

Analisis Tugas Hands-on 1

1. Membuat Gelombang Sinus dan Cosinus:

- Buat array `t` yang berkisar dari 0 hingga 2 dengan langkah sebesar 0.0001.
- Buat sinyal-sinyal berikut menggunakan array waktu ini:
 - $y_1 = 2 \cdot \sin(2\pi \cdot 3 \cdot t + 0)$
 - $y_2 = 1 \cdot \cos(2\pi \cdot 4 \cdot t + \pi/4)$
 - $y_3 = -1 \cdot \sin(2\pi \cdot 5 \cdot t + \pi/2)$
 - $y_4 = 0.5 \cdot \cos(2\pi \cdot 6 \cdot t + \pi)$

Kita dapat menggunakan `np.arange()` untuk membuat time interval dari 0 sampai detik ke-2.

```
In [17]: import numpy as np
import matplotlib.pyplot as plt

## Setup time interval
t = np.arange(0, 2, 0.0001)

## Create the first four fundamental wave
y1 = 2 * np.sin(2 * np.pi * 3 * t + 0)
y2 = 1 * np.cos(2 * np.pi * 4 * t + np.pi / 4)
y3 = -1 * np.sin(2 * np.pi * 5 * t + np.pi / 2)
y4 = 0.5 * np.cos(2 * np.pi * 6 * t + np.pi)
```

```
In [18]: ## Analisis Graph dengan Matplotlib
fig, axs = plt.subplots(2, 2, figsize=(16, 9))

# Plot y1
axs[0, 0].plot(t, y1)
axs[0, 0].set_title('y1: 2 * sin(2π * 3 * t)')

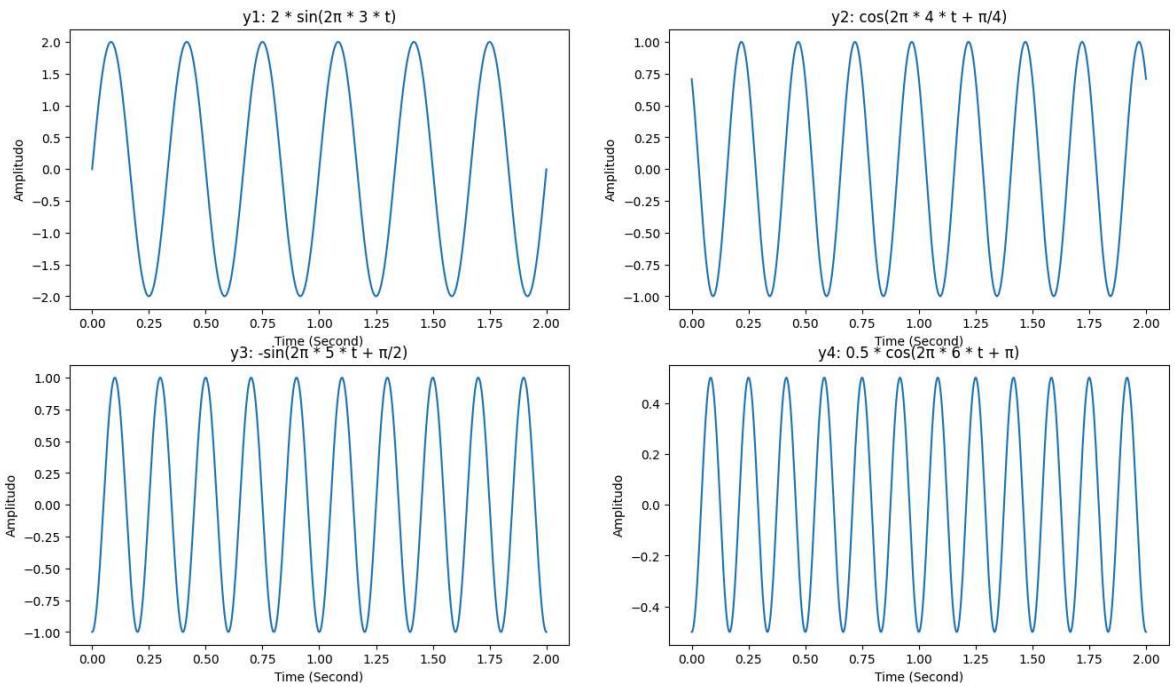
# Plot y2
axs[0, 1].plot(t, y2)
axs[0, 1].set_title('y2: cos(2π * 4 * t + π/4)')

# Plot y3
axs[1, 0].plot(t, y3)
axs[1, 0].set_title('y3: -sin(2π * 5 * t + π/2)')

# Plot y4
axs[1, 1].plot(t, y4)
axs[1, 1].set_title('y4: 0.5 * cos(2π * 6 * t + π)')

# Giving Label
for axis in axs.flat:
    axis.set(xlabel="Time (Second)", ylabel="Amplitudo")

plt.show()
```



