HW1

January 22, 2024

1 HW 1

NAME: Edwin Ruiz

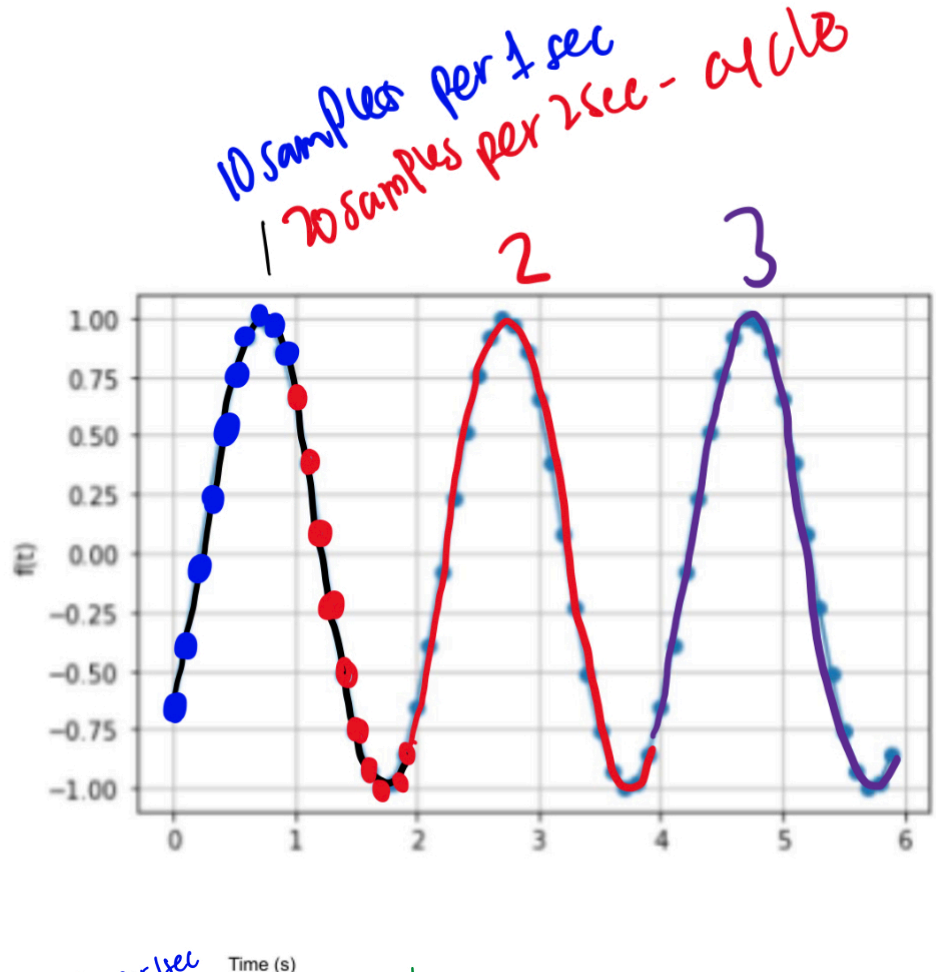
PID: A17136339

1.0.1 Note to Grader:

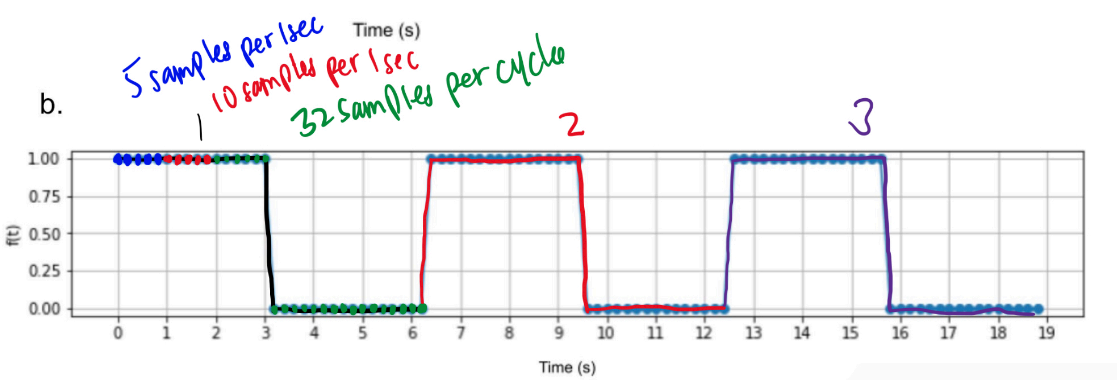
- Document contains “Notes”, “Answers”, “Reference”, Notes can be thought of as the reasoning and can be referred to when answer needs detailed explanation. Thank you.

1. For each of the plotted signals below, what is the period (T), sampling rate (F_s) and sampling time (T_s)? (Hint: Count the number of samples occurring in 1s, i.e. the interval 0 t <1.)

a.



b.



Notes: - (T) = period = one complete cycle of signal = $\frac{1}{\text{frequency (f): cycles per second}} = \frac{\text{N (number of samples in one complete cycle)}}{F_s\text{(sampling rate)}}$

1.1 Answer:

- graph a: $T = \frac{20}{10} = 2$ or $\frac{1}{0.5} = 2$
- graph b: $T = \frac{32}{\frac{96}{19}} = 6.\overline{3}$; or $\frac{32}{\frac{96}{19}} = \frac{1}{\text{frequency (f)}} \rightarrow$ frequency (f) $= \frac{3}{19} \rightarrow T = \frac{1}{\frac{3}{19}} = 6.\overline{3}$

Notes: - F_s = sampling frequency/sampling rate = samples per second (sps: interval $0 \leq t < 1$)

$= \text{Hertz(Hz)} = \text{Reciprocal of one second } (s^-1) = \frac{1}{T_s \text{ (sampling interval)}} = \frac{\text{Total Samples}}{\text{Total Time}}$

1.2 Answer:

- graph a: $F_s = \frac{20 \cdot 3 = 60}{6} = 10$ Hz
- graph b: $F_s = \frac{32 \cdot 3 = 96}{19} = 5.0526315789473684210526315789473684210526315789473$

Notes: - (T_s) = sampling time = reciprocal of F_s (sampling rate) = $\frac{1}{F_s}$:

1.3 Answer:

- graph a: $(T_s) = \frac{1}{10} = 0.1$
- graph b: $(T_s) = \frac{1}{\frac{96}{19}} = \frac{19}{96} = 0.19791\overline{6}$

---

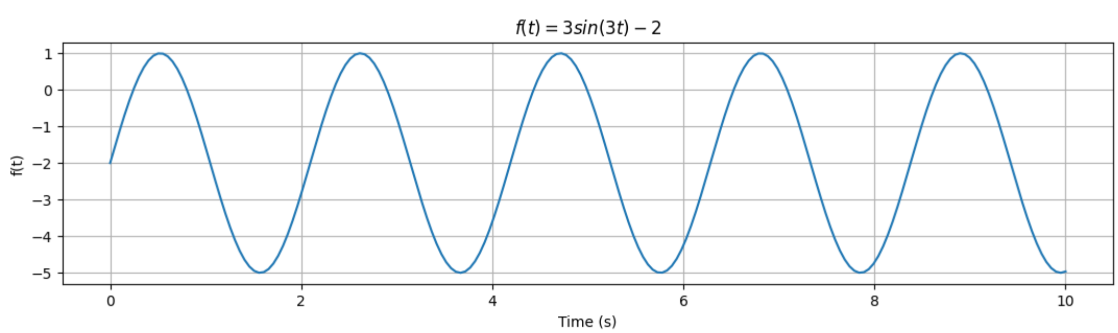
2. Use Python or MATLAB to plot each of the signals below in the range $t \in [0, 10s]$. For each signal, state whether or not it is periodic, and if it is periodic give the period. As a reminder, a periodic signal with period T is defined as a function that satisfies $f(t) = f(t + T)$ for all values of t.

