Problem Set 6 - Mitchell Kwock

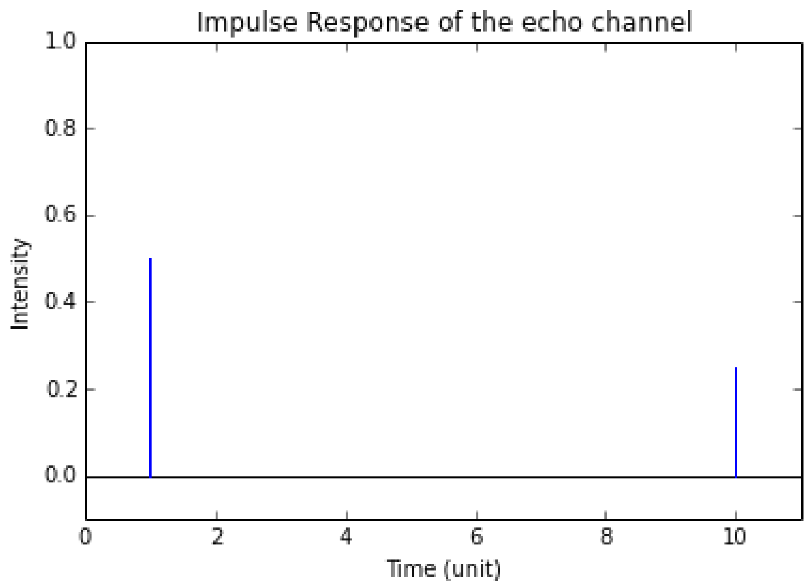
```
In [1]: # Setup Code
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline

        def plot_axes(xlim=[-1,1], ylim=[-1,1]):
            '''
            Plot the axes on a graph.  Was too lazy to type it so many times
            '''
            plt.plot(xlim, [0,0], 'k')
            plt.plot([0,0], ylim, 'k')
```

1) A shooting range can be approximated to a LTI system. A gun firing is essentially an immediate impulse, therefore, the response and echoes that it creates is the system's impulse response. Because any discretely measured samples of sound can be described as a sum of impulses, the response of the system can be modeled as the sum of those impulse responses (Shifted in time and scaled by the magnitude, of course). Conveniently, this also works wonderfully with convolution, meaning that we can multiply the frequency domain instead of convolving in the time domain.

2) We can call this an echo channel because any signal that is put in for x will return 1 step later at half its intensity and again 10 steps later at a quarter. Below, is the impulse response

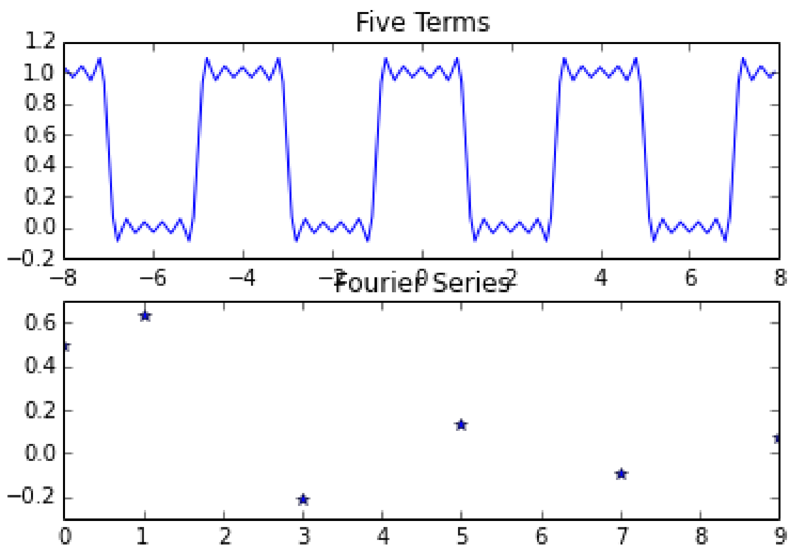
In [2]:
```
plot_axes([0, 11])

plt.plot([1, 1], [0, 0.5], 'b')
plt.plot([10, 10], [0, 0.25], 'b')
plt.title("Impulse Response of the echo channel")
plt.xlabel("Time (unit)")
plt.ylabel("Intensity")
plt.xlim([0, 11])
plt.ylim([-0.1, 1])
```

