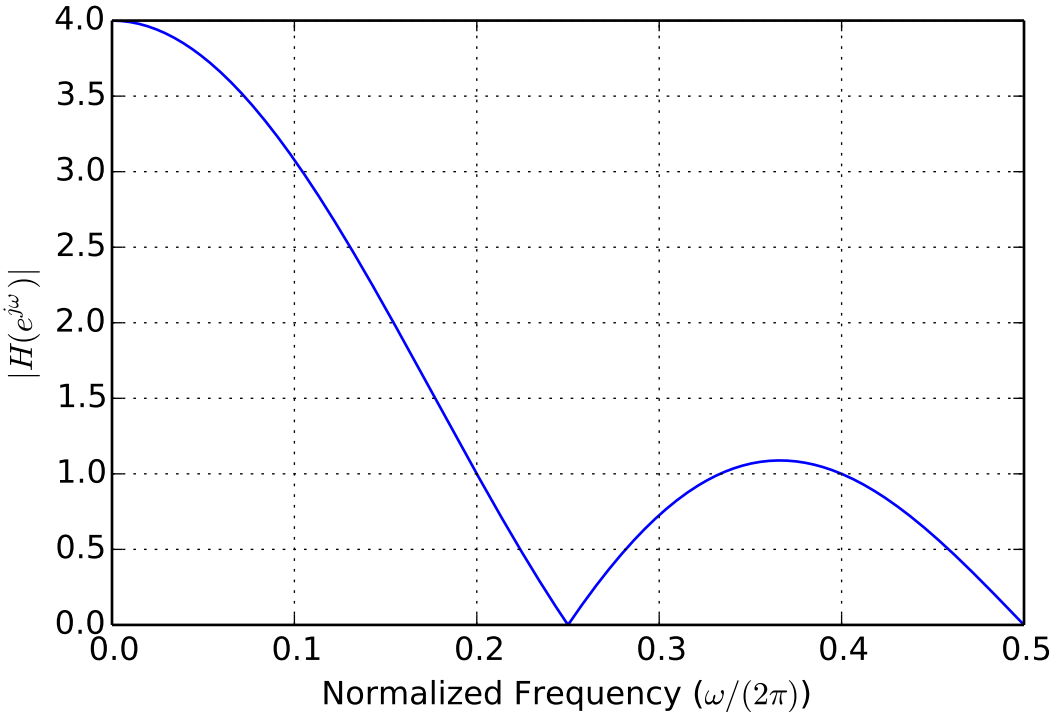
Plots of $X(e^{j\omega})$



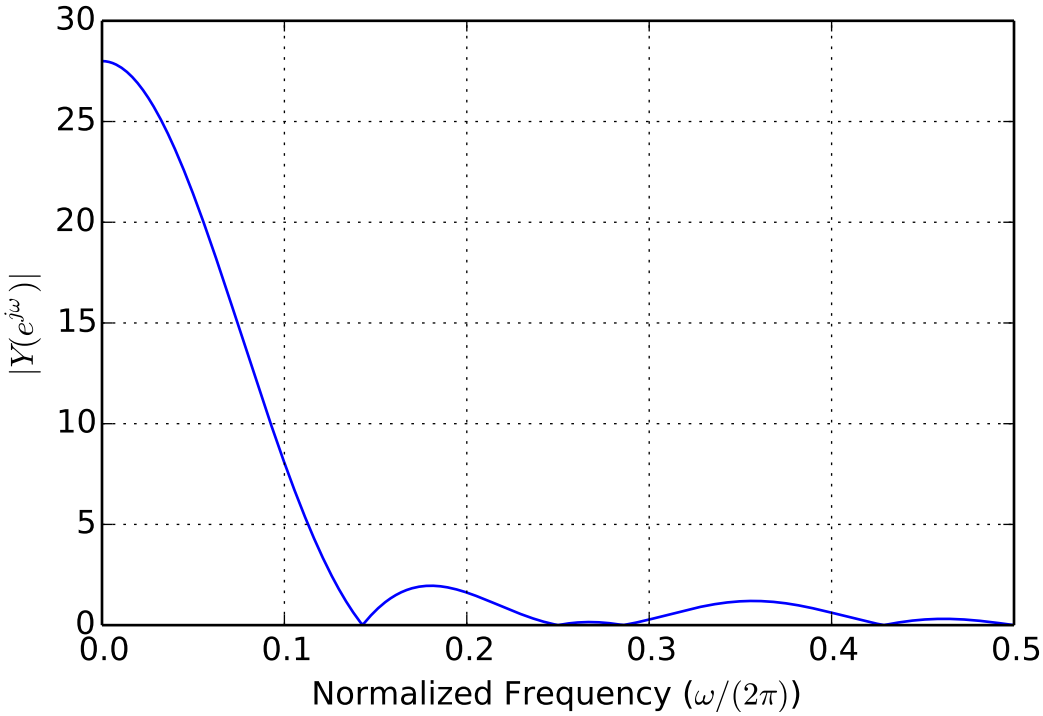

1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
Arg[$X(e^{j\omega})$|]
0.0
0.1
0.2
0.3
0.4
0.5
Normalized Frequency ($\omega/(2\pi)$)


8
6
4
2
0
−2
−4
−6
Re and Im of $X(e^{j\omega})$
Real Part
Imag Part
0.0
0.1
0.2
0.3
0.4
0.5
Normalized Frequency ($\omega/(2\pi)$)

Plot of $H(e^{j\omega})$



Plot of $Y(e^{j\omega})$



2.9 Fourier Transform Symmetry Properties and Theorems

To obtain a better understanding of the Fourier transform of sequences it is good to study some of the more important symmetry properties and useful transform theorems.

