Lab 3 Report

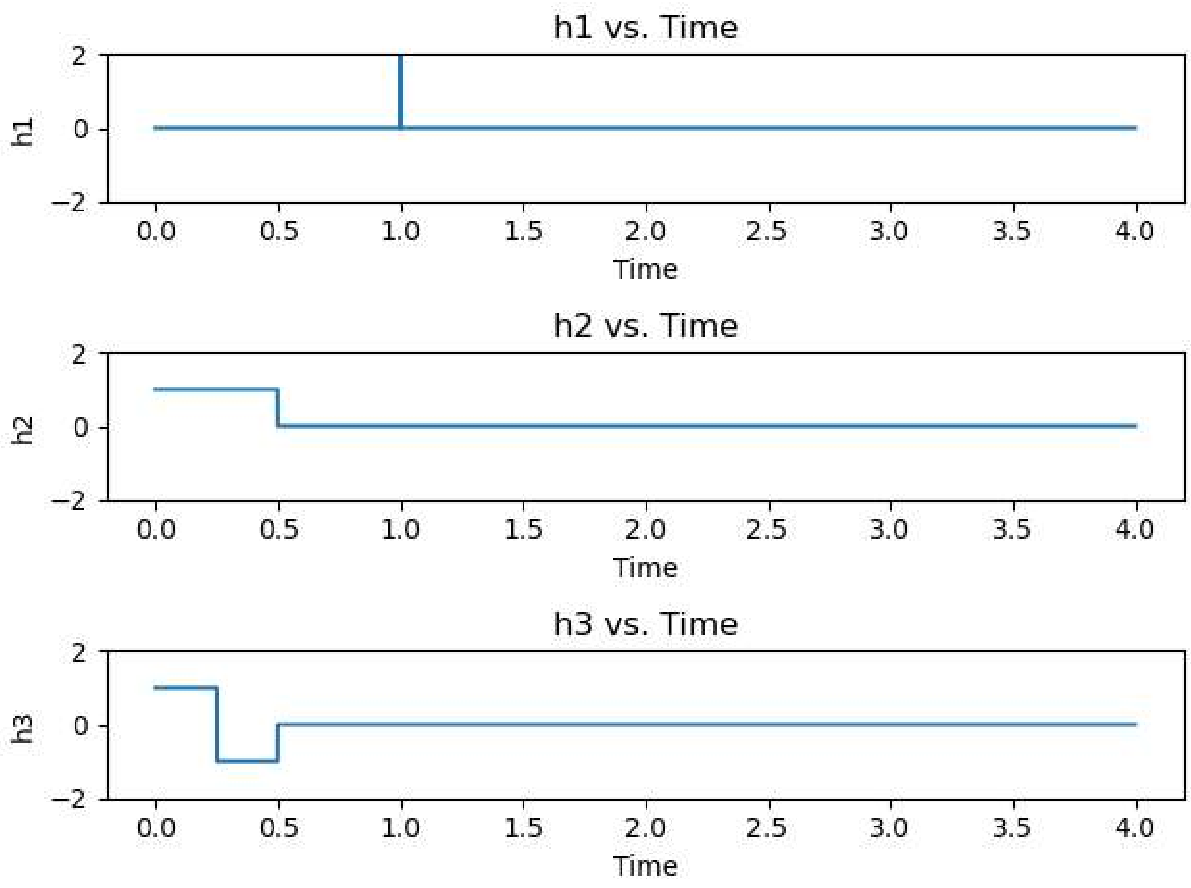
Paul Chung, Karan Singh, Samantha Pierce