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt

#range:     [0,10]s with 200 sample points
t = np.linspace(0, 10, 200)
```

a. $f(t) = 3\sin(3t) - 2$

[Hint: in Python, trig functions are in the numpy library, e.g. np.sin]

[2]:
```python
signalA_functiont = 3 * np.sin(3 * t) - 2

plt.figure(figsize = (13, 3))
plt.plot(t, signalA_functiont)
plt.title('$f(t) = 3 sin(3t) - 2$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```



Notes:

"sin" in $3\sin(3t)-2$ makes this a sinusoidal (sine) function making this function periodic and sine has a period of 2π:

Sinusoidal function form: $A\sin(wt+\phi)$ and $A\cos(wt+\phi)$ - A: amplitude - w: angular frequency - ϕ: phase shift

$3\sin(3t)-2$ to (is in) sinusoidal form $\rightarrow A\sin(wt+\phi)$

Period T Formula for Sinusoidal Function: $f(x) = sin(wx) \rightarrow \frac{2\pi}{|w|} = \frac{2\pi}{w}$ - $3\sin(3t)-2 \rightarrow A = 3, w = 3, \phi = 0$

1.4 Answer:

$f(t) = 3\sin(3t)-2$ is Periodic and its Period is: T $= \frac{2\pi}{w} = \frac{2\pi}{3}$ - $f(t) = 3\sin(3t)-2 = 3\sin\left(3\left(t+\frac{2\pi}{3}\right)\right)-2 = f\left(t+\frac{2\pi}{3}\right)$

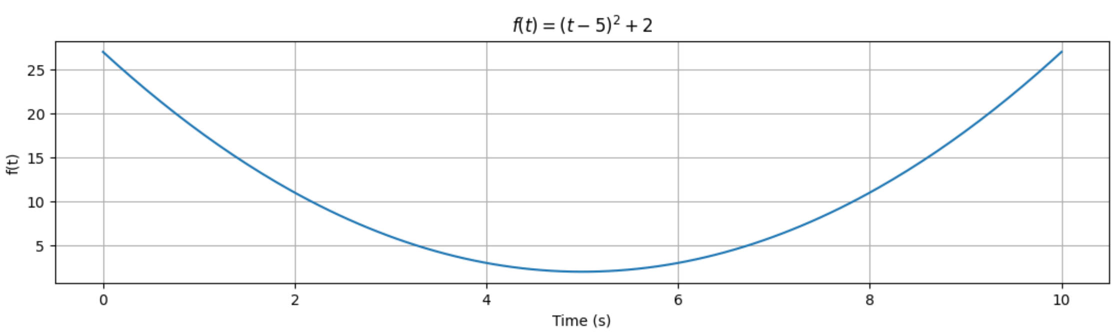
Reference:

"To find the period of a sine wave with equation $f(x) = sin(Ax)$, use the formula Period $= \frac{2\pi}{|A|}$. If $|A| = 1$, then the period of the sine wave is 2π. If $|A| < 1$, then the period will be larger, and if $|A| > 1$, then the period will be smaller." - https://study.com/academy/lesson/how-to-find-the-period-of-sine-functions.html#:~:text=To%20find%20the%20period%20of%20a%20sine%20wave%20with%20equation,sine%20w

---

b. $f(t) = (t-5)^2 + 2$

[3]:
```python
signalB_functiont = (t - 5)**2 + 2

plt.figure(figsize = (13, 3))
plt.plot(t, signalB_functiont)
plt.title('$f(t) = (t-5)^2 + 2$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```



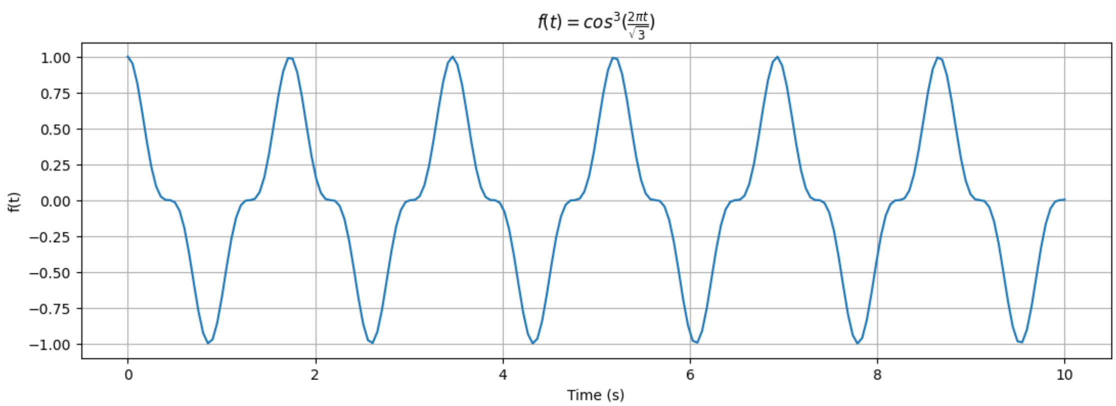
1.5 Answer:

$f(t) = (t-5)^2 + 2$ is a parabola (does not repeat at any interval = no value of T for which $f(t) = f(t+T)$ for all t) $\rightarrow$ **not periodic, so no period**.

---

c. $f(t) = \cos^3(\frac{2\pi t}{\sqrt{3}})$

[4]:
```
signalC_functiont = np.cos(2 * np.pi * t / np.sqrt(3))**3

plt.figure(figsize = (13, 4))
plt.plot(t, signalC_functiont)
plt.title('$f(t) = cos^3(\\frac{2\pi t}{\sqrt{3}})$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```



Notes:

"cos" in $\cos^3(\frac{2\pi t}{\sqrt{3}})$ also makes this a sinusoidal function making this function periodic and cos also has a period of 2π:

$\cos^3(\frac{2\pi t}{\sqrt{3}})$ to (is in) sinusoidal form $\rightarrow A\cos(wt+\phi)$

Period T Formula for Sinusoidal Function: $f(x) = cos(wx) \rightarrow \frac{2\pi}{|w|} = \frac{2\pi}{w}$ - $\cos^3(\frac{2\pi t}{\sqrt{3}}) \rightarrow A = 1, w = \frac{2\pi}{\sqrt{3}}, \phi = 0$

1.6 Answer:

$f(t) = \cos^3(\frac{2\pi t}{\sqrt{3}})$ is Periodic and its Period is: T $= \frac{2\pi}{w} = \frac{2\pi}{\frac{2\pi}{\sqrt{3}}} = 2\pi \cdot \frac{\sqrt{3}}{2\pi} = \sqrt{3}$ - $f(t) = \cos^3\left(\frac{2\pi t}{\sqrt{3}}\right) = cos^3\left(\frac{2\pi(t+\sqrt{3})}{\sqrt{3}}\right) = f(t+\sqrt{3})$

Reference:

"A function that has the same general shape as a sine or cosine function is known as a sinusoidal function."

"…the shape of the graph repeats after 2π, which means the functions are periodic with a period of 2π. A periodic function is a function for which a specific horizontal shift, P, results in a function equal to the original function: $f(x+P) = f(x)$ for all values of x in the domain of f."

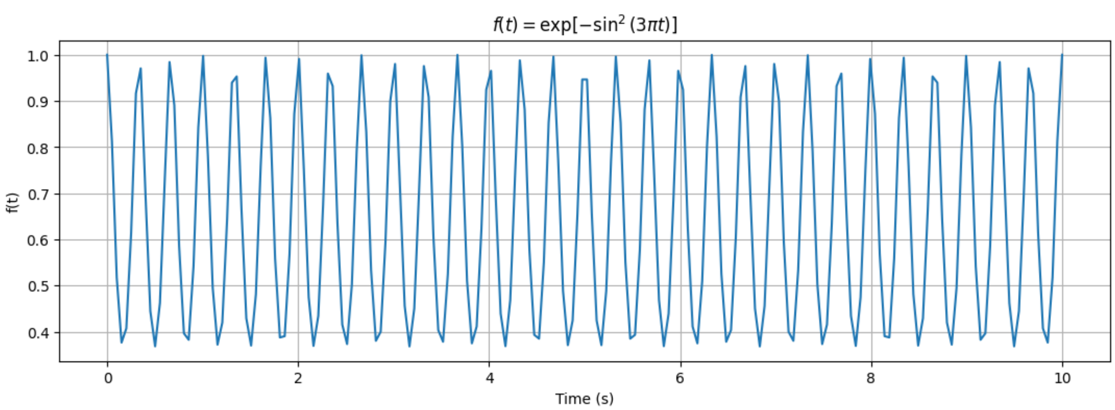
"Both sine and cosine functions have the same shape and are periodic, but they are phase-shifted relative to each other. Specifically, a cosine function is just a sine function shifted by $\frac{\pi}{2}$ radians or $90°$."

- https://courses.lumenlearning.com/suny-osalgebratrig/chapter/graphs-of-the-sine-and-cosine-functions/

---

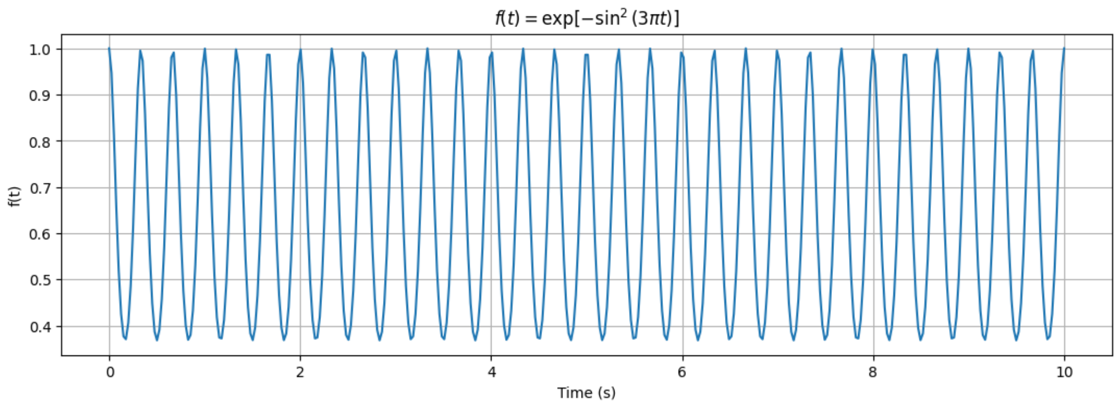
d. $f(t) = \exp\left[-\sin^2(3\pi t)\right]$