2.9.1 Symmetry Properties

The symmetry properties that we are interested in are based on the following definitions.

Sequence Decompositions

- The sequence $x_e[n]$ is *conjugate-symmetric* if $x_e[n] = x_e^*[-n]$, where * denotes conjugation
- The sequence $x_o[n]$ is *conjugate-antisymmetric* if we can write $x_o[n] = -x_o^*[-n]$
- Any sequence $x[n]$ (real or complex) can be expressed as

$$x[n] = x_e[n] + x_o[n]$$

where

$$\begin{aligned} x_e[n] &= \frac{1}{2}(x[n] + x^*[-n]) \\ x_o[n] &= \frac{1}{2}(x[n] - x^*[-n]) \end{aligned}$$

- Note if $x_e[n]$ is real then if $x_e[n] = x_e[-n]$ we say that $x_e[n]$ is an *even sequence*
- Note if $x_o[n]$ is real then if $x_o[n] = -x_o[-n]$ we say that $x_o[n]$ is an *odd sequence*

Transform Decompositions

- Any F.T. $X(e^{j\omega})$ can be written as

$$X(e^{j\omega}) = X_e(e^{j\omega}) + X_o(e^{j\omega})$$

where

$$\begin{aligned} X_e(e^{j\omega}) &= \frac{1}{2}[X(e^{j\omega}) + X^*(e^{-j\omega})] \\ X_o(e^{j\omega}) &= \frac{1}{2}[X(e^{j\omega}) - X^*(e^{-j\omega})] \end{aligned}$$

- Note that $X_e(e^{j\omega})$ is conjugate-symmetric and $X_o(e^{j\omega})$ is conjugate-antisymmetric

Special Case: *Real Signals*

- For $x[n]$ real we have the important results that:
  - $X(e^{j\omega})$ is conjugate symmetric
  - $|X(e^{j\omega})|$ is even
  - $\angle X(e^{j\omega})$ is odd
- Additionally we can show that if $x[n]$ is also even, then $X(e^{j\omega})$ is real, that is $\text{Im}[X(e^{j\omega})] = 0$
- Additionally we can show that if $x[n]$ is also odd, then $X(e^{j\omega})$ is imaginary, that is $\text{Re}[X(e^{j\omega})] = 0$

2.9.2 Symmetry Properties Summary

Sequence $x[n]$	Fourier Transform $X(e^{j\omega})$
1. $x^*[n]$	$X^*(e^{-j\omega})$
2. $x^*[-n]$	$X^*(e^{j\omega})$
3. Re$\{x[n]\}$	$X_e(e^{j\omega})$ (conjugate symmetric part of $X(e^{j\omega})$)
4. Im$\{x[n]\}$	$X_o(e^{j\omega})$ (conjugate anti - symmetric part of $X(e^{j\omega})$)
5. $x_e[n]$	$X_R(e^{j\omega})$
6. $x_o[n]$	$jX_I(e^{j\omega})$
Real $x[n]$ only	
7. Any real $x[n]$	$X(e^{j\omega}) = X^*(e^{-j\omega})$ (FT is conjuugate symmetric)
8. Any real $x[n]$	$X_R(e^{j\omega}) = X_R(e^{-j\omega})$ (real part is even)
9. Any real $x[n]$	$X_I(e^{j\omega}) = -X_I(e^{-j\omega})$ (imaginary part is odd)
10. Any real $x[n]$	$\left\|X(e^{j\omega})\right\| = \left\|X(e^{-j\omega})\right\|$ (magnitude is even)
11. Any real $x[n]$	$\angle X(e^{j\omega}) = -\angle X(e^{-j\omega})$ (phase is odd)
12. $x_e[n]$ (even part)	$X_R(e^{j\omega})$
13. $x_o[n]$ (odd part)	$jX_I(e^{j\omega})$

Example 2.19:

To provide a clear illustration of the Fourier transform symmetry properties consider the real exponential sequence $x[n] = a^n u[n]$

- By definition

$$\begin{aligned} X(e^{j\omega}) &= \sum_{n=0}^{\infty} a^n e^{-j\omega n} \\ &= \sum_{n=0}^{\infty} (ae^{-j\omega})^n \\ &= \frac{1}{1-ae^{-j\omega}} \text{ if } |ae^{-j\omega}| < 1 \text{ or } |a| < 1 \end{aligned}$$

