PPG SIMULATION

Apa itu PPG?

Photoplethysmography (PPG) is a non-invasive optical technique used to detect blood volume changes in peripheral vascular beds [1]. It has evolved significantly over seven decades, becoming indispensable in surgical procedures and patient monitoring [2]. The PPG waveform consists of a pulsatile "AC" component attributed to cardiac-synchronous blood volume changes and a slowly varying "DC" baseline influenced by respiration and sympathetic nervous system activity [3]. PPG technology has diverse applications, including measuring oxygen saturation, blood pressure, cardiac output, and detecting peripheral vascular disease [3]. It offers a promising solution for remote health examination and addressing hospital overload [4]. However, challenges remain, such as limited data availability for certain populations, sensitivity to motion and ambient conditions, and a lack of standardized criteria for sampling and interpretation [4]. Despite these issues, PPG continues to be a valuable tool in clinical physiological measurements.

Simulasi Sinyal PPG

Parameter Simulasi

- Durasi: 055 detik (mengacu pada 3 digit terakhir NIM)
- Sampling Rate: 150 Hz
- Noise Level: 0.55 (mengacu pada 2 digit terakhir NIM)
- Heart Rate: 80 BPM
- Random State: 040514 (dari tanggal lahir)

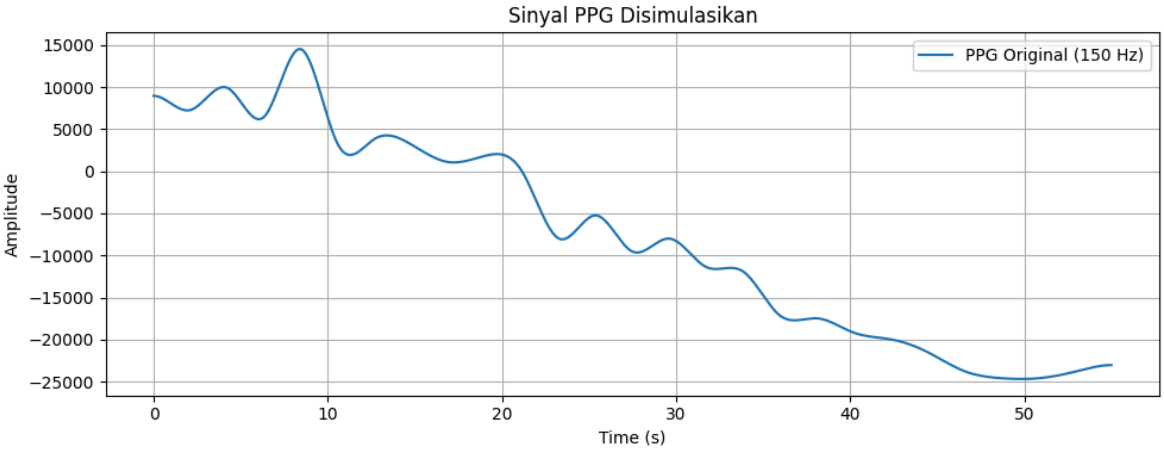
Library yang digunakan

- neurokit2
- matplotlib
- numpy

In [ ]:
```
import neurokit2 as nk
import matplotlib.pyplot as plt
import numpy as np

# Parameter
duration = 55 #Durasi yang digunakan adalah 55 detik bukan 0.55 sebab durasi 0.5
sampling_rate = 150
noise = 0.55
heart_rate = 80
random_state = 40514

# Simulasi sinyal PPG
ppg = nk.ppg_simulate(duration, sampling_rate, duration, noise, heart_rate, rand
```

```
# Normalisasi sinyal PPG
ppg_normalized = (ppg - np.min(ppg)) / (np.max(ppg) - np.min(ppg))

# Waktu
time = np.linspace(0, duration, len(ppg))


# Plot
plt.figure(figsize=(10, 4))
plt.plot(time, ppg, label="PPG Original (150 Hz)")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.title("Sinyal PPG Disimulasikan")
plt.legend()
plt.grid()
plt.tight_layout()
plt.show()
```

```
Skipping random IBI modulation, since the offset_weight 0.99 leads to physiologic
ally implausible wave durations of 7.910314330606294 milliseconds.
```