[5]:
```
signalD_functiont = np.exp(-np.sin(3 * np.pi * t)**2)

plt.figure(figsize = (13, 4))
plt.plot(t, signalD_functiont)
plt.title('$f(t) = \exp[-\sin^2(3\pi t)]$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```



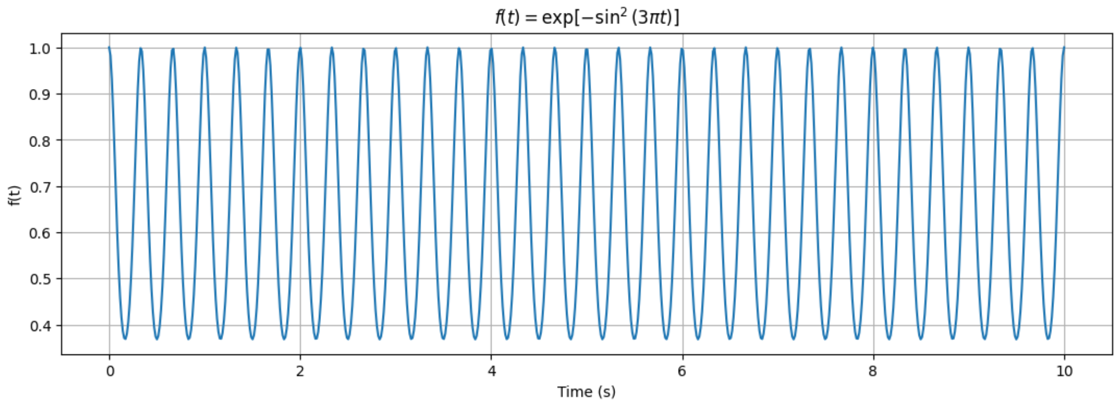
Notes: - Representation of signal seems distorted, due to sample size, thus sample size (t)/data points must increase for proper representation of the signal function.

```
[6]: t2 = np.linspace(0, 10, 400)
signalD_functiont2 = np.exp(-np.sin(3 * np.pi * t2)**2)

plt.figure(figsize = (13, 4))
plt.plot(t2, signalD_functiont2)
plt.title('$f(t) = \exp[-\sin^2(3\pi t)]$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```



Notes: - Some improvement

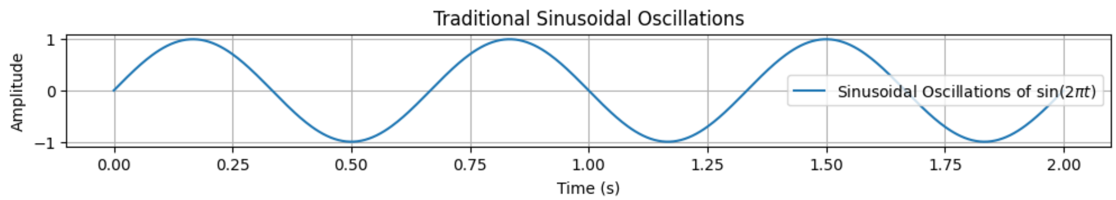
```
[7]: t3 = np.linspace(0, 10, 700)
signalD_functiont3 = np.exp(-np.sin(3 * np.pi * t3)**2)

plt.figure(figsize = (13, 4))
plt.plot(t3, signalD_functiont3)
plt.title('$f(t) = \exp[-\sin^2(3\pi t)]$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```

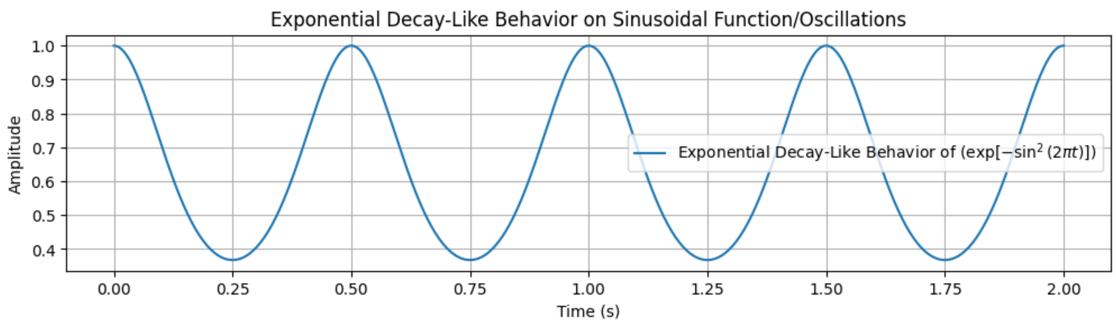


Notes: - Representation of Signal Function now looks more accurate: Threshold for exponential sinusoidal function is ≥ 700 sample points

[8]:
```
'''
Difference between traditional sinusoidal function/oscillations and Exponential␣
↪Decay-Like Behavior on Sinusoidal Function/Oscillations
'''
t_decay_example = np.linspace(0, 2, 500)

#Traditional sinusoidal function using signal d without squared sine and␣
↪exponential
y1 = np.sin(3 * np.pi * t_decay_example)
#Exponential decay-like behavior on sinusoidal function
y2 = np.exp(-np.sin(2 * np.pi * t_decay_example)**2)

plt.figure(figsize = (12, 6))
plt.subplot(4, 1, 1)
plt.plot(t_decay_example, y1, label = "Sinusoidal Oscillations of $\sin(2\pi␣
↪t)$")
plt.title("Traditional Sinusoidal Oscillations")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid(True)
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(t_decay_example, y2, label = "Exponential Decay-Like Behavior of␣
↪($\exp[-\sin^2(2\pi t)]$)")
plt.title("Exponential Decay-Like Behavior on Sinusoidal Function/Oscillations")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid(True)
```

```
plt.legend()
plt.show()
```





Notes:

Function $\exp\left[-\sin^2(3\pi t)\right]$ has sine but it is squared, which makes the signal always positive and between 0 and 1 (looses the negative portion of the sine wave (sinusoidal function)); furthermore, the exponential function reduces the amplitude of oscillations over time of the sine squared function making it show exponential decay like behavior rather than a regular oscillation. Lastly, $f(t) : \exp\left[-\sin^2(3\pi t)\right]$ can be called a composite function (having both sinusoidal and exponential elements):

Sinusoidal function form: $A\sin(wt + \phi)$

Period T Formula for Sinusoidal Function: $f(x) = sin(Ax) \rightarrow \frac{2\pi}{|A|} = \frac{2\pi}{w}$ - $\exp\left[-\sin^2(3\pi t)\right] \rightarrow A = -1, w = 3\pi, \phi = 0$

1.7 Answer:

Although, $f(t) = \exp\left[-\sin^2(3\pi t)\right]$ deviates from the typical sinusoidal function (oscillation) behavior and considering the look of the signals plotted at t = 700 samples function $\exp\left[-\sin^2(3\pi t)\right]$ is periodic and its Period: $T = \frac{2\pi}{w} = \frac{2\pi}{3\pi} = \frac{2}{3}$ - $f(t) = \exp\left[-\sin^2(3\pi t)\right] = \exp\left[-\sin^2(3\pi(t + \frac{2}{3}))\right] = f(t + \frac{2}{3})$

Notes: - If there were different values of t, exponential decay: exp(-t) will initially make the value of the function decrease fast then slow down as t increases.

References:

” … 2) As x increases, the function grows faster and faster (the rate of change increases).

3) As x decreases, the function values grow smaller, approaching zero.

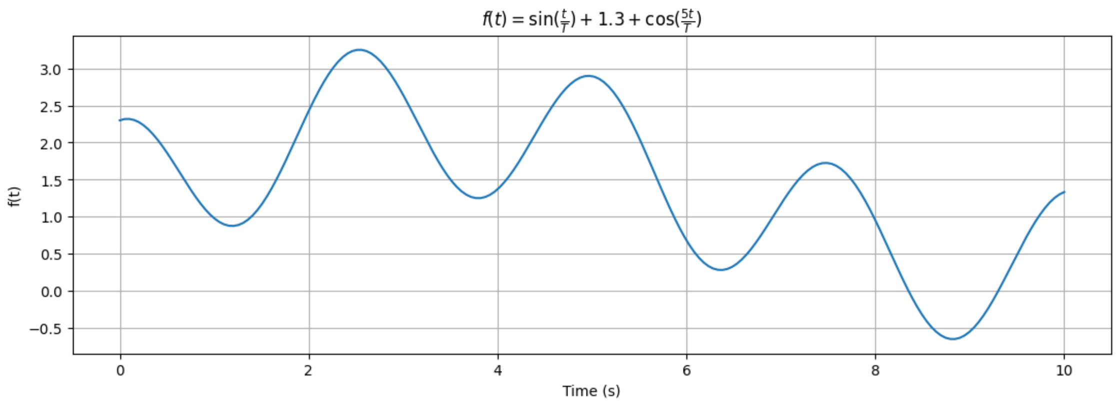
4) This is an example of exponential growth."

- http://www.opentextbookstore.com/precalc/1.4/Chapter%204.pdf

---

e. $f(t) = \sin\left(\frac{t}{T}\right) + 1.3 + \cos\left(\frac{5t}{T}\right)$

[9]:
```
T = 2
signalE_functiont = np.sin(t / T) + 1.3 + np.cos(5 * t / T)

plt.figure(figsize = (13, 4))
plt.plot(t, signalE_functiont)
plt.title('$f(t) = \sin(\\frac{t}{T}) + 1.3 + \cos(\\frac{5t}{T})$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```