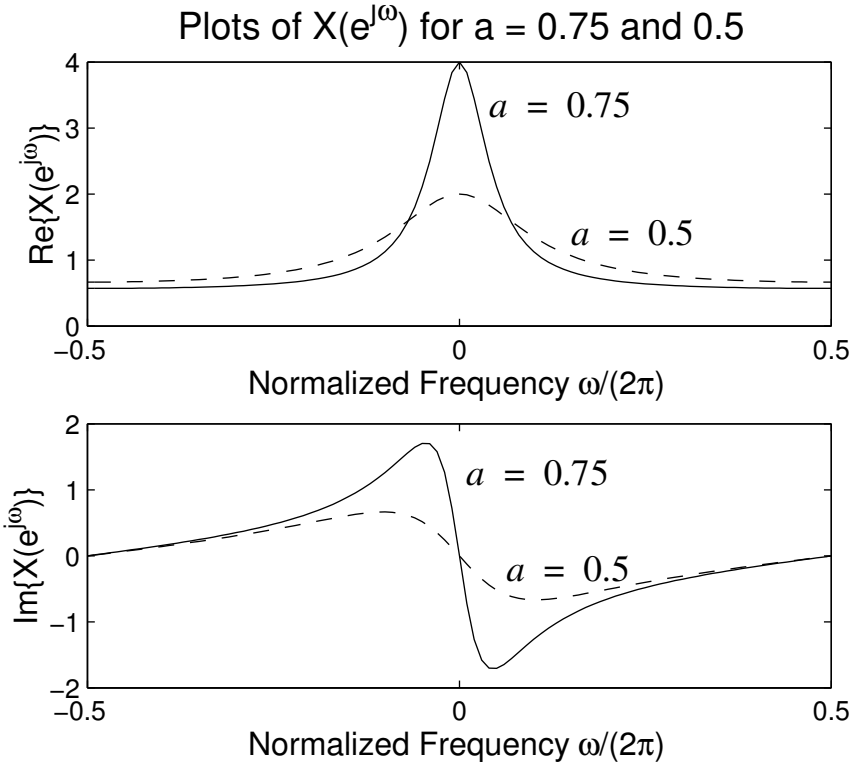
- Multiply the numerator and denominator by the conjugate of

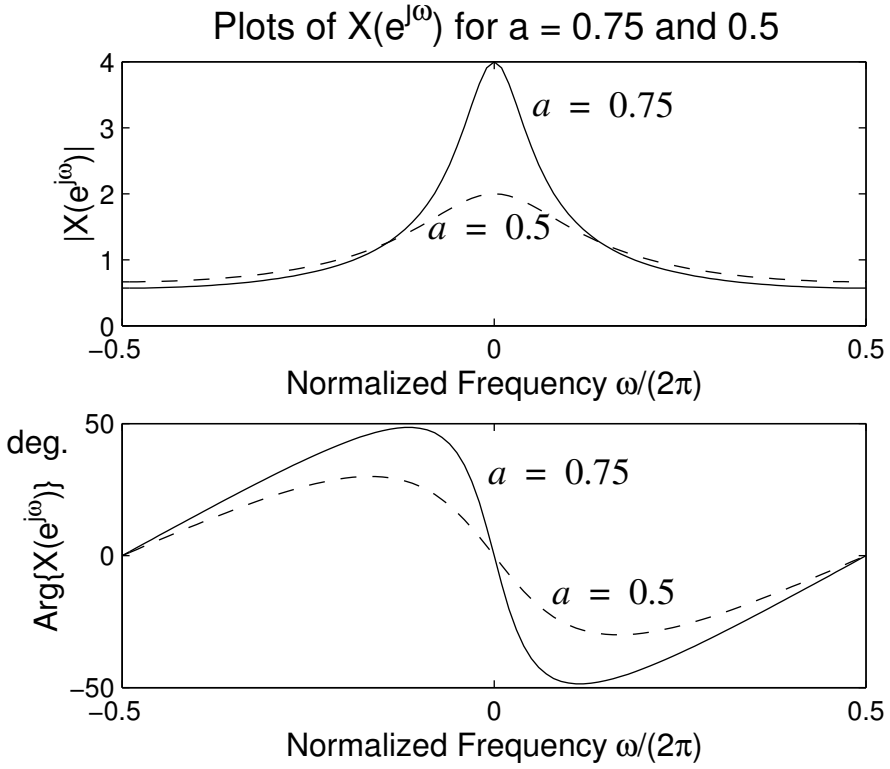
the denominator to obtain

$$X(e^{j\omega}) = \frac{1}{1-ae^{-j\omega}} \cdot \left(\frac{1-ae^{j\omega}}{1-ae^{j\omega}}\right)$$
$$= \frac{1-ae^{j\omega}}{1-a(e^{j\omega}+e^{-j\omega})+a^2}$$

$$= \underbrace{\frac{1-a\cos\omega}{1-2a\cos\omega+a^2}}_{\text{real part}} + j \underbrace{\frac{-a\sin\omega}{1-2a\cos\omega+a^2}}_{\text{imag. part}}$$

$$= \underbrace{\sqrt{\frac{1}{1+a^2-2a\cos\omega}}}_{\text{magnitude}} \exp\left[j \underbrace{\tan^{-1}\left(\frac{a\sin\omega}{1-a\cos\omega}\right)}_{\text{phase}} \right]$$





2.9.3 Transform Theorems

The Fourier transform theorems of interest for sequences are similar to the Fourier transform theorems for continuous-time signals. Several of the more important theorems will be described below. To begin with we define the following compact operator notation

$$\begin{aligned} X(e^{j\omega}) &= \mathcal{F}\{x[n]\} \text{ transform} \\ x[n] &= \mathcal{F}^{-1}\{X(e^{j\omega})\} \text{ inverse transform} \\ x[n] &\overset{\mathcal{F}}{\longleftrightarrow} X(e^{j\omega}) \text{ transform pair} \end{aligned}$$

- Linearity: If $\mathcal{F}\{x_1[n]\} = X_1(e^{j\omega})$ and $\mathcal{F}\{x_2[n]\} = X_2(e^{j\omega})$ then for arbitrary constants a and b

$$ax_1[n] + bx_2[n] \overset{\mathcal{F}}{\longleftrightarrow} aX_1(e^{j\omega}) + bX_2(e^{j\omega})$$

This result follows directly from the definition.