Out[2]:
```
(-0.1, 1)
```



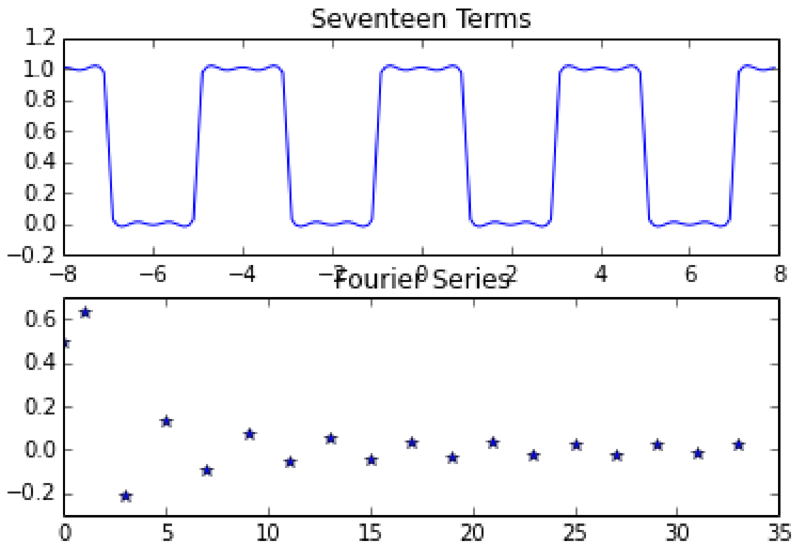
In [3]:
```
# Question 3a: Fourier Series of a Square Wave:

def square_wave_series_an(n):
    '''
    Cosine coefficient for the square wave given a frequency n
    '''
    if(n % 4 == 1):
        return 2.0 / np.pi / n
    if(n % 4 == 3):
        return -2.0 / np.pi / n
    if(n == 0):
        return 0.5
    return 0

def square_wave_series_bn(n):
    '''
    Sine coefficient for the square wave given a frequency n
    '''
    return 0

def val_at(t, freqs, an, bn, fundamental=1):
    '''
    Calculate the value at a time given a time, the frequencies, and the sine/c
osine coefficients
    t: The time to be evaluated at
    freqs: The set of included frequencies (Must be a set and integers)
    an: Function detailing the cosine coefficients
    bn: Function detailing the sine coefficients

    Example Square Wave evaluated at time 1.2:
    val_at(1.2, [1, 2, 3, 4, 5 ... n], square_wave_series_an, square_wave_serie
s_bn)
    '''
    accumulation = 0
    for freq in freqs:
        accumulation = accumulation + an(freq) * np.cos(2 * np.pi / fundamental
* freq * t) + bn(freq) * np.sin(2 * np.pi / fundamental * freq * t)
    return accumulation

# Question 3b: Plotting of 5, 17, and 257 terms

terms = 5
freqs = [0] + [2 * n + 1 for n in range(terms)]
exes = [x / 10.0 - 8 for x in range(160)]
wise = [val_at(x, freqs, square_wave_series_an, square_wave_series_bn, 4) for x
in exes]
plt.subplot(2,1,1)
plt.plot(exes, wise)
plt.title("Five Terms")
plt.subplot(2,1,2)
plt.title("Fourier Series")
plt.plot(freqs, [square_wave_series_an(freq) for freq in freqs], '*')
```

```
Out[3]: [<matplotlib.lines.Line2D at 0xae26d30>]
```