Notes:

Sinusoidal function form: $A\sin(wt+\phi)$ and $A\cos(wt+\phi)$

$\sin\left(\frac{t}{T}\right)$ - $A = 1$, $w = \frac{1}{T} = \frac{1}{2}$, $\phi = 0$ - $2\pi = \frac{t}{T} \to t = 2\pi T \to T = 2 \to t = 2\pi \cdot 2 = 4\pi$ - $T = \frac{2\pi}{w} = \frac{2\pi}{\frac{1}{2}} = 2\pi \cdot \frac{2}{1} = 4\pi$

$\cos\left(\frac{5t}{T}\right)$ - $A = 1$, $w = \frac{5}{T} = \frac{5}{2}$, $\phi = 0$ - $2\pi = \frac{5t}{T} \to t = \frac{2\pi T}{5} \to T = 2 \to t = \frac{2\pi \cdot 2}{5} = \frac{4\pi}{5}$ - $T = \frac{2\pi}{w} = \frac{2\pi}{\frac{5}{2}} = 2\pi \cdot \frac{2}{5} = \frac{4\pi}{5}$

1.8 Answer:

$f(t) = \sin\left(\frac{t}{T}\right) + 1.3 + \cos\left(\frac{5t}{T}\right)$ is periodic because both components of the function have a common multiple $\to \frac{4\pi}{\frac{4\pi}{5}} \to 4\pi \cdot \frac{5}{4\pi} = \frac{1}{5}$, this means the function repeats itself every 4π units of t, which satisfies $f(t) = f(t+T)$ for all t - $T = 4\pi \to f(t+4\pi)$ for all t

Notes:

Angular frequency (w) only applies to individual sinusoidal functions, thus periods are determined by finding the common multiple (LCM).

Reference:

"If the periods of two periodic functions do not have a common multiple, then their sum is not periodic."

"Sums of periodic functions are often periodic. The sum of two periodic functions is often periodic, but not always."

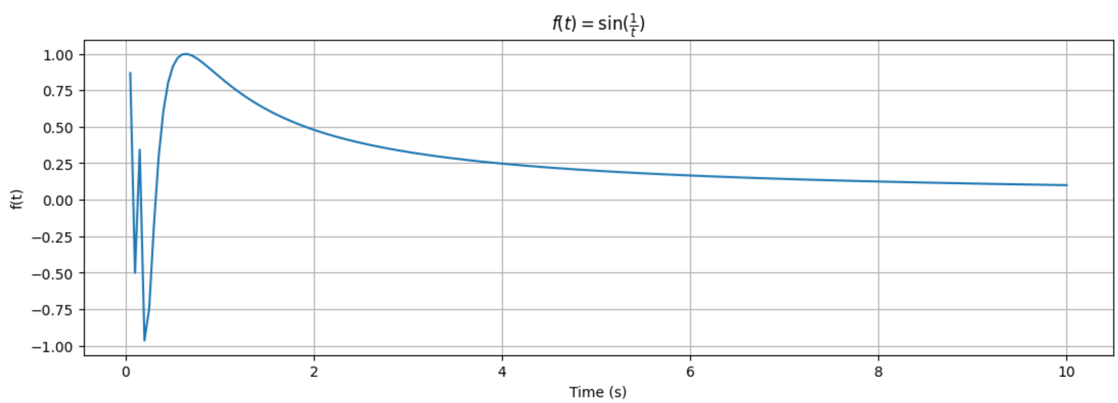
- https://mathblag.wordpress.com/2013/09/01/sums-of-periodic-functions/

---

f. $f(t) = \sin\left(\frac{1}{t}\right)$

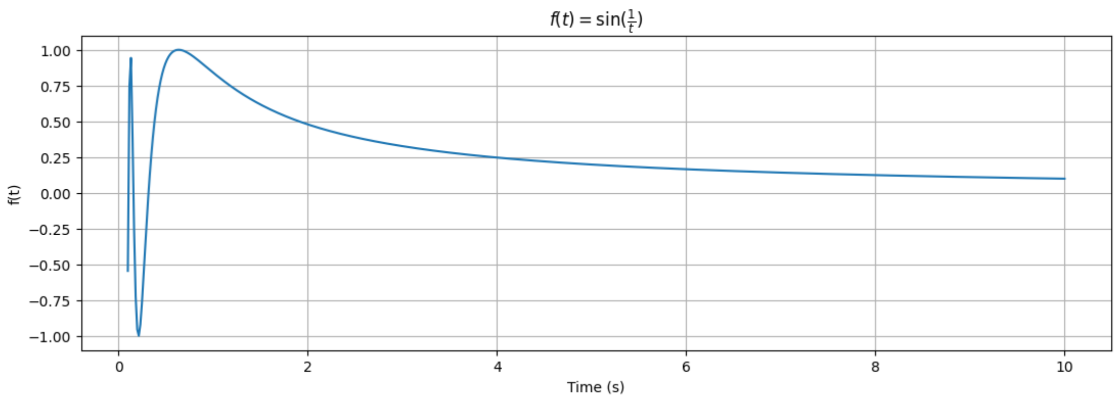
[10]:
```
signalF_functiont = np.sin(1 / t)

plt.figure(figsize = (13, 4))
plt.plot(t, signalF_functiont)
plt.title('$f(t) = \sin(\\frac{1}{t})$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```

```
/var/folders/vw/6c5wjngs433234dthdjypz800000gn/T/ipykernel_15190/1505464875.py:1
: RuntimeWarning: divide by zero encountered in divide
  signalF_functiont = np.sin(1 / t)
/var/folders/vw/6c5wjngs433234dthdjypz800000gn/T/ipykernel_15190/1505464875.py:1
: RuntimeWarning: invalid value encountered in sin
  signalF_functiont = np.sin(1 / t)
```



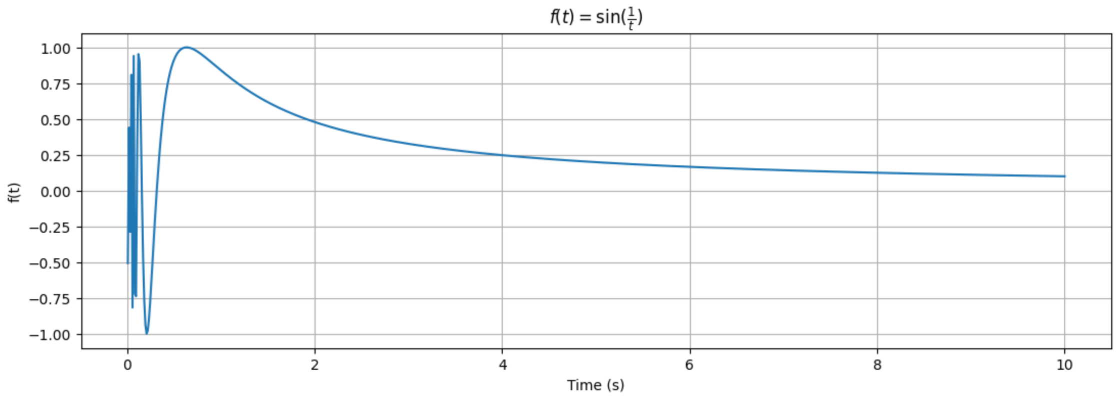
Notes: - adjusting t value to avoid dividing by 0 and increasing sample size for better representation

[11]:
```
t4 = np.linspace(0.1, 10, 600)
signalF_functiont2 = np.sin(1 / t4)

plt.figure(figsize = (13, 4))
plt.plot(t4, signalF_functiont2)
plt.title('$f(t) = \sin(\\frac{1}{t})$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```