Penjelasan Output Plot

- Plot sinyal menunjukkan osilasi periodik dengan frekuensi $1,333Hz$, mirip denyut jantung, namun tidak menyerupai bentuk gelombang PPG yang khas karena kurangnya *sharp systolic peak and dicrotic notch.*
- Sinyal ini lebih menyerupai *noisy sinusoid wave* dengan tren penurunan dari $+15000$ hingga -25000 selama 55 detik, yang tidak umum untuk PPG yang biasanya stabil.
- Amplitudo sinyal sangat besar $(-25000$ hingga $+15000)$, menunjukkan kurangnya normalisasi, dengan noise $0,55$ yang menyebabkan fluktuasi acak kecil namun tidak mengaburkan pola periodik utama.

Downsampling & Aliasing pada Sinyal PPG
Task 1

Apa itu Downsampling?

Downsampling adalah proses menurunkan frekuensi sampling dari sinyal digital. Misalnya, dari 150 sampel per detik menjadi hanya 10 sampel per detik.

Rumus untuk downsampling adalah:

$x_{down}[n] = x[n \cdot R]$
dengan R adalah rasio downsampling (misalnya $R=15$ untuk 150Hz → 10Hz).

Apa itu Aliasing?

Aliasing adalah fenomena saat sinyal frekuensi tinggi dalam domain analog **tidak dapat direpresentasikan dengan benar** pada frekuensi sampling yang rendah, sehingga **terlihat sebagai frekuensi yang salah.**