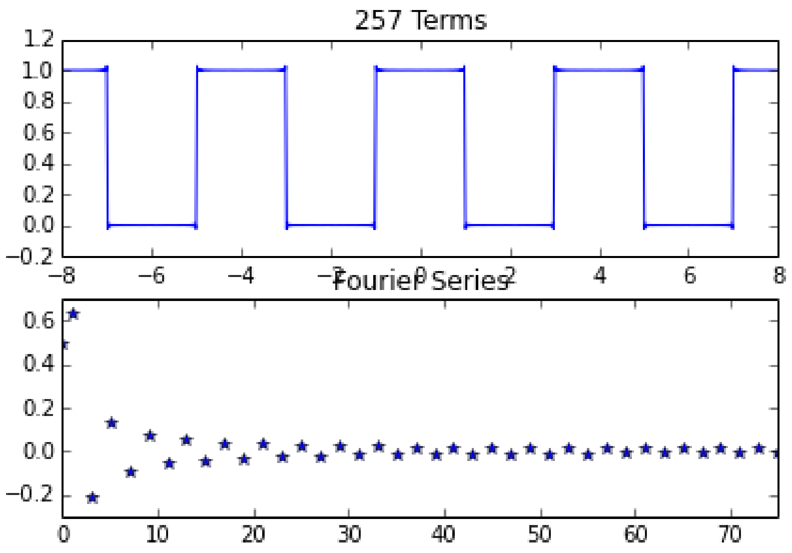
```
In [4]: terms = 17
        freqs = [0] + [2 * n + 1 for n in range(terms)]
        exes = [x / 10.0 - 8 for x in range(160)]
        wise = [val_at(x, freqs, square_wave_series_an, square_wave_series_bn, 4)for x
        in exes]
        plt.subplot(2,1,1)
        plt.plot(exes, wise)
        plt.title("Seventeen Terms")
        plt.subplot(2,1,2)
        plt.title("Fourier Series")
        plt.plot(freqs, [square_wave_series_an(freq) for freq in freqs], '*')
```

```
Out[4]: [<matplotlib.lines.Line2D at 0xb0603c8>]
```



In [5]:
```
terms = 257
freqs = [0] + [2 * n + 1 for n in range(terms)]
exes = [x / 20.0 / 4 - 8 for x in range(320 * 4)]
wise = [val_at(x, freqs, square_wave_series_an, square_wave_series_bn, 4) for x
in exes]
plt.subplot(2,1,1)
plt.plot(exes, wise)
plt.title("257 Terms")
plt.subplot(2,1,2)
plt.title("Fourier Series")
plt.plot(freqs, [square_wave_series_an(freq) for freq in freqs], '*')
plt.xlim([0, 75])
```

Out[5]:
```
(0, 75)
```



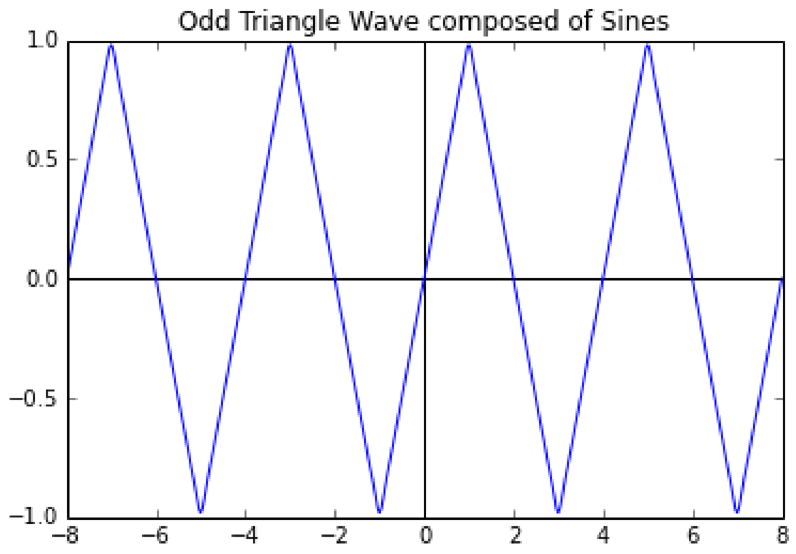
3) Describe what you see in the Fourier series representation, at the discontinuous points of the square wave, i.e. the points where the square wave goes from 1 to 0 and 0 to 1. How can you reconcile this with (10) in the Fourier series notes? Note: what you should observe is a manifestation of the Gibbs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

As we get more terms, the ears of the square waves do not disappear. They remain there no matter how many more terms are added, always overshooting the goal value of 1. The Gibbs phenomenon describes that the function will approach the original function at every point except jump discontinuities, instead offset by a finite limit, which is exactly what is happening at the edges.