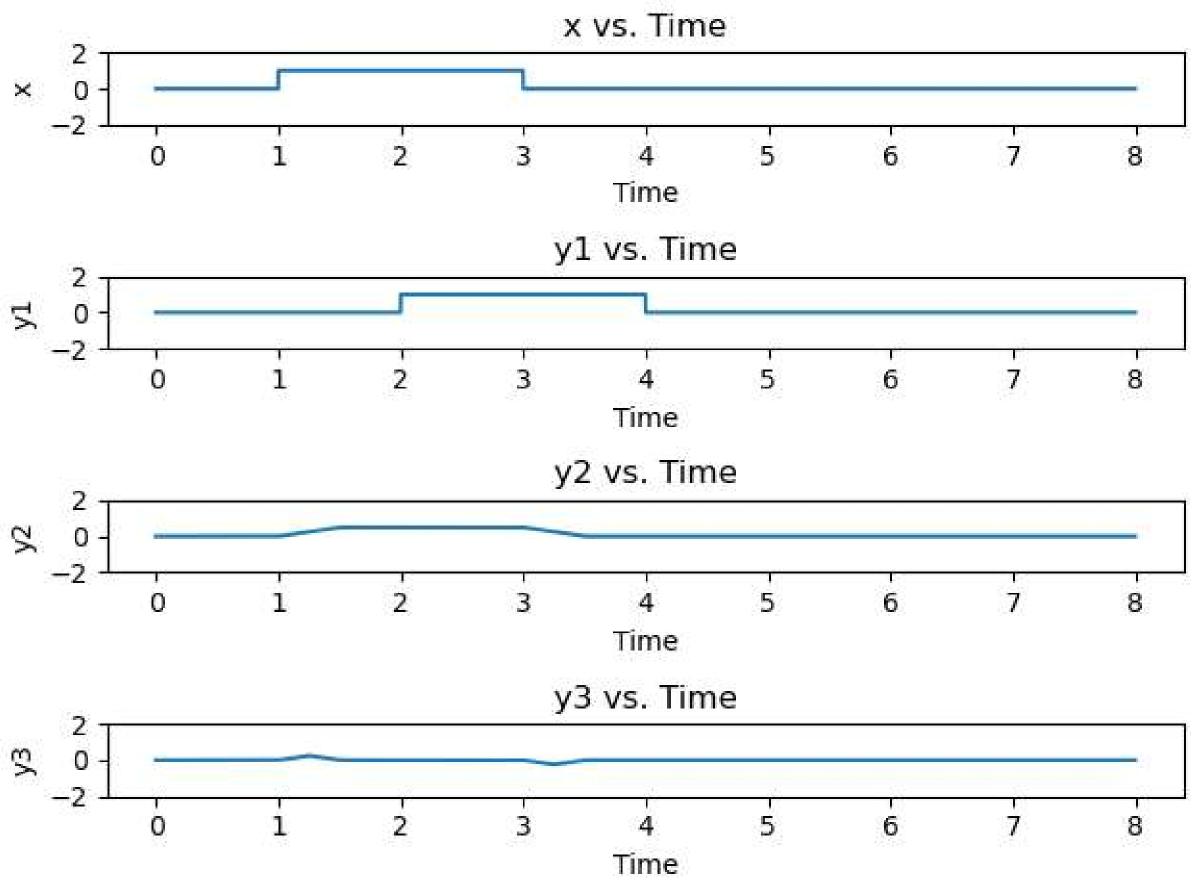
In [14]:
```
%matplotlib notebook
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from scipy.io import wavfile as wav
from scipy import signal as sig
import simpleaudio as sa
import decimal
from decimal import *
```

In [3]:
```
# Assignment 1

# Part A

end_time=4 # end time in the time interval [0,end_time]
fs=1000 # sampling frequency
ts=1/fs #sampling time

# epsilon must always be less then 1/fs!! Otherwise, the time vector will not be in ran
epsilon=1/(fs + 1)

def u(t):
    return 1.0*(t>=0)

def impulse(timestamp, t_vector, fs):
    impulse_n = int(timestamp*fs)
    h = np.zeros(len(t_vector))
    h[impulse_n] = fs
    return h

t = np.arange(0, 4 + epsilon, 1/fs)

# A.a

# create u(t-1)
u_1 = u(t-1)

# create u(t-3)
u_2 = u(t-3)

# create x(t)=u(t-1)-u(t-3)
x = u_1 - u_2

# A.b
# create h1(t)
# We can replace the first parameter with the time at which we want unit impulse
h1 = impulse(1, t, fs)

# A.c

# create u(t)
u_3 = u(t)

# create u(t-0.5)
```

```
u_4 = u(t-0.5)

# create h2(t)=u(t)-u(t-0.5)
h2 = u_3 - u_4

# A.d
h3 = u(t) - 2*u(t-0.25) + u(t-0.5)

# Part B

#th=# create the time vector
th = np.arange(0, 4 + epsilon, 1/fs)

#Plot the responses required
fig = plt.figure(1)

# B.a
plt.subplot(3, 1, 1)
plt.ylim(-2, 2)
plt.xlabel('Time')
plt.ylabel('h1')
plt.title('h1 vs. Time')
plt.plot(th, h1)

# B.b
plt.subplot(3, 1, 2)
plt.ylim(-2, 2)
plt.xlabel('Time')
plt.ylabel('h2')
plt.title('h2 vs. Time')
plt.plot(th, h2)

# B.c
plt.subplot(3, 1, 3)
plt.ylim(-2, 2)
plt.xlabel('Time')
plt.ylabel('h3')
plt.title('h3 vs. Time')
plt.plot(th, h3)

plt.tight_layout()

# Part C
y1 = np.convolve(x,h1)/fs
y2 = np.convolve(x,h2)/fs
y3 = np.convolve(x,h3)/fs

# Part D

ty = np.arange(0, 8 + epsilon, 1/fs)

#Plot the responses required
fig = plt.figure(2)

# D.a
plt.subplot(4, 1, 1)
plt.ylim(-2, 2)
plt.xlabel('Time')
plt.ylabel('x')
plt.title('x vs. Time')
x1 = np.concatenate([x, np.zeros(4000)])
```

```
plt.plot(ty, x1)

# D.b
plt.subplot(4, 1, 2)
plt.ylim(-2, 2)
plt.xlabel('Time')
plt.ylabel('y1')
plt.title('y1 vs. Time')
plt.plot(ty, y1)

# D.c
plt.subplot(4, 1, 3)
plt.ylim(-2, 2)
plt.xlabel('Time')
plt.ylabel('y2')
plt.title('y2 vs. Time')
plt.plot(ty, y2)

# D.d
plt.subplot(4, 1, 4)
plt.ylim(-2, 2)
plt.xlabel('Time')
plt.ylabel('y3')
plt.title('y3 vs. Time')
plt.plot(ty, y3)

plt.tight_layout()
```





