PS06 – Cynthia Chen

1. Earlier in the course, you saw how the recorded audio signal of a gun being fired in a shooting range can be convolved with a violin recording to approximate how the violin would sound if played in a shooting range. Please explain this using what you know about the impulse and impulse responses.

The gunshot puts an impulse into the system, and the sound is the impulse response, which simulates the effect of the room. The spectrum of the gunshot acts as the transfer function, because it encodes the response of the room and demonstrates how the system transfers the input to the output. In addition, it is in the form of an amplitude multiplier and a phase shift, so element-wise multiplication of transfer function and violin spectrum produces the effect of the violin recording characterized in the shooting range. This spectrum can then be converted back into a wave through DFT. As a result, convolution can also be implemented on the waves, because convolution in the time domain corresponds to element-wise multiplication in the frequency domain.

2. Consider a simple model of an echo channel. Suppose that the output of the echo channel is y(t) and the input is x(t), and the input and output are related as follows:

$$y(t) = \frac{1}{2}x(t-1) + \frac{1}{4}x(t-10).$$

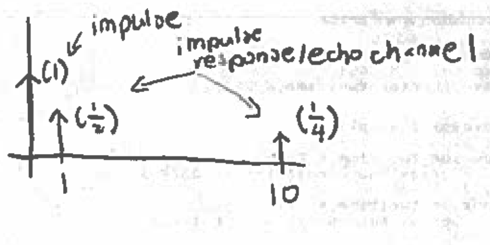
Explain why it is reasonable to call this an echo channel and find an expression for the impulse response of this system, and sketch it.

It is reasonable to call this an echo channel, because one second later, the sound output/impulse response has half the unit area of the original sound input, and ten seconds later, you will get a fourth of the original sound input's unit area. Once the components are added together, it makes an echo of the input.

The expression and sketch for the impulse response of this system:

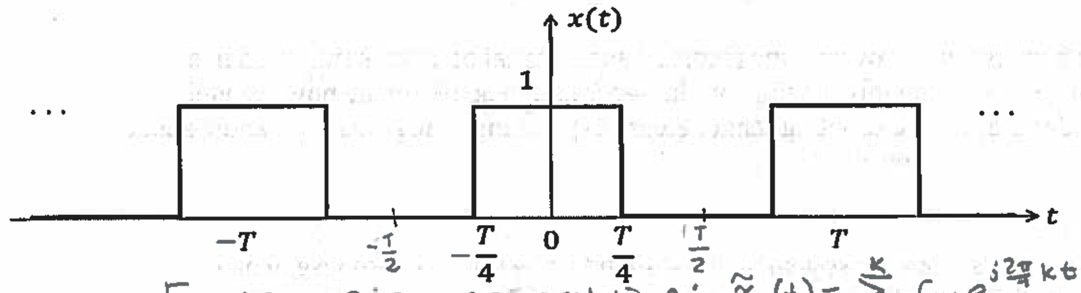
$\delta(t)$: impulse, $h(t)$: impulse response

$$h(t) = \frac{1}{2}\delta(t-1) + \frac{1}{4}\delta(t-10)$$



3.

a. Find the Fourier series representation for the square wave in Figure 1.



Fourier series representation: $\tilde{x}_K(t) = \sum_{k=-K}^{K} C_K e^{j\frac{2\pi}{T}kt}$

$C_K = \frac{1}{T}\int_{-T/2}^{T/2} x(t) e^{-j\frac{2\pi}{T}kt}\,dt$

Finding the coefficient:

from $-\frac{T}{2}$ to $\frac{T}{4}$: $C_K = 0$, $x(t)=0$

from $\frac{T}{4}$ to $\frac{T}{2}$: $C_K = 0$, $x(t) = 0$

from $-\frac{T}{4}$ to $\frac{T}{4}$: $x(t) = 0$

$$C_K = \frac{1}{T}\int_{-T/4}^{T/4} e^{-j\frac{2\pi}{T}kt}\,dt$$

$$= \frac{1}{T}\left[e^{-j\frac{2\pi}{T}kt}\left(-\frac{1}{j\frac{2\pi}{T}k}\right)\right]_{-T/4}^{T/4}$$