Notes: - Some odd difference, thus continue increasing sample size

[12]:
```
t5 = np.linspace(0.01, 10, 800)
signalF_functiont3 = np.sin(1 / t5)

plt.figure(figsize = (13, 4))
plt.plot(t5, signalF_functiont3)
plt.title('$f(t) = \sin(\\frac{1}{t})$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```



Notes: - Signal Representation looks to be greatly improved, threshold for $\sin\left(\frac{1}{t}\right)$ is ≥ 800

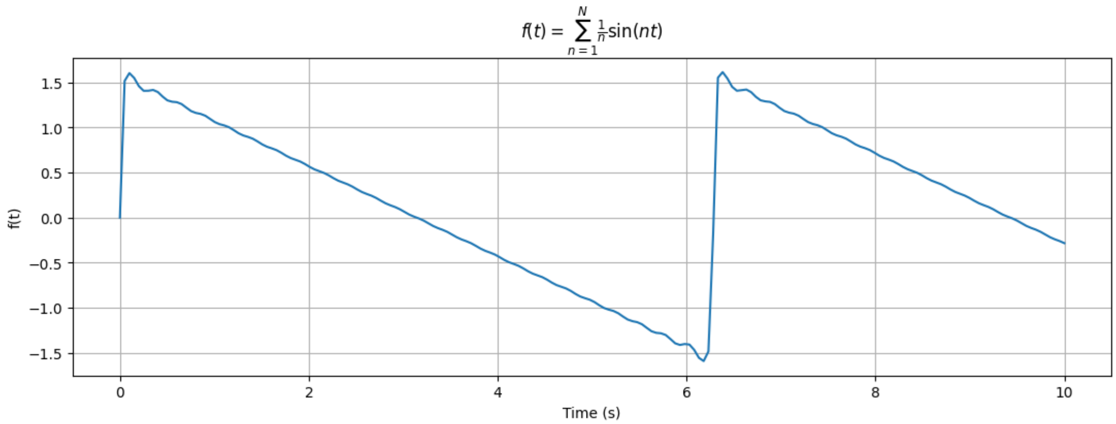
1.9 Answer:

$f(t) = \sin\left(\frac{1}{t}\right)$ is not periodic, thus no period

---

g. $f(t) = \sum_{n=1}^{N} \frac{1}{n} \sin(nt)$

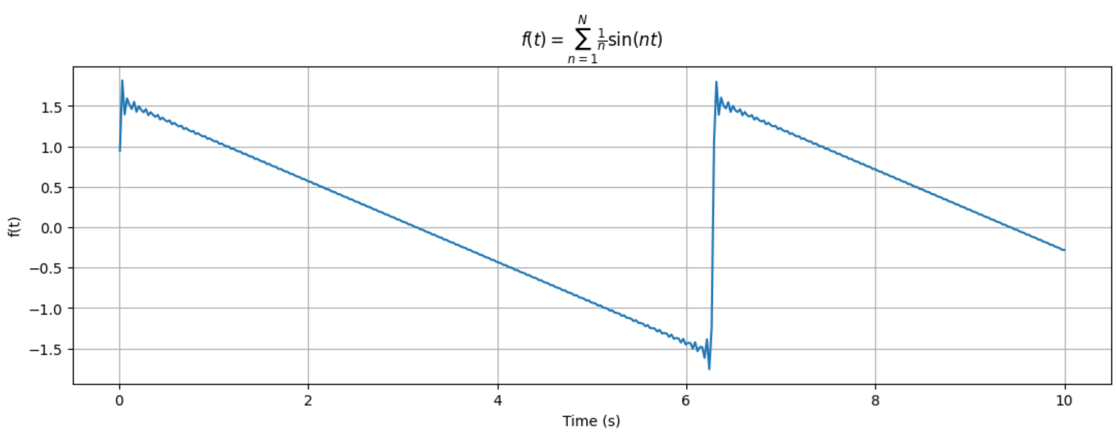
[13]:
```
N = 100
signalG_functiont = np.zeros_like(t)

for n in range(1, N + 1):
    signalG_functiont = signalG_functiont + (1 / n) * np.sin(n * t)

plt.figure(figsize = (13, 4))
plt.plot(t, signalG_functiont)
plt.title('$f(t) = \sum_{n = 1}^{N} \\frac{1}{n} \sin(nt)$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```



Notes: - Representation of signal seems distorted, due to sample size, thus sample size (t)/data points must increase for proper representation of the signal function.

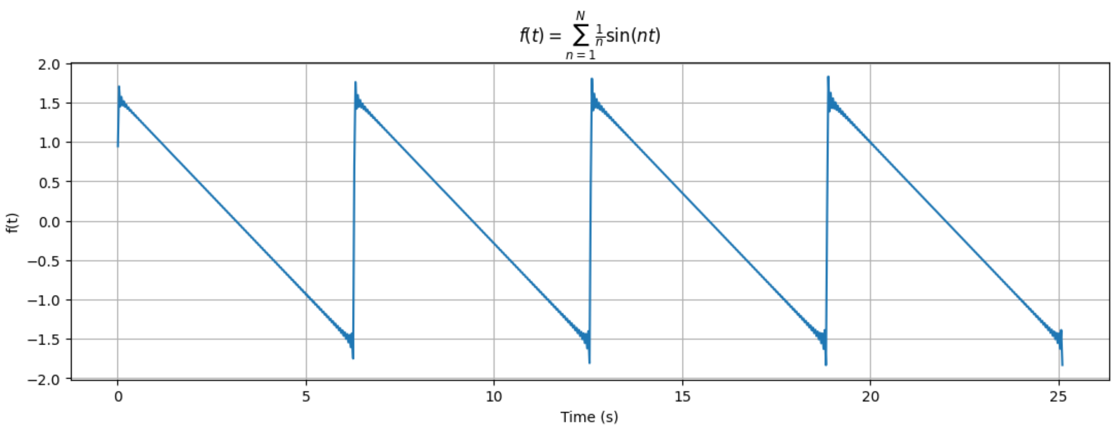
[14]:

```
t6 = np.linspace(0.01, 10, 400)
N = 100
signalG_functiont2 = np.zeros_like(t6)

for n in range(1, N + 1):
    signalG_functiont2 = signalG_functiont2 + (1 / n) * np.sin(n * t6)

plt.figure(figsize = (13, 4))
plt.plot(t6, signalG_functiont2)
plt.title('$f(t) = \sum_{n = 1}^{N} \\frac{1}{n} \sin(nt)$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```



Notes: - increasing interval and sampling to see if there is great change to better determine Periodicity:

[15]:
```
t7 = np.linspace(0.01, 25.1, 800)
N = 100
signalG_functiont3 = np.zeros_like(t7)

for n in range(1, N + 1):
    signalG_functiont3 = signalG_functiont3 + (1 / n) * np.sin(n * t7)

plt.figure(figsize = (13, 4))
plt.plot(t7, signalG_functiont3)
plt.title('$f(t) = \sum_{n = 1}^{N} \\frac{1}{n} \sin(nt)$')
plt.xlabel('Time (s)')
plt.ylabel('f(t)')
plt.grid(True)
plt.show()
```



Notes:

Using the example N = 100 we get:

$$f(t) = \sum_{n=1}^{100} \frac{1}{n} \sin(nt) \rightarrow sin(t) \rightarrow T(period) = 2\pi$$

$$sin(2\pi ft) \rightarrow f = \frac{1}{T} \rightarrow= sin(2\pi \cdot \frac{1}{T}t) \rightarrow= sin(2\pi \cdot \frac{1}{2\pi}t) = \sin(t)$$

the next is:

$$sin(2t) \rightarrow T(period) = \pi$$

$$sin(2t) = sin(2\pi \cdot \frac{1}{\pi}t)$$

…. all the way to 100 we get:

$$\rightarrow sin(100t) \rightarrow T = \frac{2\pi}{100}$$

1.10 Answer:

$f(t) = \sum_{n=1}^{N} \frac{1}{n} \sin(nt)$ appears to be periodic and the common multiple is 2π

References:

"Adding sinusoids with different frequencies results in a signal that is no longer sinusoidal. But is it periodic? (1)"

"If the frequencies of the added sinusoids are integer multiples of the fundamental, the resulting signal will be periodic. (2)"

"Adding sinusoids at 3, 6, 9 Hz produces a periodic signal at 3 Hz. (4)"

- http://musicweb.ucsd.edu/~trsmyth/addSynth171/addSynth171_4up.pdf

---

1.10.1 Question 3:

a. We can describe the contribution of one neuron to the LFP as: $f_1(t) = A\sin(2\pi ft)$. In this equation, what are A and f? Give the value and physical units. Notes: - $f_1(t) = A\sin(2\pi ft)$ is in sinusoidal form $\rightarrow A\sin(wt + \phi)$, thus $f_1(t) = A\sin(2\pi ft)$ is a sinuosoidal function. A

Sinusoidal function form: $A\sin(wt + \phi)$ and $A\cos(wt + \phi)$

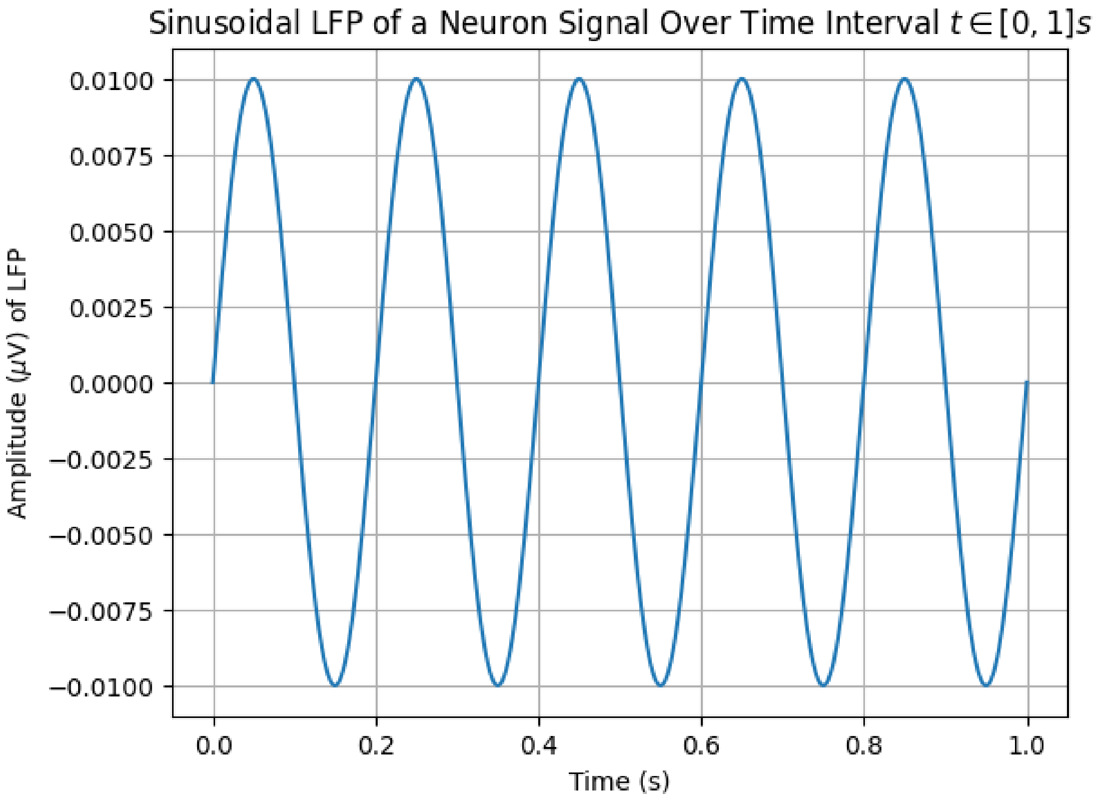
1.11 Answer:

- A: amplitude (of sinusoidal LFP signal) $= \sigma = 0.01\,\mu\text{V}\,;(given)$
- w: angular frequency
- ϕ: phase shift $= 0$
- f: frequency of sinusoidal signal: stimulus $= 5Hz\,;(stimulus flashed at 5Hz)$

b. Use Python or MATLAB to plot $f_1(t)$ as a function of time during the time interval $t \in [0, 1]s$, during which the stimulus is presented. You will need to load the libraries (numpy, matplotlib.pyplot). Make sure to label the axes (you will not get credit for plots with unlabeled axes!).

[16]:
```python
A = 0.01
f1_ofTime_t = 5