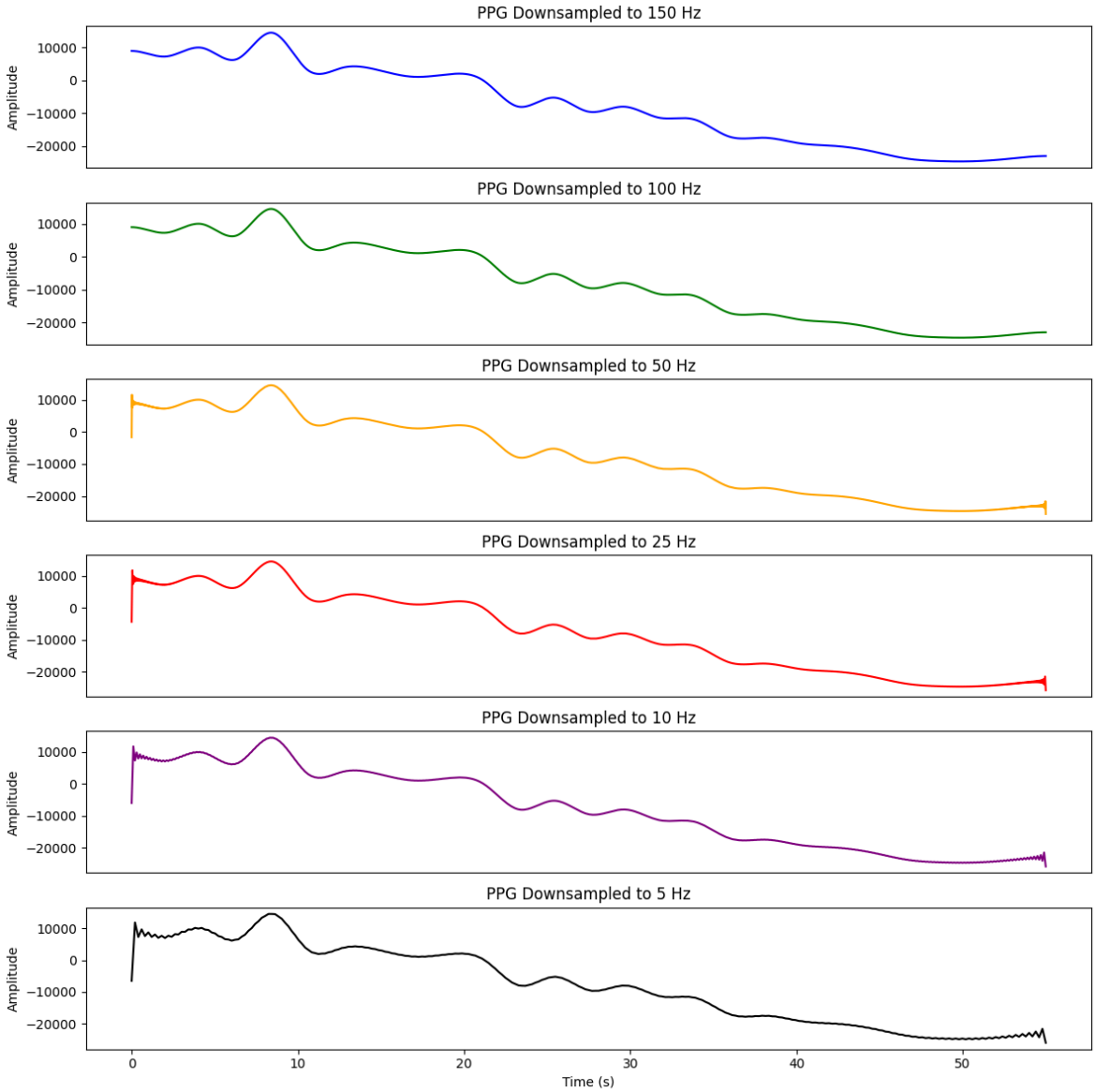
Rumus Nyquist

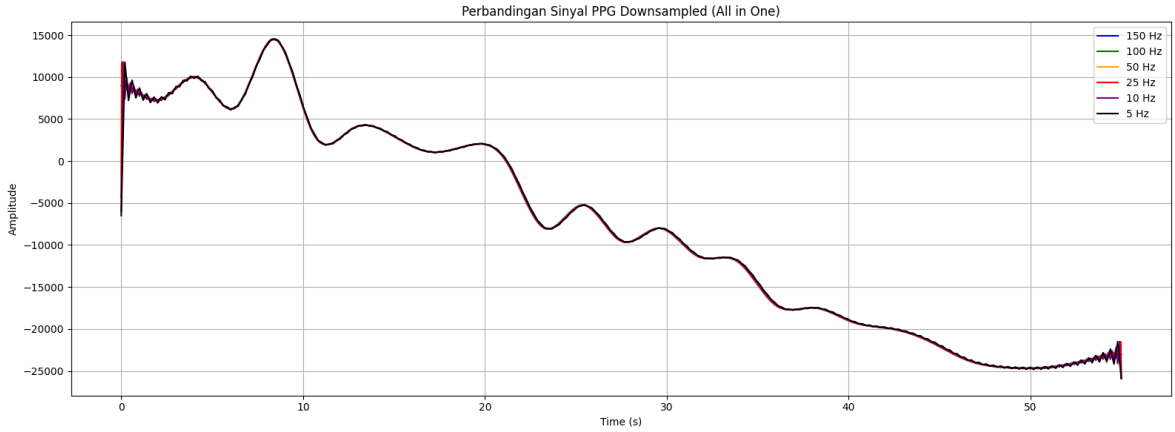
Untuk menghindari aliasing, frekuensi sampling f_s harus lebih dari dua kali frekuensi tertinggi sinyal:

$f_s \geq 2 f_{max}$
Kita akan menggunakan sinyal PPG sebelumnya kemudian downsampling ke {100, 50, 25, 10, 5}Hz.