$$= \frac{1}{T}\left(-\frac{1}{j\frac{2\pi}{T}k}\right)\left(e^{-j\frac{\pi k}{2}} - e^{j\frac{\pi k}{2}}\right)$$

$$= -\frac{e^{-j\frac{\pi k}{2}} - e^{j\frac{\pi k}{2}}}{2\pi jK}$$

$$= -\frac{1}{2j}\left(\frac{e^{-j\frac{\pi k}{2}} - e^{j\frac{\pi k}{2}}}{\pi K}\right)$$

Using $\sin(\theta) = \frac{1}{2j}e^{j\theta} - \frac{1}{2j}e^{-j\theta}$: $C_K = -\frac{\sin\left(\frac{\pi k}{2}\right)}{\pi K} = -\frac{\operatorname{sinc}\left(\frac{k}{2}\right)}{2}$

replace C_K to get

$$\boxed{\tilde{x}_K(t) = \sum_{k=-K}^{K} \frac{\operatorname{sinc}\left(\frac{k}{2}\right)}{2} e^{j\frac{2\pi}{T}kt}}$$

b. Using a computer, plot the Fourier series representation of the square wave in the previous part with fundamental period T=4, and for 5,17, and 257 terms in the Fourier series. For clarity, you should plot them in separate subplots/plots.

Function for finding Fourier Series Representation

```
T=4
def fourier_two(t,K):
    total = numpy.zeros(len(t))
    for i in range(len(K)):
        total += (0.5)*numpy.sinc(i/2.0)*numpy.exp(1j/T*2.0*numpy.pi*i*t)
    return total
```

Fourier Series Representation when k = 5

```
K_5 = numpy.arange(-2,2)
time = numpy.linspace(-6,6,100)
thinkplot.Plot(time,fourier_two(time, K_5))
```

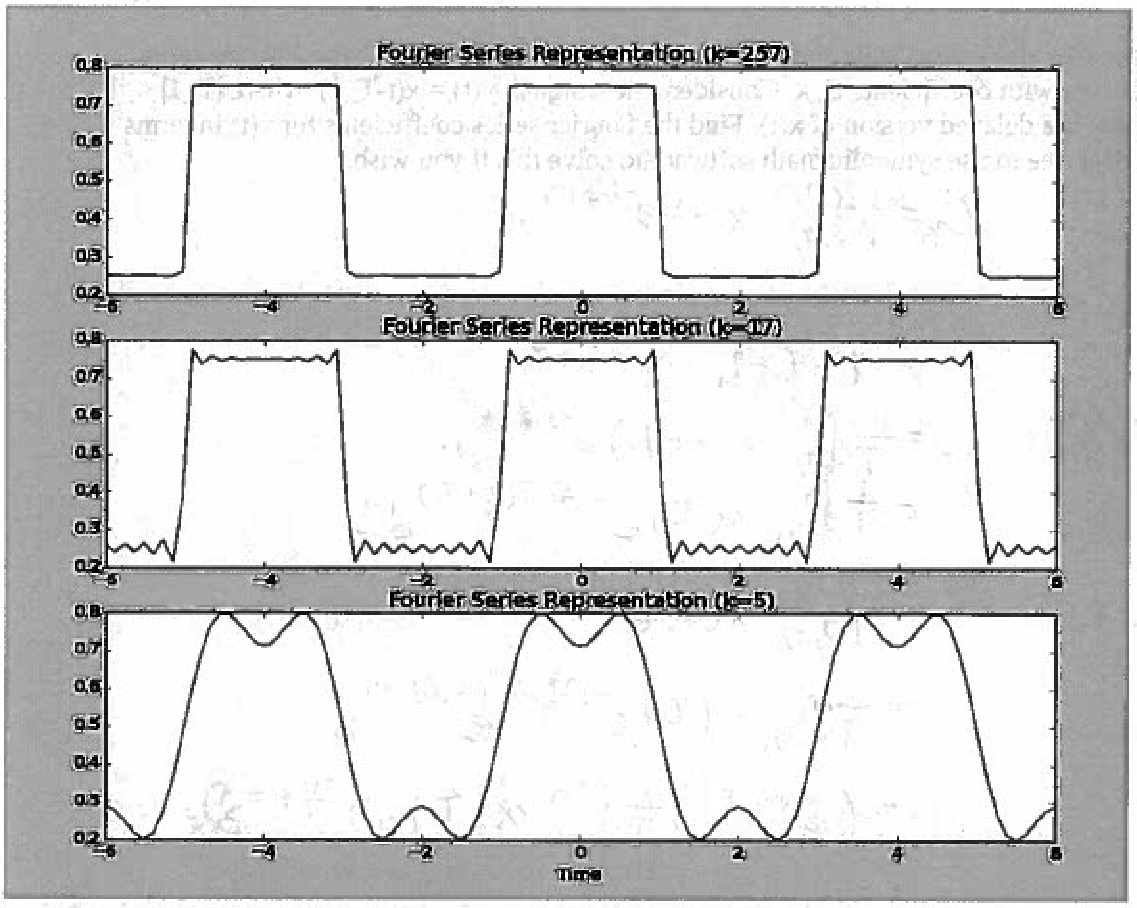
Fourier Series Representation when k=17

```
K_17 = numpy.arange(-8,8)
time = numpy.linspace(-6,6,100)
thinkplot.Plot(time,fourier_two(time,K_17))
```