In [6]:
```
def triangle_series_cs(k, L=np.pi):
    '''
    Generates an odd triangle wave about the center
    '''
    omega = 1j * k * np.pi / 2
    front = 4 / (np.pi ** 2) / (k ** 2)
    left = (-1 - omega) * np.exp(-1 * omega) # What the hell happened here?  Wh
y doesn't my math work out?
    right = (1 - omega) * np.exp( 1 * omega) # Getting rid of the omega solves
the problem.  But WHY?!

    l2 = -1 * np.exp(-1 * omega)
    r2 =  1 * np.exp( 1 * omega)
    return front * (l2 + r2)

def sum_complexes(t, freqs, comps, fundamental=1):
    '''
    Same as sum_sin_cos except uses complex exponentials and coefficients to ca
lculate
    t: Time to be evaluated at
    freqs: The frequencies checked (Must be set, skip anything that you know is
0 already to speed things up)
    comps: The Complex Coefficient generator, must be in the form func(frequenc
y_to_be_evaluated, fundamental_frequency)
    fundamental: Useful scaling factor
    '''
    accumulator = 0
    for freq in freqs:
        # added = comps(freq, fundamental) * np.exp(-2j * np.pi * freq / fundam
ental)
        # print(freq, added)
        accumulator = accumulator + comps(freq, fundamental) * np.exp(-2j * n
p.pi / fundamental * freq * t)
    return accumulator

freqs = [n + 1 for n in range(20)]
fundamental = 4
resolution = 50
exes = [1.0 * x/resolution - 2 * fundamental for x in range(4 * fundamental * i
nt(resolution) + 1)]
wise = [sum_complexes(x, freqs, triangle_series_cs, 4) for x in exes]

plot_axes([-8, 8])

plt.plot(exes, wise)
plt.title("Odd Triangle Wave composed of Sines")
```

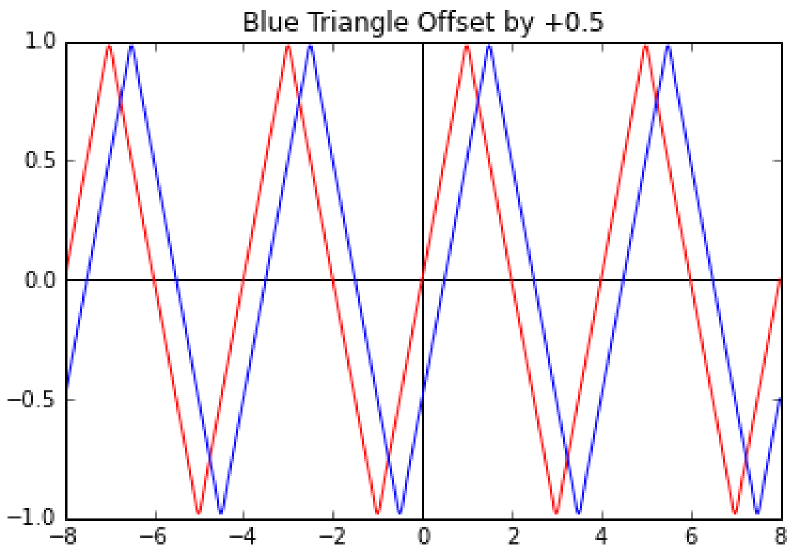
```
C:\Users\mkwock\AppData\Local\Continuum\Anaconda\lib\site-packages\numpy\core\n
umeric.py:460: ComplexWarning: Casting complex values to real discards the imag
inary part
  return array(a, dtype, copy=False, order=order)
```

```
Out[6]: <matplotlib.text.Text at 0xbcd0a58>
```



