5650 Fall 2025_Set 1_Demo

September 3, 2025

Contents

[1]:
```
# %pylab inline
from numpy import *
from matplotlib.pyplot import *
%matplotlib inline
import sk_dsp_comm.sigsys as ss
import scipy.signal as signal
from IPython.display import Audio, display
from IPython.display import Image, SVG

%config InlineBackend.figure_formats=['svg'] # SVG inline viewing
```

Section 1 of a New Notebook

Consider a simple model for the signal

$$x[n] = \delta[n-2]$$

or by another means

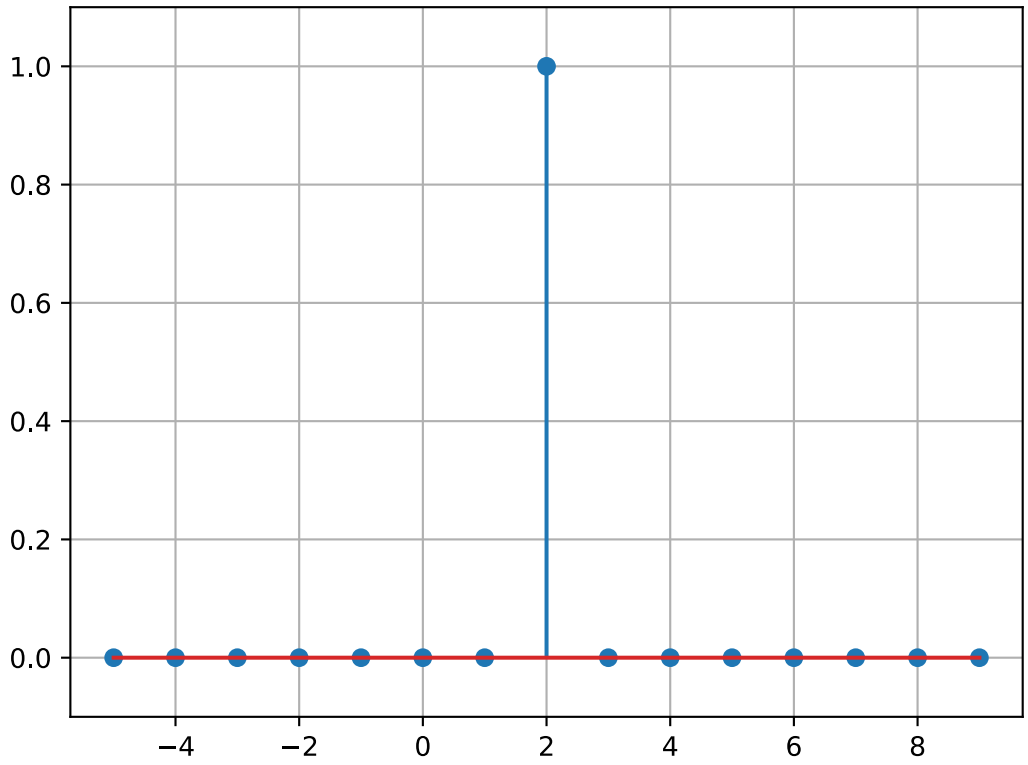
$$x[n] = u[n-2] - u[n-3] \overset{\text{also}}{=} \delta[n-2]$$

Too create a very simple Python or Julia model we can use signal primitives found in the module `scikit-dsp-comm.sigsys`. In an earlier code cell the module was loaded into the Python workspace using

`import sk_dsp_comm.sigsys as ss`

An *alias* of `ss` is created to be more convenient for typing.