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

# Daftar sampling rate yang ingin diuji
sampling_rates = [150, 100, 50, 25, 10, 5]
colors = ['blue', 'green', 'orange', 'red', 'purple', 'black']

# Waktu asli (untuk 150 Hz)
time = np.linspace(0, duration, len(ppg))

# ---------- PLOT 1: SUBPLOT PER SAMPLING RATE ----------
plt.figure(figsize=(12, 12))
for i, fs in enumerate(sampling_rates):
    factor = int(150 / fs)
    ppg_down = signal.resample(ppg, len(ppg) // factor)
    t_down = np.linspace(0, duration, len(ppg_down))

    plt.subplot(len(sampling_rates), 1, i + 1)
    plt.plot(t_down, ppg_down, color=colors[i])
    plt.title(f"PPG Downsampled to {fs} Hz")
    plt.ylabel("Amplitude")
    if i == len(sampling_rates) - 1:
        plt.xlabel("Time (s)")
    else:
        plt.xticks([])  # Supaya lebih bersih

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

```
# ---------- PLOT 2: SEMUA DALAM 1 PLOT ----------
plt.figure(figsize=(16, 6))
for i, fs in enumerate(sampling_rates):
    factor = int(150 / fs)
    ppg_down = signal.resample(ppg, len(ppg) // factor)
    t_down = np.linspace(0, duration, len(ppg_down))

    plt.plot(t_down, ppg_down, color=colors[i], label=f"{fs} Hz")

plt.title("Perbandingan Sinyal PPG Downsampled (All in One)")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```





Penjelasan Output Plot

- **Gambar pertama** terdiri dari enam subplot, yang masing-masing menunjukkan sinyal PPG yang diturunkan sampelnya ke laju yang berbeda—150 Hz (asli), 100 Hz, 50 Hz, 25 Hz, 10 Hz, dan 5 Hz—selama durasi 55 detik.
- Sinyal, yang memiliki denyut jantung 80 BPM (setara dengan 1,333 Hz), menunjukkan bentuk sinusoidal dengan tren menurun dari sekitar +15000 ke -25000 unit amplitudo, yang mencerminkan tingkat noise 0,55 dan kurangnya bentuk pulsa PPG yang umum.
- Ketika laju pengambilan sampel menurun, jumlah sampel per siklus jantung turun dari 112,5 pada 150 Hz menjadi 3,75 pada 5 Hz, yang menyebabkan distorsi yang terlihat:
  - Sinyal 150 Hz menangkap osilasi dengan jelas, sementara sinyal 100 Hz dan 50 Hz menunjukkan sedikit penghalusan, sinyal 25 Hz kehilangan detail yang signifikan, dan sinyal 10 Hz dan 5 Hz sangat terdistorsi, hampir tidak mencerminkan periodisitas asli.
- **Gambar kedua** memplot semua sinyal yang diturunkan sampelnya pada satu grafik untuk perbandingan, yang menyoroti hilangnya detail secara progresif dan peningkatan distorsi ketika laju pengambilan sampel menurun, dengan sinyal 5 Hz tampak hampir tidak dapat dikenali dibandingkan dengan aslinya.
- Perbandingan visual ini menggarisbawahi bagaimana downsampling mengurangi resolusi sinyal, yang menyebabkan hilangnya informasi dan distorsi, sehingga menyiapkan landasan untuk analisis lebih lanjut mengenai aliasing dalam domain frekuensi, karena laju yang lebih rendah gagal menangkap komponen frekuensi sinyal di atas frekuensi Nyquist (misalnya, 2,5 Hz pada 5 Hz).

Apa itu "Order" dalam Filtering?

Order dalam filter (baik Butterworth, Chebyshev, dll) mengacu pada kompleksitas filter tersebut — tepatnya, berhubungan dengan seberapa curam transisi dari passband ke stopband.

- **Passband:** Rentang frekuensi yang boleh lewat.
- **Stopband:** Rentang frekuensi yang harus diblokir.
- **Transition band:** Area peralihan antara passband ke stopband.