Analisis Gelombang

3. Pertanyaan Analisis

- Untuk masing-masing Amplitudo dan Frekuensi, dapat ditemukan dalam persamaan gelombang berikut yakni:

$$\text{signal} = A \cdot \sin(2\pi ft + \phi)$$

atau

$$\text{signal} = A \cdot \cos(2\pi ft + \phi)$$

Maka untuk hasil tersebut didapatkan

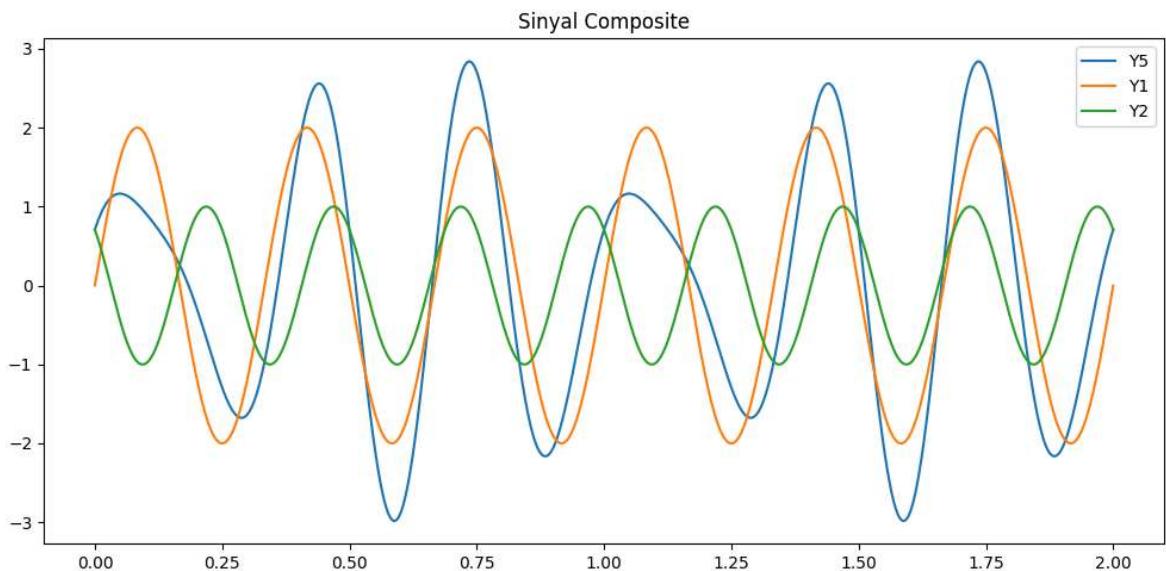
- $y1 : A = 2, f = 3$
- $y2 : A = 1, f = 4$
- $y3 : A = -1, f = 5$
- $y4 : A = 0.5, f = 6$
- Pengaruh dari *phase shift* itu sendiri membuat bagaimana gelombang tersebut dimulai. Misalnya untuk $y3$, didapatkan bahwa karena fungsi tersebut negatif (berarti terbalik, dan kita geser fungsi dari yang seharusnya dimulai dari titik 0 sebesar 90° derajat agar berada pada titik -1).
- Pengaruh amplitudo dan frekuensi disini sangat cukup penting dimana Amplitudo akan memberikan gelombang yang lebih tinggi, sebaliknya frekuensi berpengaruh dalam rapat-longgarnya gelombang-gelombang dalam kurun waktu 1 detik
- Pengaruh fase sangat penting karena mengacu bagaimana fungsi gelombang tersebut dimulai, namun tidak mengubah bentuk dari gelombang secara keseluruhan