Discussion

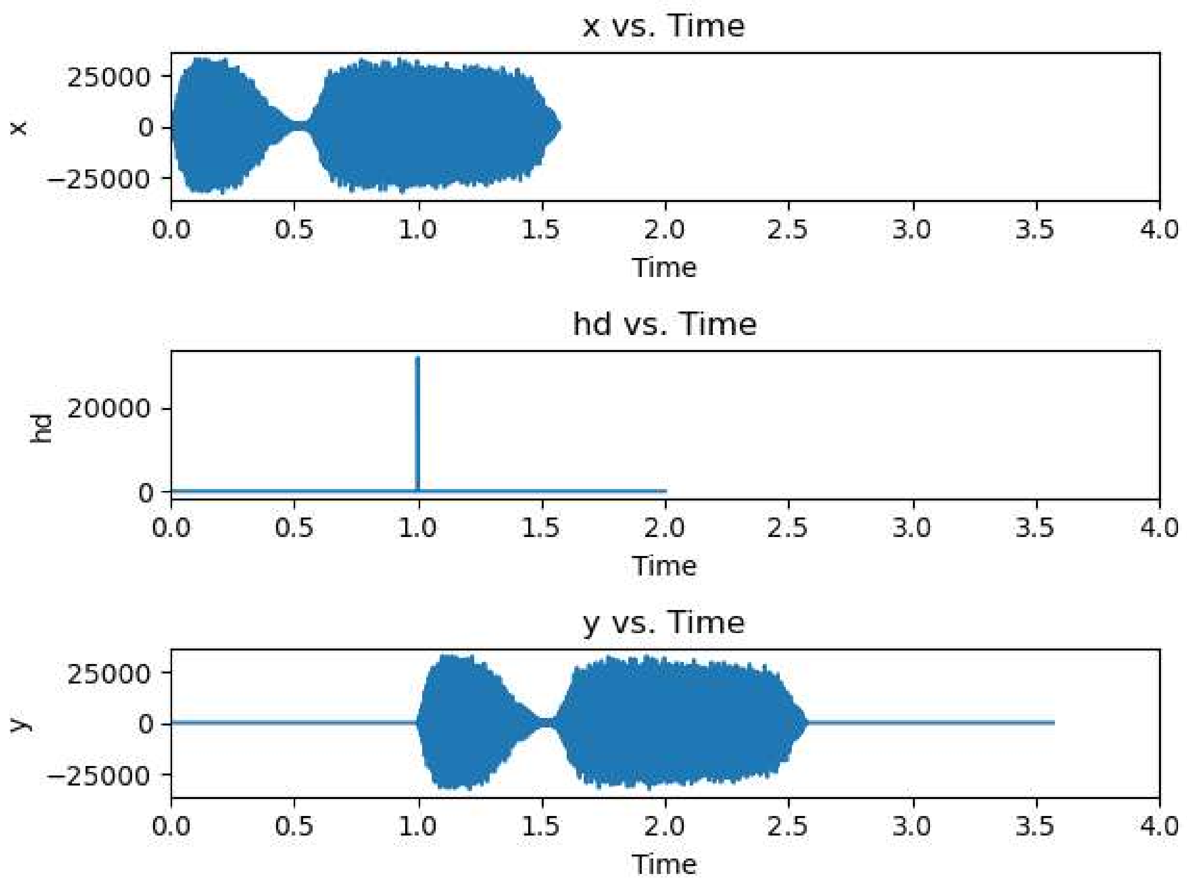
Discuss how the h plots would change if you used $fs = 10$, instead of $fs = 1000$? The systems corresponding to impulse responses $h2(t)$ and $h3(t)$ capture different information about a signal. Comment on what aspects of the input signal correspond to the largest values of $y2(t)$ and $y3(t)$.