- Time Shift: If $\mathcal{F}\{x[n]\} = X(e^{j\omega})$, then

$$x[n - n_d] \overset{\mathcal{F}}{\longleftrightarrow} e^{-j\omega n_d} X(e^{j\omega})$$

The proof follows by direct substitution into the F.T. definition.

- Frequency Shift: If $\mathcal{F}\{x[n]\} = X(e^{j\omega})$, then

$$x[n]e^{j\omega_o n} \overset{\mathcal{F}}{\longleftrightarrow} X(e^{j(\omega-\omega_o)})$$

proof

$$\mathcal{F}\{e^{j\omega_o n}x[n]\} = \sum_{n=-\infty}^{\infty} x[n] \underbrace{e^{j\omega_o n}e^{-j\omega n}}_{e^{-j(\omega-\omega_o)n}}$$

- Parseval's Theorem: If $\mathcal{F}\{x[n]\} = X(e^{j\omega})$ then

$$E = \sum_{n=-\infty}^{\infty} |x[n]|^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |X(e^{j\omega})|^2 \, d\omega$$

where E is the normalized energy in $x[n]$ and $|X(e^{j\omega})|^2$ is the *energy density spectrum*. Note that the energy density spec-

trum is defined only for finite-energy signals. *proof*

$$\begin{aligned} E &= \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})X^*(e^{j\omega})\,d\omega \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi}\left[\sum_n x[n]e^{-j\omega n}\right]\left[\sum_k x[k]e^{-j\omega k}\right]^* d\omega \\ &= \sum_n\sum_k x[n]x^*[k]\underbrace{\frac{1}{2\pi}\int_{-\pi}^{\pi} e^{-j\omega(n-k)}\,d\omega}_{\delta[n-k]} \\ &= \sum_n x[n]x^*[n] = \sum_n |x[n]|^2 \end{aligned}$$

QED

- Convolution Theorem: If $\mathcal{F}\{x[n]\} = X(e^{j\omega})$, $\mathcal{F}\{h[n]\} = H(e^{j\omega})$, and if $y[n] = x[n] * h[n]$ then

$$x[n] * h[n] \overset{\mathcal{F}}{\longleftrightarrow} Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega})$$

proof

$$\begin{aligned} y[n] &= \sum_k x[k]h[n-k] \\ &= \sum_k x[k]\left[\frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{j\omega})e^{j\omega(n-k)}\,d\omega\right] \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi}\underbrace{\left[\sum_k x[k]e^{-j\omega k}\right]}_{X(e^{j\omega})} H(e^{j\omega})e^{j\omega n}\,d\omega \\ &= \mathcal{F}^{-1}\{X(e^{j\omega})H(e^{j\omega})\} \end{aligned}$$

QED

- Modulation or Window Theorem: If $\mathcal{F}\{x[n]\} = X(e^{j\omega})$, $\mathcal{F}\{w[n]\} = W(e^{j\omega})$, and if $y[n] = x[n]w[n]$ then

$$Y(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\theta})W(e^{j(\omega-\theta)})\,d\theta$$

Note that $Y(e^{j\omega})$ can be viewed as the *periodic convolution* of $X(e^{j\omega})$ and $W(e^{j\omega})$, which is almost the dual of the convolution theorem. With discrete-time convolution recall that the Fourier transform of the convolution sum yields the multiplication of the respective periodic Fourier transforms.

proof

$$\begin{aligned}
Y(e^{j\omega}) &= \sum_n x[n]w[n]e^{-j\omega n} \\
&= \sum_n \left[\frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\theta})e^{j\theta n}\,d\theta\right] w[n]e^{-j\omega n} \\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\theta}) \underbrace{\left[\sum_n w[n]e^{-j(\omega-\theta)n}\right]}_{W(e^{j(\omega-\theta)})} d\theta
\end{aligned}$$

QED

2.9.4 Summary of Transform Theorems

Sequence	Fourier Transform
$x[n], y[n]$	$X(e^{j\omega}), Y(e^{j\omega})$
1. $ax[n] + by[n]$	$aX(e^{j\omega}) + bY(e^{j\omega})$
2. $x[n - n_d]$ (n_d an integer)	$e^{-j\omega n_d} X(e^{j\omega})$
3. $e^{j\omega_0 n} x[n]$	$X(e^{j(\omega-\omega_0)})$
4. $x[-n]$	$X(e^{-j\omega})$ if $x[n]$ real $X^*(e^{j\omega})$
5. $nx[n]$	$j\frac{dX(e^{j\omega})}{d\omega}$
6. $x[n] * y[n]$	$X(e^{j\omega})Y(e^{j\omega})$
7. $x[n]y[n]$	$\frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\theta})Y(e^{j(\omega-\theta)})d\theta$
Parsevals Theorem	
8. $\sum_{n=-\infty}^{\infty} \lvert x[n]\rvert^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} \left\lvert X(e^{j\omega})\right\rvert^2 d\omega$	
9. $\sum_{n=-\infty}^{\infty} x[n]y^*[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})Y^*(e^{j\omega})d\omega$	