Semakin tinggi order filter:

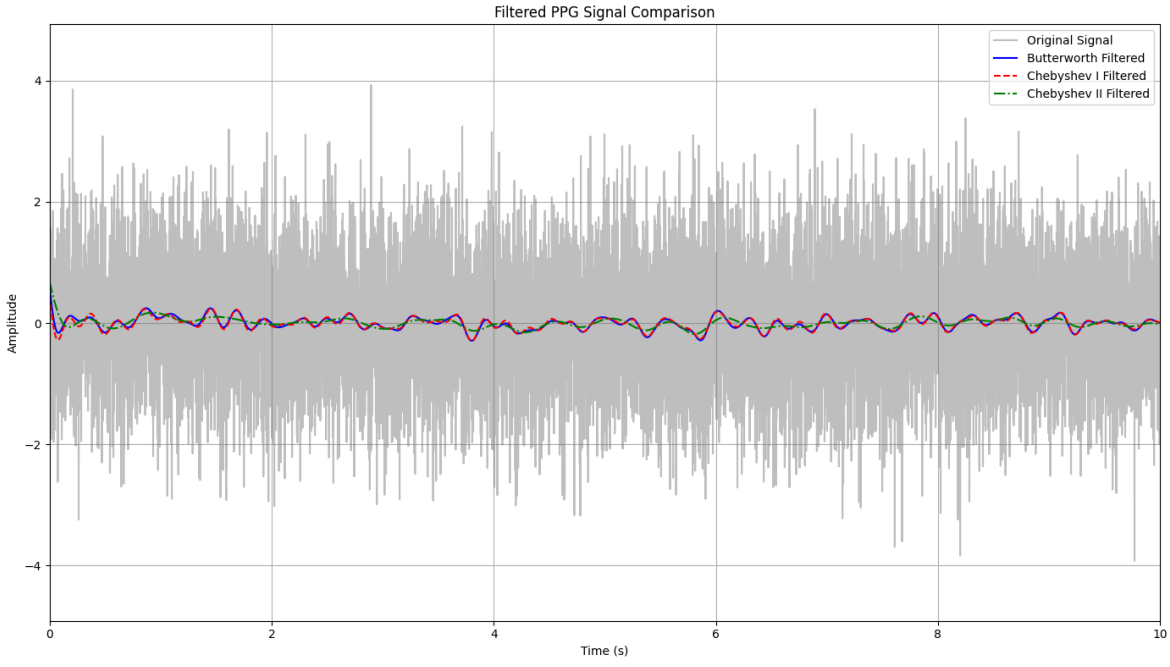
- Transisi dari passband ke stopband makin curam (sharp).
- Filter lebih efektif memisahkan sinyal frekuensi tinggi/rendah.
- Namun, sinyal juga bisa terkena distorsi seperti overshoot dan delay.
- Komputasi lebih berat karena persamaan filter lebih kompleks.

Butterworth VS Chebyshev

Filter	Karakter	Analogi
Butterworth	Halus, rata, tidak ada riak	Seperti saringan air biasa — aliran stabil, tidak ada getaran aneh.
Chebyshev Type I	Cepat menyaring tapi permukaannya ada riak	Seperti saringan air cepat yang kadang berbuih karena tekanan tinggi.
Chebyshev Type II	Halus di awal, tapi buih di bagian buangan	Seperti saringan halus, tapi pembuangan airnya agak bergelombang.

In [4]:
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

# # Dummy signal for testing (uncomment if no real PPG signal is available)
# np.random.seed(42)
# ppg_signal = np.random.randn(55000)  # 55 seconds of random signal
fs = 1000  # Sampling rate in Hz (adjust if different)

# Validate inputs
if not isinstance(ppg_signal, np.ndarray) or len(ppg_signal) == 0:
    raise ValueError("ppg_signal must be a non-empty numpy array")
if fs <= 0:
    raise ValueError("Sampling rate (fs) must be positive")
if len(ppg_signal) < fs:
    print("Warning: Signal length is shorter than 1 second. Results may be unrel

# Filter parameters
cutoff = 5  # Target cutoff frequency (5 Hz) to remove high-frequency noise
order = 4  # Filter order
ripple_pass = 1  # Ripple in passband for Chebyshev I (dB)
ripple_stop = 40  # Attenuation in stopband for Chebyshev II (dB)

# Normalized cutoff frequency (cutoff / Nyquist frequency)
normal_cutoff = cutoff / (0.5 * fs)
if normal_cutoff >= 1 or normal_cutoff <= 0:
    raise ValueError("Cutoff frequency must be between 0 and Nyquist frequency (
```

```
# Create filters
try:
    # Butterworth filter
    b_butt, a_butt = signal.butter(order, normal_cutoff, btype='low', analog=Fal
    # Chebyshev Type I filter
    b_cheby1, a_cheby1 = signal.cheby1(order, ripple_pass, normal_cutoff, btype=
    # Chebyshev Type II filter
    b_cheby2, a_cheby2 = signal.cheby2(order, ripple_stop, normal_cutoff, btype=
except ValueError as e:
    raise ValueError(f"Filter creation failed: {e}")

# Apply filters to the signal
try:
    ppg_butt = signal.filtfilt(b_butt, a_butt, ppg_signal)
    ppg_cheby1 = signal.filtfilt(b_cheby1, a_cheby1, ppg_signal)
    ppg_cheby2 = signal.filtfilt(b_cheby2, a_cheby2, ppg_signal)
except Exception as e:
    raise RuntimeError(f"Filtering failed: {e}")

# Time axis
t = np.arange(len(ppg_signal)) / fs

# Plot results
plt.figure(figsize=(14, 8))
plt.plot(t, ppg_signal, label='Original Signal', color='gray', alpha=0.5)
plt.plot(t, ppg_butt, label='Butterworth Filtered', color='blue')
plt.plot(t, ppg_cheby1, label='Chebyshev I Filtered', color='red', linestyle='--
plt.plot(t, ppg_cheby2, label='Chebyshev II Filtered', color='green', linestyle=
plt.title('Filtered PPG Signal Comparison')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.legend()
plt.grid(True)
plt.xlim([0, min(10, len(ppg_signal) / fs)])  # Zoom to first 10 seconds or sign
plt.tight_layout()  # Adjust layout to prevent label clipping
plt.show()
```