time_interval_t = np.linspace(0, 1, 800)
LFP_ofOne_neuron_f1t = A * np.sin(2 * np.pi * f1_ofTime_t * time_interval_t)

plt.plot(time_interval_t, LFP_ofOne_neuron_f1t)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude ($\mu$V) of LFP')
plt.title('Sinusoidal LFP of a Neuron Signal Over Time Interval $t\in[0, 1]s$')
```

```
plt.grid(True)
plt.show()
```



c. Calculate the RMS (root mean squared) amplitude, given by $\sqrt{\int_0^1 (f_1(t))^2\, dt}$. Hint: You can use the fact that $\sin^2(x) = \frac{1}{2}(1 - \cos(2x))$, and also $\int_0^{2\pi} \cos(x)\, dx = 0$ (since the cosine function has equal positive and negative parts). Using these two, you can show that $\int_0^1 sin^2(2\pi x)\, dx = \frac{1}{2}$

1.12 Answer:

$RMS = \sqrt{\int_b^a (f(t))^2\, dt} = \sqrt{\int_0^1 (f_1(t))^2\, dt} = \sqrt{\int_0^1 A^2 \sin^2(2\pi f t)\, dt} \rightarrow$
given: $\int_0^1 \sin^2(2\pi f t)\, dt = \frac{1}{2}$ and $A = \sqrt{1} = 1 \rightarrow A \cdot \sqrt{\frac{1}{2}} = A \cdot \frac{1}{\sqrt{2}} = \frac{A}{\sqrt{2}} \rightarrow A$:
amplitude (of sinusoidal LFP signal) = $\sigma = 0.01\, \mu\mathrm{V}$; $(given) \rightarrow \frac{0.01}{\sqrt{2}}\, \mu\mathrm{V}$

d. What is the total signal RMS as a function of r?

1.13 Answer:

$RMS = \frac{0.01}{\sqrt{2}}\, \mu\mathrm{V}$; function of r

$= r{\cdot}5000$ (active neurons: neurons in one orientation column by the fraction of neurons responding to stimu

$$\text{Total RMS} = \frac{A}{\sqrt{2}} \cdot \sqrt{5000 \cdot r} = \frac{0.01}{\sqrt{2}} \cdot \sqrt{5000 \cdot r} = \frac{0.01}{\sqrt{2}} \cdot \frac{\sqrt{5000 \cdot r}}{100} = \frac{\sqrt{5000 \cdot r}}{\sqrt{2} \cdot 100} = \frac{50\sqrt{r}}{100} = \frac{\sqrt{r}}{2}\ \mu\text{V}$$

e. Plot the signal-to-noise ratio (SNR), in decibels, as a function of the number of neurons which respond to the stimulus.

Notes:

SNR Formula: $\text{SNR} \quad (dB) = 20\log_{10}\left(\frac{\text{Total Signal RMS}}{\sigma\,(\text{standard deviation})}\right) = 20\log_{10}\left(\frac{\frac{A}{\sqrt{2}} \cdot \sqrt{5000 \cdot r}}{10}\right) = 20\log_{10}\left(\frac{\frac{0.01}{\sqrt{2}} \cdot \sqrt{5000 \cdot r}}{10}\right) = 20\left(\log\left(\frac{0.01}{\sqrt{2}} \cdot \sqrt{5000}\right) + \log(10)\right) = \log\left(\frac{r^{10}}{1048576}\right) - 20$

- r is the fraction of neurons that respond to the visual stimulus in an orientation column, which ranges from 0 to 1 (0 means no neurons responded and 1 means all neurons responded: higher r value means larger fraction of neurons responded and lower value means fewer neurons responded). Determing the proportion (r) value will require that the proportion of neurons that responded to the stimulus to be quantified using a technique like calcium imaging as suggested in the article

References:

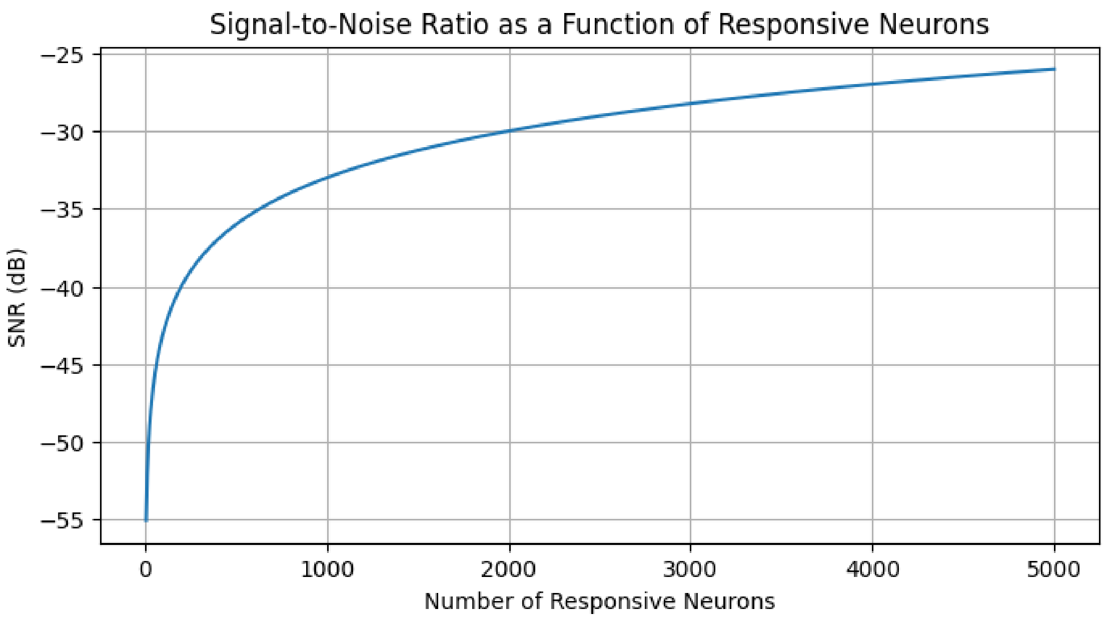
"However, when the signal and noise are measured in volts (V) or amperes (A), which are measures of amplitude,[note 1] they must first be squared to obtain a quantity proportional to power, as shown below: $\text{SNR} \quad (dB) = 10\log_{10}\left[\left(\frac{A_{\text{signal}}}{A_{\text{noise}}}\right)^2\right] = 20\log_{10}\left(\frac{A_{\text{signal}}}{A_{\text{noise}}}\right) = 2\left(A_{\text{signal, db}} - A_{\text{noise, db}}\right)$" - https://www.sciencedirect.com/topics/engineering/signal-to-noise-ratio#:~:text=SNR%20refers%20to%20the%20ratio,voltage%20and%20noise%20voltage%2C%20respectively. - https://en.wikipedia.org/wiki/Signal-to-noise_ratio

1.14 Answer:

[17]:

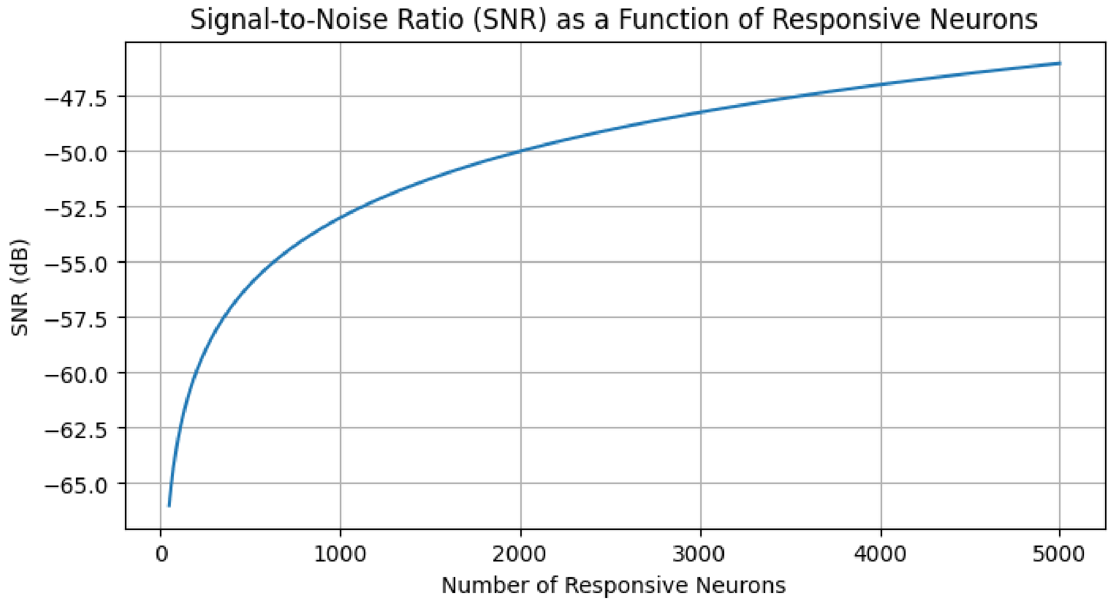
```
neurons = 5000
sd = 10
responsive_neurons_r = np.linspace(0, 1, 800)
total_signal_RMS = (A / np.sqrt(2)) * np.sqrt(neurons * responsive_neurons_r)
SNR = 20 * np.log10(total_signal_RMS / sd)