2.9.5 Useful Transform Pairs

Sequence	Fourier Transform
1. $\delta[n]$	1
2. $\delta[n-n_0]$	$e^{-j\omega n_0}$
3. $1 \ \ (-\infty < n < \infty)$	$\sum_{k=-\infty}^{\infty} 2\pi\delta(\omega+2\pi k)$
4. $a^n u[n] \ \ (\lvert a\rvert < 1)$	$\frac{1}{1-ae^{-j\omega}}$
5. $u[n]$	$\frac{1}{1-e^{-j\omega}} + \sum_{k=-\infty}^{\infty} \pi\delta(\omega+2\pi k)$
6. $(n+1)a^n u[n]$	$\frac{1}{(1-ae^{-j\omega})^2}$
7. $\frac{r^n \sin\omega_p(n+1)}{\sin\omega_p} u[n] \ (\lvert r\rvert < 1)$	$\frac{1}{1-2r\cos\omega_p e^{-j\omega}+r^2e^{-j2\omega}}$
8. $\frac{\sin\omega_c n}{\pi n}$	$X(e^{j\omega}) = \begin{cases} 1, & \lvert\omega\rvert < \omega_c \\ 0, & \omega_c < \lvert\omega\rvert < \pi \end{cases}$
9. $x[n] = \begin{cases} 1, & 0 \le n \le M \\ 0, & \text{otherwise} \end{cases}$	$\frac{\sin[\omega(M+1)/2]}{\sin(\omega/2)} e^{-j\omega M/2}$
10. $e^{j\omega_0 n}$	$\sum_{k=-\infty}^{\infty} 2\pi\delta(\omega-\omega_0+2\pi k)$
11. $\cos(\omega_0 n+\phi)$	$\pi \sum_{k=-\infty}^{\infty} \left[e^{j\phi}\delta(\omega-\omega_0+2\pi k) + e^{-j\phi}\delta(\omega+\omega_0+2\pi k)\right]$

Example 2.20:

Suppose $x[n] = a^n u[n-5]$. Find $\mathcal{F}\{x[n]\} = X(e^{j\omega})$.

- The brute-force approach will of course work, but instead consider rewriting $x[n]$ as

$$x[n] = a^5(a^{n-5}u[n-5])$$

- Now apply the Time Shift Theorem (#2 in the table) and we have

$$X(e^{j\omega}) = a^5 e^{-j5\omega}\mathcal{F}\{a^n u[n]\}$$

- As the final step apply transform pair #4 assuming $|a| < 1$

$$X(e^{j\omega}) = \frac{a^5 e^{-j5\omega}}{1 - ae^{-j\omega}}$$

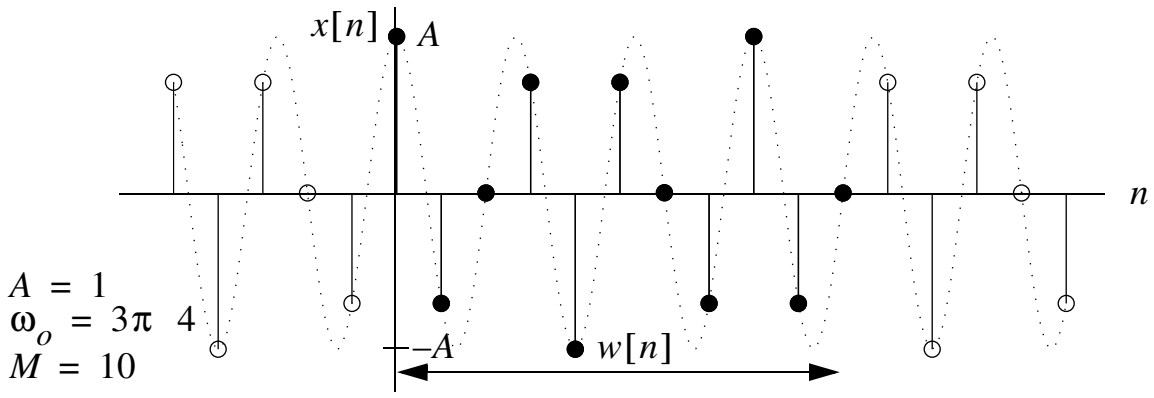
Example 2.21: Windowed Sinusoid Spectrum

Consider

$$x[n] = A\cos(\omega_o n)w[n], \;\; 0 < \omega_o < \pi$$

where

$$w[n] = \begin{cases} 1, & 0 \le n \le M \\ 0, & \text{otherwise} \end{cases}$$



The windowed sequence $x[n]$

- From the transform pair table entry #11 with $\phi = 0$

$$\mathcal{F}\{\cos(\omega_o n)\} = A\pi \sum_{k=-\infty}^{\infty} [\delta(\omega - \omega_o + 2\pi k) + \delta(\omega + \omega_o + 2\pi k)]$$