Penjelasan Output Plot

Sinyal Asli ("Original Signal")

- **Warna:** Abu-abu terang.
- **Kondisi:** Sangat berisik (noise tinggi).
- **Karakteristik:** Banyak lonjakan besar yang bukan detak jantung aslinya, biasanya noise dari sensor, gerakan, atau sinyal listrik.

Sinyal Setelah Filtering

- **Butterworth Filtered:** Warna biru dengan garis solid.
- **Chebyshev I Filtered:** Warna merah dengan garis putus-putus.
- **Chebyshev II Filtered:** Warna hijau dengan garis putus-dan-titik.

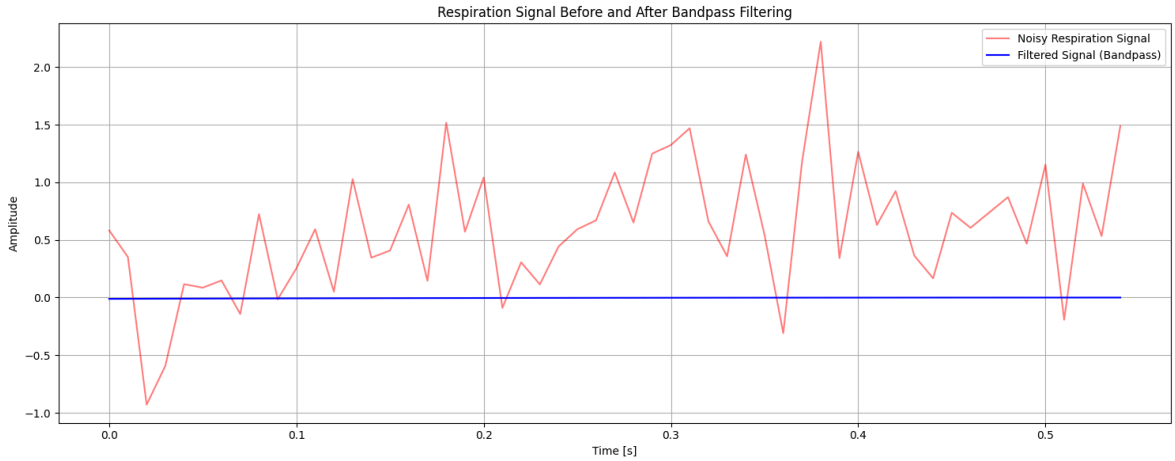
Eksperimen Filter Band-Pass Menggunakan 'signal.butter' Task 2