Fourier Series Representation when k=257

```
K_257 = numpy.arange(-128,128)
time = numpy.linspace(-6,6,100)
thinkplot.Plot(time,fourier_two(time,K_257))
```

```
import matplotlib.pyplot as plt
plt.subplot(3, 1, 1)
plt.plot(time, fourier_two(time,K_257))
plt.title('Fourier Series Representation (k=257)')
plt.subplot(3, 1, 2)
plt.plot(time, fourier_two(time,K_17))
plt.title('Fourier Series Representation (k=17)')
plt.subplot(3, 1, 3)
plt.plot(time, fourier_two(time,K_5))
plt.title('Fourier Series Representation (k=5)')
plt.xlabel('Time')
```



c. Describe what you see in the Fourier series representation, at the discontinuous points of the square wave, i.e. the points where the square wave goes from 1 to 0 and 0 to 1. How can you reconcile this with (10) in the Fourier series notes? Note: what you should observe is a manifestation of the Gibbs phenomenon, which is caused by the inability of the Fourier series to produce accurate representations of periodic signals at points of discontinuity.

At the discontinuous points of the square wave, it looks very choppy and there are sharp edges. These sharp edges denote quick changes, and, in short amount of time, corresponds to high frequency. Having too few k terms does not represent these high frequencies well enough, so there needs to be more k terms, which also reduces the squared error between the actual square wave and it's Fourier series approximation.

This error is denoted by:

$$\int_{-T/2}^{T/2} |x(t) - \tilde{x}_k(t)|^2 dt \to 0.$$

4.

a. Suppose that x(t) is a periodic signal with fundamental period T, and has a Fourier series representation with coefficients C_k. Consider a new signal, y(t) = x(t-T_1), where |T_1| < T. Thus y(t) is a delayed version of x(t). Find the Fourier series coefficients for y(t) in terms of C_k. Feel free to use symbolic math software to solve this if you wish.

$$C_k = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\, e^{-j\frac{2\pi}{T}kt}\, dt$$

$$y(t) = x(t-T_1)$$

$$\tau = t - T_1 \qquad t = \tau + T_1$$

for $y(t)$: $$C_k = \frac{1}{T}\int_{-T/2}^{T/2} x(t-T_1)\, e^{-j\frac{2\pi}{T}kt}\, dt$$

$$= \frac{1}{T}\int_{-T/2}^{T/2} x(\tau)\, e^{-j\frac{2\pi}{T}k(\tau+T_1)}\, d\tau$$

$$= \frac{1}{T}\int_{-T/2}^{T/2} x(\tau)\, e^{-j\frac{2\pi}{T}k\tau - j\frac{2\pi}{T}kT_1}\, d\tau$$

$$= \frac{1}{T}\int_{-T/2}^{T/2} x(\tau)\, e^{-j\frac{2\pi}{T}k\tau}\, e^{-j\frac{2\pi}{T}kT_1}\, d\tau$$

$$= \left(e^{-j\frac{2\pi}{T}T_1 k}\right) \underbrace{\frac{1}{T}\int_{-T/2}^{T/2} x(\tau)\, e^{-j\frac{2\pi}{T}kt}\, dt}_{C_k}$$

$$= \boxed{C_k e^{-j\frac{2\pi}{T}T_1 k}}$$

This makes so much sense in terms of phase offset shifts!!! Amplitude of origin-l needs to include the change and offset through complex exponential.

b. Using your answer above, find the Fourier series coefficients for the triangle wave in Figure 2. Verify that your answer is correct by modifying and running the code for the Fourier series of the triangle wave that you used in class. Please turn in a listing of your code and a plot of the triangle wave you generated.

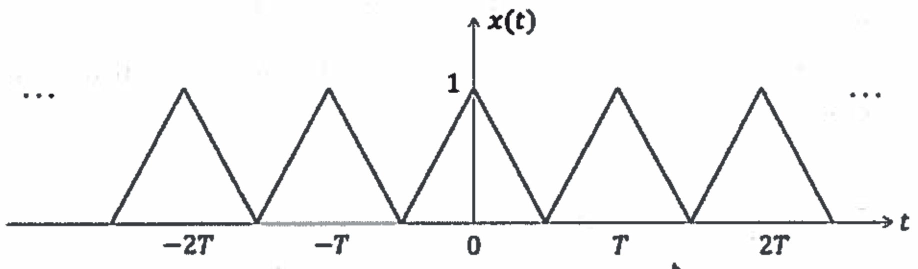


Figure 2: Triangle wave with period T.

Triangle wave:

$$C_k = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\, e^{-j\frac{2\pi}{T}kt}\,dt$$