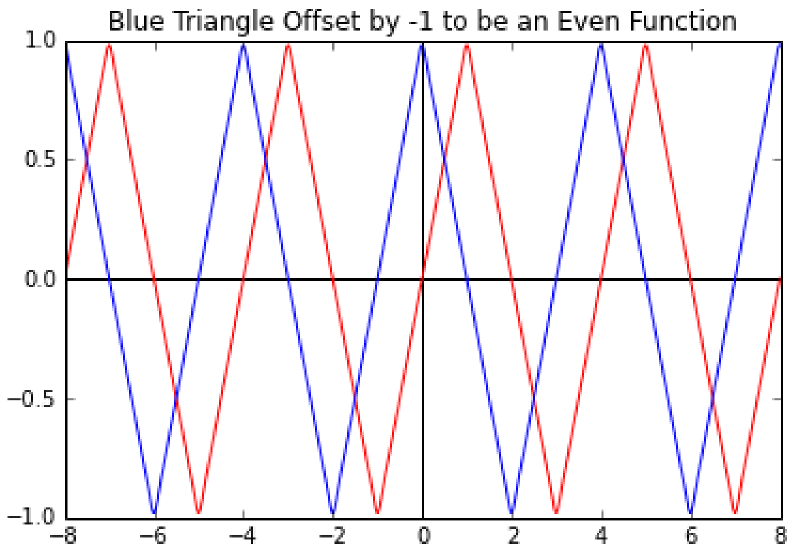
In [7]:
```
def offset_fouriers(f, k, offset, fundamental=1):
    '''
    Takes a Complex Coefficient function f and translates the wave over some of
fset by
    calculating the complex exponential for the shift and multiplying it.
    f: Original function to be offset
    k: the frequency to be evaluated
    offset: The absolute offset value
    fundamental:
    '''
    rotation = np.exp(2j * np.pi / fundamental * k * offset)
    return f(k, fundamental) * rotation

def sum_complexes_offset(t, freqs, comps, offset, fundamental=1):
    '''
    The same function as sum_complexes, only this one has an additional offset
option.
    Offset is an absolute value as opposed to a fraction of the fundamental
    '''
    accumulator = 0
    for freq in freqs:
        accumulator = accumulator + offset_fouriers(comps, freq, offset, fundam
ental) * np.exp(-2j * np.pi / fundamental * freq * t)
    return accumulator

freqs = [n + 1 for n in range(20)]
offset = 0.5
fundamental = 4
resolution = 50
exes2 = [1.0 * x/resolution - 2 * fundamental for x in range(4 * fundamental *
int(resolution) + 1)] # Display two periods to the left and right of origin
wise2 = [sum_complexes_offset(x, freqs, triangle_series_cs, offset, fundamenta
l) for x in exes]

plot_axes([-8,8])
plt.plot(exes, wise, 'r')
plt.plot(exes2, wise2, 'b')


plt.title('Blue Triangle Offset by +0.5')
```

```
Out[7]: <matplotlib.text.Text at 0xbaba128>
```



```
In [8]: freqs = [n + 1 for n in range(20)]
        offset = -1
        fundamental = 4
        resolution = 50
        exes = [1.0 * x/resolution - 2 * fundamental for x in range(4 * fundamental * i
        nt(resolution) + 1)] # Display two periods to the left and right of origin
        wise2 = [sum_complexes_offset(x, freqs, triangle_series_cs, offset, fundamenta
        l) for x in exes]

        plot_axes([-8, 8])
        plt.plot(exes, wise, 'r')
        plt.plot(exes2, wise2, 'b')

        plt.title('Blue Triangle Offset by -1 to be an Even Function')
```

```
Out[8]: <matplotlib.text.Text at 0xbe459b0>
```

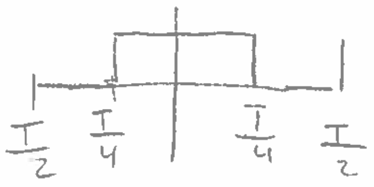


As we can see, the offset works as it is supposed to. It is surprisingly reliable and useful for any translations

```
In [8]:
```

$$a_n = \frac{1}{L}\int_{-L}^{L} f(x)\cos\left(\frac{n\pi x}{L}\right)dx \qquad f(x) = \begin{cases} 1, & |x| < \frac{L}{4} \\ 0, & \text{otherwise} \end{cases}$$

$$L = T/2$$



$$a_n = \frac{2}{L}\int_0^{T/2} f(x)\cos\left(\frac{n\pi x}{L}\right)dx$$

$$a_n = \frac{2}{L}\left(\int_0^{\frac{T}{4}}\cos\left(\frac{n\pi x}{L}\right)dx + \int_{\frac{T}{4}}^{\frac{T}{2}} \cancel{0\cos\left(\frac{n\pi x}{L}\right)dx}^{\,0}\right)$$

$$a_n = \frac{2}{\cancel{L}}\cdot\left(\frac{\cancel{L}}{n\pi}\sin\left(\frac{n\pi x}{L}\right)\right)\Bigg]_0^{\frac{T}{4}}$$

$$a_n = \frac{2}{n\pi}\left[\sin\left(\frac{n\pi T}{4L}\right) - \cancel{\sin(0)}^{\,0}\right] \qquad L = \frac{T}{2}$$

$$a_n = \frac{2}{n\pi}\sin\left(\frac{n\pi}{2}\right)$$

$$\begin{matrix} a_n = \\ \wedge \\ (\text{cosine}) \end{matrix} \begin{cases} 0, & n \text{ even} \\ \frac{2}{n\pi} & n \equiv 1 \pmod 4 \\ -\frac{2}{n\pi} & n \equiv 3 \pmod 4 \end{cases} \qquad b_n = 0, \text{ even}$$