- From the transform pair table entry #9

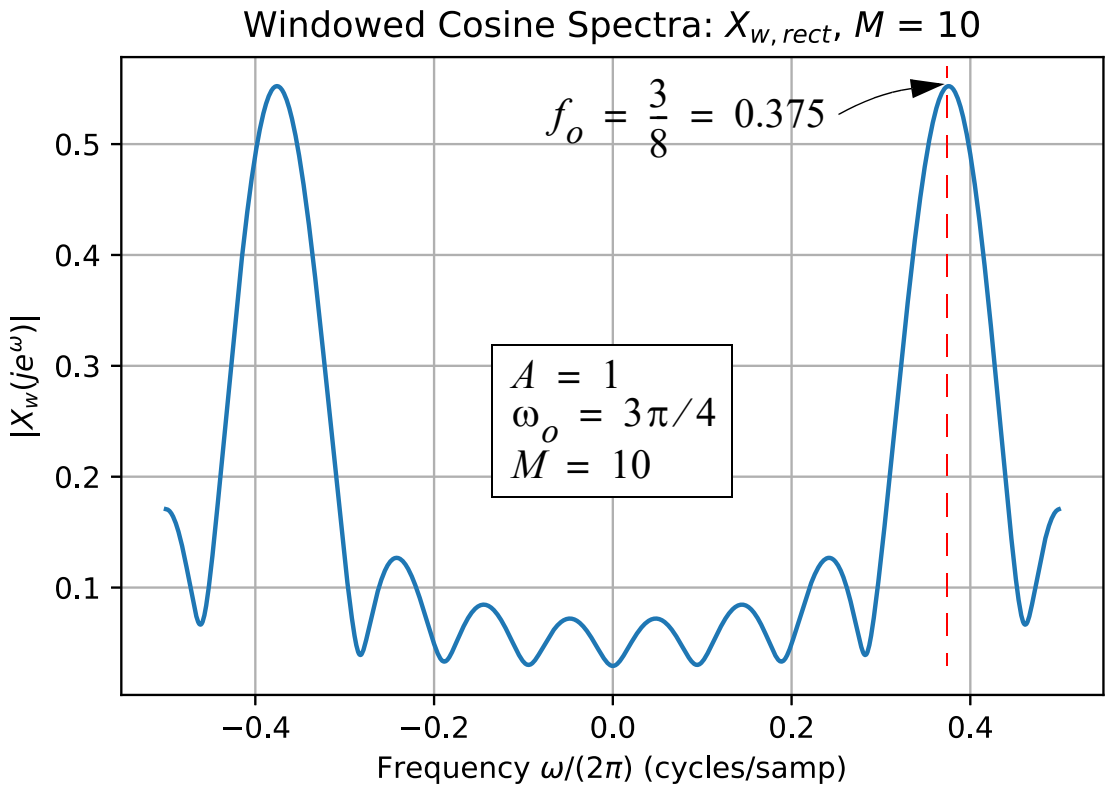
$$\mathcal{F}\{w[n]\} = \frac{\sin[\omega(M+1)/2]}{\sin(\omega/2)} e^{-j\omega M/2}$$

- To obtain $X(e^{j\omega})$ we make use of the Windowing Theorem

$$\begin{aligned} X(e^{j\omega}) &= \frac{1}{2\pi}\int_{-\pi}^{\pi} A\pi \sum_{k=-\infty}^{\infty} [\delta(\theta-\omega_o+2\pi k) \\ &\quad + \delta(\theta+\omega_o+2\pi k)] \\ &\quad \cdot \frac{\sin[(\omega-\theta)(M+1)/2]}{\sin[(\omega-\theta)/2]} e^{-j(\omega-\theta)M/2}\, d\theta \\ &= \frac{A}{2}\left[\frac{\sin[(\omega-\omega_o)(M+1)/2]}{\sin[(\omega-\omega_o)/2]} e^{-j(\omega-\omega_o)M/2}\right. \\ &\quad \left. + \frac{\sin[(\omega+\omega_o)(M+1)/2]}{\sin[(\omega+\omega_o)/2]} e^{-j(\omega+\omega_o)M/2}\right] \end{aligned}$$

- The plot of $|X(e^{j\omega})|$ below clearly shows the spectral *leakage (spreading)* that results from a length $M = 10$ window

```
M = 10
w0 = 3*pi/4
n = arange(M)
x = cos(w0*n)
w_rect = ones(M) # also signal.boxcar()
x_w_rect = x*w_rect
w,X_w_rect = signal.freqz(x_w_rect,[1],2*pi*f)
plot(f,abs(X_w_rect)/sum(w_rect)) #normalize
xlabel(r'Frequency $\omega/(2\pi)$ (cycles/samp)')
ylabel(r'$|X_w(je^{\omega})|$')
title(r'Window Function Spectra: $X_{w,rect}$, $M$ = %d'% M)
grid();
```



Example 2.22:

We can also use the Fourier transform to determine both the impulse response and frequency response, directly from a difference equation representation. Consider a system with LCCDE

$$y[n] - \frac{1}{2}y[n-1] = x[n] - \frac{1}{4}x[n-1]$$

Find $H(e^{j\omega})$ and $h[n]$.

- To find $H(e^{j\omega})$ we Fourier transform both sides of the difference equation using the Time-Delay Theorem

$$Y(e^{j\omega}) - \frac{1}{2}e^{-j\omega}Y(e^{j\omega}) = X(e^{j\omega}) - \frac{1}{4}e^{-j\omega}X(e^{j\omega})$$

- Now since $y[n] = x[n] * h[n]$ we know from the convolution theorem that $Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega})$, thus

$$H(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})} = \frac{1 - \frac{1}{4}e^{-j\omega}}{1 - \frac{1}{2}e^{-j\omega}}$$

- To obtain $h[n]$ we simply inverse transform $H(e^{j\omega})$

- To begin with we expand $H(e^{j\omega})$ as follows

$$H(e^{j\omega}) = \frac{1}{1 - \frac{1}{2}e^{-j\omega}} - \frac{\frac{1}{4}e^{-j\omega}}{1 - \frac{1}{2}e^{-j\omega}}$$

- Using the time-shift theorem and transform pair #4

$$a^n u[n] \overset{\mathcal{F}}{\longleftrightarrow} \frac{1}{1 - ae^{-j\omega}}$$

we obtain

$$h[n] = \left(\frac{1}{2}\right)^n u[n] - \left(\frac{1}{4}\right)\left(\frac{1}{2}\right)^{n-1} u[n-1]$$

2.10 The DTFT of Special Sequences

The Fourier transform of some special sequences will be presented along with simple proofs. The sequences of interest here are special in the sense that the transform will contain frequency domain delta functions, hence they do not converge uniformly.