$$x(t) = \frac{2}{T}|t|$$ ← from "Fourier Series" Reading

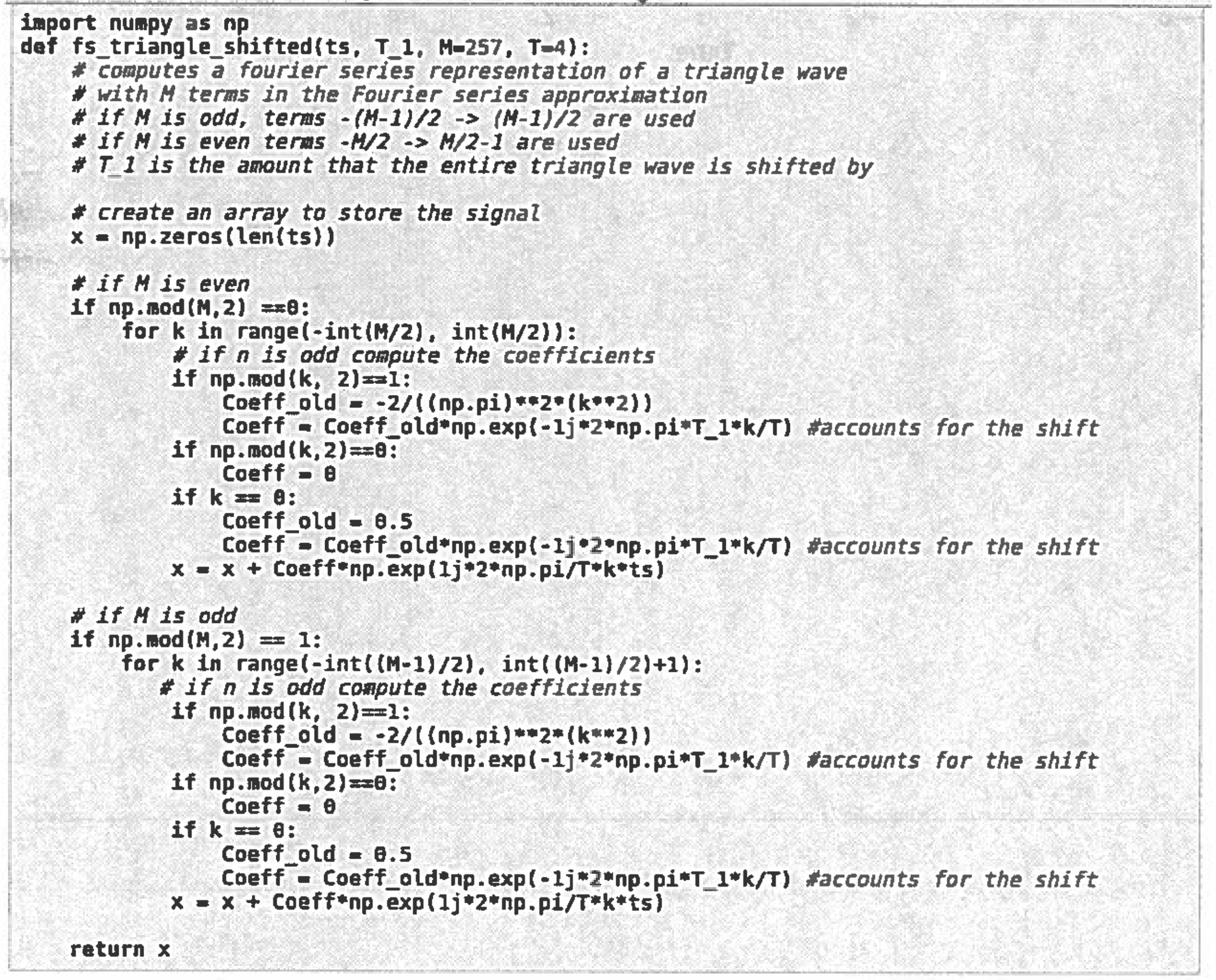
$$C_k = \begin{cases} -\frac{2}{\pi^2 k^2} \text{ if } k \text{ is odd} \\ \frac{1}{2} \text{ if } k = 0 \\ 0 \text{ otherwise} \end{cases}$$

new C_k w/ phase shift:

$$\boxed{C_k = \begin{cases} -\frac{2}{\pi^2 k^2}\left(e^{-j\frac{2\pi}{T}T_1 k}\right) \text{ if } k \text{ is odd} \\ \frac{1}{2}\left(e^{-j\frac{2\pi}{T}T_1 k}\right) \text{ if } k = 0 \\ 0 \text{ otherwise} \end{cases}}$$