Tugas Tambahan

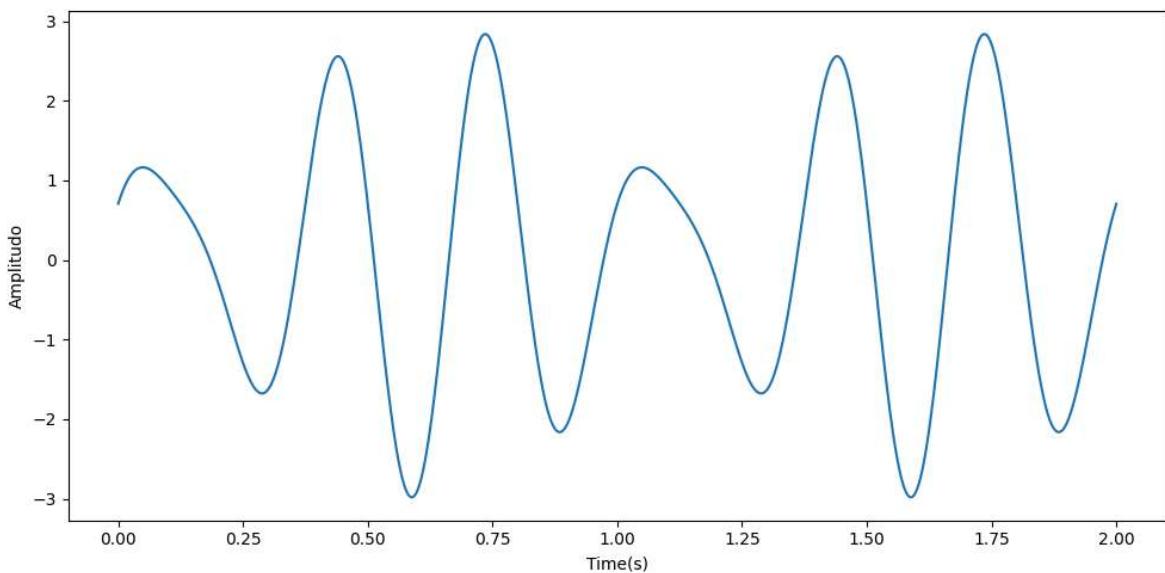
4. Penambahan sinyal kombinasi dari $y_1 + y_2$ dan analisis sederhananya.

```
In [19]: # Proses kombinasi gelombang
y5 = y1 + y2
t = np.arange(0, 2, 0.0001)

plt.figure(figsize=(10, 5))
plt.plot(t, y5)
plt.plot(t, y1)
plt.plot(t, y2)
plt.title("Sinyal Composite")
plt.legend(["Y5", "Y1", "Y2"])
plt.tight_layout()
plt.show()
```



```
In [20]: plt.figure(figsize=(10, 5))
plt.plot(t, y5)
plt.xlabel("Time(s)")
plt.ylabel("Amplitudo")
plt.tight_layout()
plt.show()
```



Analisis Sederhana

Berdasarkan kombinasi dari Sine dan Cosine wave satu ini, dapat dilihat terbentuk sebuah gelombang baru dari kedua penjumlahan gelombang tersebut. Disini juga dapat dilihat properti dari gelombang, dimana pada penjumlahan di tiap titik, kedua gelombang bisa saling memperkuat (inference konstruktif) dimana amplitudo gelombang y5 semakin besar dan saling melemahkan (inference desktruktif) dan bisa kita lihat pada bagian awal gelombang ini yang menurun.

Analisis Downsampling

5. Diberikan sebuah sinyal ECG sintensis, dan dilakukan sebuah analisis proses downsampling untuk membuktikan bahwa undersampling membuat sebuah distorsi:

Diberikan sebuah spesifikasi gelombang berikut - Durasi: Berdasarkan 3 digit terakhir nim anda - Sampling Rate: 150 Hz - Noise Level: 0.<2 digit nim terakhir> - Heart Rate: 80 BPM - Random State: tanggal bulan tahun lahir anda dengan format YYMMDD misalnya 240925

