Signals & Systems
Homework #4 Solutions

ECE 315 – Fall 2022
195 points total

Due Wednesday, November 30

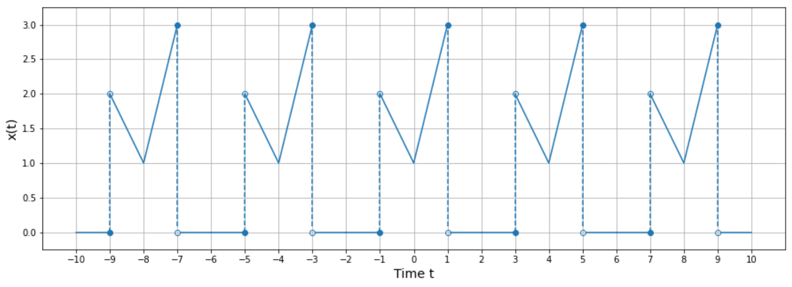
1. Consider the periodic continuous-time signal

$$x(t) = g(t) * \delta_4(t)$$

where

$$g(t) = \begin{cases} 1-t, & -1 < t \leq 0 \\ 2t+1, & 0 < t \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

The signal $x(t)$ is shown below.



(a) Determine the continuous-time Fourier series (CTFS) coefficients for $x(t)$ by hand. Hints: This will require integration by parts. The final result will be complicated. (16 points)

The period of $x(t)$ is $T = 4$ and

$$\begin{aligned}
c_x[k] &= \frac{1}{T}\int_T x(t)e^{-j2\pi kt/T}\,dt \\
&= \frac{1}{4}\int_{-2}^{2} x(t)e^{-j(\pi/2)kt}\,dt = \frac{1}{4}\int_{-1}^{1} g(t)e^{-j(\pi/2)kt}\,dt \\
&= \frac{1}{4}\left(\int_{-1}^{0}(1-t)e^{-j(\pi/2)kt}\,dt + \int_0^1 (2t+1)e^{-j(\pi/2)kt}\,dt\right) \\
&= \frac{1}{4}\left(\int_{-1}^{1} e^{-j(\pi/2)kt}\,dt - \int_{-1}^{0} te^{-j(\pi/2)kt}\,dt + 2\int_0^1 te^{-j(\pi/2)kt}\,dt\right).
\end{aligned}$$

This means that there are two integrals to consider

$$\int e^{-j(\pi/2)kt}\,dt \quad \text{and} \quad \int te^{-j(\pi/2)kt}\,dt.$$

There are also two cases.

The first case is $k = 0$. Here, the Fourier series coefficient is

$$\begin{aligned}
c_x[0] &= \frac{1}{4}\left(\int_{-1}^{1} dt - \int_{-1}^{0} t\,dt + 2\int_0^1 t\,dt\right) = \frac{1}{4}\left(t\Big|_{-1}^{1} - \frac{t^2}{2}\bigg|_{-1}^{0} + 2\left(\frac{t^2}{2}\right)\bigg|_0^1\right) \\
&= \frac{1}{4}\left(2 + \frac{1}{2} + 1\right) = \frac{7}{8}.
\end{aligned}$$

The other case is $k \neq 0$. In this case, the first integral is

$$\begin{aligned}
\int e^{-j(\pi/2)kt}\,dt &= -\frac{1}{j(\pi/2)k}e^{-j(\pi/2)kt} + c,\ \ c = \text{constant} \\
&= \frac{2j}{\pi k}e^{-j(\pi/2)kt} + c.
\end{aligned}$$

The second integral

$$\int te^{-j(\pi/2)kt}\,dt$$

can be evaluated using integration by parts. Let

$$u = t \text{ and } dv = e^{-j(\pi/2)kt}\,dt.$$

Then

$$du = dt, \quad v = -\frac{1}{j(\pi/2)k}e^{-j(\pi/2)kt} = \frac{2j}{\pi k}e^{-j(\pi/2)kt},$$

and

$$\begin{aligned}\int te^{-j(\pi/2)kt}\,dt &= \frac{2j}{\pi k}te^{-j(\pi/2)kt} - \int\left(\frac{2j}{\pi k}e^{-j(\pi/2)kt}\right)\,dt\\ &= \frac{2j}{\pi k}\left(te^{-j(\pi/2)kt} - \int e^{-j(\pi/2)kt}\,dt\right)\\ &= \frac{2j}{\pi k}\left(te^{-j(\pi/2)kt} - \left(\frac{2j}{\pi k}e^{-j(\pi/2)kt} + c\right)\right), c = \text{ constant}\\ &= \frac{2j}{\pi k}\left(te^{-j(\pi/2)kt} - \frac{2j}{\pi k}e^{-j(\pi/2)kt}\right) + c.\end{aligned}$$

Altogether,

$$\begin{aligned}c_x[k] = \frac{1}{4}&\left(\frac{2j}{\pi k}e^{-j(\pi/2)kt}\bigg|_{-1}^{1} - \frac{2j}{\pi k}\left(te^{-j(\pi/2)kt} - \frac{2j}{\pi k}e^{-j(\pi/2)kt}\right)\bigg|_{-1}^{0}\right.\\ &\left.+\frac{4j}{\pi k}\left(te^{-j(\pi/2)kt} - \frac{2j}{\pi k}e^{-j(\pi/2)kt}\right)\bigg|_{0}^{1}\right)\\ = \frac{1}{4}&\left(\frac{2j}{\pi k}\left(e^{-j(\pi/2)k} - e^{j(\pi/2)k}\right) - \frac{2j}{\pi k}\left(e^{j(\pi/2)k} - \frac{2j}{\pi k}\left(1 - e^{j(\pi/2)k}\right)\right)\right.\\ &\left.+\frac{4j}{\pi k}\left(e^{-j(\pi/2)k} - \frac{2j}{\pi k}\left(e^{-j(\pi/2)k} - 1\right)\right)\right)\end{aligned}$$

Here, the first term is

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

Furthermore,

$$e^{j(\pi/2)k} = \left(e^{j(\pi/2)}\right)^k, \quad e^{-j(\pi/2)k} = \left(e^{-j(\pi/2)}\right)^k,$$

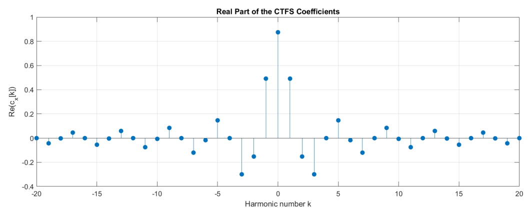
$$\begin{aligned}e^{j(\pi/2)} &= \cos\left(\frac{\pi}{2}\right) + j\sin\left(\frac{\pi}{2}\right) = j,\\ e^{-j(\pi/2)} &= \cos\left(\frac{\pi}{2}\right) - j\sin\left(\frac{\pi}{2}\right) = -j,\end{aligned}$$

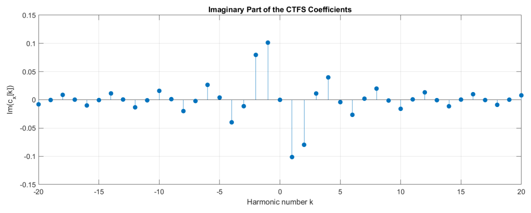
so that

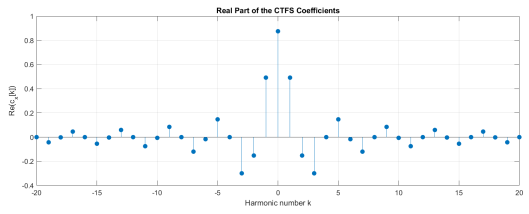
$$\begin{aligned}c_x[k] &= \frac{1}{4}\left(\frac{4}{\pi k}\sin\left(\frac{\pi}{2}k\right) - \frac{2j}{\pi k}\left(j^k - \frac{2j}{\pi k}\left(1 - j^k\right)\right) + \frac{4j}{\pi k}\left((-j)^k - \frac{2j}{\pi k}\left((-j)^k - 1\right)\right)\right)\\ &= \frac{\sin\left(\frac{\pi}{2}k\right)}{\pi k} - \frac{1}{2\pi k}\left(j^{k+1} + \frac{2}{\pi k}\left(1 - j^k\right)\right) + \frac{1}{\pi k}\left(j(-j)^k + \frac{2}{\pi k}\left((-j)^k - 1\right)\right).\end{aligned}$$

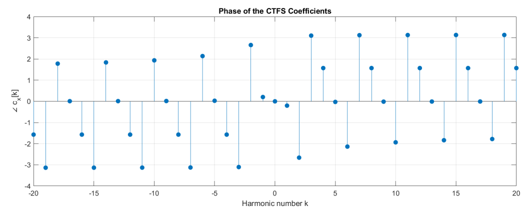
(b) Use MATLAB or python to plot the real and imaginary parts and the magnitude and phase of the CTFS coefficients for harmonic numbers $k = -20, -19, \cdots, 20$. Include your code as part of your solution. (8 points)

MATLAB code for plotting the real and imaginary parts and magnitude and phase of $c_x[k]$ is in Appendix A. The output from this code is below. Notice that these quantities have the expected symmetries.

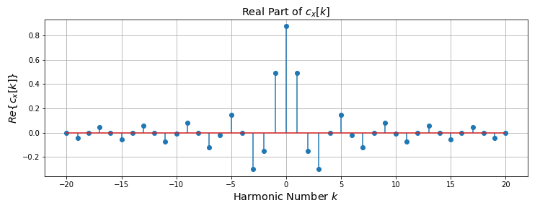


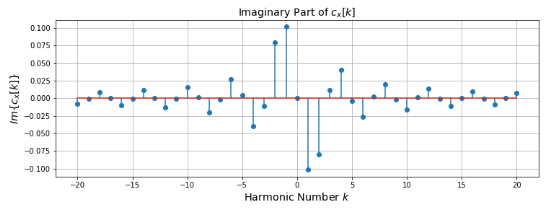


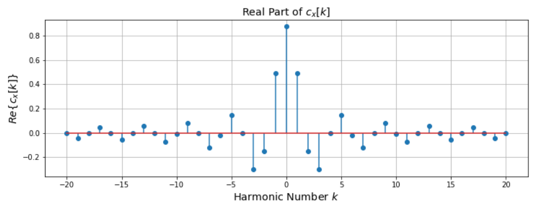


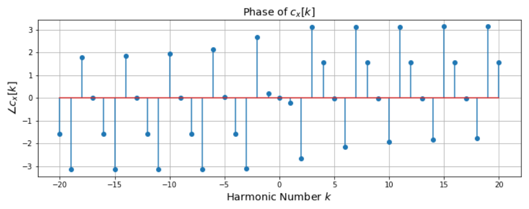


Python code for plotting the real and imaginary parts and magnitude and phase of $c_x[k]$ is in Appendix B. The output from this code is below. Notice that these quantities have the expected symmetries and match the plots created using MATLAB.