[5]:
```
# A code cell with code
n = arange(-5,10)
# x = ss.dstep(n-2) - ss.dstep(n-3)
x = ss.dimpulse(n-2)
stem(n,x)
ylim([-.1,1.1])
grid();
```



Worked Problem

Text 2.3

Consider

$$y[n] = x[n] * h[n] \tag{1}$$

where

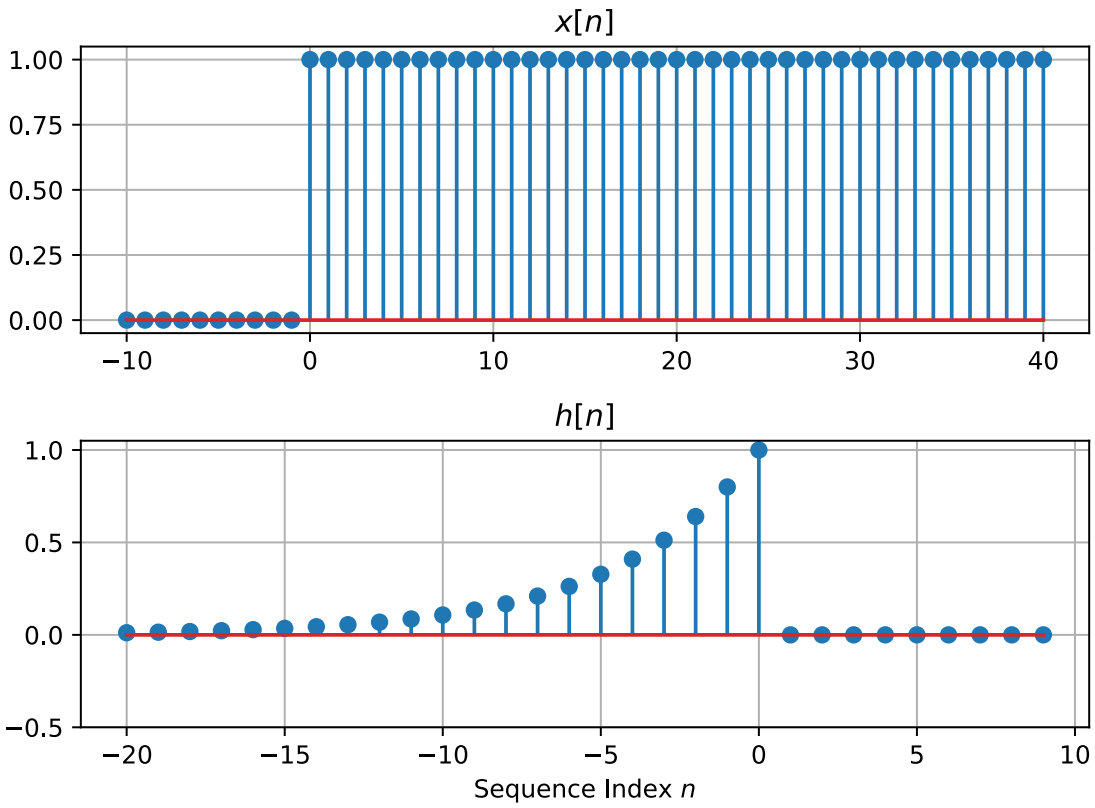
$$x[n] = u[n] \tag{2}$$
$$h[n] = a^{-n}u[-n], \text{ with } a = 0.8 \tag{3}$$

Create the Signals Using Functions from `sigsys.py` (aliased as `ss`)

[6]:
```
a = 0.8
nx = arange(-10,40+1)
x = ss.dstep(nx)
nh = arange(-20,10)
h = a**(-nh)*ss.dstep(-nh)
```

