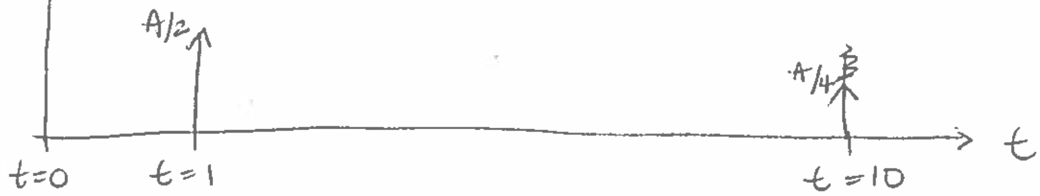
Meg McCauley March 6, 2015

# PS06

① You put the impulse into the system and the impulse response is the sound you hear. Because the gunshot is so loud it has a very large amplitude. This amplitude provides a fully characterized response to a system. If you convolve the impulse response from the gunshot with the violin recordings, the overlapping ~~[illegible]~~ amplitudes multiply together to produce the final impulse response. This final impulse response has the sound of the violin in the firing range.

② This is an echo channel because the echo comes back after a time delay (due to offset) and with a lower volume (due to amplitude). The expression can be represented as:

$$h(t) = \tfrac{1}{2}\Delta(t-1) + \tfrac{1}{4}\Delta(t-10)$$




3a next page

③ $\omega_0 = \frac{2\pi}{T}$

a. $C_k = \frac{1}{T}\int_0^T x(t)e^{-jk\omega_0 t}\,dt = \frac{1}{T}\int_0^{DT} A e^{-jk\omega_0 t}\,dt$

$= \frac{A}{T}\cdot\frac{1}{-jk\omega_0}e^{-jk\omega_0 t}\Big|_0^{DT}$

$= \frac{A}{T(-jk\omega_0)}\left[e^{-jk\frac{2\pi}{T}DT} - e^{-jk\omega_0 0}\right]$ $\left(\omega_0 = \frac{2\pi}{T}\right)$ $(e^0 = 1)$

$= \frac{A}{jk2\pi}\left[1 - e^{-jk2\pi D}\right]\cdot \underbrace{e^{j\pi kD}\,e^{-j\pi kD}}_{\text{fancy way of representing 1}}$

$= \frac{AD}{k\pi D}\cdot\underbrace{\frac{1}{2j}\left[e^{jk\pi D} - e^{-jk\pi D}\right]}_{\frac{\sin kD}{kD}} e^{-jk\pi D}$

$= AD\cdot \operatorname{Sinc} kD \quad e^{-jk\pi D}$

$\boxed{C_k = AD\cdot\operatorname{Sinc}(kD)\cdot e^{-jk\pi D}}$

③ a. $C_k = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} A e^{-jk\omega t}\,dt$

$$= \frac{A}{T}\cdot\frac{1}{-jk\omega} e^{-jk\omega t}\Big|_{-\frac{T}{2}}^{\frac{T}{2}} = \frac{A}{T}\cdot\frac{1}{-jk\omega}\left[e^{\frac{-jk\omega T}{2}} - e^{\frac{jk\omega T}{2}}\right]$$

$$\omega = \frac{2\pi}{T} \Rightarrow \frac{A}{T}\cdot\frac{1}{jk\frac{2\pi}{T}}\left[e^{\frac{-jk\frac{2\pi}{T}T}{2}} - e^{\frac{jk\frac{2\pi}{T}T}{2}}\right]$$

$$= \frac{A}{-jk2\pi}\left[e^{-jk\pi} - e^{jk\pi}\right]\cdot -1 = \frac{A}{jk2\pi}\left[e^{jk\pi} - e^{-jk\pi}\right]$$

$$\sin(k\pi) = \frac{1}{2j}\left(e^{jk\pi} - e^{-jk\pi}\right) \Rightarrow \frac{A}{k\pi}\cdot \sin k\pi = \frac{A \cdot \sin k\pi}{k\pi}$$

$$\boxed{C_k = A\cdot \mathrm{Sinc}(k)}$$

b. See graphs at end

c. At discontinuous points of the square wave, the signal overshoots the proper amplitude. This is a demonstration of the Gibbs phenomenon and is likely due to faults in measurement tools.

④ a. $C_k = \frac{1}{T}\int_{-\frac{T}{2}-T_1}^{\frac{T}{2}-T_1} x(t-T_1)e^{\left(\frac{-jk(2\pi)}{T}kt\right)}dt$

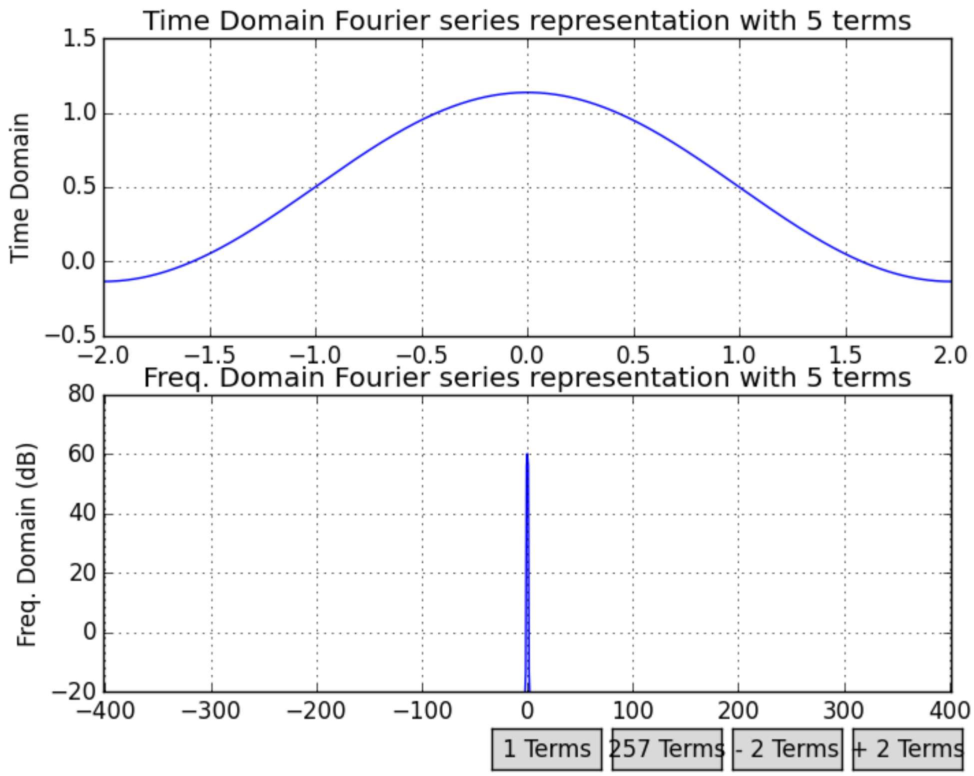
Using u substitution let $u = t - T_1$

$$C_k = \frac{1}{T}\int_{-\frac{T}{2}-(u-t)}^{\frac{T}{2}-(u-t)} x(u)e^{\left(\frac{-j(2\pi)}{T}k(u+T_1)\right)}du$$

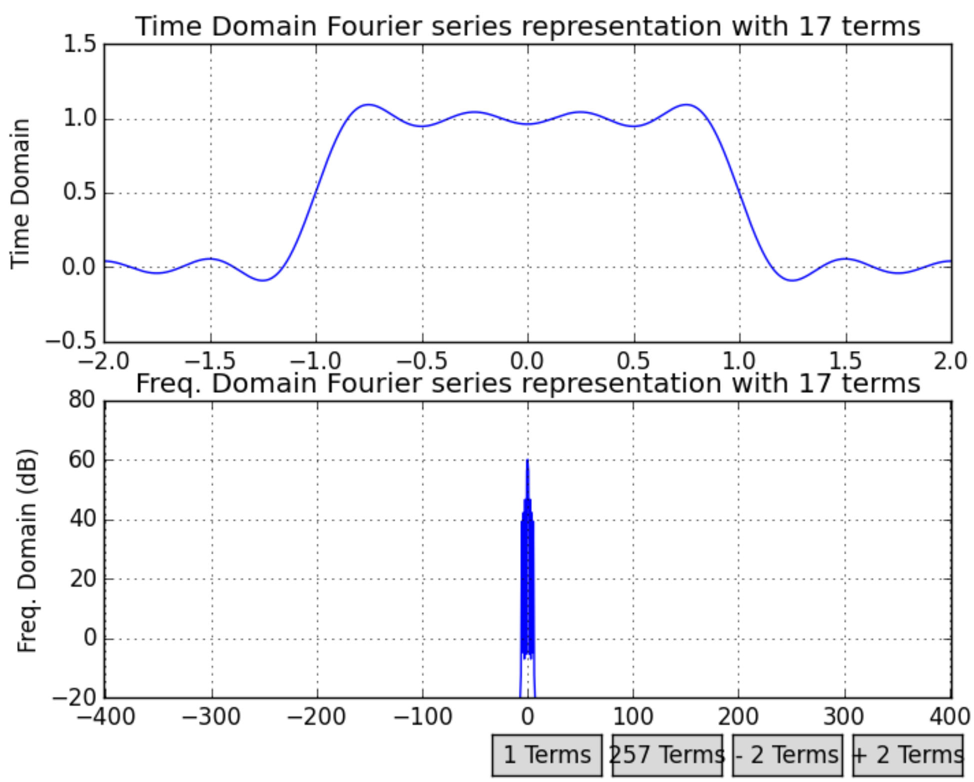
$$= e^{\frac{-j2\pi k T_1}{T}}\frac{1}{T}\int_{-\frac{T}{2}-(u-t)}^{\frac{T}{2}-(u-t)} x(u)e^{\left(\frac{-j(2\pi)}{T}k(u)\right)}du$$

b. See graph and code at end

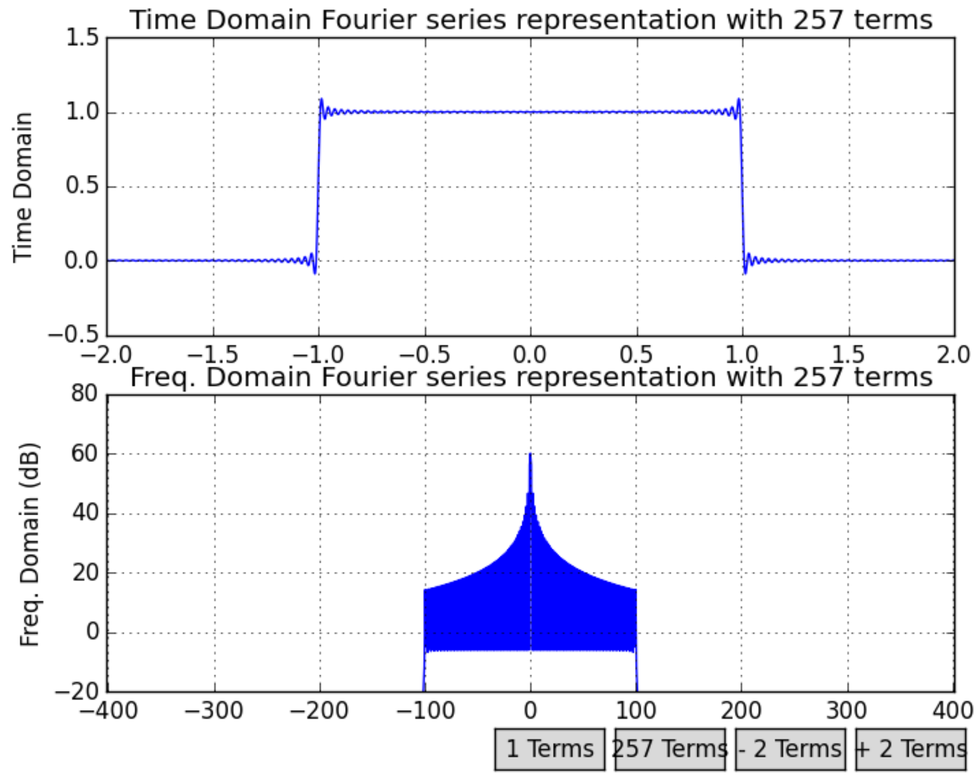
## Question 3b



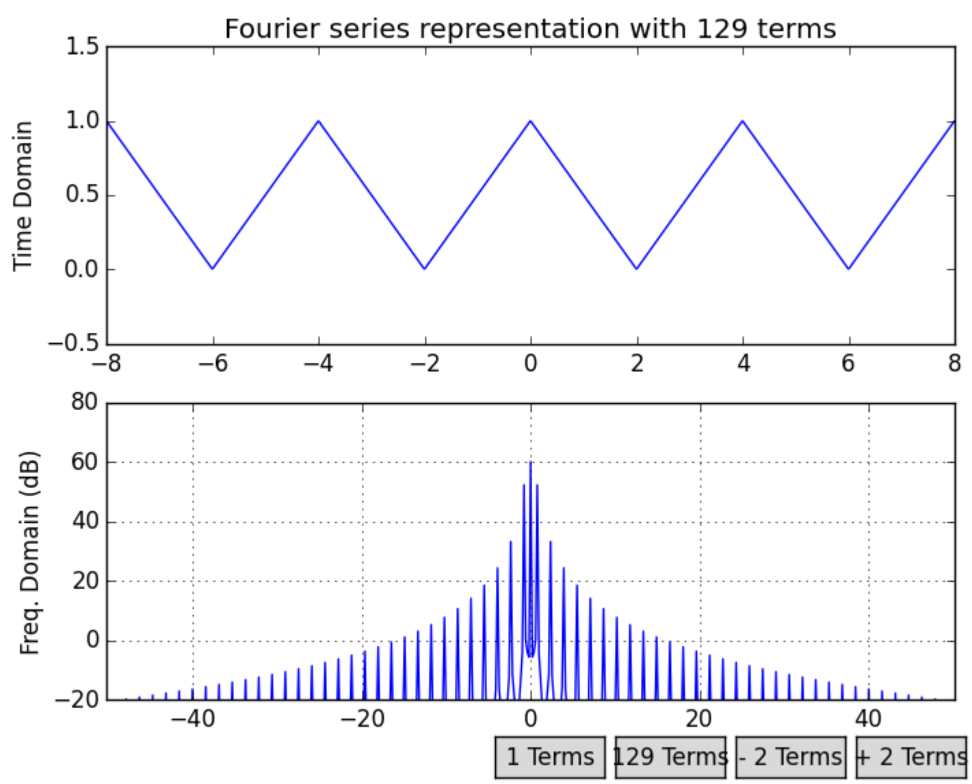
Time Domain Fourier series representation with 5 terms
Time Domain
Freq. Domain Fourier series representation with 5 terms
Freq. Domain (dB)
1 Terms
257 Terms
- 2 Terms
+ 2 Terms

Time Domain Fourier series representation with 17 terms
Time Domain
Freq. Domain Fourier series representation with 17 terms
Freq. Domain (dB)
1 Terms
257 Terms
- 2 Terms
+ 2 Terms


Time Domain Fourier series representation with 257 terms
Time Domain
Freq. Domain Fourier series representation with 257 terms
Freq. Domain (dB)
1 Terms
257 Terms
- 2 Terms
+ 2 Terms

## Question 4b


Fourier series representation with 129 terms
Time Domain
Freq. Domain (dB)
1 Terms
129 Terms
- 2 Terms
+ 2 Terms

```python
# Edited for ps06, question 4
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))

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

    # 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) * np.exp(1j*np.pi*k)

    return x
```