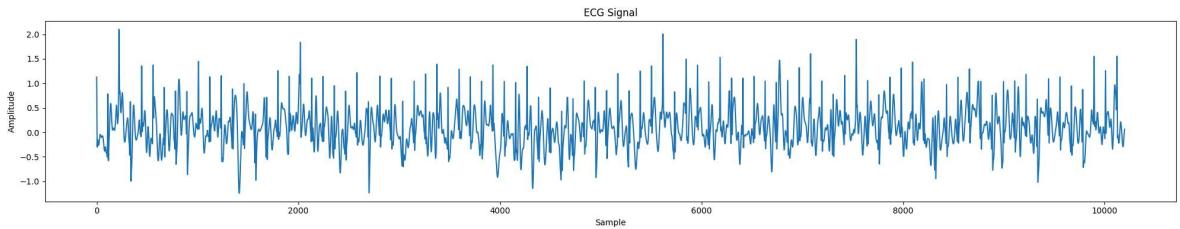
```
In [21]: import neurokit2 as nk

time_interval = 68 # 068 as the time interval
sampling_rate_150 = 150 # Sampling rate
noise_level = 0.68 # Noise level
bpm = 80 # Heart rate
random_state = 171003 # Id that got used to be an identifier as the random state

## Original wave
original_wave = nk.ecg_simulate(duration = time_interval, sampling_rate=sampling_rate_150)

plt.figure(figsize=(20, 4))
plt.plot(original_wave)
plt.title("ECG Signal")
plt.ylabel("Amplitude")
plt.xlabel("Sample")
```

```
plt.tight_layout()
plt.show()
```



Pada tugas kali ini akan dilakukan proses downsampling dari 150Hz ke 100Hz, 50Hz, 25Hz, 10Hz, hingga 5Hz.

Seperti yang kalian ketahui terdapat beberapa relasi antara sampling_rate (frekuensi), jumlah sampel data dan waktu interval yang ditentukan dalam formula berikut

$$\text{time length} = \frac{\text{length of sound data (samples)}}{\text{sampling rates}}$$

```
In [22]: ## 100 Hz as the new sampling_rates
sampling_rate_100 = 100
modified_length_100 = len(original_wave) * (sampling_rate_100 / sampling_rate_150)
resampled_signal_100 = nk.signal_resample(original_wave, desired_length=int(modified_length_100))

## 50 Hz as the second sampling_rates
sampling_rate_50 = 50
modified_length_50 = len(original_wave) * (sampling_rate_50 / sampling_rate_150)
resampled_signal_50 = nk.signal_resample(original_wave, desired_length=int(modified_length_50))

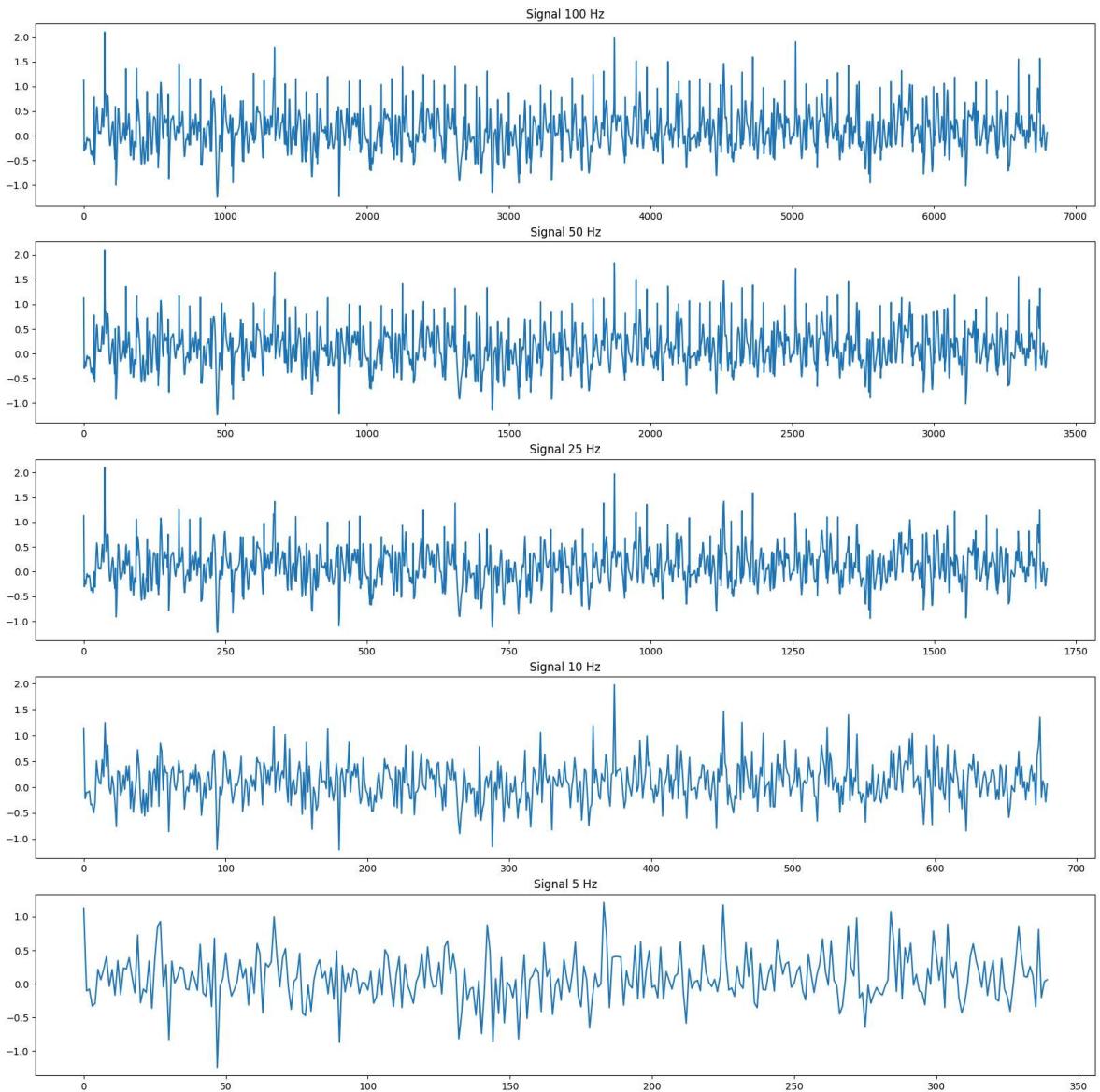
## 25 Hz as the third sampling_rates
sampling_rate_25 = 25
modified_length_25 = len(original_wave) * (sampling_rate_25 / sampling_rate_150)
resampled_signal_25 = nk.signal_resample(original_wave, desired_length=int(modified_length_25))

## 10 Hz as the fourth sampling_rates
sampling_rate_10 = 10
modified_length_10 = len(original_wave) * (sampling_rate_10 / sampling_rate_150)
resampled_signal_10 = nk.signal_resample(original_wave, desired_length=int(modified_length_10))

## 5 Hz as the fifth sampling_rates (extreme one)
sampling_rate_5 = 5
modified_length_5 = len(original_wave) * (sampling_rate_5 / sampling_rate_150)
resampled_signal_5 = nk.signal_resample(original_wave, desired_length=int(modified_length_5))

ax, fig = plt.subplots(5 ,1, figsize=(20, 20))
fig[0].plot(resampled_signal_100)
fig[0].set_title("Signal 100 Hz")
fig[1].plot(resampled_signal_50)
fig[1].set_title("Signal 50 Hz")
fig[2].plot(resampled_signal_25)
fig[2].set_title("Signal 25 Hz")
fig[3].plot(resampled_signal_10)
fig[3].set_title("Signal 10 Hz")
fig[4].plot(resampled_signal_5)
fig[4].set_title("Signal 5 Hz")
```

```
plt.show()
```



Analisis Gelombang

Berdasarkan hasil dari lima gelombang yang telah melalui proses downsampling, terjadi aliasing yang cukup ekstrem pada sinyal dengan sampling rate 5 Hz dan 10 Hz. Ini terlihat dari sinyal yang terlihat putus-putus. Namun, karena adanya noise yang cukup besar pada sinyal asli (~68%), sulit untuk mengamati aliasing pada sinyal dengan frekuensi sampling 100 Hz, 50 Hz, dan 25 Hz.

- Aliasing sendiri merupakan sebuah konsep dimana tidak akuratnya data (distorsi), sehingga menyebabkan kesalahan dalam rekonstruksi (pembuatan) sinyal tersebut. Aliasing sendiri biasanya terjadi karena sampel yang diambil memiliki frekuensi yang rendah (undersampling), sehingga menyebabkan sinyal yang tidak cukup mulus (patah-patah) ketika di visualisasikan [1].
- Teorema Nyquist-Shannon Sampling menyatakan bahwa untuk merekonstruksi sinyal tanpa aliasing, sampling rate harus minimal dua kali lipat dari frekuensi maksimum sinyal tersebut agar data dapat diambil secara akurat.

Satu cara untuk visualiasi Nyquist Theorem adalah sebagai berikut:

- Misalnya, batas pendengaran manusia berada dalam rentang 20 Hz hingga 20.000 Hz (20 kHz). Untuk merekam frekuensi ini tanpa aliasing, dibutuhkan minimal sampling rate 40 kHz. Namun, dalam praktiknya, biasanya digunakan 44.1 kHz.

$$\text{sampling rate} \geq \text{maximum frequencies} \cdot 2$$

- Selain itu, terdapat konsep frekuensi Nyquist, di mana frekuensi di bawah nilai ini akan ditangkap dengan akurat tanpa aliasing. Sebaliknya, frekuensi di atasnya akan mengalami aliasing.

$$\text{Nyquist frequencies} = \frac{\text{sampling rate}}{2}$$

Untuk ECG sendiri yang mengukur aktivitas otot jantung, didapatkan bahwa nilai minimal dari frekuensi maksimum otot jantung adalah 50 hz sehingga nilai minimum sampling rate yang dibutuhkan agar tidak aliasing adalah 100 Hz [2].

Berdasarkan penjelasan di atas, alasan terjadinya aliasing yang parah pada sinyal dengan sampling rate 5 Hz dan 10 Hz adalah karena sinyal ECG asli memiliki komponen frekuensi antara 5 hingga 10 Hz. Saat dilakukan downsampling dengan sampling rate 5 Hz dan 10 Hz, frekuensi Nyquist yang berada pada 2.5 Hz dan 5 Hz tidak mampu menangkap sinyal frekuensi yang lebih tinggi. Akibatnya, sinyal di atas frekuensi Nyquist terlipat kembali (folded) dan menyebabkan aliasing yang terlihat pada visualisasi.

Notes: Sebagai catatan, untuk melakukan visualisasi aliasing, menurut saya lebih baik agar durasi dari proses sinyal di kurangi menjadi 10-20 detik agar terlihat jelas efek dari undersampling tersebut. (Silahkan revisi jika ada kesalahan pada analisis saya).

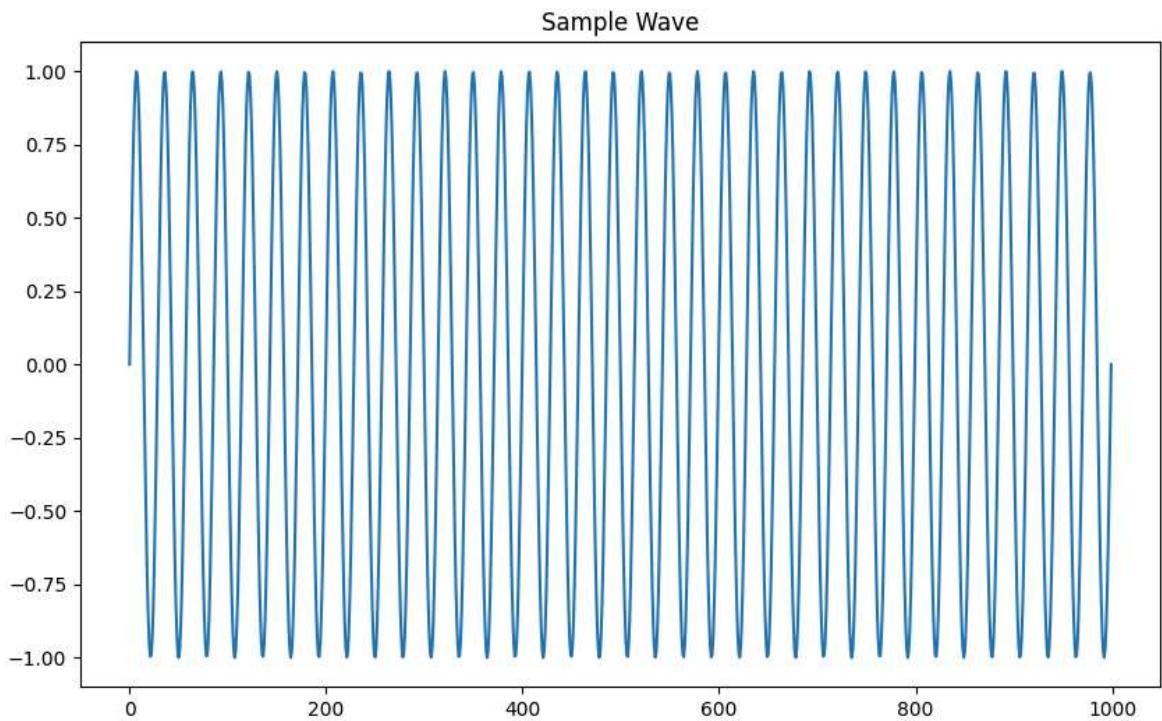
Filtering

Mari kita buat sebuah kasus filtering sederhana dimana kita akan menggunakan 15 sebagai frekuensi cutoff dengan tipe filter low-pass filter

```
In [23]: fs = 1000
time_axis = np.linspace(0, 1, fs)

sample_signal = np.sin(2 * np.pi * 35 * time_axis)

plt.figure(figsize=(10,6))
plt.plot(sample_signal)
plt.title("Sample Wave")
plt.show()
```

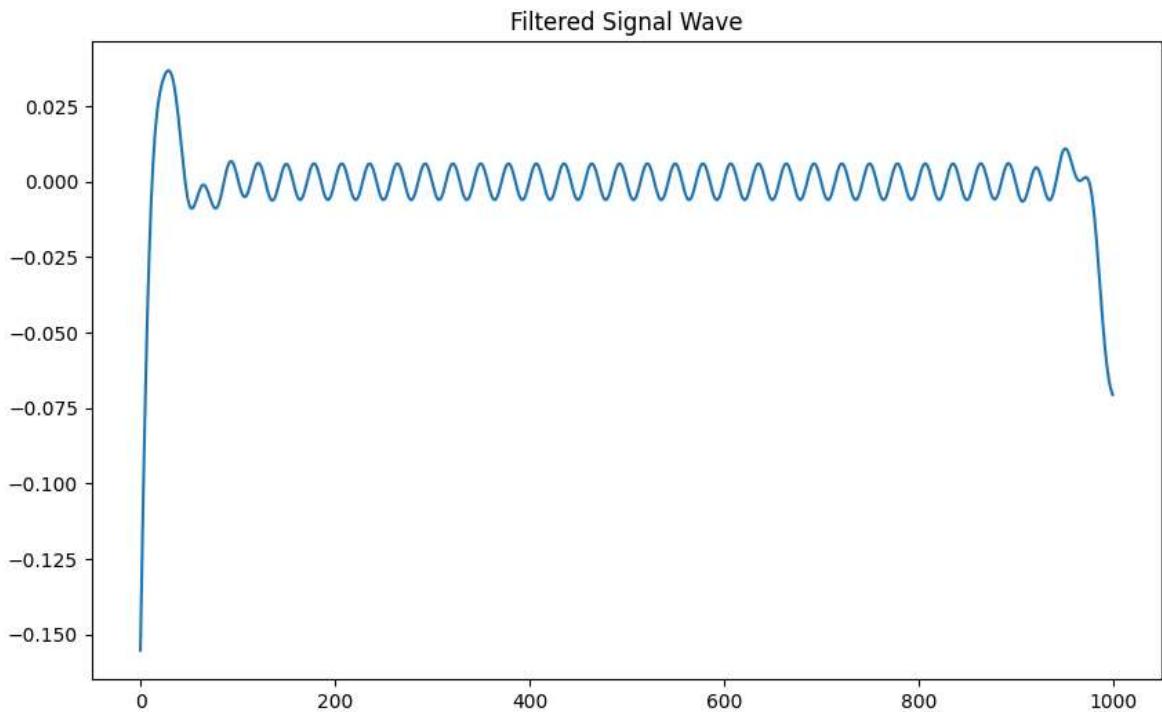


```
In [24]: from scipy import signal
## Creating Filter
cutoff_frequencies = 15 # Frekuensi cutoff dalam Hz
order_attenuation = 3 # Orde filter

b, a = signal.butter(order_attenuation, cutoff_frequencies, fs=fs, btype='low',
# atau cara lain menuliskannya
# cutoff_nyquist = cutoff / (fs / 2) ## Multimedia one
# b, a = signal.butter(order, cutoff_nyquist, btype='Low')

## Applied Filter
filtered_signal = signal.filtfilt(b, a, sample_signal)

plt.figure(figsize=(10,6))
plt.plot(filtered_signal)
plt.title("Filtered Signal Wave")
plt.show()
```



Notes

Bisa dilihat dari kedua grafik tersebut, terdapat perbedaan dari amplitudo dari sinyal original dan sinyal yang telah dilakukan filter. Hal ini disebabkan karena adanya **order** yang akan membuat setiap 2 kali frekuensi (atau disebut dengan oktave / oktaf) yang membuat amplitudo yang semakin menurun seiring meningkatnya frekuensi (semakin pelan suara seiring meningkatnya frekuensi pada sampel yang sudah dilakukan filter). Berikut tabel relasi antara frekuensi dengan laju penurunan amplitudo

Filter Order	Attenuation Rate	Description
1st-order	6 dB per octave (20 dB/decade)	Gentle roll-off, basic filtering.
2nd-order	12 dB per octave (40 dB/decade)	Sharper roll-off, used in EQs and audio enhancement.
3rd-order	18 dB per octave (60 dB/decade)	More aggressive filtering, tighter frequency control.
4th-order	24 dB per octave (80 dB/decade)	Steep roll-off, often used in advanced noise suppression.
Nth-order	6n dB per octave	Steeper roll-off with each increase in order (6 dB × order).

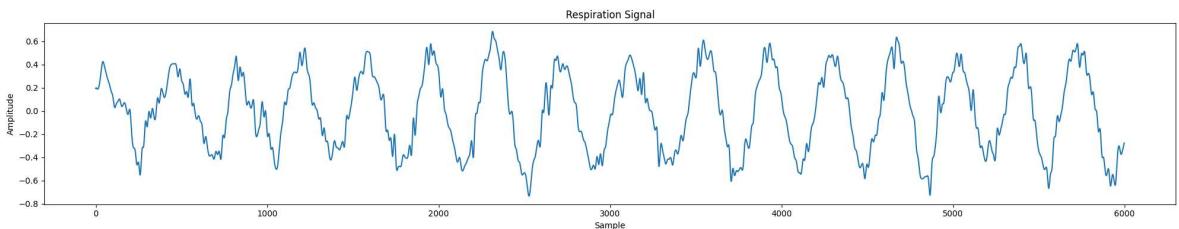
Band Pass on Respiration Signal

Untuk hands-on kali ini, akan dilakukan proses simulasi sinyal pernafasan dan akan dilakukan proses band pass filtering. Untuk kasus ini akan ditentukan untuk batas frekuensi cutoff untuk bagian bawah dan atas **0.1** dan **0.5** hz karena rata-rata orang

bernafas itu sebanyak 12 - 20 kali per menit, jadi kita bagi dengan 60, kita mendapat 0.1 dan 0.2 sebagai hz.

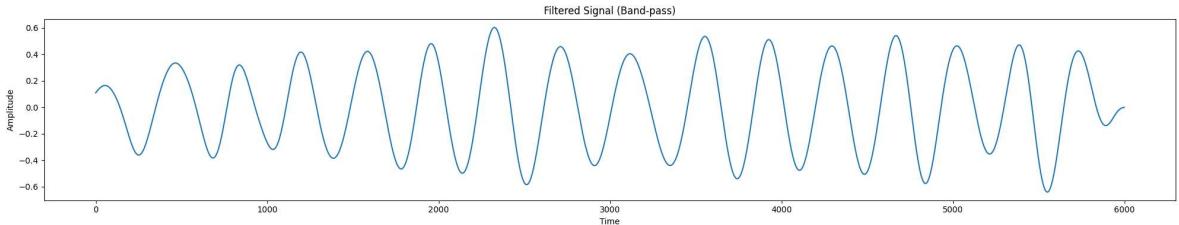
```
In [25]: # Simulate a respiration signal
sampling_rate = 100 # Common for respiration
respiratory_rate = 15 # In breaths per minute
respiration_signal = nk.rsp_simulate(duration=60, sampling_rate=sampling_rate, r

plt.figure(figsize=(20, 4))
plt.plot(respiration_signal)
plt.title("Respiration Signal")
plt.ylabel("Amplitude")
plt.xlabel("Sample")
plt.tight_layout()
plt.show()
```



```
In [26]: ## Filtering
cutoff_low = 0.1
cutoff_high = 0.5
order = 3
b, a = signal.butter(order, [cutoff_low, cutoff_high], fs=sampling_rate, btype='
filtered_resp_signal = signal.filtfilt(b, a, respiration_signal)

plt.figure(figsize=(20, 4))
plt.plot(filtered_resp_signal)
plt.title("Filtered Signal (Band-pass)")
plt.ylabel("Amplitude")
plt.xlabel("Time")
plt.tight_layout()
plt.show()
```



Hasil Percakapan AI-LLM :

- 1. Downsampling & Nyquist Theorem
- 2. Order and Attenuation Rate
- 3. Band-pass filter with Resp Signal