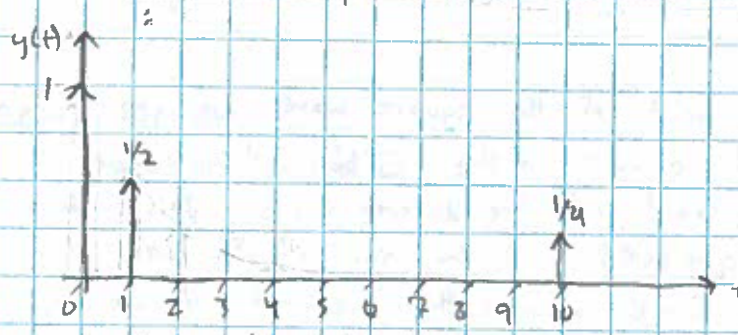
PROBLEM SET 6

Zuher Ghadyali
Sig Sys 295
03/03/2015

1. The room the gun is fired in is a linear time-invariant system that acts as an operator. That is to say that the gunshot itself is an impulse to the system and the echo of the gunshot is the impulse response. Now if we take the room as an operator, this operator is applied to the impulse to produce the impulse response. Convolving the violin recording with the impulse response applies this operator to the violin recording which produces what the violin sounds like if played in the same room.

2. To test if this is an echo channel, let's apply an impulse at $t=0$ and see what happens if we follow the echo channel model. The impulse is the input $x(t)$.



We see that both terms in this equation, $\frac{1}{2}x(t-1)$ and $\frac{1}{4}x(t-10)$ are "echoing" the original impulse at $t=0$. The summation of both terms is also an echo channel. The impulse response is:

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

where $h(t)$ is the impulse response and $\delta(t)$ is the impulse

3. a. $$\tilde{x}_k(t) = \sum_{k=-K}^{K} C_k e^{j\frac{2\pi}{T}kt} \Leftarrow \text{Fourier Series}$$

We need to determine the coefficients C_k for each value of k.

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

We know that over the interval from $-\frac{T}{2}$ to $-\frac{T}{4}$, the value of $x(t)$ is 0 so:

$$C_k = \frac{1}{T}\int_{-T/2}^{-T/4} (0)\, e^{-j\frac{2\pi}{T}kt} = 0 \qquad C_k = \frac{1}{T}\int_{T/4}^{T/2} (0)\, e^{-j\frac{2\pi}{T}kt} = 0$$

We need to determine C_k in the range from $-\frac{T}{4}$ to $\frac{T}{4}$

$$C_k = \frac{1}{T}\int_{-T/4}^{T/4} (1)\, e^{-j\frac{2\pi}{T}kt} = \frac{1}{T}\left(\frac{1}{-j\frac{2\pi}{T}k} e^{-j\frac{2\pi}{T}kt}\Big|_{-T/4}^{T/4}\right) =$$

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

$$\frac{1}{-j2\pi k} e^{-1/2\, j\pi k} + \frac{1}{j2\pi k} e^{\frac{1}{2}j\pi k} = \frac{1}{j2\pi k} e^{1/2\, j\pi k} - \frac{1}{j2\pi k} e^{-1/2\, j\pi k} =$$

$$\frac{1}{\pi k}\left(\frac{1}{2j} e^{1/2\, j\pi k} - \frac{1}{2j} e^{-1/2\, j\pi k}\right) = \frac{1}{\pi k}\left(\sin\left(\tfrac{1}{2}\pi k\right)\right) = \frac{\sin\left(\frac{1}{2}\pi k\right)}{\pi k}$$

If we let $x = \frac{k}{2}$ then:

$$C_k = \frac{\sin(\pi x)}{\pi(2x)} = \frac{1}{2}\,\mathrm{sinc}(x) = \frac{1}{2}\,\mathrm{sinc}\left(\frac{k}{2}\right)$$

3. c. At the discontinuous points of the square wave we see large oscillations which are expected because of the Gibbs phenomenon. The Fourier series overshoots at the discontinuities. This is because corners represent high frequencies and the sum of all these high frequencies at the corner of the wave creates the overshoot. Reconciling the Gibbs Phenomenon with Equation 10 in the Fourier Series, we see that increasing the number of terms in the Fourier Series smooths out the overshoot, which makes the integral of the squared error approach 0.