plt.figure(figsize = (8, 4))
plt.plot(neurons * responsive_neurons_r, SNR)
plt.xlabel('Number of Responsive Neurons')
plt.ylabel('SNR (dB)')
plt.title('Signal-to-Noise Ratio as a Function of Responsive Neurons')
plt.grid(True)
plt.show()
```

```
/var/folders/vw/6c5wjngs433234dthdjypz800000gn/T/ipykernel_15190/4104557351.py:5
: RuntimeWarning: divide by zero encountered in log10
  SNR = 20 * np.log10(total_signal_RMS / sd)
```



Notes: - Increasing t to prevent division from 0

[18]:
```
#neurons = 5000
sd = 10
A = 0.05
responsive_neurons_r = np.linspace(0.01, 1, 800)
total_signal_RMS = (A / np.sqrt(2)) * np.sqrt(2 * responsive_neurons_r)
SNR = 20 * np.log10(total_signal_RMS / sd)

plt.figure(figsize = (8, 4))
plt.plot(neurons * responsive_neurons_r, SNR)
plt.xlabel('Number of Responsive Neurons')
plt.ylabel('SNR (dB)')
plt.title('Signal-to-Noise Ratio (SNR) as a Function of Responsive Neurons')
plt.grid(True)
plt.show()
```



Notes:

Conclusion: closer SNR is to 0 the more the noise and the more difficult it is to distinguish signal from noise, the less the SNR is to 0 the more accurate the signal measure and the more the noise is filtered.

References:

"A high SNR means that the signal is clear and easy to detect or interpret, while a low SNR means that the signal is corrupted or obscured by noise and may be difficult to distinguish or recover."

"Since SNR is a signal divided by noise, we can directly improve it by either increasing the signal or reducing the noise. Also, the SNR can be improved by accumulating the total signal count. This is because the shot noise is proportional to the square root of the photon counts."

"Many midbrain neurons showed a decreasing response rate as TIN signal-to-noise ratio (SNR) was increased by elevating the tone level"

- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKI to-noise_ratio%23%3A~%3Atext%3DA%2520high%2520SNR%2520means%2520that%2Cdifficult%2520to%
- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKI improve-signal-to-noise-ratio-x-ray-ct-images-aya-takase%23%3A~%3Atext%3DSince%2520SNR%2520is%25
- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKI

f. Based on the different organization of V1 neurons in rat and cat observed by Ohki et al. (Nature 2005), in which species would you expect to observe a larger field potential SNR? For simplicity, you may assume that all other parameters (cell density, cortical thickness, etc.) are equivalent between these two species (though this is far from true!).

Notes:

Quote from Abstract of the article: - "In rat primary visual cortex, neurons had robust orientation selectivity but there was no discernible local structure; neighbouring neurons often responded to different orientations. In area 18 of cat visual cortex, functional maps were organized at a fine scale. Neurons with opposite preferences for stimulus direction were segregated with extraordinary spatial precision in three dimensions, with columnar borders one to two cells wide. These results indicate that cortical maps can be built with single-cell precision."

1.15 Answer:

Cats are expected to observe a larger field potential SNR because V1 neurons (visual cortex neurons: area 18) in cats are systematically and precisely organized by different directional preferences separated leading to a stronger summed signals, whereas V1 neurons in rats are selective in their response and lack an organized pattern.

4. You are considering which amplifier to buy for your electrophysiology recording. The deluxe model offers 10-fold lower noise amplitude (standard deviation) compared to the basic version. How much larger will the SNR be with the deluxe amplifier? Give your answer in dB.

Notes:

$SNR_{\text{db}} = 20\log_{10}\left(\frac{A_{\text{signal}}}{A_{\text{noise}}}\right) = 20\log_{10}\left(\frac{\text{Amplitude of Signal}}{\sigma\text{ of Noise}}\right) \rightarrow$ basic version ; deluxe version $= 20\log_{10}\left(\frac{\text{Amplitude of Signal}}{\frac{\sigma}{10}}\right)$

$SNR_{\text{db}} = 20\log_{10}\left(\frac{\text{Total Signal RMS}}{\sigma\text{ (standard deviation)}}\right) = 20\log_{10}\left(\frac{\frac{A}{\sqrt{2}}\cdot\sqrt{5000\cdot r}}{10}\right) = 20\log_{10}\left(\frac{\frac{0.01}{\sqrt{2}}\cdot\sqrt{5000\cdot r}}{10}\right) = 20\left(\log\left(\frac{0.01}{\sqrt{2}}\cdot\sqrt{5000}\right) + \log(10)\right) = \log\left(\frac{r^{10}}{1048576}\right) - 20$

1.16 Answer:

$20\log_{10}\left(\frac{\text{Amplitude of Signal}}{\frac{\sigma}{10}}\right) - 20\log_{10}\left(\frac{\text{Amplitude of Signal}}{\sigma\text{ of Noise}}\right) = \log\left(\frac{100000000000000000000A^{20}x^{20}}{A^{20}x^{20}}\right) = \log(100000000000000000000) = 20\,\text{dB}$

Verifying answer: $20\log_{10}\left(\frac{\frac{0.01}{\sqrt{2}}\cdot\sqrt{5000\cdot r}}{\frac{10}{10}}\right) - 20\log_{10}\left(\frac{\frac{0.01}{\sqrt{2}}\cdot\sqrt{5000\cdot r}}{10}\right) = \log\left(\frac{r^{10}}{1048576}\right) - \log\left(\frac{r^{10}}{1048576}\right) + 20 = \log\left(\frac{\frac{x^{10}}{1048576}}{\frac{x^{10}}{1048576}}\right) + 20 \rightarrow \log\left(\frac{x}{y}\right) = \log(x) - \log(y) \rightarrow = \log\left(\frac{x^{10}}{1048576}\right) - \log\left(\frac{x^{10}}{1048576}\right) + 20 = 0 + 20 = 20\,\text{dB}$

---

5. Aliasing (coding exercise – you may use Python, Matlab, or other scientific software).

[19]:
```python
import scipy.io.wavfile as wavfile