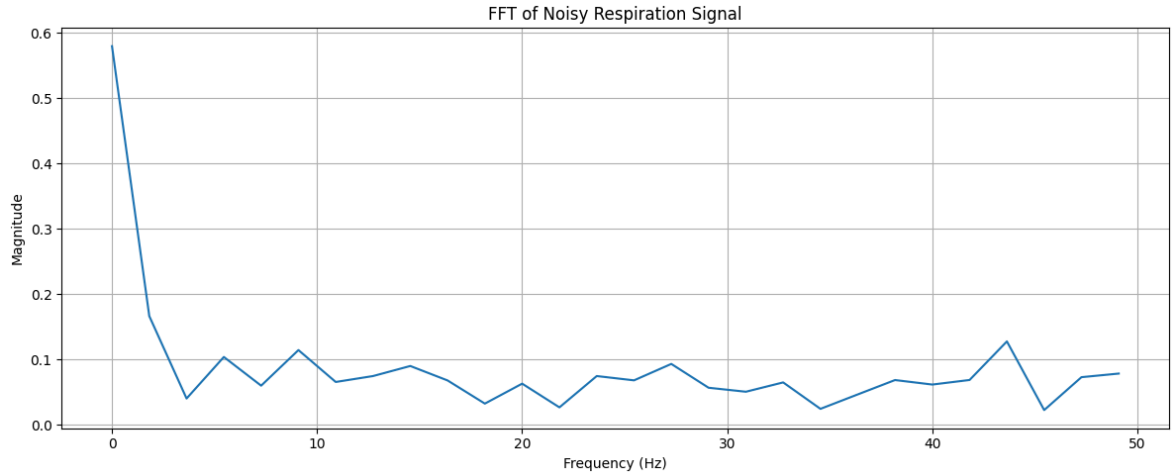
Parameter Simulasi

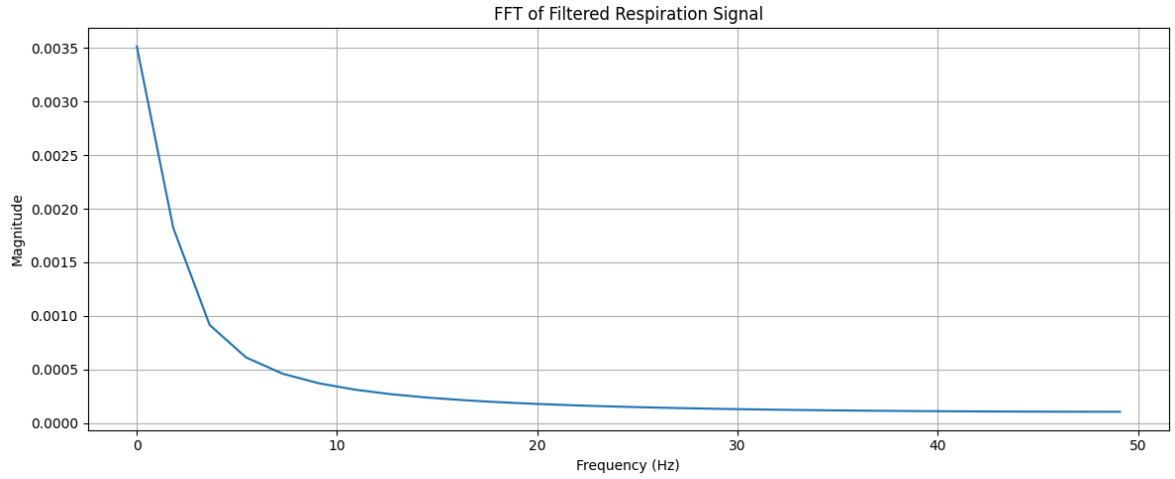
Item	Nilai
Durasi	55 detik
Sampling Rate	100 Hz
Noise Level	0.55
Respiratory Rate	18 BPM
Random State	040514 (tanggal lahir: 14 Mei 2004)