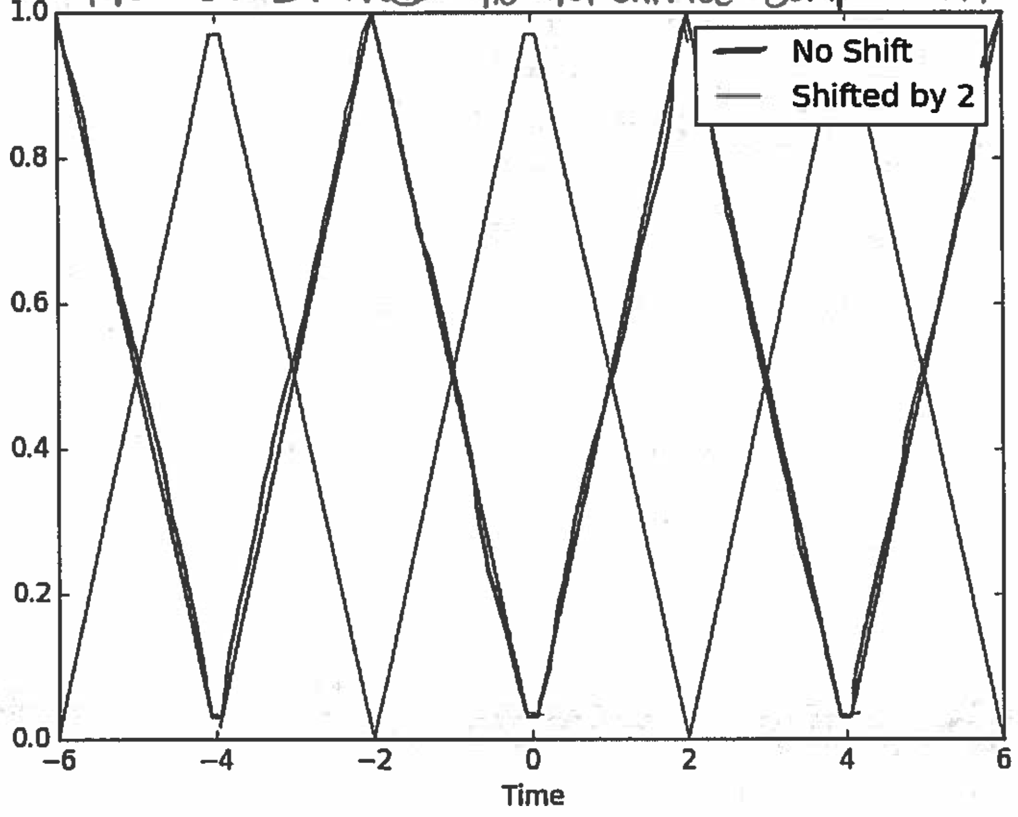
where T_1 is the shifted amount

Code for shifted triangle wave:



```
import numpy as np
def fs_triangle_shifted(ts, T_1, M=257, T=4):
    # computes a fourier series representation of a triangle wave
    # with M terms in the Fourier series approximation
    # if M is odd, terms -(M-1)/2 -> (M-1)/2 are used
    # if M is even terms -M/2 -> M/2-1 are used
    # T_1 is the amount that the entire triangle wave is shifted by

    # create an array to store the signal
    x = np.zeros(len(ts))

    # if M is even
    if np.mod(M,2) ==0:
        for k in range(-int(M/2), int(M/2)):
            # if n is odd compute the coefficients
            if np.mod(k, 2)==1:
                Coeff_old = -2/((np.pi)**2*(k**2))
                Coeff = Coeff_old*np.exp(-1j*2*np.pi*T_1*k/T) #accounts for the shift
            if np.mod(k,2)==0:
                Coeff = 0
            if k == 0:
                Coeff_old = 0.5
                Coeff = Coeff_old*np.exp(-1j*2*np.pi*T_1*k/T) #accounts for the shift
            x = x + Coeff*np.exp(1j*2*np.pi/T*k*ts)

    # if M is odd
    if np.mod(M,2) == 1:
        for k in range(-int((M-1)/2), int((M-1)/2)+1):
           # if n is odd compute the coefficients
            if np.mod(k, 2)==1:
                Coeff_old = -2/((np.pi)**2*(k**2))
                Coeff = Coeff_old*np.exp(-1j*2*np.pi*T_1*k/T) #accounts for the shift
            if np.mod(k,2)==0:
                Coeff = 0
            if k == 0:
                Coeff_old = 0.5
                Coeff = Coeff_old*np.exp(-1j*2*np.pi*T_1*k/T) #accounts for the shift
            x = x + Coeff*np.exp(1j*2*np.pi/T*k*ts)

    return x
```

Plot of shifted and not shifted comparison:


No Shift
Shifted by 2
Time