2.10.1 Zero Padded Impulse Train

The sequence of interest is

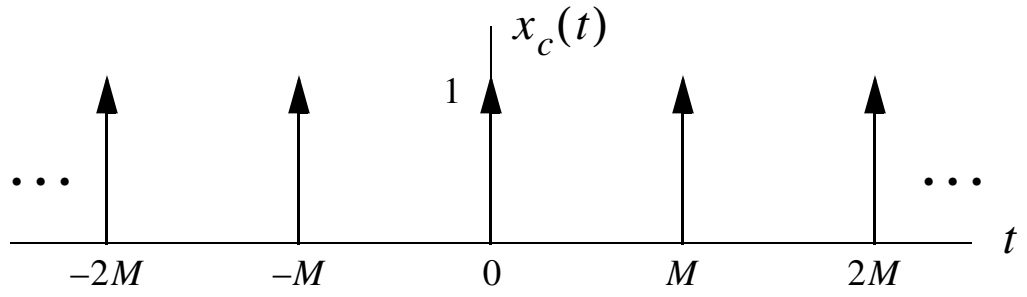
$$x[n] = \sum_{k=-\infty}^{\infty} \delta[n - M\,k], \quad M \text{ an integer}$$

- By definition the Fourier transform of this sequence is

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \left\{ \sum_{k=-\infty}^{\infty} \delta[n - Mk] \right\} e^{-j\omega n} = \sum_{k=-\infty}^{\infty} e^{-j\omega Mk}$$

- The above can be evaluated by using the Poisson sum formula
- An alternate solution can be obtained by first considering the continuous-time signal $x_c(t)$ given by

$$x_c(t) = \sum_{k=-\infty}^{\infty} \delta(t - MTk)\Big|_{T=1} = \sum_{k=-\infty}^{\infty} \delta(t - Mk)$$



A plot of $x_c(t)$ with $T = 1$.

- The continuous-time Fourier transform of $x_c(t)$ is just

$$X_c(j\,\Omega) = \frac{2\pi}{M} \sum_{n=-\infty}^{\infty} \delta\left(\Omega - \frac{2\pi}{M} n\right)$$

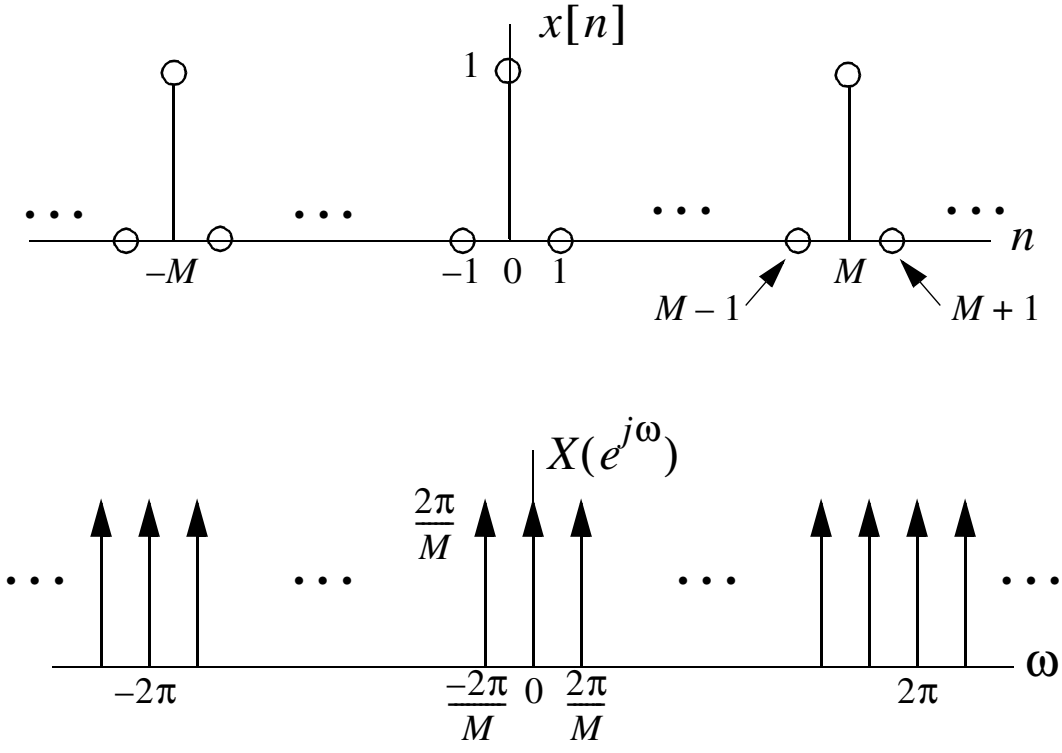
- If $x_c(t)$ is sampled with sampling period $T = 1$, then the sequence $x[n]$ is the sequence we are studying

- We know that the Fourier transform of $x_c(t)$ and $x[n]$ are related by

$$X(e^{j\omega}) = X_c(j\Omega)|_{\Omega=\omega/T} \overset{\text{here}}{=} X_c(j\omega)$$

- Thus it follows that

$$\sum_{k=-\infty}^{\infty} \delta[n - M\,k] \quad \overset{\mathcal{F}}{\Leftrightarrow} \quad \frac{2\pi}{M} \sum_{n=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi}{M} n\right)$$