h1 has a larger width of where the impulse would occur due to the smaller frequency value of 10. With a smaller frequency, we would have larger time samples; thus, we would have a wider impulse spike rather than a vertical line. In regards to h2 and h3, in areas we would see a vertical jump discontinuity, we see a slope due to the smaller frequency of 10.

To get the largest values of y2 and y3, we would have to change the input signal to find the maximum overlapping regions. This is because area of the overlapping regions correspond to the values of y2 and y3.

In [13]:
```
# Assignment 2

# Part A

# read in train audio
fs, x = wav.read('train32.wav')
t_x = np.arange(0, len(x)/fs, (1/fs))
```

```
# Part B

# time delay LTI system impulse response = hd
t_h = np.arange(0, 2, 1/fs)
hd = impulse(1, t_h, fs)

# Part C
y = np.convolve(hd, x)/fs
t_y = np.arange(0, (len(y)/fs), (1/fs))

# Part D
#Plot the responses required
fig = plt.figure(3)

# D.a
plt.subplot(3, 1, 1)
plt.xlim(0, 4)
plt.xlabel('Time')
plt.ylabel('x')
plt.title('x vs. Time')
plt.plot(t_x, x)
# D.b
plt.subplot(3, 1, 2)
plt.xlim(0, 4)
plt.xlabel('Time')
plt.ylabel('hd')
plt.title('hd vs. Time')
plt.plot(t_h, hd)

# D.c
plt.subplot(3, 1, 3)
plt.xlim(0, 4)
plt.xlabel('Time')
plt.ylabel('y')
plt.title('y vs. Time')
plt.plot(t_y, y)


plt.tight_layout()
```



Discussion

Suppose we modify (C) by not scaling according to the sampling time. Describe (in words) what would happen to output signal $y(t)$ if you used this $h(t)$ instead. How will the graph of $y(t)$ differ from the original and how will the sound differ?

If we do not scale according to the sampling time, the output signal y(t) will have it's y-axis scaled incorrectly. In the case that fs = 25,000, the max magnitude of amplitude will be 25,000 if we scale accordingly rather than 1E9 if we do not scale at all. If we do scale accordingly, the output volume will decrease due to a decrease in amplitude and sound like a train whistle. If we were to play the output sound without scaling, it sounds very scratchy and loud.

In [17]:
```
# Assignment 3

# Part A

# The below files are stereo and must be converted to mono for the convolutions
fs, x1 = wav.read('s1_1.wav')
fs, x2 = wav.read('s1_2.wav')
fs, x3 = wav.read('s1_3.wav')
# Convert to mono
x1=x1[:,0]
x2=x2[:,0]
x3=x3[:,0]
```

```
# Part B
t = np.arange(0, 3, 1/fs)
# create h1(t)
h1 = impulse(1, t, fs)

# create h2(t)
h2 = 10*impulse(0, t, fs)

# create h3(t)
h3 = impulse(2, t, fs)

# Part C

# Decode each time slice by convolving with the impuse response of the appropriate syst
y1 = np.convolve(x1, h1)/fs
y2 = np.convolve(x2, h2)/fs
y3 = np.convolve(x3, h3)/fs

# Part D

# store the original lengths of all the time slices
y1_len = len(y1)
y2_len = len(y2)
y3_len = len(y3)

# zero pad the others
y1 = np.concatenate([y1, np.zeros(y3_len - y1_len)])
y2 = np.concatenate([y2, np.zeros(y3_len - y2_len)])

# Print the length of all the outputs just to be sure
print(len(y1))
print(len(y2))
print(len(y3))

y = y1 + y2 + y3

outfile1 = 'y.wav'
wav.write(outfile1,fs,y.astype('int16'))
play_obj1 = sa.WaveObject.from_wave_file('y.wav').play()
play_obj1.wait_done()
```

```
221271
221271
221271
```

Discussion

What would the result sound like if you accidentally put $x_2(t)$ into all three filters?

If we put x2 into all three filters, we get the same clip repeated three times in sequence. The first sound clip is distinctly louder than the last two clips due to the 10*delta(t).

Can you name the character or TV show for your selected sound file?

The selected sound file is from the movie Toy Story from the character Buzz Lightyear.

```
In [ ]:
```