# Load audio file. This code will read the wav file
Fs, aud = wavfile.read('HW1_audio.wav')

# The audio file has two channels (stereo - Left and Right).
# For simplicity, we will keep just one channel.
```

```
aud = aud[:,0]

print(aud)
print(aud.shape)
```

```
[   -1    -3    -6 … -1376 -1069  -277]
(638632,)
```

a. What is the sampling rate (Fs) and what is the duration of the audio recording?

1.17 Answer:

Duration $= \frac{\text{Total Samples}}{F_s} = \frac{\text{Length of Audio}}{F_s} = \frac{638632}{48000} = 13.30483\overline{3}$ - frequency (sampling rate) is given as $Fs = 48,000\,Hz$ - total sample is the length of $aud = 13.30\,seconds$

[20]:
```
total_sample = len(aud)
duration = total_sample / Fs

print("Sampling Rate (Fs): ", Fs)
print("Duration (Hz): ", duration)
```

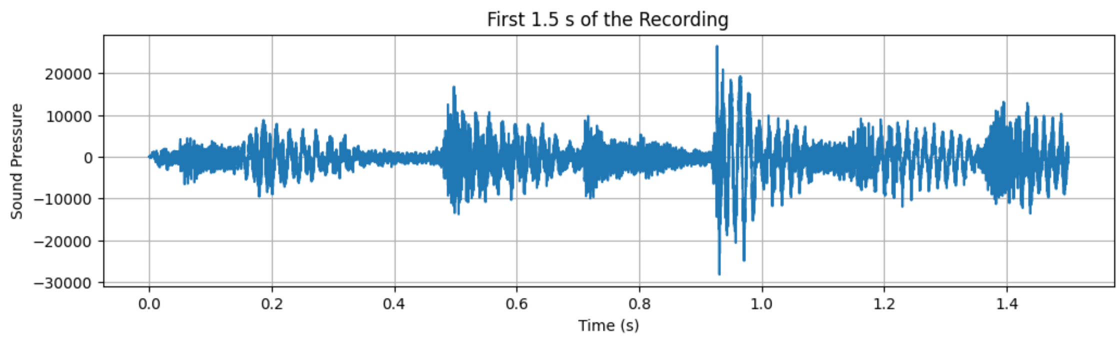
```
Sampling Rate (Fs):  48000
Duration (Hz):  13.304833333333333
```

b. Make a plot showing the first 1.5 s of the recording. Label the axes (x axis: Time (s), y-axis: Sound pressure).

Notes: - $interval = [0, 1.5]\,s$ - total samples = frequency $(F_s) \cdot \text{time} = 48000 \cdot 1.5 = 72000$

1.18 Answer:

[21]:
```
total_samples = 72000
time = np.linspace(0, 1.5, total_samples)
sound_pressure = aud[:total_samples]

plt.figure(figsize = (12, 3))
plt.plot(time, sound_pressure)
plt.xlabel('Time (s)')
plt.ylabel('Sound Pressure')
plt.title('First 1.5 s of the Recording')
plt.grid(True)
plt.show()
```



Notes: - Values in an digital audio file are the amplitude of the sound pressure wave at each sample point, thus higher amplitudes make a stronger pressure (louder sound) and lower amplitudes make a weaker pressure (lower sound)

References:

"The size of these pressure variations is called pressure amplitude. In general, the larger the pressure amplitude a sound has, the louder it seems."

- https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKI 9vug_CDAxXWJUQIHSmJDhAQFnoECBAQAQ&url=https%3A%2F%2Fpressbooks.pub%2Fsound%2Fch amplitude%2F&usg=AOvVaw2013T9ctQmOiaFdtgmb03v&opi=89978449

c. Downsample the recording by a factor of 15 (see Hints in the python notebook). What is the new sampling rate, Fs_dsamp, of the downsampled recording?

(c) Downsample the recording by a factor of 15, i.e. select every 15th sample of the recorded signal and throw away the rest. Hint: you can do this using numpy array "slicing" (https://docs.scipy.org/doc/numpy/reference/arrays.indexing.html):

The basic slice syntax is i:j:k where i is the starting index, j is the stopping index, and k is the step (k 0). This selects the m elements (in the corresponding dimension) with index values i, i + k, ..., i + (m - 1) k where m = q + (r 0) and q and r are the quotient and remainder obtained by dividing j - i by k: j - i = q k + r, so that i + (m - 1) k < j.

For example, the syntax x[0:10:2] would select every 2nd element of x starting with the first (0) and ending with the 10th.

Another example: x[::2] would give every 2nd element of x, starting with 0 and ending with the last element of x. In this case, the "missing" values of i and j are automatically interpreted as the first and last elements

What is the new sampling rate, Fs_dsamp?

[22]:
```
downsample_factor = 15

aud_dsamp = aud[::downsample_factor]
Fs_dsamp = Fs / downsample_factor
```

```
Fs_dsamp
```

[22]: 3200.0

1.19 Answer:

new sampling rate = Fs_damp = 3200 Hz

d. Plot the first 1.5 s of the downsampled recording. Does it appear to be very different from the original?

- $interval = [0, 1.5]\, s$
- total samples = new frequency $(F_{s(dsamp)}) \cdot$ time $= 3200 \cdot 1.5 = 4800$

1.20 Answer:

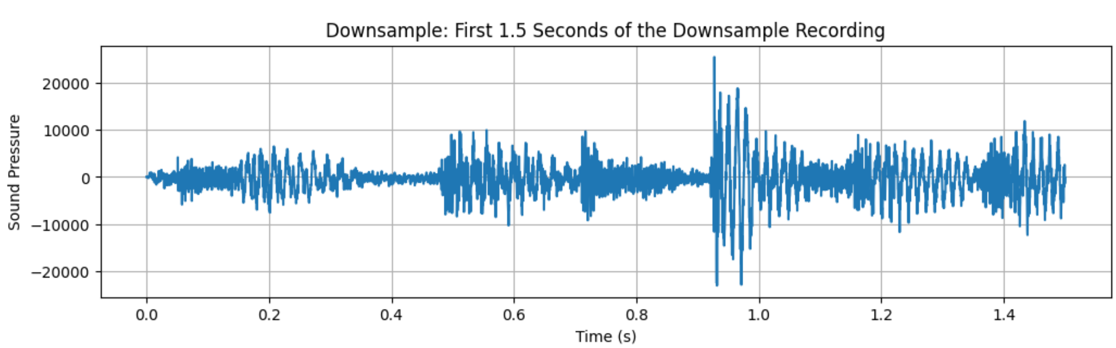
The downsampled recording is not obviously different but one can observe some changes. There appears to be some slight compression (or filtering). Downsampling captures the signal sampling at a lower rate. Hence, it does not appear to be very different from the original but it is.

Reference:

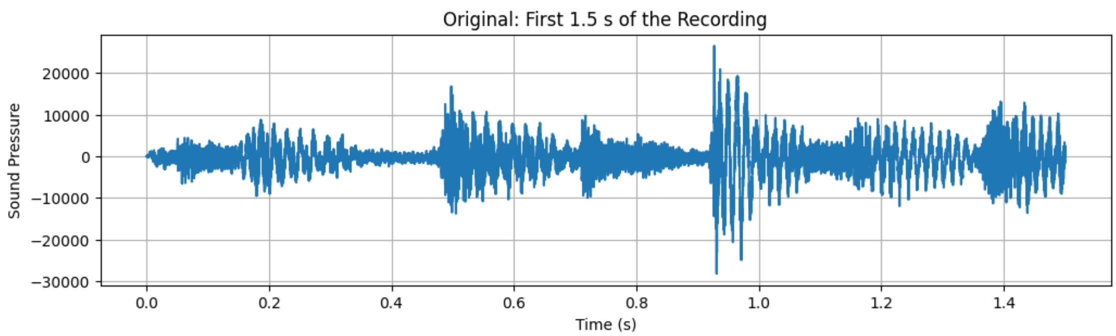
"Both downsampling and decimation can be synonymous with compression, or they can describe an entire process of bandwidth reduction (filtering) and sample-rate reduction … it produces an approximation of the sequence that would have been obtained by sampling the signal at a lower rate (or density, as in the case of a photograph)."

- https://en.wikipedia.org/wiki/Downsampling_(signal_processing)

[23]:
```
downsampled_total = 4800
time_dsamp = np.linspace(0, 1.5, downsampled_total)

plt.figure(figsize = (12, 3))
plt.plot(time_dsamp, aud_dsamp[:downsampled_total])
plt.xlabel('Time (s)')
plt.ylabel('Sound Pressure')
plt.title('Downsample: First 1.5 Seconds of the Downsample Recording')
plt.grid(True)
plt.show()
```



[24]:
```
total_samples = 72000
time = np.linspace(0, 1.5, total_samples)
sound_pressure = aud[:total_samples]

plt.figure(figsize = (12, 3))
plt.plot(time, sound_pressure)
plt.xlabel('Time (s)')
plt.ylabel('Sound Pressure')
plt.title('Original: First 1.5 s of the Recording')
plt.grid(True)
plt.show()
```



e. Now save the downsampled signal to an audio file. Download the file and listen to it. Describe how it sounds. What differences do you notice compared to the original audio file?

[25]:
```
# (e) Save the audio file
wavfile.write('HW1_dsamp.wav', int(Fs_dsamp), aud_dsamp)
```

1.21 Answer:

The voice of the audio file now sounds lower and the audio file in general now has more static (voice volume lowered and static noise increased). This means that the downsample decreased the number of samples, which alters the pitch (loss of high frequency) and is the result of improper anti-aliasing filtering.

f. Given the new sampling frequency (Fs_dsamp), would you expect that audio frequencies around 10,000 Hz (10 kHz) are distorted by the downsampling? Why or why not?

Notes:

Nyquist Theorem: $F_s \leq 2 \cdot F_{max} \rightarrow \frac{F_{s(dsamp)}}{2}$ - F_s: Sampling Rate - F_{max}: Highest Frequency in signal

$F_{s(dsamp)} = 3200$ - $\frac{F_{s(dsamp)}}{2} = \frac{3200}{2} = 1600\,\text{Hz} \rightarrow F_{max}$ - Given $10,000\,\text{Hz} \rightarrow F_s$

1.22 Answer:

$F_s > F_{max}$ (Nyquist) $\rightarrow 10,000\,\text{Hz} \ > \ 1600\,\text{Hz}$

Given sample is higher than the maximum frequency that can be accurately represented by downsampled sampling rate, thus the given 10,000 Hz frequency (frequencies) will be distorted by downsampling and subject to aliasing (higher frequencies misrepresented as lower frequencies)