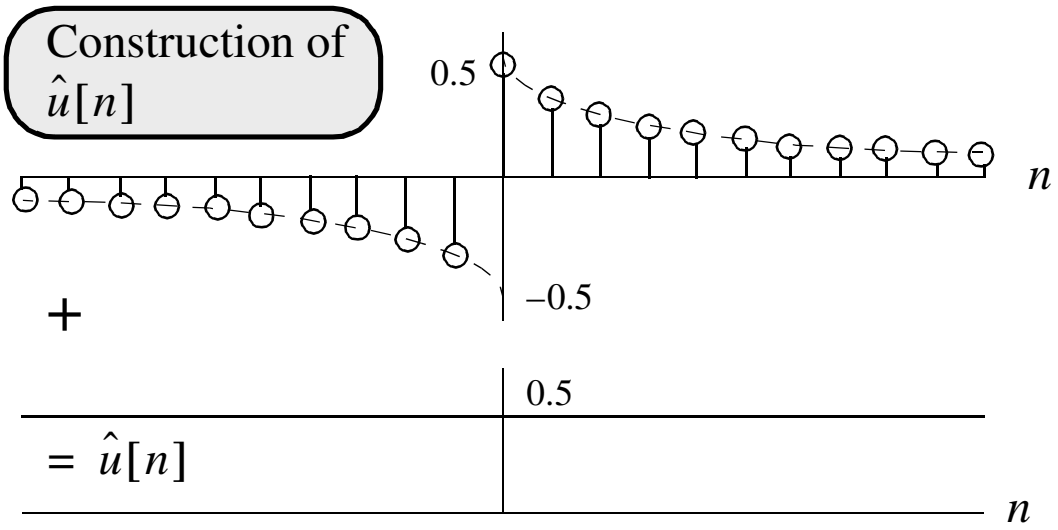
Zero padded impulse train

2.10.2 The Unit Step Sequence

The sequence of interest is the unit step function $u[n]$.

- Direct application of the definition is not very satisfying
- The proper approach is to consider the Fourier transform in the limit of a test function such as

$$\hat{u}[n] = \frac{1}{2}\left[1 + a^n u[n] - a^{-n} u[-1-n]\right]$$



The composition of the test function $\hat{u}[n]$.

- Note that

$$\lim_{a \to 1} \hat{u}[n] = \frac{1}{2} + \frac{1}{2}\text{sgn}[n] = u[n]$$

where

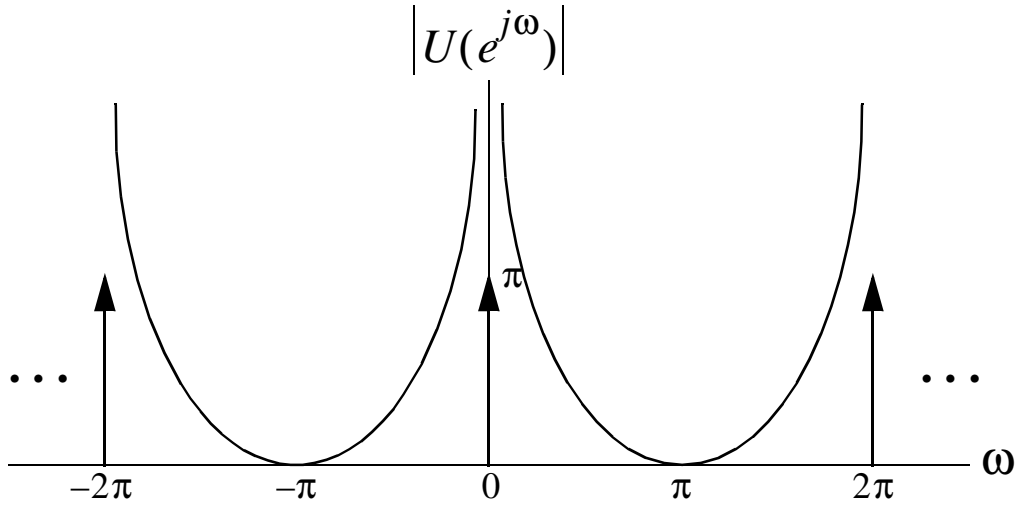
$$\text{sgn}[n] = \begin{cases} 1, & n \geq 0 \\ -1, & n < 0 \end{cases}$$

- We can now take the Fourier transform of $\hat{u}[n]$ term-by-term to obtain

$$\hat{U}(e^{j\omega}) = \pi \sum_{k=-\infty}^{\infty} \delta[\omega - 2\pi k]$$
$$+ \left[\frac{1}{1 - ae^{-j\omega}} - \underbrace{\sum_{n=-\infty}^{-1} a^{-n} e^{-j\omega n}}_{a/(a-e^{-j\omega})} \right]$$
$$= \pi \sum_{k=-\infty}^{\infty} \delta[\omega - 2\pi k] + \frac{1}{2}\left[\frac{1}{1 - ae^{-j\omega}} + \frac{a}{a - e^{-j\omega}} \right]$$

- Finally using the Fourier transform in the limit we obtain

$$U(e^{j\omega}) = \lim_{a \to 1} \hat{U}(e^{j\omega})$$
$$= \frac{1}{1 - e^{-j\omega}} + \pi \sum_{k=-\infty}^{\infty} \delta[\omega - 2\pi k]$$



The magnitude spectrum of u[n].

- In summary

$$u[n] \overset{\mathcal{F}}{\Leftrightarrow} \frac{1}{1-e^{-j\omega}} + \pi \sum_{k=-\infty}^{\infty} \delta[\omega - 2\pi k]$$