Alexander Crease

SigSys Problem Set 6

1. The sound of the gun in the shooting range represents the impulse response of the shooting range. By convolving it with the violin playing, we are basically applying the shooting range impulse response to the waveform of the violin.
2. It is reasonable to call this an echo channel because the output of the signal has a decreased amplitude and is shifted forward in time; one impulse at 1 second and one impulse at 10 seconds. The impulse response can be expressed as $.5\delta(t-1)+.25\delta(t-10)$.

3. A.

$$f(x)=\sum_{k=0}^{\infty}\left(c_k e^{\frac{j2\pi kt}{T}}\right)$$

Because the function is 0 from –T/4 to T/4, and T = 4, we can say:

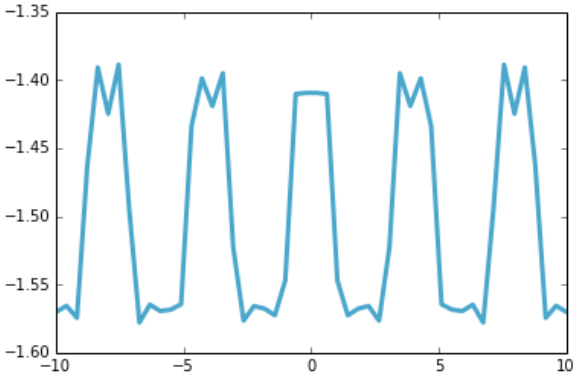
$$c_k=\frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(x)\,e^{-\frac{j2\pi kt}{T}}dt=\frac{1}{T}\int_{-T/4}^{T/4} e^{-\frac{j2\pi kt}{T}}dt=\frac{1}{T}\left(\frac{T}{j2\pi kt}\right)e^{\frac{-j2\pi kt}{T}}\Big]_{-T/4}^{T/4}$$

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

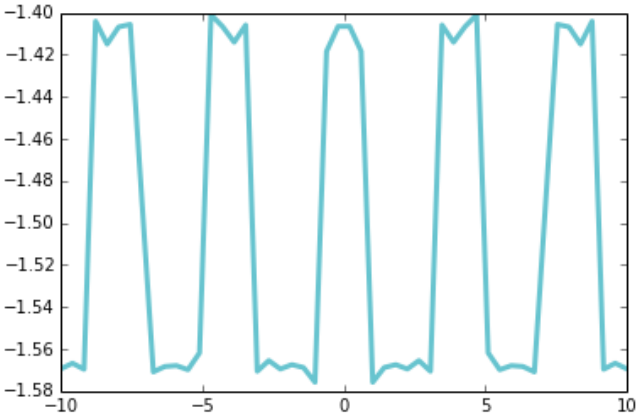
$$f(x)=\sum_{k=0}^{\infty}\frac{1}{\pi k}\sin\left(\frac{\pi k}{2}\right)\left(e^{\frac{j2\pi kt}{T}}\right)=\sum_{k=0}^{\infty}\frac{1}{2}\operatorname{sinc}\left(\frac{\pi k}{2}\right)\left(e^{\frac{j\pi kt}{2}}\right)$$

B.

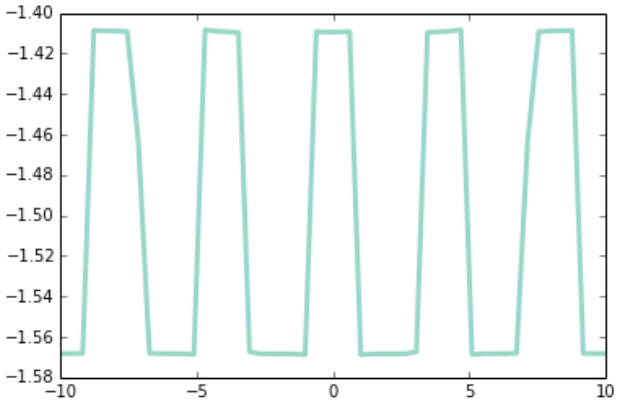
K = 5



K = 17



K = 271



C. The discontinuity of the square signal makes it impossible to represent as a continuous signal. The Fourier signal can keep adding on more terms to approximate the function, but all this will do is reduce the wavering at the discontinuous points. This can only occur when the sum is infinite, which we cannot physically reproduce. The discontinuity means that the periodic signal will always overshoot the top or bottom of the square wave a little bit, and then dampen down to the correct place. As we increase the length of the sum, the amount of overshoot produced decreases.

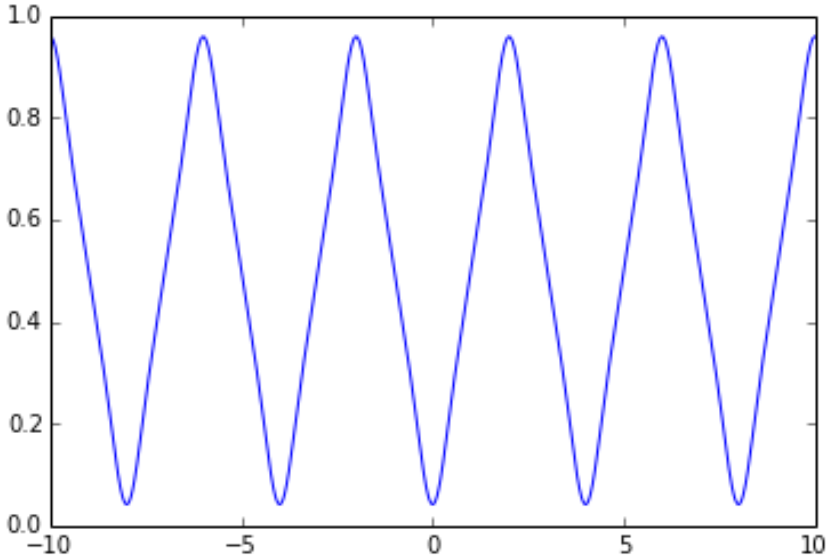
4. A.

$$C_{ky} = \frac{1}{T}\int_{-\frac{T}{2}-T_1}^{\frac{T}{2}-T_1} x(t-T_1)e^{-\frac{j2\pi k(t-T_1)}{T}}dt \quad \frac{1}{T}\int_{-\frac{T}{2}-T_1}^{\frac{T}{2}-T_1} x(t-T_1)e^{-\frac{j2\pi kt}{T}}e^{\frac{j2\pi kT_1}{T}}dt \; = \; C_k e^{\frac{j2\pi kT_1}{T}}$$

From this, we can obtain:

$$C_{ky} = \begin{cases} -\frac{2}{\pi^2 k^2}e^{\frac{j\pi kT_1}{T}} \; if \; K \; is \; odd \\ \frac{e^{\frac{j\pi kT_1}{T}}}{2} \; if \; K = 0 \\ 0 \; otherwise \end{cases}$$

B.



The Fourier series approximation matches that of the triangle wave in the prompt.

Appendix

Code for Problem 3

```
def square_fourier(k):
   t = numpy.linspace(-10, 10)
   total_wave = 0
   for i in range(0,k):
      wave = -.5*numpy.sinc(math.pi*i/2)*numpy.exp(j*math.pi*i*t/2)
      total_wave += wave
   thinkplot.plot(t,total_wave -1)
   thinkplot.config(label = False, xlim = [-10,10])
```

```
square_fourier(5)
square_fourier(17)
square_fourier(271)
```

Code for Problem 4

Modified from Siddhartan's Code

```
def fs_triangle(ts, M=3, 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))
   Ycoef=np.exp(j*math.pi*M)

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

   return x

# create an array of time samples from -8 to 8 and use 2048 points to represent the wave ts =
np.linspace(-10,10,2048)

x = fs_triangle(ts,M=10)
```

```
mplib.plot()
# set the axis range for the main plot

# plot the FS representation
line1, = mplib.plot(ts, x, lw=1)

mplib.show()
```