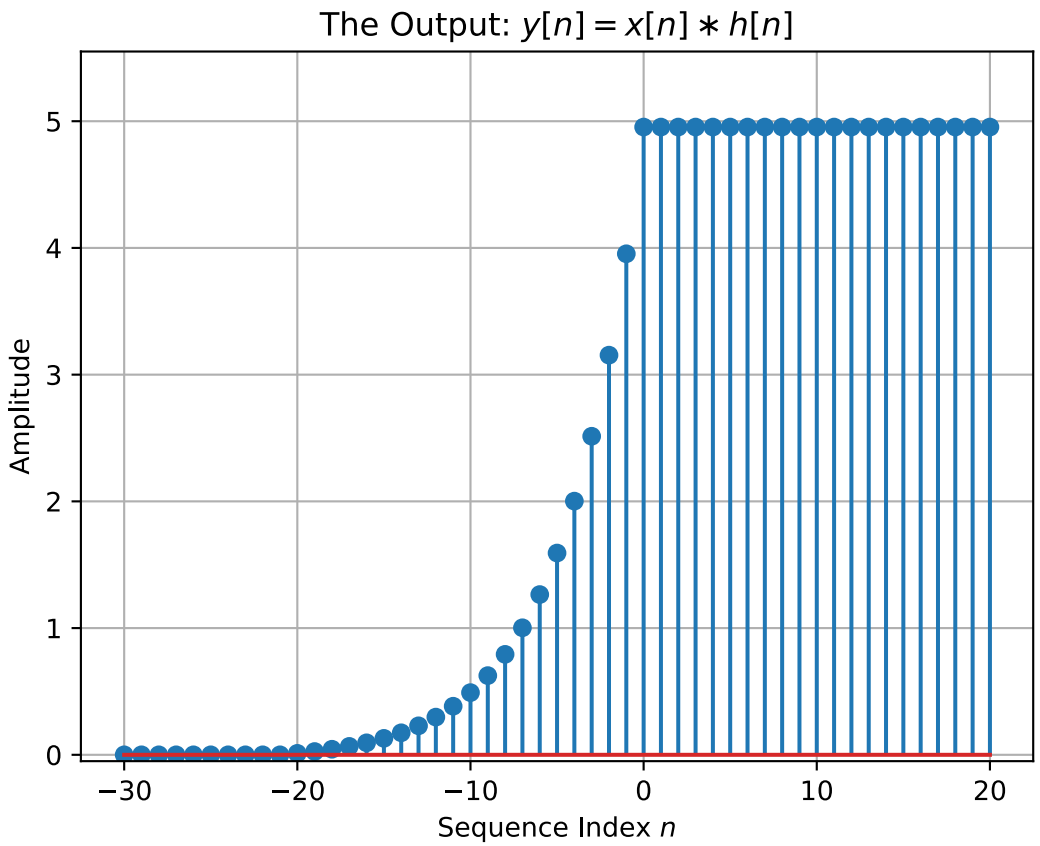
[8]:
```
subplot(211)
stem(nx,x)
```

```
ylim([-.05,1.05])
title(r'$x[n]$')
grid();
subplot(212)
stem(nh,h)
ylim([-0.5,1.05])
title(r'$h[n]$')
xlabel(r'Sequence Index $n$')
grid();
tight_layout()
```



[10]:
```
y, ny = ss.conv_sum(x,nx,h,nh,extent=('r','f'))
```

[11]:
```
stem(ny,y)
#xlim([-20,10])
ylim([-0.05,5.55])
title(r'The Output: $y[n] = x[n] \ast h[n]$')
ylabel(r'Amplitude')
xlabel(r'Sequence Index $n$')
grid();
```



Text 2.30

In this problem you consider a system that is a cascade of the two LTI systems:

$$h_1[n] = u[n] - u[n-4] \tag{4}$$
$$h_2[n] = u[n+3] - u[n-1] \tag{5}$$

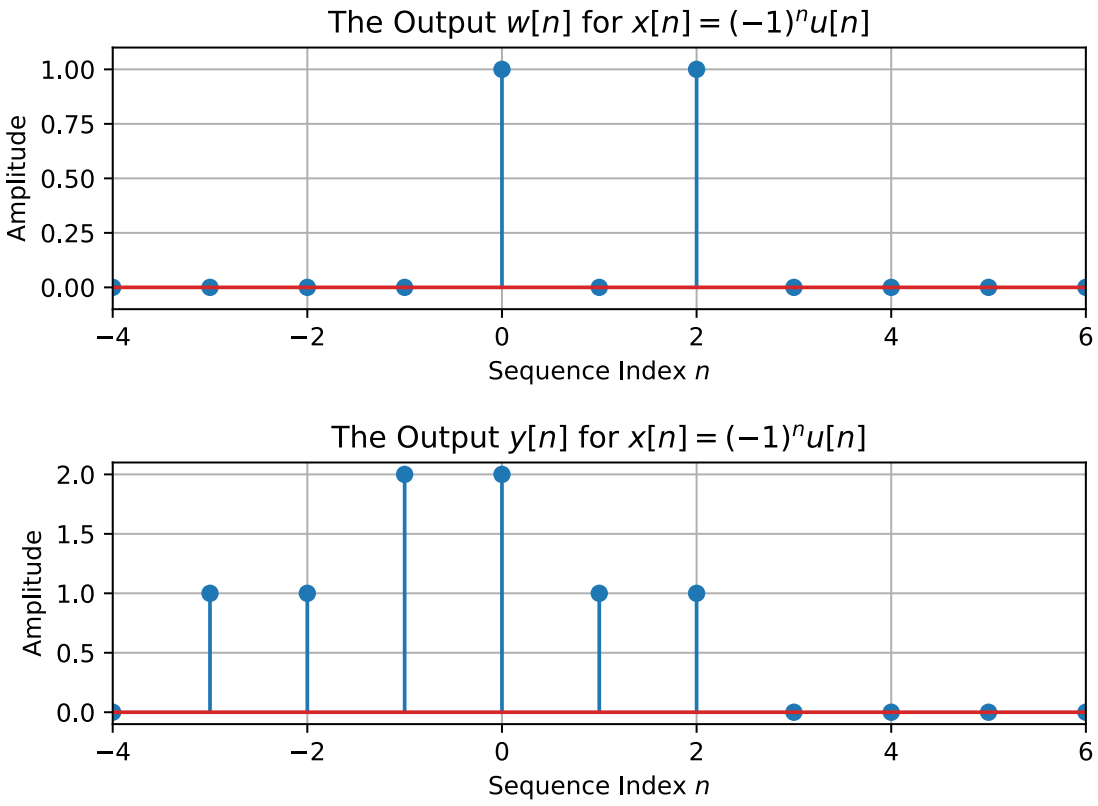
This time the simulation will be done using the `scipy.signal` difference equation engine `y = lfilter(b,a,x)`. Since $h_2[n]$ is non-causal (why?) we will have to adjust the time axis to work the problem assuming causal filters, but then shifting the axis to the left by 3 when considering outputs $w[n]$ and $y[n]$. So, in modeling the two filter impulse responses we start by shifting $h_2[n]$ to the right by 3 samples:

[12]:
```
# Causal impulse responses start at n = 0
h1 = [1, 1, 1, 1]
h2 = [1,1,1,1]
```

Part a

Here we let $x[n] = (-1)^n u[n]$

[13]:
```
# The input does not shift, just the system impulse responses
n = arange(-4,13+1)
xa = (-1.0)**n * ss.dstep(n)
w = signal.lfilter(h1,1,xa)
y = signal.lfilter(h2,1,w)
# Plot the outputs but shift the time axis back to the left by 3 after h2
subplot(211)
stem(n,w)
axis([-4,6,-0.1,1.1])
title(r'The Output $w[n]$ for $x[n] = (-1)^n u[n]$')
ylabel(r'Amplitude')
xlabel(r'Sequence Index $n$')
grid();
subplot(212)
axis([-4,6,-0.1,2.1])
stem(n-3,y)
title(r'The Output $y[n]$ for $x[n] = (-1)^n u[n]$')
ylabel(r'Amplitude')
xlabel(r'Sequence Index $n$')
grid();
tight_layout()
```



Part b

Carry on with $x[n] = \delta[n]$ and find the overall impulse response.

[ ]:

Part c & d

Carry on with $x[n] = 2\delta[n] + 4\delta[n-4] - 2\delta[n-12]$ and plot $w[n]$ and $y[n]$.

[ ]: