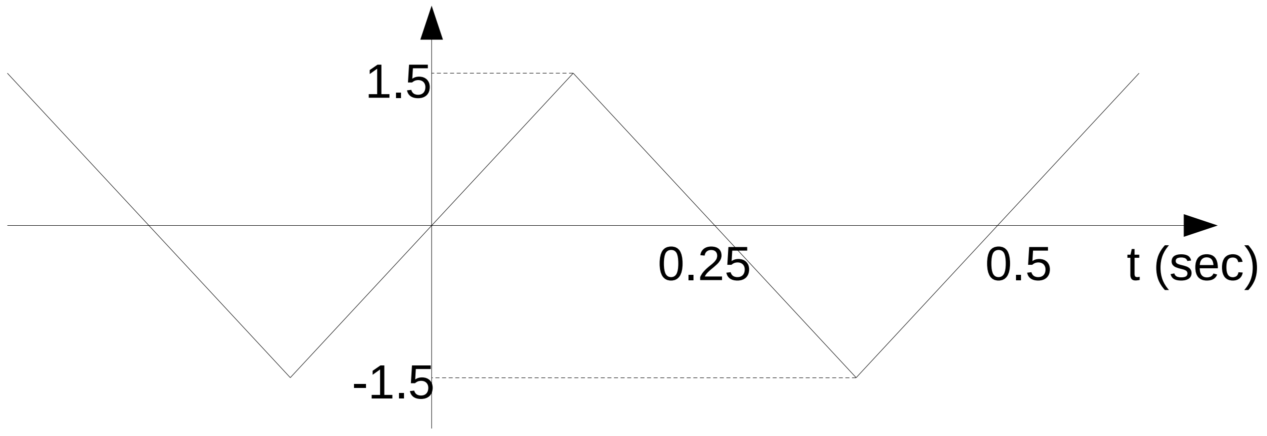
EE395 Intro to DSP
Homework #1
(due Friday, Sep. 5, 2025, 11:59:00 pm)

Text Problems:
T 01-02 The analog signal $x(t)=8\sin(2\pi t)+8\sin(10\pi t)+4\sin(14\pi t)$, where t is in seconds, is sampled at a rate of $f_s=4$ Hz. Determine the signal $x_a(t)$ aliased with $x(t)$. Show that the two signals have the same sample values, that is show that $x(nT)=x_a(nT)$. Repeat the above questions if the sampling rate is $f_s=12$ Hz.
Hint: Be very careful about positive versus negative frequencies (a sinusoid of frequency -1 Hz is NOT the same as a sinusoid of frequency +1 Hz)

T 01-04 Let $x(t)=3\cos(10\pi t)+2\cos(6\pi t)\cos(8\pi t)$, where t is is seconds. Determine the signal $x_a(t)$ aliased with $x(t)$, if the sampling rate is 7 Hz. Repeat for a sampling rate of 12 Hz.
Hint: You will probably want to consider using a trig identity to simplify.

T 01-07 Consider the periodic triangular waveform with period $T_0=0.5$ sec shown below. The waveform is sampled at rate $f_s=16$ Hz and the resulting samples are reconstructed by an *ideal* reconstructor. Show that the signal $x_{rec}(t)$ that will appear at the output of the reconstructor will have the form: $x_{rec}(t)=A\sin(2\pi f_1 t)+B\sin(2\pi f_2 t)$ and determine the numerical values of the frequencies f_1 and f_2 and amplitudes A and B .
Hints:
- Example 1.4.6 may be helpful here.
- To help you solve for f_1 and f_2 , note that $x(t)$ is an odd periodic signal. We know something specific about the harmonics that appear in odd periodic signals...
- To help you solve for A and B , use the fact that $x(nT)=x_a(nT)$ and choose two values of n to set up two equations to solve for A and B .



T 01-26 Consider a speech wave $x(t)$ with frequency content in the range $|f| \leq 20{,}000$ Hz. Assume that the maximum gain of the speech wave $x(t)$ is 0 dB in the range $0 \leq |f| \leq 1000$ Hz and that the gain decays at a rate of 30 dB/decade for $|f| > 1000$ Hz.

(a) Assume that you implement an antialiasing filter with cutoff frequency $f_c = 3000$ Hz and rolloff of 60 dB/decade, and that you sample the signal at a rate of $f_s = 8000$ Hz. What is the amplitude in dB of the worst case aliasing present in the sampled signal $x_a(t)$?

Notes:

- Since you can easily hear up to $f_s/2 = 4000$ Hz, this is the frequency at which the worst aliasing distortion will occur.
- Remember that gains and attenuations are additive in dB since $\log(xy) = \log(x) + \log(y)$ and that you have two attenuations (one due to the signal itself $A_{x_{in}}(f)$ and one due to the antialiasing filter $A(f)$)
- A quick sketch similar to that in Example 1.5.3 may be helpful here, but remember that you are dealing with two rolloffs, each of which begins at a different frequency.

(b) The human ear has a dynamic range of ~100 dB. Since the speech wave $x(t)$ has a max gain of 0 dB, this means that the human ear could hear sounds attenuated by up to 100 dB. For the scenario in part (a), you should have found that the aliasing has not been attenuated by more than 100 dB. Assuming that you are stuck with the same sampling frequency $f_s = 8000$ Hz and the same antialiasing filter cutoff $f_c = 3000$ Hz, what is the **attenuation** needed in the antialiasing filter to result in worst case aliasing attenuated by at least 100 dB?

(c) For the same scenario as in part (b), what is the **rolloff** needed in the antialiasing filter for the worst case aliasing attenuated by at least 100 dB?

(d) Repeat parts (a)-(c) assuming that the signal is sampled at a rate of $f_s = 15000$ Hz. This is an example of *oversampling* in that the signal is sampled at greater than twice the cutoff of the antialiasing filter.

Programming Problems:

Matlab commands are specified in the text and in `blue Courier font` and python commands in parenthesis and in `red Courier font`; variables and filenames are specified in `black Courier font`. In the following python syntax, I have used `import matplotlib.pyplot as plt`, and `import numpy as np`, and you probably want to use `%matplotlib inline` to display plots inline in the jupyter notebook.

You are expected to carefully document your code and results throughout these programming problems per the format specified. I include specific reminders in each part about expected outputs, but well documented code, discussions about the code and results, and answers to specific questions in each part are implied even when not explicitly stated.

P 01-01 Aliasing with Sinusoids

Consider an analog signal consisting of three sinusoids $x(t) = 2\sin(2\pi f_1 t) + 2\sin(2\pi f_2 t) + 2\sin(2\pi f_3 t)$ where $f_1 = 1000$, $f_2 = 4000$, and $f_3 = 6000$.

(a) We cannot directly represent an analog signal on a digital computer, but we can approximate one by using a closely spaced time vector, e.g., `tc=0:1e-5:2e-3` (`tc=np.arange(0,2e-3+1e-5,1e-5)`) which specifies a time vector from 0 to 2 ms with 10 μs spacing. Define variable `x` to correspond to the analog signal described above and

plot `x` versus `tc` in blue using the `plot` (`plt.plot`) command. Title your plot and label your axes.
Expected output: The output for this part should be the labeled plot of `x`.

(b) Define vector `nT` to be the sample points for $f_s = 5000$ over the time range 0 to 2 ms using the command `nT=0:1/fs:2e-3` (`nT=np.arange(0,2e-3+1./fs,1./fs)`). Define the sampled signal `xs` using this time vector. Create another plot with `x` versus `tc` in blue and plot `xs` versus `nT` in red on top using the `stem` (`plt.stem`) command. Title your plot, label your axes, and add a legend. The sampled signal `xs` should match `x` at sample points `nT`—does it?
Expected output: The output for this part should be the labeled plot of `x` and `xs` along with answers to questions as requested.

(c) Since sampling frequency $f_s = 5000$ does not satisfy the sampling theorem, you will alias when sampling. Define frequencies `fa1`, `fa2`, and `fa3` to correspond to the aliased versions of frequencies f_1, f_2, and f_3, respectively (you can simply define these—you do not need to compute them in the code). Define variable `xa` to be the aliased version of `x`. Since `xa` is an analog signal, you want to use time vector `tc` here. Create another plot with `x` versus `tc` in blue and a stem plot of `xs` versus `nT` in red. Plot aliased signal `xa` versus `tc` on top with a green dashed line. Title your plot, label your axes, and add a legend. The aliased signal `xa` should match both the original signal `x` and the sampled signal `xs` at the time points `nT`—does it? Which of the three sinusoids have aliased?
Expected output: The output for this part should be the labeled plot of `xa`, `x`, and `xs` along with answers to questions as requested.

(d) Repeat part (c) for $f_s = 9000$. Remember that you will have to redefine the vector `nT` for this new sampling frequency f_s.
Expected output: The output for this part should be the labeled plot of `xa`, `x`, and `xs` along with answers to questions as requested.

(e) Repeat part (c) for $f_s = 20{,}000$. Remember that you will have to redefine the vector `nT` for this new sampling frequency f_s.
Expected output: The output for this part should be the labeled plot of `xa`, `x`, and `xs` along with answers to questions as requested.

P 01-02 Aliasing with a Triangular Wave

In this problem, you will plot the solution to text problem 01-07 using Matlab. Note that the programming for this problem can be completed by using dummy values for `A`, `B`, `f1`, and `f2` and the correct values can be plugged in once you have completed the text problem.

(a) Define time vector `t=-0.125:0.01:1.125` (`t=np.arange(-0.125,1.13,0.01)`). Since the triangular waveform is linear, we can use the usual linear vector creation tools in Matlab (python) to stitch together a triangle wave. We can define the first "leg" of the signal by noting that the time range is -0.125 to 0.125 and x ranges from -1.5 to 1.5. Define the first leg as `x=(-0.125:0.01:0.125)*12` (`x=np.arange(-0.125,0.13,0.01)*12`). Now we can use the flipping commands and concatenation to define the signal `x` over the full range of interest $t \in [-0.125, 1.125]$. Define `x=[x, fliplr(x(1:end-1)), x(2:end), fliplr(x(1:end-1)), x(2:end)]` (`x=np.concatenate((x, x[-2:0:-1], x, x[-2:0:-1], x))`). Plot `x` versus `t`. Title your plot and label your axes.
Expected output: The output for this part should be the labeled plot of `x`.

(b) From text problem 01-07, the reconstructed signal will be $x_{rec}(t) = A\sin(2\pi f_1 t) + B\sin(2\pi f_2 t)$. Define variables `A`, `B`, `f1`, and `f2` corresponding to

the values you solved for (or to dummy values in the meantime). Define signal `xrec`. Plot `x` versus `t` and `xrec` versus `t` on the same axes. Use different line styles to distinguish between your plot of `x` and `xrec`. Title your plot, label your axes, and provide a legend.
Expected output: The output for this part should be the labeled plot of `x` and `xrec`.

(c) The signal `xrec` is an aliased version of the triangle wave. Is it a reasonable approximation? Comment on the appearance of `xrec` versus `x`. As a sanity check, you know that the aliased signal should be identical to the original signal at sample points $t=nT, T=1/f_s, n=0,\pm 1,\pm 2,\ldots$.
Expected output: The output for this part should be discussion as requested.

The following table is provided to let you know the point allocation for each problem. More details about point allocations for problem parts are included in gradescope.

Problem	**Points**
Text 01-02	/10
Text 01-04	/10
Text 01-07	/20
Text 01-26	/25
Programming 01-01	/25
Programming 01-02	/15
TOTAL	/105

T01 - 02

$f_s = 4 Hz$ $\quad x(t) = 8\sin(2\pi t) + 8\sin(10\pi t) + 4\sin(14\pi t)$

$x_a(t)$

$$x(nT) \overset{?}{\doteq} x_a(nT)$$

$f_1 = 1Hz \quad f_2 = 5Hz \quad f_3 = 7Hz$

$f_{a1} = 1Hz \quad f_{a2} = 5-4 \quad f_{a3} = 7-4$

$= 1Hz \quad = 3Hz - 4 = -1Hz$

$x_a(t) = 8\sin(2\pi t) + 8\sin(2\pi t) - 4\sin(2\pi t)$

$x_a(t) = 12\sin(2\pi t)$

$x(nT) = x_a(nT), \quad T = \frac{1}{4}$

$x(n\frac{1}{4}) = x_a(n\frac{1}{4})$

$8\sin(2\pi(n\frac{1}{4})) + 8\sin(10\pi(n\frac{1}{4})) + 4\sin(14\pi(n\frac{1}{4})) = 12\sin(2\pi(n\frac{1}{4}))$

$n=0 \Rightarrow 0 = 0$

$n=1 \Rightarrow 12 = 12$

$n=2 \Rightarrow 0 = 0$

$n=3 \Rightarrow -12 = -12$

$n=4 \Rightarrow 0 = 0$

$n=5 \Rightarrow 12 = 12$

$\downarrow$ cyclical, and always equal $x(nT) = x_a(nT)$

$f_s = 12 \quad f_{a1} = 1Hz \quad f_{a2} = 5Hz \quad f_{a3} = 7-12$

$= -5$

$x_a = 8\sin(2\pi t) + 8\sin(10\pi t) - 4\sin(10\pi t) = 8\sin(2\pi t) + 4\sin(10\pi t)$

$x(nT) \doteq x_a(nT) \Rightarrow T = 1/12 \Rightarrow x(n 1/12) \overset{?}{\doteq} x_a(n 1/12)$

$8\sin(\frac{1}{6}\pi n) + 8\sin(\frac{5}{6}\pi n) + 4\sin(\frac{7}{6}\pi n) = 8\sin(\frac{1}{6}\pi n) + 4\sin(5/6\,\pi n)$

$n=0 \Rightarrow$	$0 = 0$	$n=7 \Rightarrow$	$-6 = -6$
$n=1 \Rightarrow$	$6 = 6$	$n=8 \Rightarrow$	$-3.46 = -3.46$
$n=2 \Rightarrow$	$3.46 = 3.46$	$n=9 \Rightarrow$	$-12 = -12$
$n=3 \Rightarrow$	$12 = 12$	$n=10 \Rightarrow$	$-3.46 = -3.46$
$n=4 \Rightarrow$	$3.46 = 3.46$	$n=11 \Rightarrow$	$-6 = -6$
$n=5 \Rightarrow$	$6 = 6$	$n=12 \Rightarrow$	$0 = 0$
$n=6 \Rightarrow$	$0 = 0$	$n=13 \Rightarrow$	$6 = 6$

Cycles every 12 samples with both x and x_a having the same values

T01-04 $x(t)=3\cos(10\pi t)+2\cos(6\pi t)\cos(8\pi t)$, $f_s=7Hz, 12Hz$

$\cos a\cdot\cos b=\frac{\cos(a-b)+\cos(a+b)}{2}$

$x(t)=3\cos(10\pi t)+\cos(2\pi t)+\cos(14\pi t)$ $f_s=7$

$f_1=5\ Hz$ $f_2=1Hz$ $f_3=7Hz$

$f_{a1}=5-7=-2$ $f_{a2}=1$ $f_{a3}=7-7=0$

$x_a(t)=3\cos(4\pi t)+\cos(2\pi t)+1$

$f_s=12\ Hz$ $f_{a1}=5Hz$ $f_{a2}=1Hz$ $f_{a3}=7-12$
$=-5Hz$

$x_a(t)=3\cos(10\pi t)+\cos(2\pi t)+\cos(10\pi t)$

$x_a(t)=4\cos(10\pi t)+\cos(2\pi t)$

T01-07 $T_0=0.5s$ $f_s=16\ Hz$

$x_{rec}(t)=A\sin(2\pi f_1 t)+B\sin(2\pi f_2 t)$

odd periodic signals have odd harmonics

$$x(t)=\sum_{m=1,3,5\ldots} b_m\sin(2\pi m t)=b_1\sin(2\pi t)+b_3\sin(6\pi t)+\ldots$$

$f_0=\frac{1}{T_0}=\frac{1}{0.5}=2$

$f_m=f_0\cdot m$

$f_1=2Hz$
$f_2=6Hz$

$f_1=2$	$f_{a1}=2$
$f_3=6$	$f_{a3}=6$
$f_5=10$	$f_{a5}=\lvert 10-16=-6$
$f_7=14$	$f_{a7}=14-16=-2$

$x_{rec}=A\sin(2\pi f_1 t)+B\sin(2\pi f_2 t)$

$$x(t)=\begin{cases}12t, & -0.125<t\le 0.125\\ -12t, & 0.125<t\le 0.375\end{cases}$$

$T=1/16$

$x(nT)=x_a(nT)$

$x(1T)=x(1/16)=x_a(1/16)\Rightarrow 3/4=A\sin(2\pi 2(1/16))+B\sin(2\pi 6(1/16))$

$x(2T)=x(1/8)=x_a(1/8)\Rightarrow 1.5=A\sin(2\pi 2(1/8))+B\sin(2\pi 6(1/8))$

$\hookrightarrow 3/4=A\frac{\sqrt2}{2}+B\frac{\sqrt2}{2}$

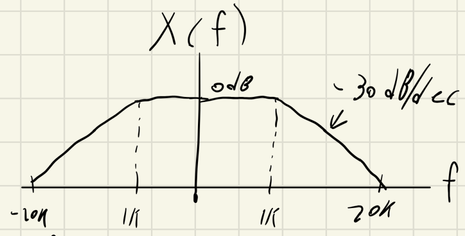
$3/2=A-B\longrightarrow A=3/2+B$

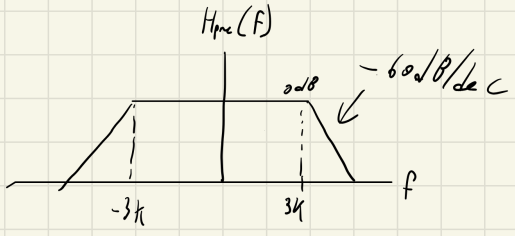
$\frac34=\left(\frac32+B\right)\frac{\sqrt2}{2}+B\frac{\sqrt2}{2}\Rightarrow\frac34=\frac{3\sqrt2}{4}+B\sqrt2\Rightarrow B=-0.220$

$A = 1.5 - 0.220 = 1.28$ $A = 1.28$

$x_{rec}(t) = 1.28 \sin(2\pi 2t) - 0.220 \sin(2\pi 6t)$

T01-26 a





$f_s = 8 kHz$

$f_a = 4 kHz$

$f_{xdist} = \log_{10}\left(\frac{4kHz}{1kHz}\right)$
$f_{xdist} = 0.602 dec$
$A_x(f) = 30 \cdot 0.602$
$A_x = 1.806 dB$

$f_{Hdist} = \log_{10}\left(\frac{4k}{3k}\right)$
$f_{Hdist} = 0.125 dec$
$A(f) = 60 \cdot 0.125$
$A = 7.496 dB$

$|x_a(f_a)| = 0 - A_x - A$
$|x_a(f_a)| = -9.303 dB$

b

$|x_a(f_a)| = -100 dB = -A_x - A$
$-100 dB = -1.806 dB - A$
$A = 98.19 dB$

c

$A = 98.19 dB = f_{Hdist} \cdot rolloff$
$98.19 dB = 0.125 dec \cdot r \frac{dB}{dec}$
$785.6 \frac{dB}{dec} = rolloff$

d

$f_s = 15 kHz$ worst aliasing will occur at $f_a = 7.5 kHz$

$f_{xdist} = \log_{10}\left(\frac{7.5k}{1k}\right)$
$= 0.875 dec$
$A_x = 30 \cdot 0.875$
$A_x = 26.25$

$f_{Hdist} = \log_{10}\left(\frac{7.5}{3.5}\right)$
$= 0.398 dec$
$A_H = 60 \cdot 0.398$
$A_H = 23.88 dB$

$$|X_a(f_a)| = -A_X - A_H$$

$$|X_a(f_a)| = -50.13\,dB$$

$$|X_a(f_a)| = -100 = -A_X - A_H$$

$$-100 = -26.25 - A_H$$

$$A_H = 73.75$$

$$\text{rolloff} = \frac{A_H}{f_{H\,dist}} = \frac{73.75}{0.398}$$

$$\text{rolloff} = 185.3\,\frac{dB}{dec}$$

EE495 HW 1

Problem 01-01 Aliasing with sinusoids

Kevin Morales

Part A

In part A, we're to plot a sinusoidal signal x(t), using numpy and matplotlib libraries.

In [225…

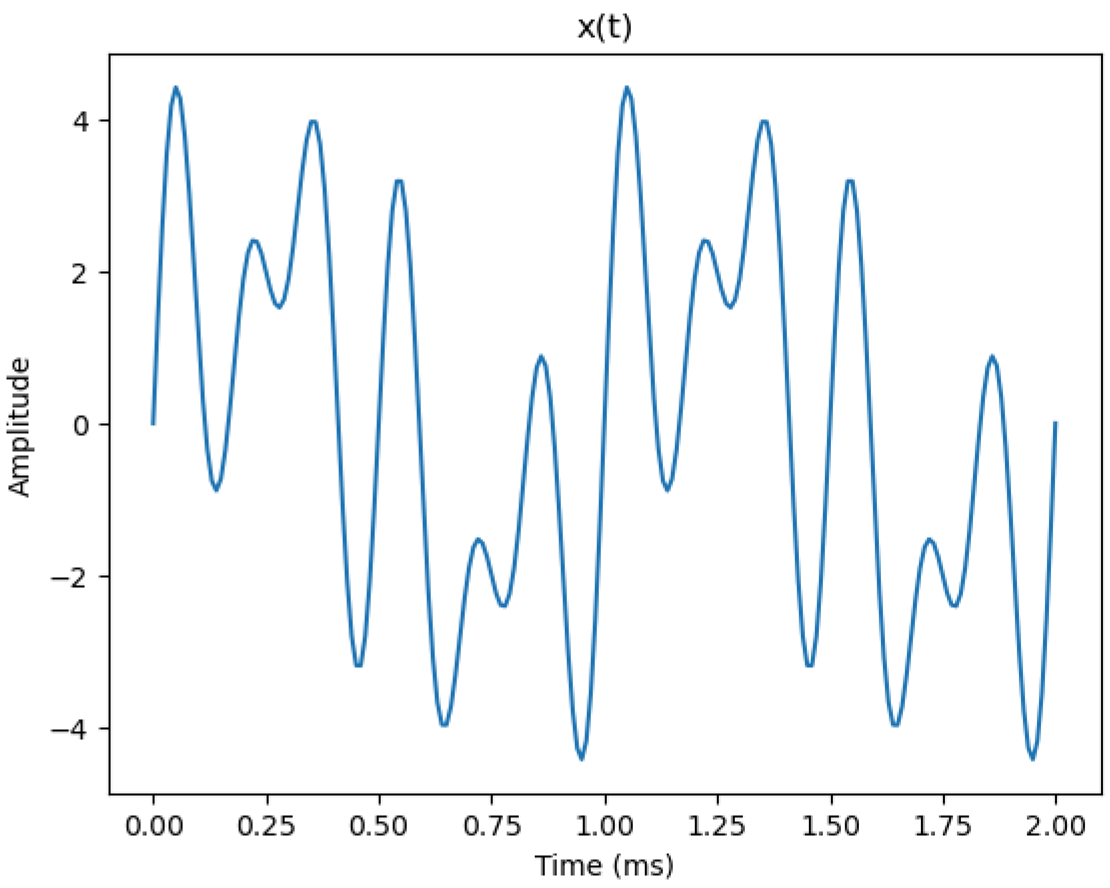
```
import matplotlib.pyplot as plt
import numpy as np
```

In [227…

```
tc = np.arange(0,2e-3+1e-5,1e-5)
f1 = 1000
f2 = 4000
f3 = 6000
x = 2*np.sin(2*np.pi*f1*tc) + 2*np.sin(2*np.pi*f2*tc) + 2*np.sin(2*np.pi*f3*tc)
plt.plot(tc * 1000,x)
plt.title("x(t)")
plt.ylabel("Amplitude")
plt.xlabel("Time (ms)")
```

Out[227…

```
Text(0.5, 0, 'Time (ms)')
```



Seen above is the expected output, a sinusoid, with 3 different frequencies all of which have an amplitude of 2.

Part B

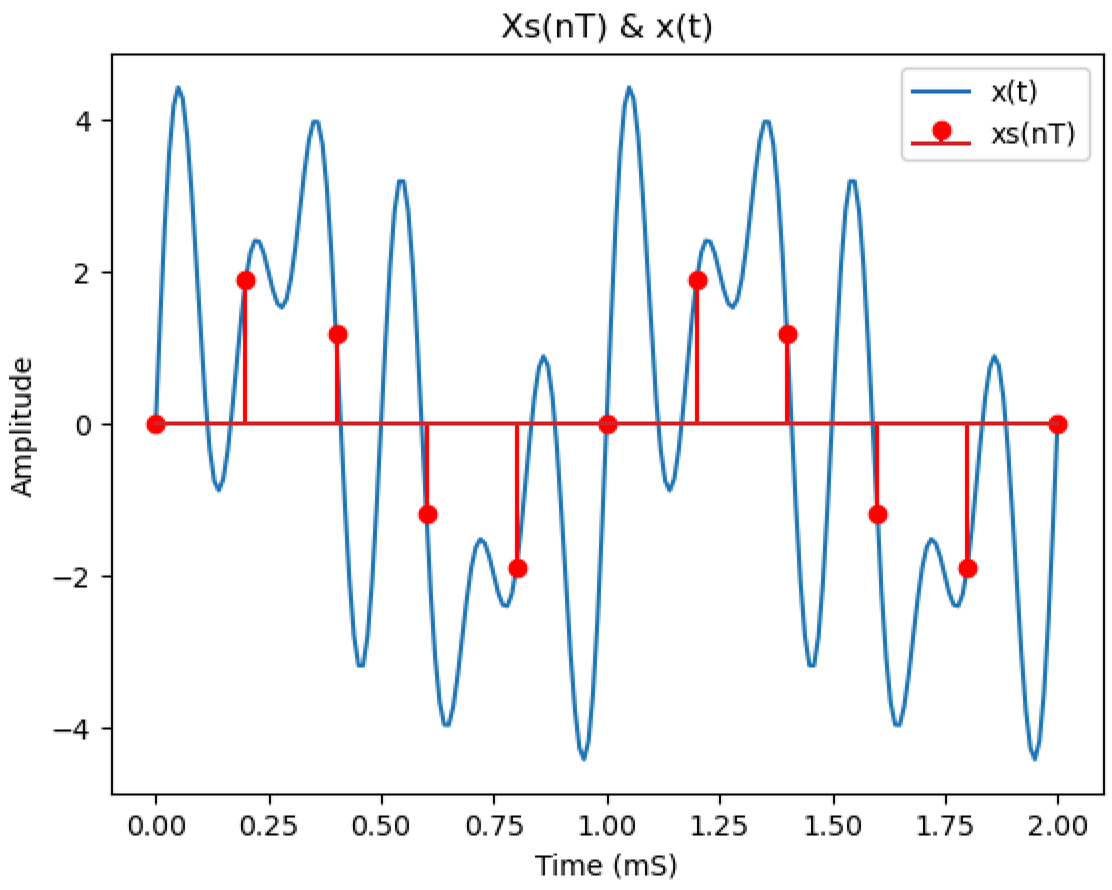
In part two we're tasked with sampling the sinusoidal signal x(t) generated in part A. This sampling will create a new discrete signal xs(nT).

In [232…

```
fs = 5000
Ts = 1 / fs
nT = np.arange(0,2e-3+Ts, Ts)
xs =  2*np.sin(2*np.pi*f1*nT) + 2*np.sin(2*np.pi*f2*nT) + 2*np.sin(2*np.pi*f3*nT)
plt.plot(tc *1000, x)
plt.stem(nT*1000, xs, "r")
plt.title("Xs(nT) & x(t)")
plt.ylabel("Amplitude")
plt.xlabel("Time (mS)")
plt.legend(['x(t)', 'xs(nT)'])
```

Out[232…

```
<matplotlib.legend.Legend at 0x23d3f776c90>
```



Seen above is the sampled signal xs(nT) overlayed with the original signal x(t). As you can see every point of xs corresponds with a point on x. This leads me to belive that the signal x(t) was sampled properly.

Part C

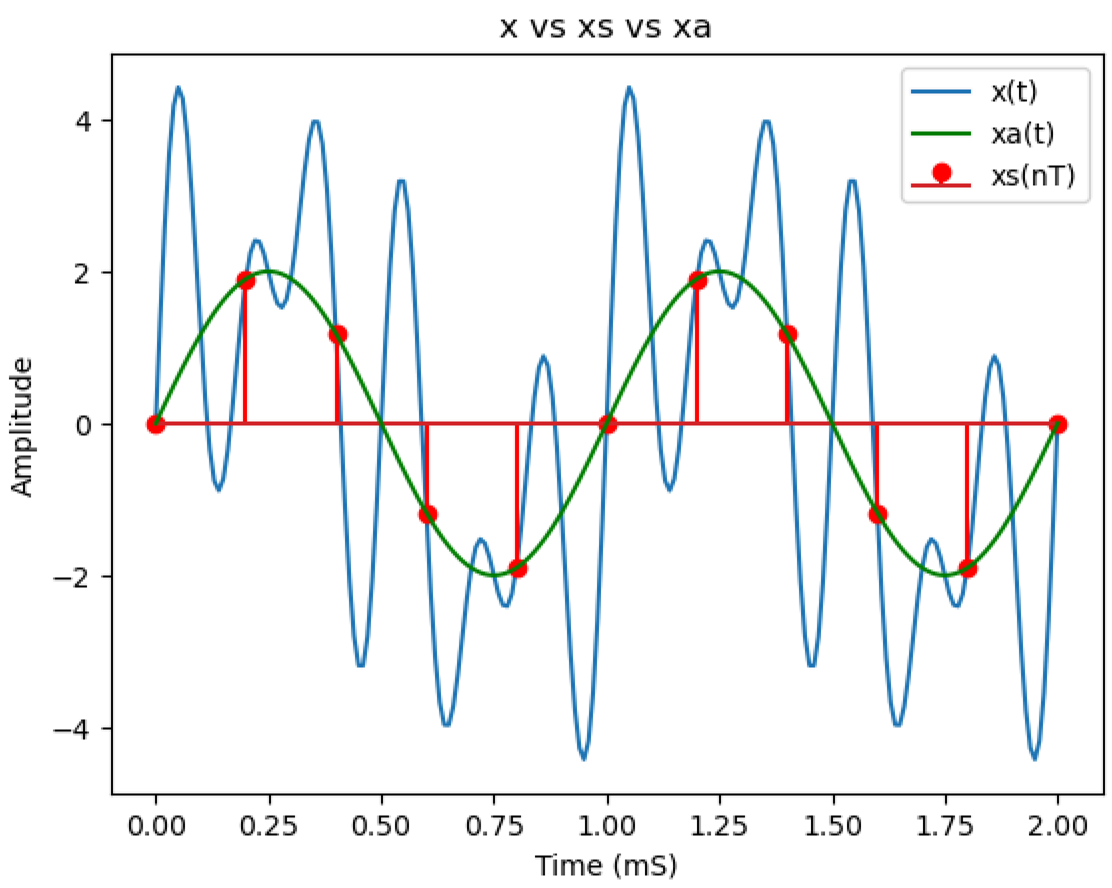
In this part, we're tasked with generating the signal which will result from aliasing. This signal will be called xa(t).

In [237…

```
fa1 = 1000
fa2 = -1000
fa3 = 1000

xa = 2*np.sin(2*np.pi*fa1*tc) + 2*np.sin(2*np.pi*fa2*tc) + 2*np.sin(2*np.pi*fa3*tc)
plt.plot(tc*1000, x)
plt.stem(nT*1000,xs, "r")
plt.plot(tc*1000, xa,"g")
plt.title("x vs xs vs xa")
plt.xlabel("Time (mS)")
plt.ylabel("Amplitude")
plt.legend(['x(t)', 'xa(t)', 'xs(nT)'])
```

Out[237…

```
<matplotlib.legend.Legend at 0x23d3f6bd3a0>
```



Seen above is the resulting waveform xa(t) overlayed onto the plot from part b. The resulting wave only has one frequency, 1000 Hz. Although the signal xa(t) looks nothing like our original signal x(t). The sampled points seen in xs(nT) all line up with x and xa, showing proper reconstruction from the aliased signal.

Part D

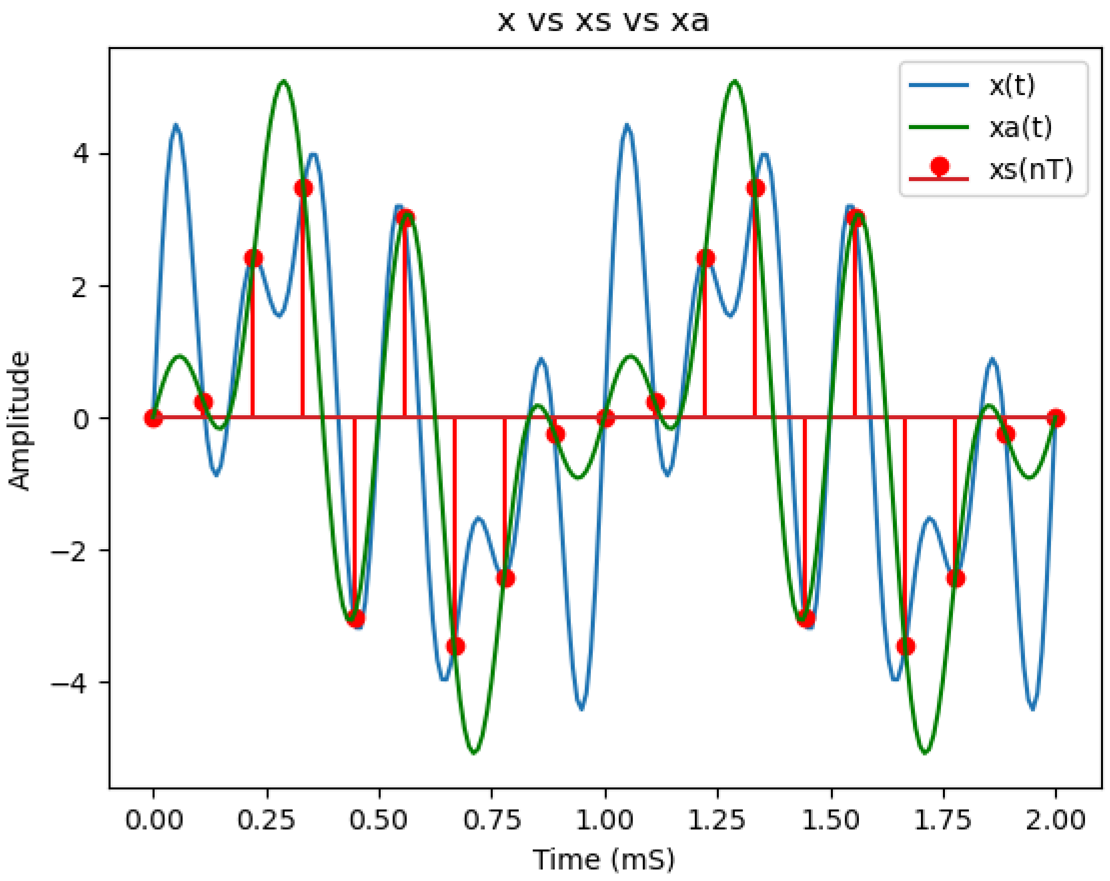
This part will have the same process as part c, with the exception being the sampling frequency increasing to 9000 Hz.

In [245…

```
fs = 9000
Ts = 1 / fs
nT = np.arange(0,2e-3+Ts, Ts)
xs =  2*np.sin(2*np.pi*f1*nT) + 2*np.sin(2*np.pi*f2*nT) + 2*np.sin(2*np.pi*f3*nT)

fa1 = 1000
fa2 = 4000
fa3 = -3000

xa = 2*np.sin(2*np.pi*fa1*tc) + 2*np.sin(2*np.pi*fa2*tc) + 2*np.sin(2*np.pi*fa3*tc)

plt.plot(tc*1000, x)
plt.stem(nT*1000,xs, "r")
plt.plot(tc*1000, xa,"g")
plt.title("x vs xs vs xa")
plt.xlabel("Time (mS)")
plt.ylabel("Amplitude")
plt.legend(['x(t)', 'xa(t)', 'xs(nT)'])
```

Out[245…

```
<matplotlib.legend.Legend at 0x23d3fc34170>
```



The result of the same process with the higher frequency sampling is similar. The resulting waveform matches all of the sampled points, but the recreated signal doesn't match the original. In this case, we have 3 seperate frequencies, resulting in higher amplitudes, showing a bit more fidelity to the original but not close enough.

Part E

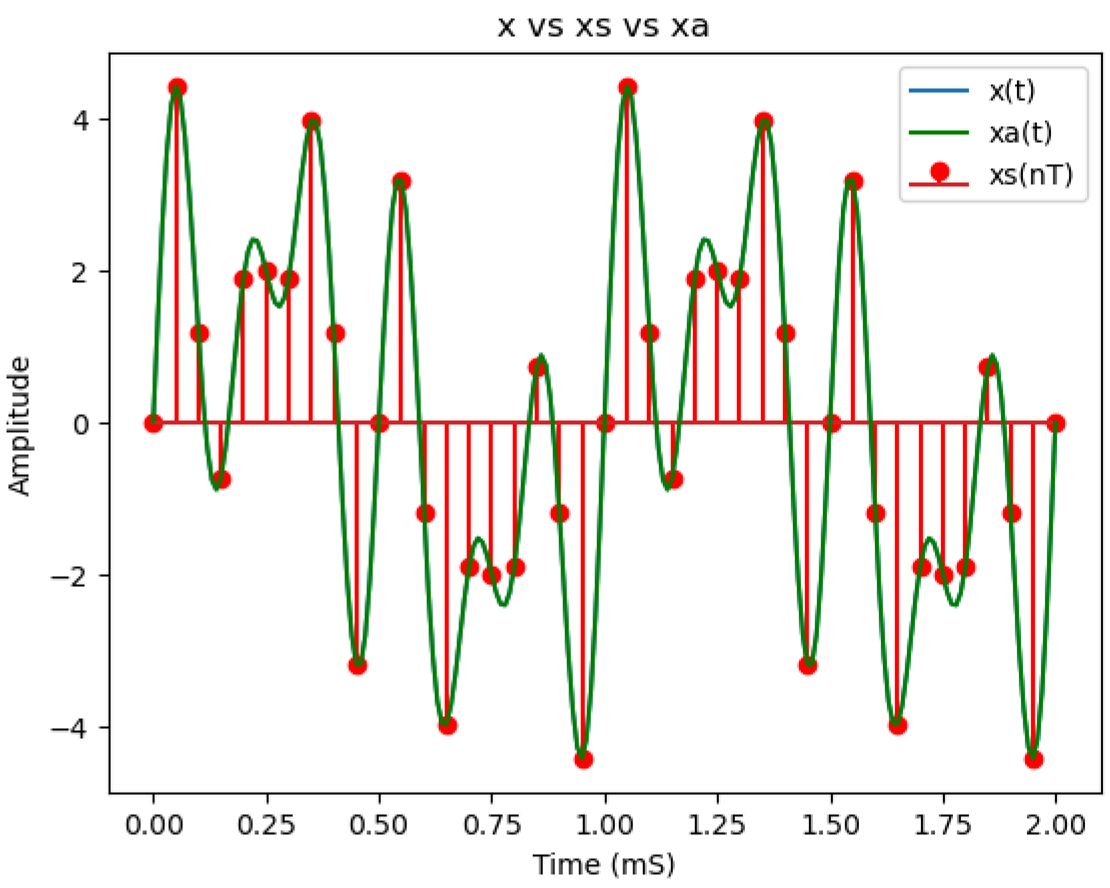
In this final part, we further increase the sampling frequency of 20000 HZ, an extent that it could be considered oversampled.

In [254…

```
fs = 20000
Ts = 1 / fs
nT = np.arange(0,2e-3+Ts, Ts)
xs =  2*np.sin(2*np.pi*f1*nT) + 2*np.sin(2*np.pi*f2*nT) + 2*np.sin(2*np.pi*f3*nT)

fa1 = 1000
fa2 = 4000
fa3 = 6000

xa = 2*np.sin(2*np.pi*fa1*tc) + 2*np.sin(2*np.pi*fa2*tc) + 2*np.sin(2*np.pi*fa3*tc)
```

```
plt.plot(tc*1000, x)
plt.stem(nT*1000,xs, "r")
plt.plot(tc*1000, xa,"g")
plt.title("x vs xs vs xa")
plt.xlabel("Time (mS)")
plt.ylabel("Amplitude")
plt.legend(['x(t)', 'xa(t)', 'xs(nT)'])
```

```
Out[254… <matplotlib.legend.Legend at 0x23d3fbee300>
```



The result of this attempt is a recreation that looks exactly like the original signal. This is due to the sampling frequency 20 kHz being more than twice the highest frequency of 6 kHz. On top of having the same amplitude at every sampled point, every point in between has the same value as well. This shows the right execution of sampling a signal at a high enough frequency and recreating it without having any aliasing.

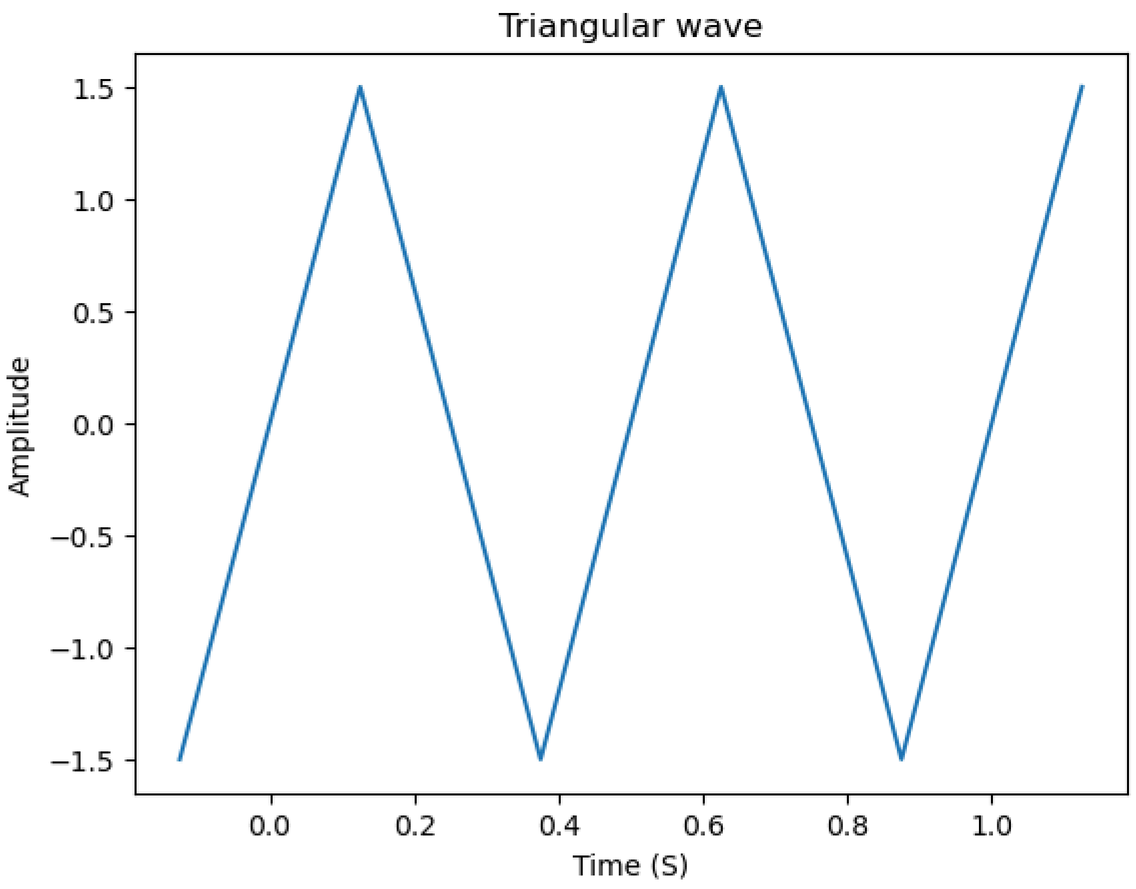
Problem 01-02 Aliasing with a triangular wave

Part A

In this first part, we have to create a triangular wave, exactly like the one seen on problem T01-07. This is to be done by creating the first segment of x, then concatenating it together and flipping it repeatedly until we have the right signal.

In [279…
```
t = np.arange(-0.125, 1.13, 0.01)
x = np.arange(-0.125,0.13,0.01)*12
x = np.concatenate((x,x[-2:0:-1],x,x[-2:0:-1],x))
plt.plot(t,x)
plt.title("Triangular wave")
plt.ylabel("Amplitude")
plt.xlabel("Time (S)")
```

Out[279…
```
Text(0.5, 0, 'Time (S)')
```



Seen above is the signal x(t), which is exaclt the same signal as the one seen on the problem T01-07.

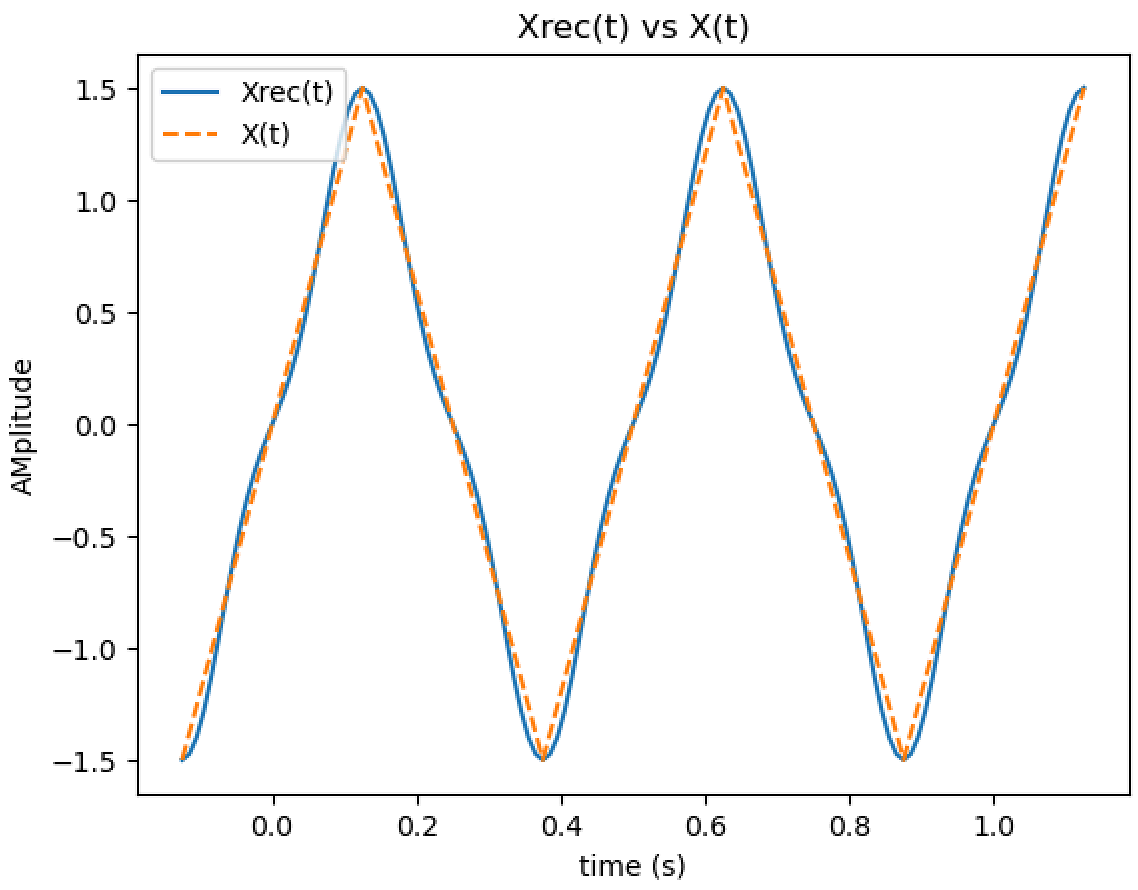
Part B

In this part, we're going to use the results from our calculations in problem T01-07 to plot Xrec(t) the recreation of the signal after sampling at 16 Hz and recreating it.

In [286…
```
A = 1.28
B = -0.220
f1 = 2
f2 = 6

Xrec = A*np.sin(2*np.pi*f1*t)+B*np.sin(2*np.pi*f2*t)
plt.plot(t,Xrec)
plt.plot(t,x,'--')
plt.title('Xrec(t) vs X(t)')
plt.xlabel('time (s)')
plt.ylabel('AMplitude')
plt.legend(['Xrec(t)', 'X(t)'])
```

Out[286…

```
<matplotlib.legend.Legend at 0x23d418bbd40>
```



As we can see the recreated signal looks very close to the original signal.

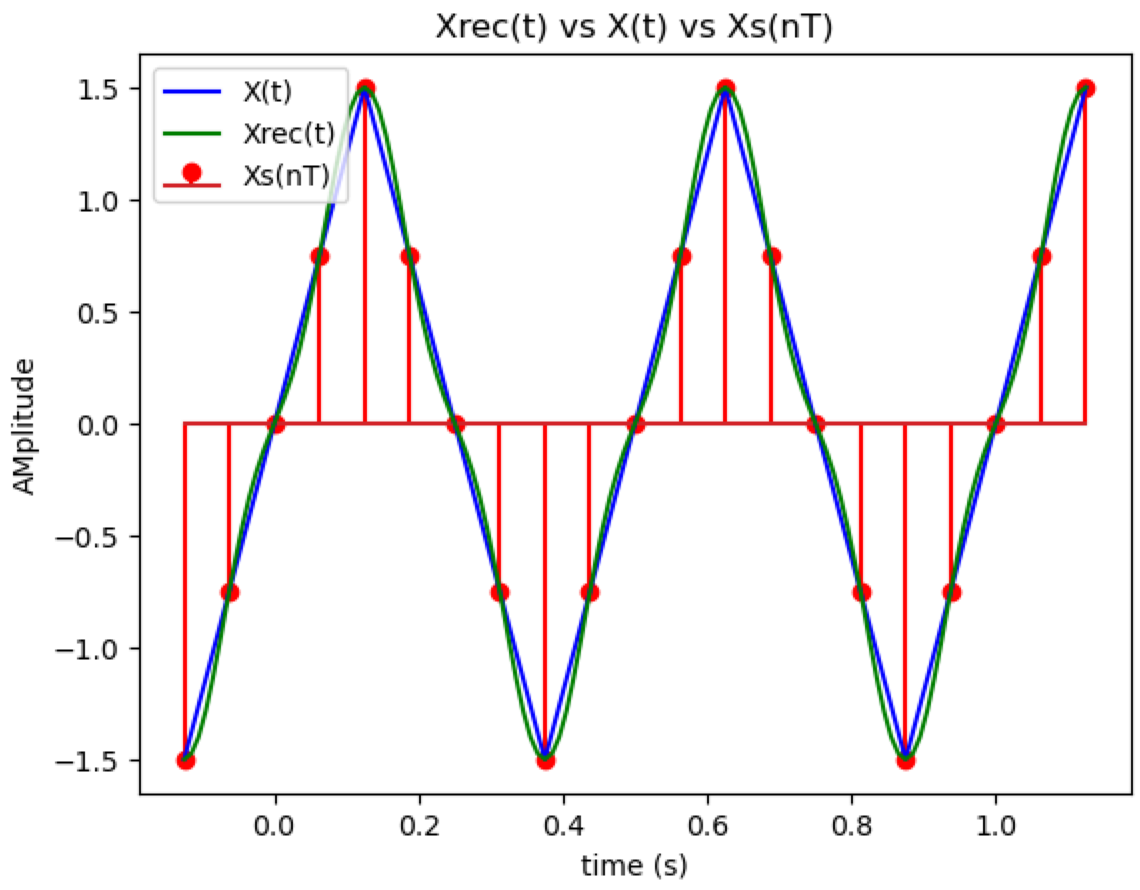
Part c

For the final part, we simply have to verify that the approximation is reasonable. By plotting the sampled points as well we can do a quick sanity check.

In [297…

```
fs = 16
Ts = 1 / fs
nT = np.arange(-0.125,1.125+Ts, Ts)
xs = A*np.sin(2*np.pi*f1*nT)+B*np.sin(2*np.pi*f2*nT)

plt.stem(nT,xs, 'r')
plt.plot(t,x, 'b')
plt.plot(t,Xrec, 'g')
plt.title('Xrec(t) vs X(t) vs Xs(nT)')
plt.xlabel('time (s)')
plt.ylabel('AMplitude')
plt.legend(['X(t)','Xrec(t)', 'Xs(nT)'])
```

```
Out[297…   <matplotlib.legend.Legend at 0x23d4185e300>
```



As we can see in the above plot, the sampled points correspond to both the original signal and the sampled signal. This proves that our Xrec was generated correctly.