$\nwarrow$ Square Wave

$$a_n = \frac{1}{L}\int_{-L}^{L} f(x)$$

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

~~$u =$~~ $\frac{T-2W}{2}$ $\frac{-T-2W}{2}$

$u_0 = t - T_1$

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

$$\frac{1}{T}\int_{-\frac{T}{2}-W}^{T/2-W} x(t-T_1)\, e^{-j\frac{2\pi}{T}k(t-T_1)}\, dt$$

$$\frac{1}{T}\int$$

$W = T_1$

$$C_k = \frac{1}{T}\int_{-\frac{T_2}{2}-T_1}^{\frac{T_2}{2}-T_1} x(t-T_1)\, e^{-j\frac{2\pi}{T}k(t-T_1)}\, dt$$

$$L^2 \cdot \frac{\left(\frac{1}{2} - -j\pi k - 1\right)}{-\pi^2 k^2}$$

multiples by $e^{+j\frac{2\pi}{T}kT_1}$

$\text{cis}\left(\frac{2\pi k}{T} T_1\right)$

~~$C_y = C_k$~~

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

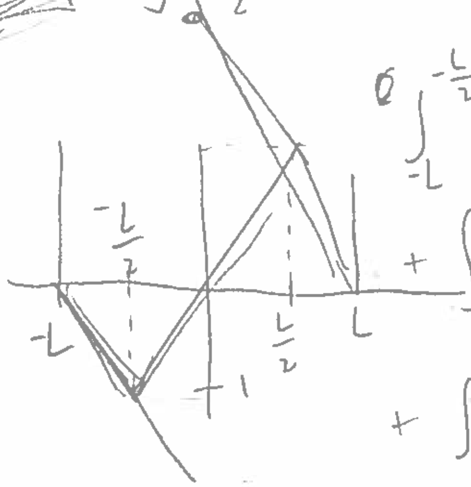
T_1

$$C_k = \frac{1}{L}\int_{-L}^{L} f(x) e^{-j\frac{2\pi}{T}kt}\, dL$$

~~$C_y e^{j\frac{2\pi}{T}T_1 y}$~~ 2^2

$$C_k = \frac{C_y}{e^{+j\frac{2\pi}{T}T_1}} =$$

$$C_y = C_k e^{j\frac{2\pi}{T}yT_1}$$



$$\int_{-L}^{-\frac{L}{2}} \left(-\frac{2}{L}x - \frac{4}{L}\right) e^{\sim}\, dt$$

$$+ \int_{-\frac{L}{2}}^{\frac{L}{2}} \frac{2}{L}x\, e^{\sim}\, dt$$

$$+ \int$$

$$\frac{2}{L}\left[\int_{-\frac{L}{2}}^{-L} t e^{-jk\frac{1}{2L}t}\right. \qquad \omega = \frac{1}{2L}$$

$$-j\omega t k$$

$$\frac{1}{L}\left[\int_{-L}^{-\frac{L}{2}}\left(-\frac{2}{L}t-2\right)e^{-j\frac{2\pi}{2L}kt}dt + \int_{-\frac{L}{2}}^{\frac{L}{2}}\left(\frac{2}{L}t\right)e^{-j\frac{2\pi}{2L}kt}dt\right. \qquad \omega = \frac{-jk}{2L}\cdot 2\pi$$

$$\left.+\int_{\frac{L}{2}}^{L}\left(-\frac{2}{L}+2\right)e^{-j\frac{2\pi}{2L}kt}dt\right]$$

$$-\frac{2}{L}\int_{-L}^{-\frac{L}{2}}\frac{\cancel{2}t}{\cancel{L}}e^{-j\frac{\pi}{L}kt}dt + \boxed{\int_{\frac{L}{2}}^{\frac{L}{2}}\frac{-4}{L}e^{j\frac{\pi}{L}kt}dt}$$

$$-\frac{4}{2}$$

$$\frac{2}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}}\frac{\cancel{2}t}{\cancel{L}}e^{-j\frac{\pi}{L}kt}dt \qquad \text{Cancels}$$

$$-\frac{2}{L}\int_{\frac{L}{2}}^{L}\frac{\cancel{2}t}{\cancel{L}}e^{-j\frac{\pi}{L}kt}dt + \boxed{\int_{\frac{L}{2}}^{L}\frac{+4}{L}e^{j\frac{\pi}{L}kt}}$$

~~$u = e^{\omega t}$ $dv = t\,dt$~~

~~$du = \omega e^{\omega t}$ $v = \frac{1}{2}t^2$~~

~~$\frac{1}{2}t^2 e^{\omega t} \int$~~

$$\omega = -j\frac{\pi}{L}k$$

$$u = t \qquad dv = e^{\omega t}$$

$$du = dt \qquad v = \frac{1}{\omega}e^{\omega t}$$

$$\frac{t}{\omega}e^{\omega t} - \int\frac{1}{\omega}e^{\omega t} = \frac{t}{\omega}e^{\omega t} - \frac{1}{\omega^2}e^{\omega t}$$

$$= \left(\frac{t\omega - 1}{\omega^2}\right)e^{\omega t}\Big]$$

$$\omega^2 = -\frac{\pi^2}{L^2}k^2$$

$$-\frac{L}{2} \qquad -L$$

$$-L \to -\frac{L}{2} \quad \left(\frac{\frac{L}{2}\cdot -j\frac{\pi}{L}k - 1}{-\frac{\pi^2}{L^2}k^2}e^{+j\frac{\pi}{2}k}\right) - \left(\frac{+j\pi k - 1}{-\frac{\pi^2}{L^2}k^2}e^{+j\pi k}\right)\cdot -1 \qquad \frac{L}{2}\cdot\frac{-jk}{2L} \quad \frac{-jk}{4}$$

$$\uparrow$$

$$\frac{+j\pi k/2 - 1}{-\frac{\pi^2}{L^2}k^2} \qquad \left(\frac{-\frac{L}{2}\omega - 1}{\omega^2}e^{\omega\frac{L}{2}}\right) - \left(\frac{-L\omega - 1}{\omega^2}e^{\omega L}\right) \qquad \frac{\frac{L}{2}\omega - 1}{\omega}$$

$$-\frac{L}{2} \to \frac{L}{2} \quad \left(\frac{-j\pi k/2 - 1}{\omega^2}e^{-j\frac{\pi}{2}k}\right) - \left(\frac{j\pi k/2 - 1}{\omega^2}e^{j\frac{\pi}{2}k}\right)\cdot\cancel{2} \qquad \begin{matrix}-\left(-\frac{L}{2}\right)+\left(\cancel{-L}\right)\\ \left(\frac{L}{2}\right)-\left(-\frac{L}{2}\right)\\ -\left(\cancel{L}\right)+\left(\frac{L}{2}\right)\end{matrix}$$

$$\frac{L}{2} \to L \quad \left(\frac{-j\pi k - 1}{\omega^2}e^{j\pi k}\right) - \left(\frac{-j\pi k/2 - 1}{\omega^2}e^{-j\frac{\pi}{2}k}\right)\cdot -1 \qquad 2\left(\frac{L}{2}\right) \cdot 2\left(\frac{L}{2}\right)$$

$$\frac{2}{L}\left[2\left(\frac{-jk\pi/2 \quad -1}{-\pi^2 k^2/L^2}\, e^{-jk\pi/2} \right) - 2\left(\frac{jk\pi/2 \quad -1}{-\pi^2 k^2/L^2}\, e^{jk\pi/2} \right) \right]$$

$$\frac{4}{L} \cdot \frac{L^2}{\pi^2 k^2}\left[(-jk\pi/2 \; -1)\, e^{-jk\pi/2} + (1 - jk\pi/2)\, e^{jk\pi/2} \right]$$

Triangle Wave?

$$\boxed{C_k = \frac{4}{\pi^2 k^2}\left[-1 \cdot e^{-jk\pi/2} + e^{jk\pi/2} \right]}$$