4. a. $C_{ky} = \frac{1}{T}\int_{-T/2}^{T/2} y(t)\, e^{-j\frac{2\pi}{T}kt}\, dt$ $\qquad y(t) = x(t - T_1)$ $\qquad \tau = t - T_1$, $t = \tau + T_1$

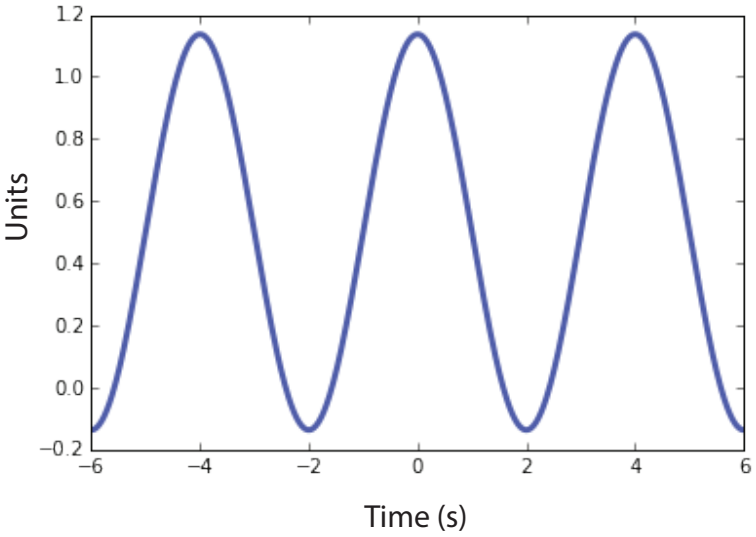
$$= \frac{1}{T}\int_{-T/2-T_1}^{T/2-T_1} x(t-T_1)\, e^{-j\frac{2\pi}{T}kt}\, dt = \frac{1}{T}\int_{-T/2-T_1}^{T/2-T_1} x(\tau)\, e^{-j\frac{2\pi}{T}k(\tau+T_1)}\, d\tau$$

$$= \frac{1}{T}\int_{-T/2-T_1}^{T/2-T_1} x(\tau)\, e^{-j\frac{2\pi}{T}k\tau}\, e^{-j\frac{2\pi}{T}kT_1}\, d\tau = C_k\, e^{-j\frac{2\pi}{T}kT_1}$$

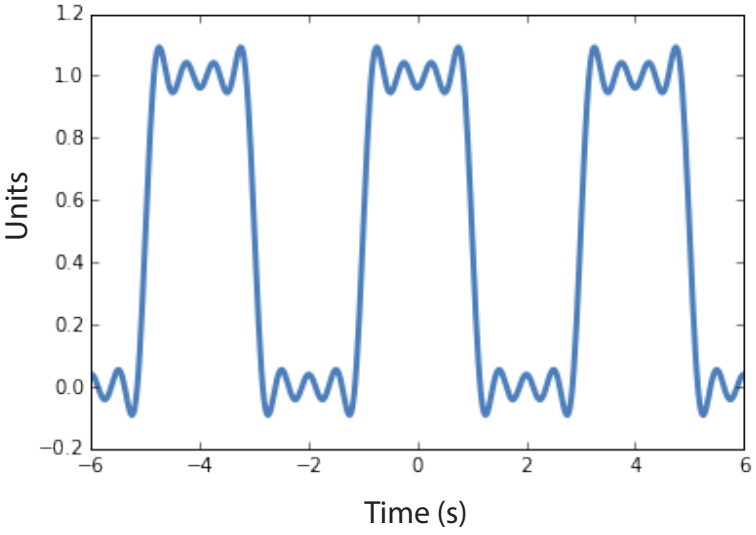
b. $$C_{ky} = \begin{cases} -\frac{2}{\pi^2 k^2} e^{-j\frac{2\pi}{T}kT_1} & \text{if } k \text{ is odd} \\ \frac{1}{2} e^{-j\frac{2\pi}{T}kT_1} & \text{if } k = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Zoher Ghadyali
PS06
3b Plots

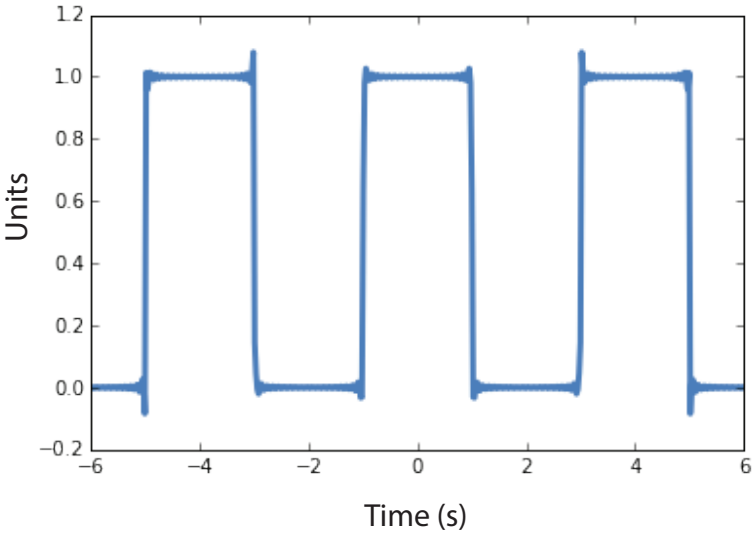
Fourier Series with
5 Terms



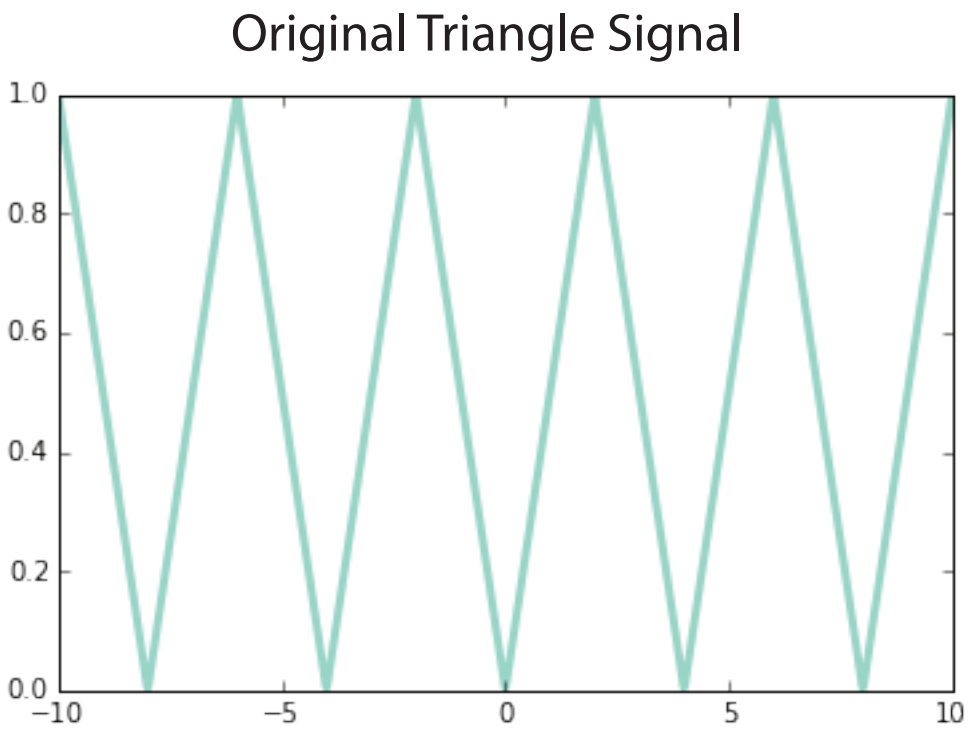
Fourier Series with
17 Terms

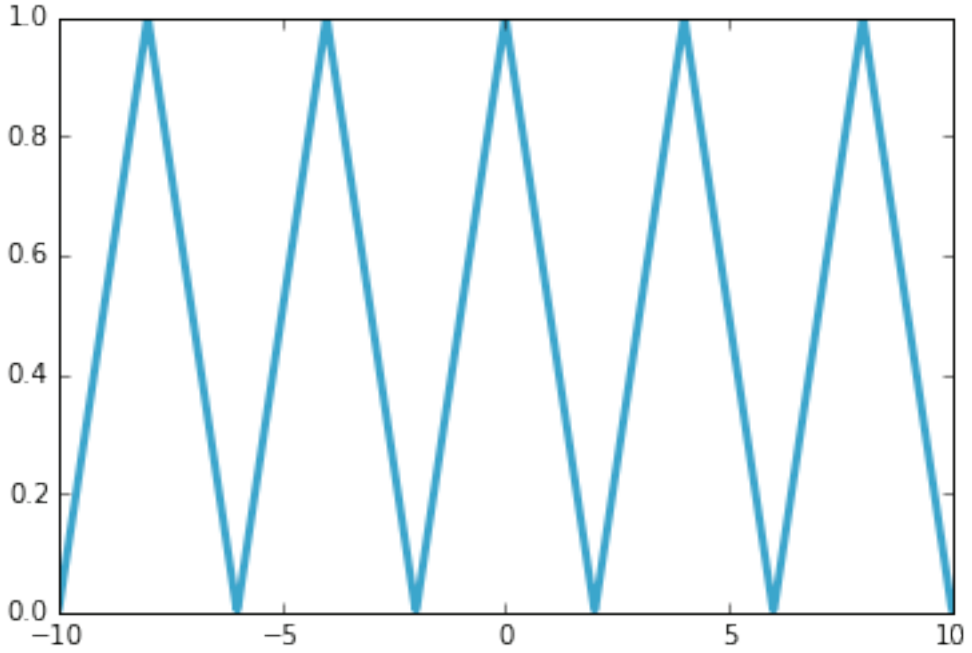


Fourier Series with
255 Terms



Zoher Ghadyali
PS06
4b plots





Zoher Ghadyali
PS06
3b and 4b original code

3b code

```
def approx_square_wave(T, ts, n):
    """T is the period and n is the number of complex exponential terms"""
    ks = numpy.arange(-n/2+1, n/2+1)
    total = 0
    for i in range(len(ks)):
        total += 0.5*numpy.sinc(ks[i]/2.0) * numpy.exp(1j*(2.0*math.pi/4.0)*ks[i]*ts)
    return total

T = 4
ts = numpy.linspace(-6, 6, 500)

total_5 = approx_square_wave(T, ts, 5)
total_17 = approx_square_wave(T, ts, 17)
total_255 = approx_square_wave(T, ts, 255)
```

4b code that generates original triangle signal

```
def fs_triangle(ts, M=100, 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

    # 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 = -2/((np.pi)**2*(k**2))
            if np.mod(k,2)==0:
                Coeff = 0
            if k == 0:
                Coeff = 0.5
            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 = -2/((np.pi)**2*(k**2))
            if np.mod(k,2)==0:
                Coeff = 0
            if k == 0:
                Coeff = 0.5
            x = x + Coeff*np.exp(1j*2*np.pi/T*k*ts)

    return x

ts = numpy.linspace(-10, 10, 1000)
res = fs_triangle(ts)

thinkplot.plot(ts, res)
```

Zoher Ghadyali
PS06
4b shift code

4b code that generates shifted triangle signal

```
def fs_triangle2(ts, M=100, T=4, T1=2):
    # 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

    # 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 = -2/((np.pi)**2*(k**2)) * numpy.exp(-1j*(2.0*math.pi/T)*k*T1)
            if np.mod(k,2)==0:
                Coeff = 0 * numpy.exp(-1j*(2.0*math.pi/T)*k*T1)
            if k == 0:
                Coeff = 0.5 * numpy.exp(-1j*(2.0*math.pi/T)*k*T1)
            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 = -2/((np.pi)**2*(k**2)) * numpy.exp(-1j*(2.0*math.pi/T)*k*T1)
            if np.mod(k,2)==0:
                Coeff = 0 * numpy.exp(-1j*(2.0*math.pi/T)*k*T1)
            if k == 0:
                Coeff = 0.5 * numpy.exp(-1j*(2.0*math.pi/T)*k*T1)
            x = x + Coeff*np.exp(1j*2*np.pi/T*k*ts)

    return x

ts2 = numpy.linspace(-10, 10, 1000)
res2 = fs_triangle2(ts2)

thinkplot.plot(ts2, res2)
```