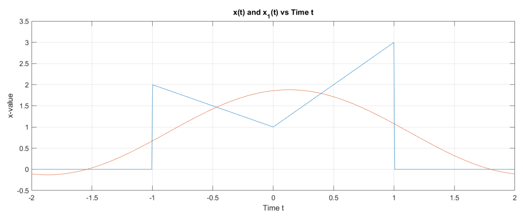


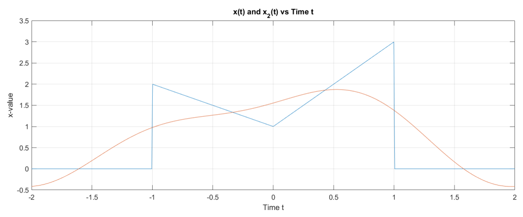
(c) Use MATLAB or python to form the partial sums $x_N(t)$ of the CTFS and plot the partial sums with the signal $x(t)$ for $N = 1, 2, 5, 10, 50$ and for $-2 \leq t \leq 2$. Include your MATLAB or python code as part of your solution. (16 points)

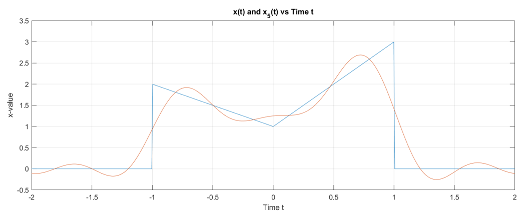
The N^{th} partial sum is

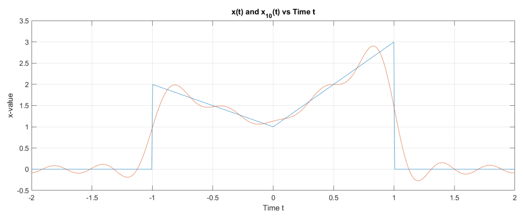
$$x_N(t) = \sum_{k=-N}^{N} c_x[k] e^{j2\pi kt/T} = \sum_{k=-N}^{N} c_x[k] e^{j(\pi/2)kt}$$

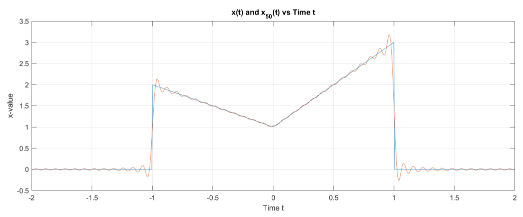
MATLAB code for evaluating the partial sums is in Appendix C. The output from this code is below.



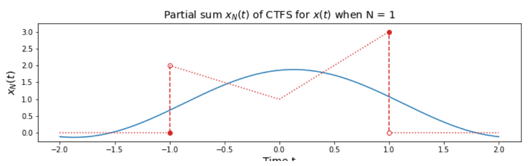


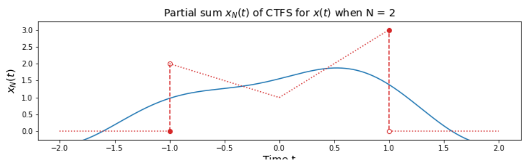


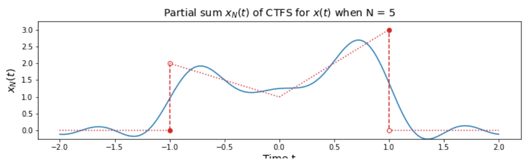


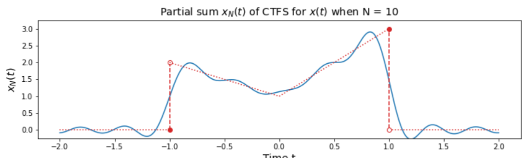


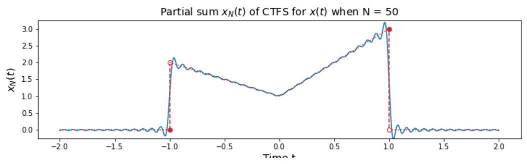
Python code for evaluating the partial sums is in Appendix D. The output from this code is below.











2. Consider the system

$$y''(t) - 2y'(t) - 3y(t) = x(t).$$

(a) Use the Fourier properties to determine the harmonic response for the system. (The "harmonic response" is defined in Example 6.2 on p. 246 of the textbook.) (6 points)

Suppose that the input signal $x(t)$ is periodic with period T and Fourier series coefficients $c_x[k]$. Then the output signal $y(t)$ is periodic with the same period. The properties needed are listed in Table 6.1 on p. 245 of the textbook. Specifically, these are the linearity and time differentiation properties.

Let the Fourier series coefficients for $y(t)$ be $c_y[k]$. The derivative $\frac{d}{dt}y(t)$ is also periodic with period T and its Fourier series coefficients are

$$\frac{j2\pi k}{T}c_y[k].$$

Similarly, since $\frac{d^2}{dt^2}y(t) = \frac{d}{dt}\left(\frac{d}{dt}y(t)\right)$, the second derivative is periodic with period T, as well, and its Fourier series coefficients are

$$\frac{j2\pi k}{T}\left(\frac{j2\pi k}{T}c_y[k]\right) = -\frac{4\pi^2k^2}{T^2}c_y[k]$$

Altogether, the Fourier series coefficients associated with both sides of the system equation satisfy

$$-\frac{4\pi^2k^2}{T^2}c_y[k] - 2\frac{j2\pi k}{T}c_y[k] - 3c_y[k] = c_x[k]$$

or, equivalently,

$$\left(-\frac{4\pi^2k^2}{T^2} - 2\frac{j2\pi k}{T} - 3\right)c_y[k] = c_x[k].$$

Therefore, the harmonic response is

$$H[k] = \frac{c_y[k]}{c_x[k]} = \frac{1}{-\frac{4\pi^2k^2}{T^2} - 2\frac{j2\pi k}{T} - 3}.$$

(b) Determine the frequency response of the system. How is the harmonic response related to the frequency response? (6 points)

For this system,

$$a_2 = 1, a_1 = -2, a_0 = -3, \text{ and } b_0 = 1.$$

Therefore, the transfer function for the system is

$$H(s) = \frac{\sum_{k=0}^{M} b_k s^k}{\sum_{k=0}^{N} a_k s^k} = \frac{1}{s^2 - 2s - 3}$$

and the frequency response is

$$H(j\omega) = \frac{1}{(j\omega)^2 - 2(j\omega) - 3} = \frac{1}{-\omega^2 - 2j\omega - 3}.$$

The harmonic response is the frequency response evaluated at the frequencies

$$\omega_k = \frac{2\pi k}{T}.$$

(c) Use the result from part (a) and the Fourier series pairs to determine the response of the system to the signal

$$x(t) = \operatorname{tri}(t/2) * \delta_5(t). \text{ (6 points)}$$

Based on an entry in Table 6.2 on p. 245 of the textbook, the Fourier series coefficients of the input signal $x(t)$ are

$$c_x[k] = \left(\frac{2}{T}\right) \operatorname{sinc}^2\left(\frac{2k}{T}\right) \delta_1[k] = \left(\frac{2}{T}\right) \operatorname{sinc}^2\left(\frac{2k}{T}\right),$$

where the values $w = 2$ and assuming that T is the fundamental period of $x(t)$ so that $m = 1$. (Here, $\delta_1[k]$ is the discrete-time unit impulse train with period 1, which has the value 1 at all k.) Therefore, the Fourier series coefficients $c_y[k]$ of the output of the system are

$$c_y[k] = H[k]c_x[k] = \frac{\left(\frac{2}{T}\right) \operatorname{sinc}^2\left(\frac{2k}{T}\right)}{-\frac{4\pi^2k^2}{T^2} - 2\frac{j2\pi k}{T} - 3}$$

and the output of the system is

$$y(t) = \sum_{k=-\infty}^{\infty} c_y[k]e^{j2\pi kt/T} = \sum_{k=-\infty}^{\infty} \frac{\left(\frac{2}{T}\right) \operatorname{sinc}^2\left(\frac{2k}{T}\right)}{-\frac{4\pi^2k^2}{T^2} - 2\frac{j2\pi k}{T} - 3} e^{j2\pi kt/T}.$$

3. Consider the continuous-time signal

$$x(t) = u(t)t^3e^{-t}.$$

(a) Derive the continuous-time Fourier transform (CTFT) of $x(t)$ as a function of the angular frequency ω, as listed in Table 6.3 on p. 260 in the textbook. (10 points)

By definition, the Fourier transform of $x(t)$ is

$$\begin{aligned} X(j\omega) &= \int_{-\infty}^{\infty} x(t)e^{-j\omega t}\, dt \\ &= \int_0^{\infty} t^3e^{-t}e^{-j\omega t}\, dt = \int_0^{\infty} t^3e^{-(1+j\omega)t}\, dt. \end{aligned}$$

This can be evaluated using integration by parts. Let

$$u = t^3 \quad \text{and} \quad dv = e^{-(1+j\omega)t}\,dt.$$

Then

$$du = 3t^2 dt, \quad v = -\frac{1}{1+j\omega}e^{-(1+j\omega)t},$$

and

$$\begin{aligned} X(j\omega) &= -\frac{1}{1+j\omega}t^3 e^{-(1+j\omega)t}\bigg|_0^\infty - \int_0^\infty \left(-\frac{1}{1+j\omega}e^{-(1+j\omega)t}\right) 3t^2\,dt \\ &= \frac{3}{1+j\omega}\int_0^\infty t^2 e^{-(1+j\omega)t}\,dt. \end{aligned}$$

This time, let

$$u = t^2 \quad \text{and} \quad dv = e^{-(1+j\omega)t}\,dt.$$

Then

$$du = 2tdt, \quad v = -\frac{1}{1+j\omega}e^{-(1+j\omega)t},$$

and

$$\begin{aligned} X(j\omega) &= \frac{3}{1+j\omega}\left(-\frac{1}{1+j\omega}t^2 e^{-(1+j\omega)t}\bigg|_0^\infty - \int_0^\infty \left(-\frac{1}{1+j\omega}e^{-(1+j\omega)t}2t\,dt\right)\right) \\ &= \frac{6}{(1+j\omega)^2}\int_0^\infty t e^{-(1+j\omega)t}\,dt. \end{aligned}$$

Finally, let

$$u = t \quad \text{and} \quad dv = e^{-(1+j\omega)t}\,dt.$$

Then

$$du = dt, \quad v = -\frac{1}{1+j\omega}e^{-(1+j\omega)t},$$

and

$$\begin{aligned} X(j\omega) &= \frac{6}{(1+j\omega)^2}\left(-\frac{1}{1+j\omega}te^{-(1+j\omega)t}\bigg|_0^\infty - \int_0^\infty \left(-\frac{1}{1+j\omega}e^{-(1+j\omega)t}\right)\,dt\right) \\ &= \frac{6}{(1+j\omega)^3}\int_0^\infty e^{-(1+j\omega)t}\,dt = \frac{6}{(1+j\omega)^3}\left(-\frac{1}{1+j\omega}e^{-(1+j\omega)t}\bigg|_0^\infty\right) \\ &= \frac{6}{(1+j\omega)^4}. \end{aligned}$$

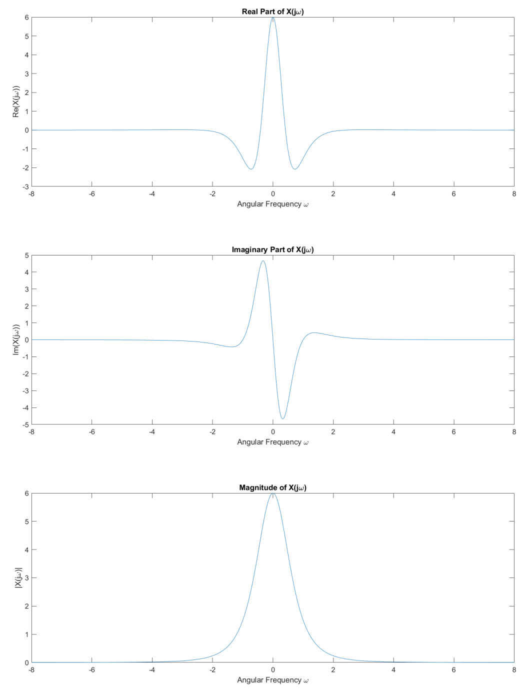
Table 6.3 contains the Fourier pair

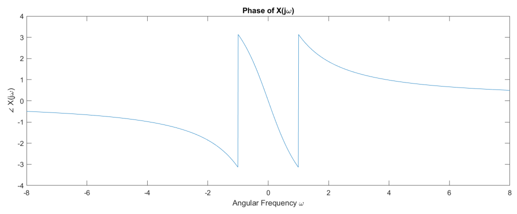
$$t^n e^{-\alpha t} u(t) \overset{\mathcal{F}}{\longleftrightarrow} \frac{n!}{(j\omega+\alpha)^{n+1}}, \alpha > 0.$$

The time-domain signal matches $x(t)$ with $n = 3$ and $\alpha = 1$. Substituting these values into the frequency-domain signal on the right-hand side of the Fourier pair reproduces the expression for $X(j\omega)$ derived above.

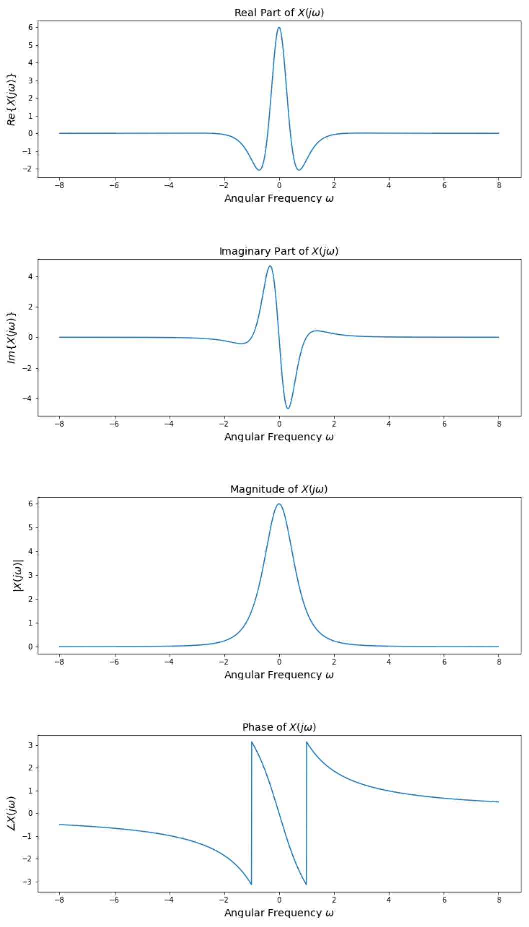
(b) Use MATLAB or python to determine the real and imaginary parts and the magnitude and phase of the CTFT and plot them as functions of the angular frequency ω for $-8 \leq \omega \leq 8$. Include your code as part of your solution. (20 points)

MATLAB code for plotting the CTFT is in Appendix E. The output from this code is below.



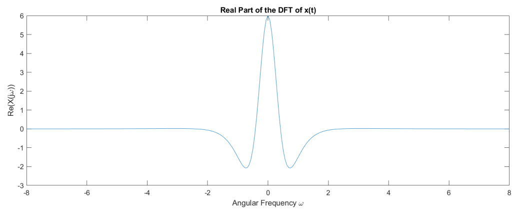


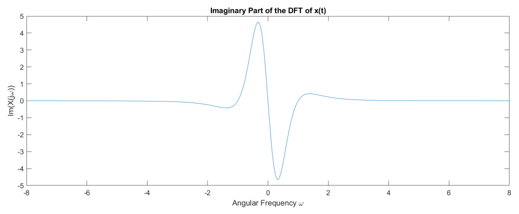
Python code for plotting the CTFT is in Appendix F. The output from this code is below.

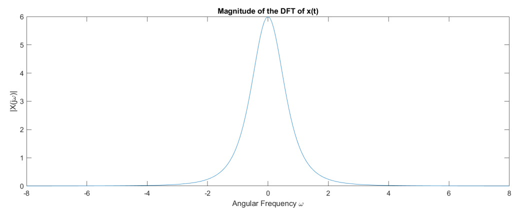


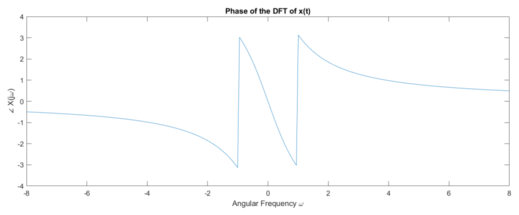
(c) Use the MATLAB function `fft` or the NumPy function `fft`, which calculate the discrete Fourier transform, to approximate the CTFT of $x(t)$. Create the approximation using $N = 262,144$ samples of $x(t)$ over the time interval $0 \leq t \leq 100$. Use MATLAB or python to plot the real and imaginary parts and the magnitude and phase of the approximation as a function of angular frequency ω for $-8 \leq \omega \leq 8$. Include your code in your solution. (10 points)

MATLAB code for calculating and plotting the DFT is in Appendix G. The output from this code follows. Notice that these graphs are very similar to the graphs plotted in Part (b), with small differences near sharp transitions.

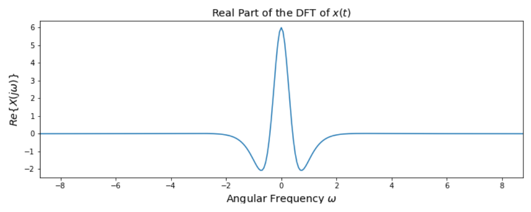


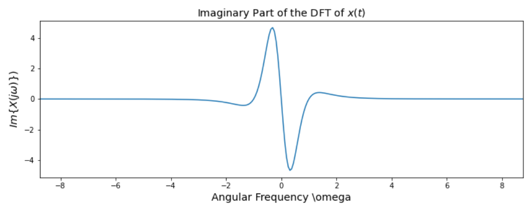


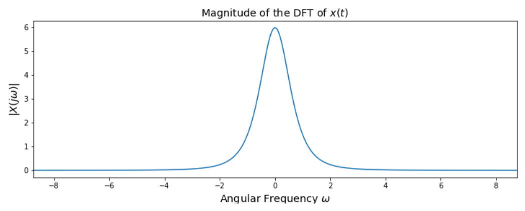


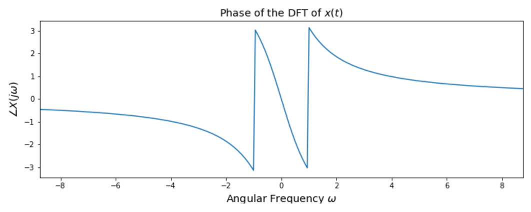


Python code for calculating and plotting the DFT is in Appendix H. The output from this code follows. Notice that these graphs are very similar to the graphs plotted in Part (b), with small differences near sharp transitions.









4. Consider the system

$$y''(t) - 5y'(t) + 6y(t) = 2x'(t) - 3x(t).$$

(a) Find the frequency response for the system. (6 points)

For this system,

$$a_2 = 1, a_1 = -5, a_0 = 6, b_1 = 2, b_0 = -3, N = 2, \text{ and } M = 1.$$

So, the transfer function is

$$H(s) = \frac{\sum_{k=0}^{M} b_k s^k}{\sum_{k=0}^{N} a_k s^k} = \frac{2s - 3}{s^2 - 5s + 6}$$

and the frequency response is

$$H(j\omega) = \frac{2j\omega - 3}{(j\omega)^2 - 5j\omega + 6} = \frac{2j\omega - 3}{-\omega^2 - 5j\omega + 6}.$$

(b) Use a partial fractions expansion and a table of CTFT pairs from the textbook to determine the response $y(t)$ of the system when the input is

$$x(t) = u(t)e^{-\frac{t}{5}} \sin(2\pi t). \text{ (16 points)}$$

Table 6.3 on p. 260 contains the Fourier transform pair

$$e^{-\alpha t} \sin(\omega_0 t)u(t) \overset{\mathcal{F}}{\longleftrightarrow} \frac{\omega_0}{(j\omega + \alpha)^2 + \omega_0^2}, \alpha > 0.$$

The signal $x(t)$ has the form of the time-domain signal in this pair with

$$\alpha = \frac{1}{5}, \omega_0 = 2\pi.$$

Therefore,

$$X(j\omega) = \frac{2\pi}{(j\omega + \frac{1}{5})^2 + (2\pi)^2}.$$

The Fourier transform of the output is

$$Y(j\omega) = H(j\omega)X(j\omega) = \frac{2\pi(2j\omega - 3)}{((j\omega)^2 - 5j\omega + 6)\left((j\omega + \frac{1}{5})^2 + (2\pi)^2\right)}.$$

It is easier to find a partial fractions expansion for $Y(j\omega)$ when it is expressed in terms of $s = j\omega$.. In this case,

$$\begin{aligned} Y(s) &= \frac{2\pi(2s - 3)}{(s^2 - 5s + 6)\left((s + \frac{1}{5})^2 + (2\pi)^2\right)} \\ &= \frac{2\pi(2s - 3)}{(s - 2)(s - 3)\left(s + \frac{1}{5} + j2\pi\right)\left(s + \frac{1}{5} - j2\pi\right)} \\ &= 2\pi Y_1(s), \end{aligned}$$

where

$$Y_1(s) = \frac{2s - 3}{(s - 2)(s - 3)\left(s + \frac{1}{5} + j2\pi\right)\left(s + \frac{1}{5} - j2\pi\right)}.$$

The goal is to write $Y_1(s)$ in the form

$$Y_1(s) = \frac{A_0}{s-2} + \frac{A_1}{s-3} + \frac{A_2}{s+\frac{1}{5}+j2\pi} + \frac{A_3}{s+\frac{1}{5}-j2\pi},$$

where A_0, A_1, A_2, and A_3 are constants. Then

$$\begin{aligned}
y(t) &= \mathcal{F}^{-1}\left\{Y(j\omega)\right\} \\
&= 2\pi\mathcal{F}^{-1}\left\{Y_1(j\omega)\right\} \\
&= 2\pi\mathcal{F}^{-1}\left\{\frac{A_0}{j\omega-2} + \frac{A_1}{j\omega-3} + \frac{A_2}{j\omega+\frac{1}{5}+j2\pi} + \frac{A_3}{j\omega+\frac{1}{5}-j2\pi}\right\} \\
&= 2\pi\left(A_0\mathcal{F}^{-1}\left\{\frac{1}{j\omega-2}\right\} + A_1\mathcal{F}^{-1}\left\{\frac{1}{j\omega-3}\right\}\right. \\
&\qquad \left. +A_2\mathcal{F}^{-1}\left\{\frac{1}{j\omega+\frac{1}{5}+j2\pi}\right\} + A_3\mathcal{F}^{-1}\left\{\frac{1}{j\omega+\frac{1}{5}-j2\pi}\right\}\right) \\
&= 2\pi\left(-A_0e^{2t}u(-t) - A_1e^{3t}u(-t)\right. \\
&\qquad \left. +A_2\mathcal{F}^{-1}\left\{\frac{1}{j\omega+\frac{1}{5}+j2\pi}\right\} + A_3\mathcal{F}^{-1}\left\{\frac{1}{j\omega+\frac{1}{5}-j2\pi}\right\}\right)
\end{aligned}$$

Here,

$$\begin{aligned}
A_0 &= \left.(s-2)Y_1(s)\right|_{s=2} \\
&= \left.\frac{2s-3}{(s-3)\left(s+\frac{1}{5}+j2\pi\right)\left(s+\frac{1}{5}-j2\pi\right)}\right|_{s=2} \\
&= \frac{1}{(-1)\left(\frac{11}{5}+j2\pi\right)\left(\frac{11}{5}-j2\pi\right)} \\
&= -\frac{1}{\left(\frac{121}{25}+4\pi^2\right)},
\end{aligned}$$

$$\begin{aligned}
A_1 &= (s-3)Y_1(s)\big|_{s=3} \\
&= \left.\frac{2s-3}{(s-2)\left(s+\frac{1}{5}+j2\pi\right)\left(s+\frac{1}{5}-j2\pi\right)}\right|_{s=3} \\
&= \frac{3}{\left(\frac{16}{5}+j2\pi\right)\left(\frac{16}{5}-j2\pi\right)} \\
&= \frac{3}{\left(\frac{256}{25}+4\pi^2\right)},
\end{aligned}$$

$$\begin{aligned}
A_2 &= \left(s+\frac{1}{5}+j2\pi\right)Y_1(s)\Big|_{s=-\frac{1}{5}-j2\pi} \\
&= \left.\frac{2s-3}{(s-2)(s-3)\left(s+\frac{1}{5}-j2\pi\right)}\right|_{s=-\frac{1}{5}-j2\pi} \\
&= \frac{2\left(-\frac{1}{5}-j2\pi\right)-3}{\left(-\frac{1}{5}-j2\pi-2\right)\left(-\frac{1}{5}-j2\pi-3\right)\left(-\frac{1}{5}-j2\pi+\frac{1}{5}-j2\pi\right)} \\
&= \frac{-\frac{17}{5}-j4\pi}{\left(-\frac{11}{5}-j2\pi\right)\left(-\frac{16}{5}-j2\pi\right)(-j4\pi)} \\
&= \frac{\frac{17}{5}+j4\pi}{\left(\frac{11}{5}+j2\pi\right)\left(\frac{16}{5}+j2\pi\right)(j4\pi)} \\
&= \frac{\frac{17}{5}+j4\pi}{j4\pi\left(\frac{176}{25}-4\pi^2+\frac{54}{5}\pi j\right)} \\
&= \frac{\frac{17}{20\pi}+j}{\left(-\frac{54}{5}\pi+\left(\frac{176}{25}-4\pi^2\right)j\right)},
\end{aligned}$$

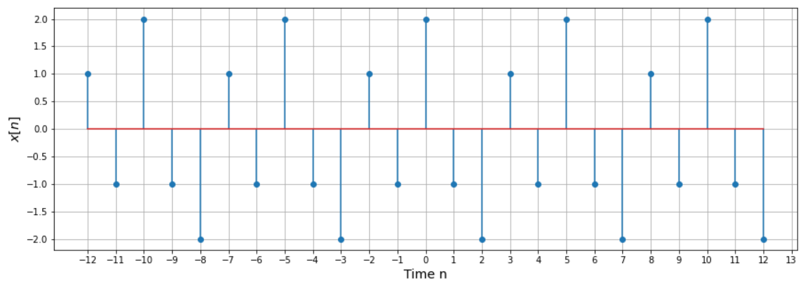
and

$$\begin{aligned}
A_3 &= \left(s + \frac{1}{5} - j2\pi\right) Y_1(s)\bigg|_{s=-\frac{1}{5}+j2\pi} \\
&= \frac{2s-3}{(s-2)(s-3)\left(s+\frac{1}{5}+j2\pi\right)}\Bigg|_{s=-\frac{1}{5}+j2\pi} \\
&= \frac{2\left(-\frac{1}{5}+j2\pi\right)-3}{\left(-\frac{1}{5}+j2\pi-2\right)\left(-\frac{1}{5}+j2\pi-3\right)\left(-\frac{1}{5}+j2\pi+\frac{1}{5}+j2\pi\right)} \\
&= \frac{-\frac{17}{5}+j4\pi}{\left(-\frac{11}{5}+j2\pi\right)\left(-\frac{16}{5}+j2\pi\right)(j4\pi)} \\
&= \frac{-\frac{17}{20\pi}+j}{j\left(\frac{176}{25}-4\pi^2-\frac{54}{5}\pi j\right)} \\
&= \frac{\frac{17}{20\pi}-j}{\left(-\frac{54}{5}\pi-\left(\frac{176}{25}-4\pi^2\right)j\right)} \\
&= A_2^*.
\end{aligned}$$

Also, notice that

$$\begin{aligned}
\frac{A_2}{s+\frac{1}{5}+j2\pi}+\frac{A_3}{s+\frac{1}{5}-j2\pi} &= \frac{A_2}{s+\frac{1}{5}+j2\pi}+\frac{A_2^*}{s+\frac{1}{5}-j2\pi} \\
&= \frac{A_2\left(s+\frac{1}{5}-j2\pi\right)+A_2^*\left(s+\frac{1}{5}+j2\pi\right)}{\left(s+\frac{1}{5}\right)^2+4\pi^2} \\
&= \frac{\operatorname{Re}\left\{A_2\left(s+\frac{1}{5}-j2\pi\right)\right\}}{\left(s+\frac{1}{5}\right)^2+4\pi^2}.
\end{aligned}$$

5. Consider the periodic discrete-time signal $x[n]$ shown below.

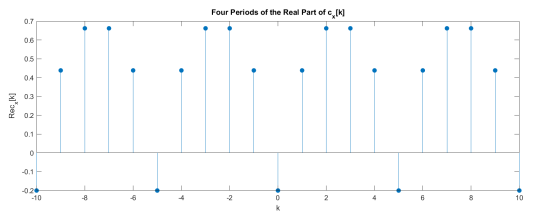


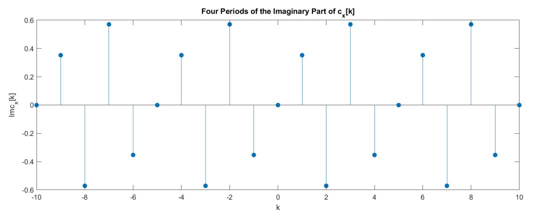
(a) Calculate the DTFS coefficients of $x[n]$ by hand and plot them using MATLAB or python. If they're complex-valued, plot the real and imaginary parts and the magnitude and phase. Include your MATLAB or python code in your answer. (16 points)

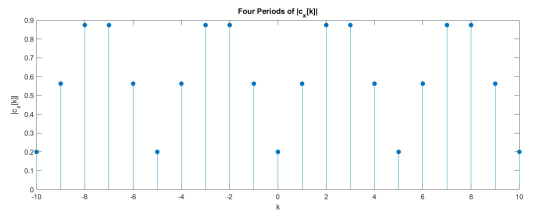
The first step is to determine a period, preferably the fundamental period, of $x[n]$. In this case, the fundamental period is $N = 5$. The Fourier series coefficients are

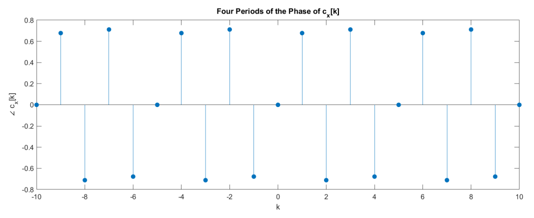
$$\begin{aligned} c_x[k] &= \frac{1}{N} \sum_{n=<N>} x[n]e^{-j2\pi kn/N} \\ &= \frac{1}{5} \sum_{n=0}^{4} x[n]e^{-j(2\pi/5)kn} \\ &= \frac{1}{5} \left(x[0] + x[1]e^{-j(2\pi/5)k} + x[2]e^{-j(4\pi/5)k} + x[3]e^{-j(6\pi/5)k} + x[4]e^{-j(8\pi/5)k} \right) \\ &= \frac{1}{5} \left(2 - e^{-j(2\pi/5)k} - 2e^{-j(4\pi/5)k} + e^{-j(6\pi/5)k} - e^{-j(8\pi/5)k} \right). \end{aligned}$$

MATLAB code for plotting the coefficients is in Appendix I. The output from this code is below. In this case, for the calculation of the phase, the `angle` function returned incorrect values. This was apparent since the phase at 0 was not 0, which meant that the calculated phase wasn't odd. Using the `atan` function instead resulted in the correct plot. This provided an example where it was helpful to know what symmetry the plots should have.

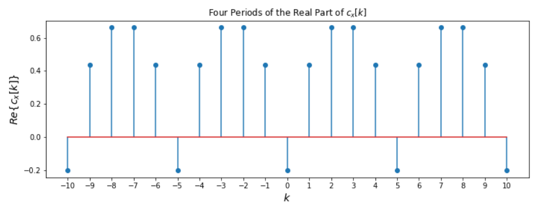


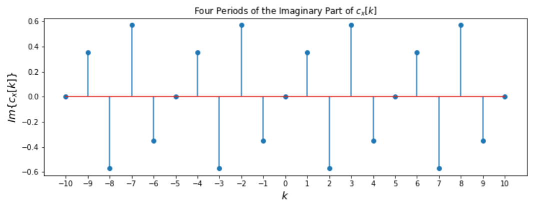


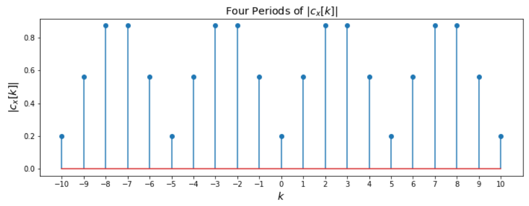


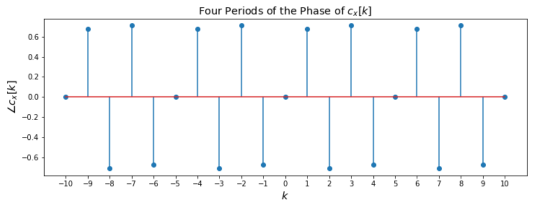


Python code for plotting the coefficients is in Appendix J. The output from this code is below. As for the MATLAB script, the NumPy `angle` and `arctan2` functions returned incorrect values. This was apparent since the phase at 0 was not 0, which meant that the calculated phase wasn't odd. Other values were also incorrect and contrary to the symmetry expected for the phase. The `angle` function is probably using the `arctan2` function and there might be a bug in the latter.









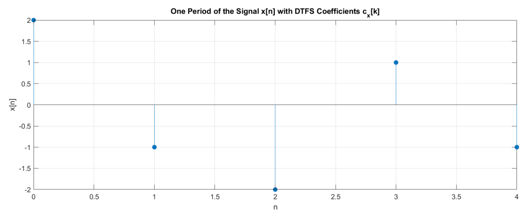
(b) Use MATLAB or python to show that the Fourier series representation of $x[n]$ has the expected values for one period of $x[n]$. Include your MATLAB or python code in your answer. (14 points)

The Fourier series representation of $x[n]$ is

$$x[n] = \sum_{k=0}^{N-1} c_x[k] e^{j2\pi kn/N} = \sum_{k=0}^{4} c_x[k] \left(e^{j(2\pi/5)k}\right)^n .$$

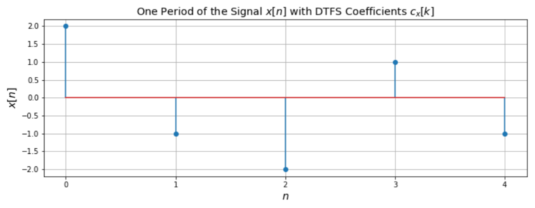
MATLAB code for evaluating this representation for $n = 0, 1, \cdots, 4$ is in Appendix K. The output from this code follows. Notice that the calculated values match the original signal values and that the imaginary parts of the calculated values are appropriately small.

```
The maximum absolute value of the imaginary part of x[n]
is 3.33067e-16
```

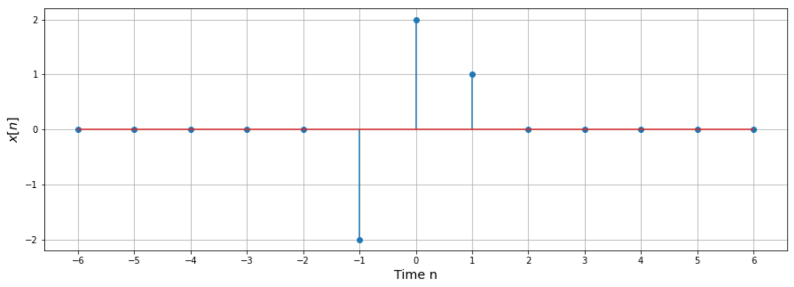


Python code for evaluating this representation for $n = 0, 1, \cdots, 4$ is in Appendix L. The output from this code follows. Notice that the calculated values match the original signal values and that the imaginary parts of the calculated values are appropriately small.

```
The max absolute value of the imaginary part of x[n] is
4.2433236152133335e-16
```



6. Consider the discrete-time signal $x[n]$ shown below, where $x[n] = 0$ for $n \leq -2$ and for $n \geq 2$.



(a) Find the DTFT for the signal $x[n]$ by hand. (8 points)

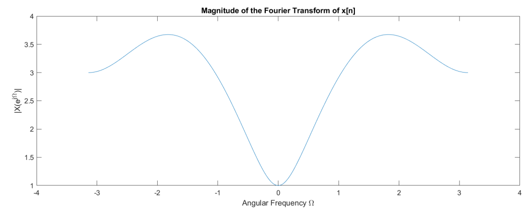
Here,

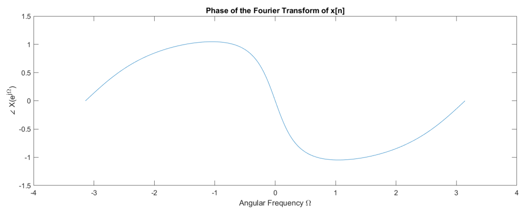
$$X\left(e^{j\Omega}\right) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\Omega n} = -2e^{j\Omega} + 2 + e^{-j\Omega}.$$

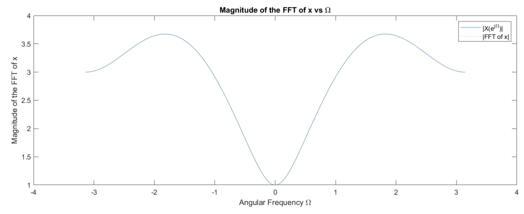
(b) In this part, you'll be comparing the DTFT that you calculated and the DFT calculated using MATLAB or python.

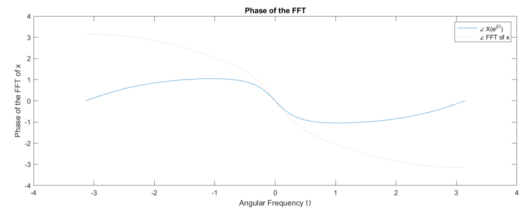
   i. Use MATLAB or python to plot the magnitude and phase of the DTFT of $x[n]$ that you calculated in Part (a) as functions of angular frequency Ω. (6 points)
   ii. Next, create a MATLAB or python/NumPy vector of length 1024 that starts with the nonzero values of $x[n]$ and is otherwise 0. (This is called zero-padding and is often done when working with short signals. In this case, you're just adding values of $x[n]$ at times n when $x[n] = 0$.) Use the MATLAB or NumPy function `fft` to calculate the DFT of this signal and plot the magnitude and phase of the DFT as functions of angular frequency Ω. (10 points)
   iii. How do the magnitudes and phases of the DTFT and the DFT compare to each other? If they are different, use the properties of the DTFT or the properties of the DFT to explain why. (5 points)

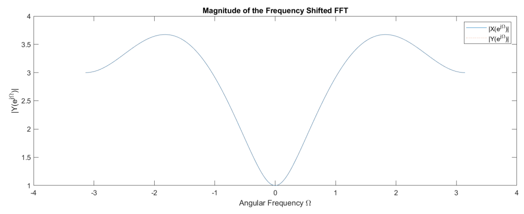
MATLAB code for all of these steps is in Appendix M. The output of the code is below. I plotted the DTFT and the FFT on the same plots to show the differences. Notice that the DTFT and the FFT have matching magnitudes but their phases differ. The phase difference is due to the fact that the FFT assumes that all signals start at $n = 0$ whereas the original signal starts at $n = -1$. It's possible to compensate for this difference by using the time-shift property of the DFT, which means shifting the frequencies of the FFT. The final two plots show that the magnitudes and phases match after applying the frequency shift.

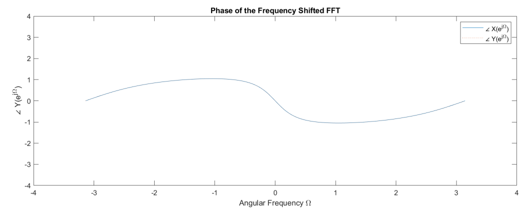






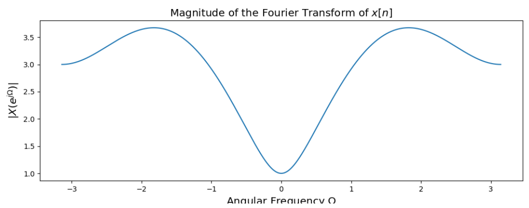


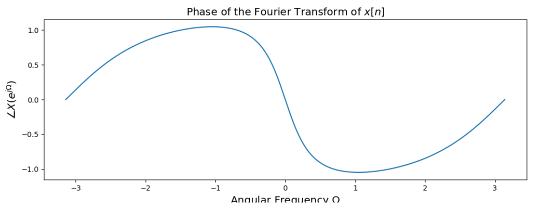


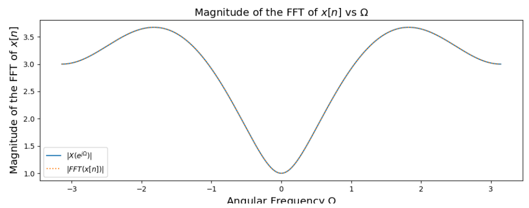


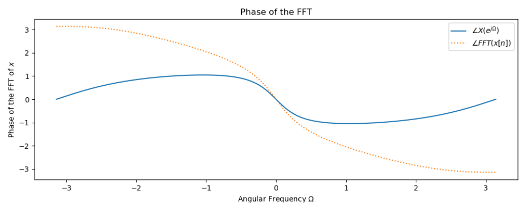
Python code for all of these steps is in Appendix N. The output of the code is below. I plotted the DTFT and the FFT on the same plots to show the differences. Notice that the DTFT and the FFT have matching magnitudes but their phases differ. The phase difference is due to the fact that the FFT assumes that all signals start at $n = 0$ whereas the original signal starts at $n = -1$.

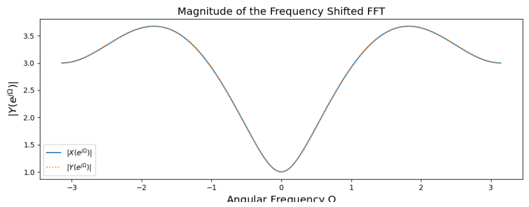
It's possible to compensate for this difference by using the time-shift property of the DFT, which means shifting the frequencies of the FFT. The final two plots show that the magnitudes and phases match after applying the frequency shift.

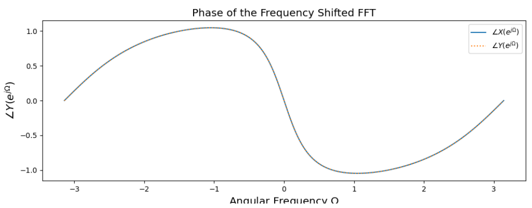












(c) Show that the inverse DTFT produces the expected values of $x[n]$ for all integers n. This calculation should be done by hand. (16 points)

The inverse DTFT is the synthesis equation

$$\begin{aligned}
x[n] &= \frac{1}{2\pi}\int_{2\pi} X\left(e^{j\Omega}\right) e^{j\Omega n}\, d\Omega \\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi} \left(-2e^{j\Omega} + 2 + e^{-j\Omega}\right) e^{j\Omega n}\, d\Omega \\
&= \frac{1}{2\pi}\left(-2\int_{-\pi}^{\pi} e^{j\Omega} e^{j\Omega n}\, d\Omega + 2\int_{-\pi}^{\pi} e^{j\Omega n}\, d\Omega + \int_{-\pi}^{\pi} e^{-j\Omega} e^{j\Omega n}\, d\Omega\right) \\
&= \frac{1}{2\pi}\left(-2\int_{-\pi}^{\pi} e^{j\Omega(n+1)}\, d\Omega + 2\int_{-\pi}^{\pi} e^{j\Omega n}\, d\Omega + \int_{-\pi}^{\pi} e^{j\Omega(n-1)}\, d\Omega\right).
\end{aligned}$$

All of these integrals have the form

$$\int_{-\pi}^{\pi} e^{j\Omega(n-m)}\, d\Omega$$

for some integer m. If $n = m$, then

$$\int_{-\pi}^{\pi} e^{j\Omega(n-m)}\, d\Omega = \int_{-\pi}^{\pi} d\Omega = 2\pi.$$

Otherwise, if $n \neq m$, then

$$\begin{aligned}
\int_{-\pi}^{\pi} e^{j\Omega(n-m)}\, d\Omega &= \frac{1}{j(n-m)}\, e^{j\Omega(n-m)}\Big|_{-\pi}^{\pi} = \frac{1}{j(n-m)}\left(e^{j\pi(n-m)} - e^{-j\pi(n-m)}\right) \\
&= \frac{2}{n-m}\left(\frac{e^{j\pi(n-m)} - e^{-j\pi(n-m)}}{2j}\right) = \frac{2}{n-m}\sin\left((n-m)\pi\right).
\end{aligned}$$

Since $n - m$ is an integer,

$$\int_{-\pi}^{\pi} e^{j\Omega(n-m)}\, d\Omega = 0.$$

Therefore,

$$\begin{aligned}
n = -1 &\Longrightarrow x[-1] = \frac{1}{2\pi}(-2(2\pi) + 0 + 0) = -2 \\
n = 0 &\Longrightarrow x[0] = \frac{1}{2\pi}(0 + 2(2\pi) + 0) = 2 \\
n = 1 &\Longrightarrow x[1] = \frac{1}{2\pi}(0 + 0 + 2\pi) = 1 \\
n \neq -1, 0, 1 &\Longrightarrow x[n] = \frac{1}{2\pi}(0 + 0 + 0) = 0,
\end{aligned}$$

which matches the original signal.

A MATLAB Code for Problem 1, Part (b)

```
% Script for plotting the continuous-time Fourier series coefficients for
% Problem 1(b).

kmin = -20;
kmax = -kmin;
k = kmin:kmax;

c = complex(zeros(size(k)));

omega0 = 0.5*pi;
omegak = omega0*k;
sinTerm = sin(omegak);

zeroIdx = 1 - kmin;
c(zeroIdx) = 7.0/8.0;
for m = 1:kmax

    idx = m - kmin + 1;
    kval = k(idx);
    pik = kval*pi;

    c(idx) = sinTerm(idx)/pik - (0.5/pik)*((1j).^(kval+1)...
             + (1.0 - (1j).^kval)/omegak(idx))...
```

```
            + (1.0/pik)*(1j*(-1j).^kval + ((-1j).^kval - 1.0)/omegak(idx));

    negIdx = zeroIdx - m;
    kval = k(negIdx);
    pik = kval*pi;

    c(negIdx) = sinTerm(negIdx)/pik - (0.5/pik)*((1j).^(kval+1)...
                + (1.0 - (1j).^kval)/omegak(negIdx))...
                + (1.0/pik)*(1j*(-1j).^kval + ((-1j).^kval - 1.0)/omegak(negIdx));

end

figure('Position', [100, 100, 1200, 400])
stem(k, real(c), 'filled')
title('Real Part of the CTFS Coefficients')
xlabel('Harmonic number k')
ylabel('Re(c_{x}[k])')
grid on
saveas(gcf, 'prob1b_real_MATLAB.png')

figure('Position', [200, 200, 1200, 400])
stem(k, imag(c), 'filled')
title('Imaginary Part of the CTFS Coefficients')
xlabel('Harmonic number k')
ylabel('Im(c_{x}[k])')
grid on
saveas(gcf, 'prob1b_imag_MATLAB.png')

figure('Position', [300, 300, 1200, 400])
stem(k, abs(c), 'filled')
title('Magnitude of the CTFS Coefficients')
xlabel('Harmonic number k')
ylabel('|c_{x}[k]|')
grid on
saveas(gcf, 'prob1b_mag_MATLAB.png')

figure('Position', [400, 400, 1200, 400])
stem(k, angle(c), 'filled')
```

```
title('Phase of the CTFS Coefficients')
xlabel('Harmonic number k')
ylabel('\angle c_{x}[k]')
grid on
saveas(gcf, 'prob1b_phase_MATLAB.png')
```

B Python Code for Problem 1, Part (b)

```
#-----------------------------------------------------#
# CTFS coefficients for the signal x(t) in Problem 1 #
#-----------------------------------------------------#

import numpy as np
import matplotlib.pyplot as plt

numPosCoeffs = 20
totalNumCoeffs = 2*numPosCoeffs + 1

omega0 = 0.5*np.pi

coeffs = np.zeros(totalNumCoeffs, dtype=complex)
harmNum = np.arange(-numPosCoeffs, numPosCoeffs + 1)
for k in harmNum:
    index = k + numPosCoeffs
    if k != 0:
        pik = np.pi*k
        omegak = omega0*k
        coeffs[index] = np.sin(omegak)/pik\
                        - (1.0/2.0/pik)*((1j)**(k+1) + (1.0 - (1j)**k)/omegak)\
                        + (1.0/pik)*(1j*(-1j)**k + ((-1j)**k - 1.0)/omegak)
        if k == 0:
            coeffs[index] = 7.0/8.0 + 0.0*1j

fig, ax = plt.subplots(1, figsize=(12,4))
ax.stem(harmNum, np.real(coeffs))
ax.set_title('Real Part of $c_{x}[k]$', fontsize="x-large")
ax.set_xlabel('Harmonic Number $k$', fontsize="x-large")
ax.set_ylabel('$Re\\{c_{x}[k]\\}$', fontsize="x-large")
ax.grid('True')
plt.savefig('prob1b_real_python.png')
plt.show()

fig, ax = plt.subplots(1, figsize=(12,4))
ax.stem(harmNum, np.imag(coeffs))
ax.set_title('Imaginary Part of $c_{x}[k]$', fontsize="x-large")
ax.set_xlabel('Harmonic Number $k$', fontsize="x-large")
ax.set_ylabel('$Im\\{c_{x}[k]\\}$', fontsize="x-large")
ax.grid('True')
plt.savefig('prob1b_imag_python.png')
plt.show()

fig, ax = plt.subplots(1, figsize=(12,4))
ax.stem(harmNum, np.abs(coeffs))
ax.set_title('Magnitude of $c_{x}[k]$', fontsize="x-large")
ax.set_xlabel('Harmonic Number $k$', fontsize="x-large")
ax.set_ylabel('$|c_{x}[k]|$', fontsize="x-large")
ax.grid('True')
plt.savefig('prob1b_mag_python.png')
plt.show()

fig, ax = plt.subplots(1, figsize=(12,4))
ax.stem(harmNum, np.angle(coeffs))
ax.set_title('Phase of $c_{x}[k]$', fontsize="x-large")
ax.set_xlabel('Harmonic Number $k$', fontsize="x-large")
ax.set_ylabel('$\\angle c_{x}[k]$', fontsize="x-large")
ax.grid('True')
plt.savefig('prob1b_phase_python.png')
plt.show()
```

C MATLAB Code from Problem 1, Part (c)

```
% Script for plotting the partial sums of the Fourier series of x(t) for
% Problem 1(c).
% Calculate all of the Fourier series coefficients that will be needed when
% calculating the partial sums up to the maximum number needed.

Nmax = 50;

kmin = -Nmax;
kmax =  Nmax;
k = kmin:kmax;

c = complex(zeros(size(k)));

omega0 = 0.5*pi;
omegak = omega0*k;
sinTerm = sin(omegak);

zeroIdx = 1 - kmin;
c(zeroIdx) = 7.0/8.0;
for m = 1:kmax

    idx = m - kmin + 1;
    kval = k(idx);
    pik = kval*pi;

    c(idx) = sinTerm(idx)/pik - (0.5/pik)*((1j).^(kval+1)...
             + (1.0 - (1j).^kval)/omegak(idx))...
             + (1.0/pik)*(1j*(-1j).^kval + ((-1j).^kval - 1.0)/omegak(idx));

    negIdx = zeroIdx - m;
    kval = k(negIdx);
    pik = kval*pi;

    c(negIdx) = sinTerm(negIdx)/pik - (0.5/pik)*((1j).^(kval+1)...
                + (1.0 - (1j).^kval)/omegak(negIdx))...
                + (1.0/pik)*(1j*(-1j).^kval + ((-1j).^kval - 1.0)/omegak(negIdx));

end

% Calculate and plot the partial sums x_{N}(t) for N = 1, 2, 5, 10, 50.
```

```
T0 = 4.0;

% The kth basis function has the form e^{j*2*pi*k*t/T0} = (e^{j*2*pi*t/T0})^{k}.
t = linspace(-2.0, 2.0, 500);
base = exp((2j*pi/T0)*t);    % = e^{j*2*pi*t/T0}

% The original signal
x = zeros(size(t));
for m = 1:length(t)
    tval = t(m);
    if tval <= -1.0
        continue
    elseif tval <= 0.0
        x(m) = 1.0 - tval;
    elseif tval <= 1.0
        x(m) = 2.0*tval + 1.0;
    else
        break
    end
end

% Set the initial partial sum to the partial sum for N = 1.
xN0 = c(zeroIdx) + c(zeroIdx + 1)*base + c(zeroIdx - 1)*conj(base);

% The imaginary part of the partial sum is 0.
fprintf('The maximum abs imaginary part of the partial sum x1(t) is %g.\n',...
max(abs(imag(xN0))))

figure('Position', [100, 100, 1200, 400])
plot(t, x, t, real(xN0))
ylim([-0.5, 3.5])
title('x(t) and x_{1}(t) vs Time t')
xlabel('Time t')
ylabel('x-value')
grid on
saveas(gcf, 'prob1cN1_MATLAB.png')
```

```
for N = [2, 5, 10, 50]

    xN = xN0;
    for k = 2:N
        basisFn = base.^k;
        xN = xN + c(zeroIdx + k)*basisFn;
        xN = xN + c(zeroIdx - k)*conj(basisFn);
    end

    % The imaginary part of the partial sum is 0.
    fprintf('The maximum abs imaginary part of the partial sum x%d(t) is %g.\n',..
            N, max(abs(imag(xN))))

    figure('Position', [100, 100, 1200, 400])
    plot(t, x, t, real(xN))
    ylim([-0.5, 3.5])
    title(sprintf('x(t) and x_{%d}(t) vs Time t', N))
    xlabel('Time t')
    ylabel('x-value')
    grid on
    filename = sprintf('prob1cN%d_MATLAB.png', N);
    saveas(gcf, filename)

end
```

D Python Code from Problem 1, Part (c)

```
#--------------------------#
# Partial sums of the CTFS #
#--------------------------#

import numpy as np
import matplotlib.pyplot as plt

numPosCoeffs = 50
totalNumCoeffs = 2*numPosCoeffs + 1
```

```
# Calculate the CTFS coefficients.

omega0 = 0.5*np.pi

coeffs = np.zeros(totalNumCoeffs, dtype=complex)
harmNum = np.arange(-numPosCoeffs, numPosCoeffs + 1)
for k in harmNum:
    index = k + numPosCoeffs
    if k != 0:
        pik = np.pi*k
        omegak = omega0*k
        coeffs[index] = np.sin(omegak)/pik\
                        - (1.0/2.0/pik)*((1j)**(k+1) + (1.0 - (1j)**k)/omegak)\
                        + (1.0/pik)*(1j*(-1j)**k + ((-1j)**k - 1.0)/omegak)
    if k == 0:
        coeffs[index] = 7.0/8.0 + 0.0*1j

# Calculate the values of the original signal.

txleft  = np.linspace(-1.0, 0.0, 500)
xleft   = 1.0 - txleft
txright = np.linspace( 0.0, 1.0, 500)
xright  = 2.0*txright + 1.0

# Calculate the values of the partial sums.

numTimes = 1000
t = np.linspace(-2.0, 2.0, numTimes) # time vector

omega0jt = (1j*omega0)*t

dt = 0.025
dx = 0.04

print('')
for N in [1, 2, 5, 10, 50]:
    xN = np.zeros(np.size(t), dtype=complex)
    for k in np.arange(-N, N + 1):
        xN += coeffs[k + numPosCoeffs]*np.exp(k*omega0jt)

    # As a check, print the maximum absolute value of the imaginary part
    # of the Nth partial sum. This value should be around 10^-16 or less.
    print('The maximum absolute value of the imaginary part of x{0}(t) is {1}.'\
          .format(N, np.max(np.abs(np.imag(xN)))))

    # Plot the Nth partial sum.
    fig, ax = plt.subplots(1, figsize=(12,3))
    ax.plot(t, np.real(xN)) # The imaginary part should be 0.
    ax.set_title('Partial sum $x_N(t)$ of CTFS for $x(t)$ when N = {0}'.format(N),
                 fontsize="x-large")
    ax.set_ylim([-0.25, 3.25])
    ax.set_xlabel('Time t', fontsize="x-large")
    ax.set_ylabel('$x_{N}(t)$', fontsize="x-large")

    # Plot the original signal
    ax.plot([-2.0, -1.0], [0.0, 0.0], ':', color='tab:red')
    ax.plot([-1.0], [0.0], 'o', color='tab:red', fillstyle='full')
    ax.plot([-1.0, -1.0], [0.0, 2.0 - dx], '--', color='tab:red')
    ax.plot([-1.0], [2.0], 'o', color='tab:red', fillstyle='none')
    ax.plot(txleft,  xleft,  ':', color='tab:red')
    ax.plot(txright, xright, ':', color='tab:red')
    ax.plot([1.0], [3.0], 'o', color='tab:red', fillstyle='full')
    ax.plot([1.0, 1.0], [dx, 3.0], '--', color='tab:red')
    ax.plot([1.0], [0.0], 'o', color='tab:red', fillstyle='none')
    ax.plot([1.0 + dt, 2.0], [0.0, 0.0], ':', color='tab:red')
    plt.savefig('prob1cN{0}_python.png'.format(N))
    plt.show()
```

E MATLAB Code for Problem 3, Part (b)

```
% Script for plotting the real and imaginary parts and the magnitude and
% phase of the Fourier transform of x(t) = t^3*e^{-t}*u(t) in Problem 3(b).

% F{x(t)} = 6/(j*omega + 1)^4
```

```
omega = linspace(-8.0, 8.0, 2000);
factor = 1.0 + 1j*omega;

X = 6.0./(factor.*factor.*factor.*factor);

figure('Position', [200, 200, 1200, 400])
plot(omega, real(X))
title('Real Part of X(j\omega)')
xlabel('Angular Frequency \omega')
ylabel('Re(X(j\omega))')
saveas(gcf, 'prob3b_real_MATLAB.png')

figure('Position', [200, 300, 1200, 400])
plot(omega, imag(X))
title('Imaginary Part of X(j\omega)')
xlabel('Angular Frequency \omega')
ylabel('Im(X(j\omega))')
saveas(gcf, 'prob3b_imag_MATLAB.png')

figure('Position', [200, 400, 1200, 400])
plot(omega, abs(X))
title('Magnitude of X(j\omega)')
xlabel('Angular Frequency \omega')
ylabel('|X(j\omega)|')
saveas(gcf, 'prob3b_mag_MATLAB.png')

figure('Position', [200, 500, 1200, 400])
plot(omega, angle(X))
title('Phase of X(j\omega)')
xlabel('Angular Frequency \omega')
ylabel('\angle X(j\omega)')
saveas(gcf, 'prob3b_phase_MATLAB.png')
```

F Python Code for Problem 3, Part (b)

```
#-----------------------------------------------------------------------------#
# Script for plotting the real and imaginary parts and the magnitude and    #
# phase of the Fourier transform of x(t) = t^3*e^{-t}*u(t) in Problem 3(b). #
#--------------------------------------------------------------------------#

import numpy as np
import matplotlib.pyplot as plt

# F{x(t)} = 6/(j*omega + 1)^4

omega = np.linspace(-8.0, 8.0, 2000)
factor = 1.0 + 1j*omega

X = np.divide(6.0,np.multiply(factor,np.multiply(factor,np.multiply(factor,factor)

fig, ax = plt.subplots(1, figsize=(12,4))
ax.plot(omega, np.real(X))
ax.set_title('Real Part of $X(j\\omega)$', fontsize="x-large")
ax.set_xlabel('Angular Frequency $\\omega$', fontsize="x-large")
ax.set_ylabel('$Re\\{X(j\\omega)\\}$', fontsize="x-large")
plt.savefig('prob3b_real_python.png')
plt.show()

fig, ax = plt.subplots(1, figsize=(12,4))
ax.plot(omega, np.imag(X))
ax.set_title('Imaginary Part of $X(j\\omega)$', fontsize="x-large")
ax.set_xlabel('Angular Frequency $\\omega$', fontsize="x-large")
ax.set_ylabel('$Im\\{X(j\\omega)\\}$', fontsize="x-large")
plt.savefig('prob3b_imag_python.png')
plt.show()

fig, ax = plt.subplots(1, figsize=(12,4))
ax.plot(omega, np.abs(X))
ax.set_title('Magnitude of $X(j\\omega)$', fontsize="x-large")
ax.set_xlabel('Angular Frequency $\\omega$', fontsize="x-large")
ax.set_ylabel('$|X(j\\omega)|$', fontsize="x-large")
plt.savefig('prob3b_mag_python.png')
plt.show()

fig, ax = plt.subplots(1, figsize=(12,4))
```

```
ax.plot(omega, np.angle(X))
ax.set_title('Phase of $X(j\\omega)$', fontsize="x-large")
ax.set_xlabel('Angular Frequency $\\omega$', fontsize="x-large")
ax.set_ylabel('$\\angle X(j\\omega)$', fontsize="x-large")
plt.savefig('prob3b_phase_python.png')
plt.show()
```

G MATLAB Code for Problem 3, Part (c)

```
% Script for plotting the real and imaginary parts and the magnitude and
% phase of the DFT approximation of X(j*omega) in Problem 3(c).

N = 262144;
T = 100.0;

Ts = T/N;    % sample period
fs = 1.0/Ts; % sample rate
df = fs/N;   % = 1/T = frequency increment

n = 0:N-1;
t = Ts*n;
x = t.*t.*t.*exp(-t);

X = fftshift(Ts*fft(x));
k = -N/2:N/2-1;
omega = 2.0*pi*k*df;

figure('Position', [200, 200, 1200, 400])
plot(omega, real(X))
xlim([-8.0, 8.0])
title('Real Part of the DFT of x(t)')
xlabel('Angular Frequency \omega')
ylabel('Re(X(j\omega))')
saveas(gcf, 'prob3c_real_MATLAB.png')

figure('Position', [200, 300, 1200, 400])
plot(omega, imag(X))
```

```
xlim([-8.0, 8.0])
title('Imaginary Part of the DFT of x(t)')
xlabel('Angular Frequency \omega')
ylabel('Im(X(j\omega))')
saveas(gcf, 'prob3c_imag_MATLAB.png')

figure('Position', [200, 400, 1200, 400])
plot(omega, abs(X))
xlim([-8.0, 8.0])
title('Magnitude of the DFT of x(t)')
xlabel('Angular Frequency \omega')
ylabel('|X(j\omega)|')
saveas(gcf, 'prob3c_mag_MATLAB.png')

figure('Position', [200, 500, 1200, 400])
plot(omega, angle(X))
xlim([-8.0, 8.0])
title('Phase of the DFT of x(t)')
xlabel('Angular Frequency \omega')
ylabel('\angle X(j\omega)')
saveas(gcf, 'prob3c_phase_MATLAB.png')
```

H Python Code for Problem 3, Part (c)

```
#----------------------------------------------------------------------#
# Script for plotting the real and imaginary parts and the magnitude #
# and phase of the DFT approximation of X(j*omega) in Problem 3(c).  #
#----------------------------------------------------------------------#

import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftshift

N = 262144
T = 100.0

Ts = T/N    # sample period
```

```
fs = 1.0/Ts # sample rate
df = fs/N   # = 1/T = frequency increment

n = np.arange(0,N)
t = Ts*n
x = np.multiply(np.multiply(t,np.multiply(t,t)),np.exp(-t))

X = fftshift(Ts*fft(x))
k = np.arange(-N/2,N/2)
omega = 2.0*np.pi*df*k

fig, ax = plt.subplots(1, figsize=(12,4))
ax.plot(omega, np.real(X))
ax.set_xlim([-8.75, 8.75])
ax.set_title('Real Part of the DFT of $x(t)$', fontsize="x-large")
ax.set_xlabel('Angular Frequency $\\omega$', fontsize="x-large")
ax.set_ylabel('$Re\\{X(j\\omega)\\}$', fontsize="x-large")
plt.savefig('prob3c_real_python.png')
plt.show()

fig, ax = plt.subplots(1, figsize=(12,4))
ax.plot(omega, np.imag(X))
ax.set_xlim([-8.75, 8.75])
ax.set_title('Imaginary Part of the DFT of $x(t)$', fontsize="x-large")
ax.set_xlabel('Angular Frequency \omega', fontsize="x-large")
ax.set_ylabel('$Im\\{X(j\omega)\\})$', fontsize="x-large")
plt.savefig('prob3c_imag_python.png')
plt.show()

fig, ax = plt.subplots(1, figsize=(12,4))
ax.plot(omega, np.abs(X))
ax.set_xlim([-8.75, 8.75])
ax.set_title('Magnitude of the DFT of $x(t)$', fontsize="x-large")
ax.set_xlabel('Angular Frequency $\\omega$', fontsize="x-large")
ax.set_ylabel('$|X(j\\omega)|$', fontsize="x-large")
plt.savefig('prob3c_mag_python.png')
plt.show()
```

```
fig, ax = plt.subplots(1, figsize=(12,4))
ax.plot(omega, np.angle(X))
ax.set_xlim([-8.75, 8.75])
ax.set_title('Phase of the DFT of $x(t)$', fontsize="x-large")
ax.set_xlabel('Angular Frequency $\\omega$', fontsize="x-large")
ax.set_ylabel('$\\angle X(j\\omega)$', fontsize="x-large")
plt.savefig('prob3c_phase_python.png')
plt.show()
```

I MATLAB Code for Problem 5, Part (a)

```
% Script for plotting the DTFS coefficients for the signal x[n]
% in Problem 5, Part (a).

Omega0 = 2.0*pi/5.0; % fundamental angular frequency of x[n]
k = -10:10;          % frequency indices
minusjOmega0k = -1j*Omega0*k;

cx = 0.2*(2.0 - exp(minusjOmega0k) - 2.0*exp(2.0*minusjOmega0k)...
     + exp(3.0*minusjOmega0k) - exp(4.0*minusjOmega0k));

figure('Position', [100, 100, 1200, 400])
stem(k, real(cx), 'filled')
title('Four Periods of the Real Part of c_x[k]')
xlabel('k')
ylabel('Re{c_x[k]}')
saveas(gcf, 'prob5a_real_MATLAB.png')

figure('Position', [200, 200, 1200, 400])
stem(k, imag(cx), 'filled')
title('Four Periods of the Imaginary Part of c_x[k]')
xlabel('k')
ylabel('Im{c_x[k]}')
saveas(gcf, 'prob5a_imag_MATLAB.png')

figure('Position', [300, 300, 1200, 400])
stem(k, abs(cx), 'filled')
```

```
title('Four Periods of |c_x[k]|')
xlabel('k')
ylabel('|c_x[k]|')
saveas(gcf, 'prob5a_mag_MATLAB.png')

figure('Position', [400, 400, 1200, 400])
%stem(k, angle(cx), 'filled')
stem(k, atan(imag(cx)./real(cx)), 'filled')
title('Four Periods of the Phase of c_x[k]')
xlabel('k')
ylabel('\angle c_x[k]')
saveas(gcf, 'prob5a_phase_MATLAB.png')
```

J Python Code for Problem 5, Part (a)

```
#--------------------------------------------------------------------#
# Plotting the DTFS coefficients for the signal x[n] in Problem 5. #
#--------------------------------------------------------------------#

import numpy as np
import matplotlib.pyplot as plt

Omega0 = 2.0*np.pi/5.0 # fundamental angular frequency of x[n]
k = np.arange(-10, 11) # frequency indices
minusjOmega0k = -1j*Omega0*k

cx = 0.2*(2.0 - np.exp(minusjOmega0k) - 2.0*np.exp(2.0*minusjOmega0k)\
          + np.exp(3.0*minusjOmega0k) - np.exp(4.0*minusjOmega0k))

fig, ax = plt.subplots(1, figsize=(12,4))
ax.stem(k, np.real(cx))
ax.set_xticks(k)
ax.set_title('Four Periods of the Real Part of $c_x[k]$')
ax.set_xlabel('$k$', fontsize="x-large")
ax.set_ylabel('$Re\{c_x[k]\}$', fontsize="x-large")
plt.savefig('prob5a_real_python.png')
```

```
fig, ax = plt.subplots(1, figsize=(12,4))
ax.stem(k, np.imag(cx))
ax.set_xticks(k)
ax.set_title('Four Periods of the Imaginary Part of $c_x[k]$')
ax.set_xlabel('$k$', fontsize="x-large")
ax.set_ylabel('$Im\{c_x[k]\}$', fontsize="x-large")
plt.savefig('prob5a_imag_python.png')

fig, ax = plt.subplots(1, figsize=(12,4))
ax.stem(k, np.abs(cx))
ax.set_xticks(k)
ax.set_title('Four Periods of $|c_x[k]|$', fontsize="x-large")
ax.set_xlabel('$k$', fontsize="x-large")
ax.set_ylabel('$|c_x[k]|$', fontsize="x-large")
plt.savefig('prob5a_mag_python.png')

fig, ax = plt.subplots(1, figsize=(12,4))
# In this case, the angle function returns the wrong result.
# It's wrong because it returns a nonzero value for cx[0].
# It seems to use the arctan2 function, which also returns the wrong result.
#ax.stem(k, np.angle(cx))
#ax.stem(k, np.arctan2(np.imag(cx), np.real(cx)))
ax.stem(k, np.arctan(np.divide(np.imag(cx), np.real(cx))))
ax.set_xticks(k)
ax.set_title('Four Periods of the Phase of $c_x[k]$', fontsize="x-large")
ax.set_xlabel('$k$', fontsize="x-large")
ax.set_ylabel('$\\angle c_x[k]$', fontsize="x-large")
plt.savefig('prob5a_phase_python.png')
```

K MATLAB Code for Problem 5, Part (b)

```
% This script calculates values of the inverse discrete-time Fourier series
% for Problem 5, Part (c).

N0 = 5;             % fundamental period of x[n]
Omega0 = 2.0*pi/N0; % fundamental angular frequency of x[n]
k0 = 0;             % initial frequency index
```

```
k = k0:k0 + N0 - 1; % frequency indices
Omega0k = Omega0*k;
minusjOmega0k = -1j*Omega0*k;

cx = 0.2*(2.0 - exp(minusjOmega0k) - 2.0*exp(2.0*minusjOmega0k)...
     + exp(3.0*minusjOmega0k) - exp(4.0*minusjOmega0k));

w = exp(1j*Omega0k);

x = zeros([1,N0]);
for n = k
    wn = w.^n;
    idx = n - k0 + 1;
    x(idx) = 0.0 + 0.0*1j;
    for m = 1:N0
        x(idx) = x(idx) + cx(m)*wn(m);
    end
end

fprintf('The maximum absolute value of the imaginary part of x[n] is %g\n', ...
max(abs(imag(x))))

figure('Position', [200, 200, 1200, 400])
stem(k, real(x), 'filled') % the time indices n are the same as the freq indices k
title('One Period of the Signal x[n] with DTFS Coefficients c_x[k]')
xlabel('n')
ylabel('x[n]')
grid on
saveas(gcf, 'prob5b_MATLAB.png')
```

L Python Code for Problem 5, Part (b)

```
#--------------------------------------------------------#
# Reconstructing the time-domain signal x[n] in Problem 5 #
# using the DTFS coefficients.                            #
#--------------------------------------------------------#
```

```
import numpy as np
import matplotlib.pyplot as plt

N0 = 5                      # fundamental period of x[n]
Omega0 = 2.0*np.pi/N0       # fundamental angular frequency of x[n]
k0 = 0                      # initial frequency index
k = np.arange(k0, k0 + N0) # frequency indices
Omega0k = Omega0*k
minusjOmega0k = -1j*Omega0k

cx = 0.2*(2.0 - np.exp(minusjOmega0k) - 2.0*np.exp(2.0*minusjOmega0k)\
          + np.exp(3.0*minusjOmega0k) - np.exp(4.0*minusjOmega0k))

w = np.exp(1j*Omega0k)

x = np.zeros(N0, dtype=complex)
for n in np.arange(k0, k0 + N0):
    # Calculate the Fourier weights.
    wn = np.power(w, n)
    idx = n - k0
    # Add the products of the coefficients with the weights.
    x[idx] = np.dot(cx, wn)

# This is a sanity check.
print('The max absolute value of the imaginary part of x[n] is {0}'\
      .format(np.max(np.abs(np.imag(x)))))

fig, ax = plt.subplots(1, figsize=(12,3))
ax.stem(k, np.real(x)) # The time indices n are the same as the freq indices k.
ax.set_xticks(k)
ax.set_title('One Period of the Signal $x[n]$ with DTFS Coefficients $c_x[k]$',\
             fontsize="x-large")
ax.set_xlabel('$n$', fontsize="x-large")
ax.set_ylabel('$x[n]$', fontsize="x-large")
ax.grid(True)
plt.savefig('prob5b_python.png')
```

M MATLAB Code for Problem 6, Part (b)

```
% This script plots the DTFT of the nonperiodic signal x[n] in Problem 6(b)
% and uses the function fft to approximate the DTFT of x[n].

Omega = linspace(-pi, pi, 1000);
X = -2.0*exp(1j*Omega) + 2.0 + exp(-1j*Omega);

figure('Position', [100, 100, 1200, 400])
plot(Omega, abs(X))
xlabel('Angular Frequency \Omega')
ylabel('|X(e^{j\Omega})|')
title('Magnitude of the Fourier Transform of x[n]')
saveas(gcf, 'prob6b_DTFT_mag_MATLAB.png')

figure('Position', [200, 200, 1200, 400])
plot(Omega, angle(complex(X)))
xlabel('Angular Frequency \Omega')
ylabel('\angle X(e^{j\Omega})')
title('Phase of the Fourier Transform of x[n]')
saveas(gcf, 'prob6b_DTFT_phase_MATLAB.png')

L = 1024;
x = [-2.0, 2.0, 1.0];

% Zero-pad the signal
z = zeros(1,L);
for n = 1:length(x)
z(n) = x(n);
end

OmegaL = -pi + (0:L-1)*(2.0*pi/L);

xfft = fftshift(fft(z));

figure('Position', [300, 300, 1200, 400])
plot(Omega, abs(X), '-', OmegaL, abs(xfft), ':')
xlabel('Angular Frequency \Omega')
```

```
ylabel('Magnitude of the FFT of x')
title('Magnitude of the FFT of x vs \Omega')
legend('|X(e^{j\Omega})|', '|FFT of x|')
saveas(gcf, 'prob6b_FFT_mag_MATLAB.png')

figure('Position', [400, 400, 1200, 400])
plot(Omega, angle(X), '-', OmegaL, angle(xfft), ':')
xlabel('Angular Frequency \Omega')
ylabel('Phase of the FFT of x')
title('Phase of the FFT')
legend('\angle X(e^{j\Omega})', '\angle FFT of x')
saveas(gcf, 'prob6b_FFT_phase_MATLAB.png')

Y = exp(1j*OmegaL).*xfft;

figure('Position', [500, 500, 1200, 400])
plot(Omega, abs(X), '-', OmegaL, abs(Y), ':')
xlabel('Angular Frequency \Omega')
ylabel('|Y(e^{j\Omega})|')
title('Magnitude of the Frequency Shifted FFT')
legend('|X(e^{j\Omega})|', '|Y(e^{j\Omega})|')
saveas(gcf, 'prob6b_FFT_shifted_mag_MATLAB.png')

figure('Position', [600, 600, 1200, 400])
plot(Omega, angle(X), '-', OmegaL, angle(Y), ':')
ylim([-4, 4])
xlabel('Angular Frequency \Omega')
ylabel('\angle Y(e^{j\Omega})')
title('Phase of the Frequency Shifted FFT')
legend('\angle X(e^{j\Omega})', '\angle Y(e^{j\Omega})')
saveas(gcf, 'prob6b_FFT_shifted_phase_MATLAB.png')
```

N Python Code for Problem 6, Part (b)

```
#-----------------------------------------------------------------------#
# The DTFT of the nonperiodic signal x[n] in Problem 6(b) and the FFT #
# of a zero-padded version of x[n].                                   #
```

```
#----------------------------------------------------------------------#

import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftshift

Omega = np.linspace(-np.pi, np.pi, 1000)
X = -2.0*np.exp(1j*Omega) + 2.0 + np.exp(-1j*Omega)

fig, ax = plt.subplots(1, figsize=(12,4))
ax.plot(Omega, np.abs(X))
ax.set_xlabel('Angular Frequency $\\Omega$', fontsize="x-large")
ax.set_ylabel('$|X(e^{j\\Omega})|$', fontsize="x-large")
ax.set_title('Magnitude of the Fourier Transform of $x[n]$', fontsize="x-large")
plt.savefig('prob6b_DTFT_mag_python.png')

fig, ax = plt.subplots(1, figsize=(12,4))
ax.plot(Omega, np.angle(X))
ax.set_xlabel('Angular Frequency $\\Omega$', fontsize="x-large")
ax.set_ylabel('$\\angle X(e^{j\\Omega})$', fontsize="x-large")
ax.set_title('Phase of the Fourier Transform of $x[n]$', fontsize="x-large")
plt.savefig('prob6b_DTFT_phase_python.png')

L = 1024
x = [-2.0, 2.0, 1.0]

# Zero-pad the signal
z = np.zeros(L, dtype=complex)
for n in np.arange(0,3):
    z[n] = x[n]

OmegaL = -np.pi + np.arange(0,L)*(2.0*np.pi/L)

xfft = fftshift(fft(z))

fig, ax = plt.subplots(1, figsize=(12,4))
magX,   = ax.plot(Omega,  np.abs(X), label='$|X(e^{j\\Omega})|$', linestyle='-')
magFFT, = ax.plot(OmegaL, np.abs(xfft), label='$|FFT(x[n])|$', linestyle=':')
```

```
ax.set_xlabel('Angular Frequency $\\Omega$', fontsize="x-large")
ax.set_ylabel('Magnitude of the FFT of $x[n]$', fontsize="x-large")
ax.set_title('Magnitude of the FFT of $x[n]$ vs $\\Omega$', fontsize="x-large")
ax.legend(handles=[magX, magFFT])
plt.savefig('prob6b_FFT_mag_python.png')

fig, ax = plt.subplots(1, figsize=(12,4))
angX,   = ax.plot(Omega,  np.angle(X), label='$\\angle X(e^{j\\Omega})$', linestyl
angFFT, = ax.plot(OmegaL, np.angle(xfft), label='$\\angle FFT(x[n])$', linestyle='
ax.set_xlabel('Angular Frequency $\\Omega$')
ax.set_ylabel('Phase of the FFT of $x$')
ax.set_title('Phase of the FFT')
ax.legend(handles=[angX, angFFT])
plt.savefig('prob6b_FFT_phase_python.png')

Y = np.multiply(np.exp(1j*OmegaL),xfft)

fig, ax = plt.subplots(1, figsize=(12,4))
magX, = ax.plot(Omega,  np.abs(X), label='$|X(e^{j\\Omega})|$', linestyle='-')
magY, = ax.plot(OmegaL, np.abs(Y), label='$|Y(e^{j\\Omega})|$', linestyle=':')
ax.set_xlabel('Angular Frequency $\\Omega$', fontsize="x-large")
ax.set_ylabel('$|Y(e^{j\\Omega})|$', fontsize="x-large")
ax.set_title('Magnitude of the Frequency Shifted FFT', fontsize="x-large")
ax.legend(handles=[magX, magY])
plt.savefig('prob6b_FFT_shifted_mag_python.png')

fig, ax = plt.subplots(1, figsize=(12,4))
angX, = ax.plot(Omega,  np.angle(X), label='$\\angle X(e^{j\\Omega})$', linestyle=
angY, = ax.plot(OmegaL, np.angle(Y), label='$\\angle Y(e^{j\\Omega})$', linestyle=
ax.set_xlabel('Angular Frequency $\\Omega$', fontsize="x-large")
ax.set_ylabel('$\\angle Y(e^{j\\Omega})$', fontsize="x-large")
ax.set_title('Phase of the Frequency Shifted FFT', fontsize="x-large")
ax.legend(handles=[angX, angY])
plt.savefig('prob6b_FFT_shifted_phase_python.png')
```