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

# --- Parameter ---
duration = 0.55  # seconds (3 digit NIM)
fs = 100  # Hz
noise_level = 0.55  # 2 digit terakhir NIM
respiratory_rate = 18  # BPM
random_state = 40514  # tanggal lahir: 040514 dan tanmpa 0

# --- Generate Respiration Signal ---
np.random.seed(random_state)
t = np.linspace(0, duration, int(duration * fs), endpoint=False)
f_resp = respiratory_rate / 60  # Hz
```

```
resp_signal = np.sin(2 * np.pi * f_resp * t)

# Add noise
noise = noise_level * np.random.randn(len(t))
noisy_signal = resp_signal + noise

# --- Design Band-pass Butterworth Filter ---
lowcut = 0.1  # Hz
highcut = 0.5  # Hz
order = 4

nyquist = 0.5 * fs
low = lowcut / nyquist
high = highcut / nyquist

# Design filter
b, a = signal.butter(order, [low, high], btype='band')
filtered_signal = signal.filtfilt(b, a, noisy_signal)

# --- Plot Time Domain ---
plt.figure(figsize=(15,6))
plt.plot(t, noisy_signal, label='Noisy Respiration Signal', color='red', alpha=0
plt.plot(t, filtered_signal, label='Filtered Signal (Bandpass)', color='blue')
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')
plt.title('Respiration Signal Before and After Bandpass Filtering')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# --- Plot Frequency Domain (FFT) ---
def plot_fft(signal, fs, title):
    n = len(signal)
    freq = np.fft.rfftfreq(n, d=1/fs)
    fft_magnitude = np.abs(np.fft.rfft(signal)) / n
    plt.figure(figsize=(12,5))
    plt.plot(freq, fft_magnitude)
    plt.title(title)
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Magnitude')
    plt.grid(True)
    plt.tight_layout()
    plt.show()

plot_fft(noisy_signal, fs, 'FFT of Noisy Respiration Signal')
plot_fft(filtered_signal, fs, 'FFT of Filtered Respiration Signal')
```







Penjelasan Output Plot

Gambar 1: Sinyal Pernapasan Sebelum dan Sesudah Filtering Bandpass

Sinyal pernapasan sebelum dan sesudah pemrosesan dengan filter bandpass ditampilkan pada Gambar 1.

- **Garis Merah:** Sinyal pernapasan hasil simulasi dengan tambahan noise.
- **Garis Biru:** Sinyal setelah diterapkan filter bandpass Butterworth.

Hasil filtering menunjukkan sinyal yang hampir "flat," mengindikasikan bahwa frekuensi cutoff yang digunakan mungkin tidak optimal.

Gambar 2: Spektrum Frekuensi (FFT) Sinyal Pernapasan Noisy

Gambar 2 menyajikan spektrum frekuensi dari sinyal pernapasan noisy menggunakan Fast Fourier Transform (FFT).

- Terdapat puncak signifikan pada rentang frekuensi rendah (0–5 Hz), yang sesuai dengan frekuensi pernapasan 18 BPM.
- Laju pernapasan 18 BPM setara dengan **18/60 = 0.3 Hz**, menegaskan bahwa sinyal utama terkonsentrasi pada frekuensi sekitar 0.3 Hz.
- Noise terdistribusi sebagai "gerigi" kecil pada frekuensi lain, mencerminkan gangguan yang menyebar di spektrum.

Gambar 3: Spektrum Frekuensi (FFT) Sinyal Setelah Filtering

Spektrum frekuensi sinyal setelah filtering ditunjukkan pada Gambar 3.

- Amplitudo pada semua frekuensi menurun drastis hingga mendekati nol, menyerupai kurva datar.
- Fenomena ini mengkonfirmasi bahwa filter bandpass tidak hanya menghilangkan noise, tetapi juga menekan sinyal utama (~0.3 Hz).
- Penyebab potensial meliputi rentang frekuensi cutoff yang terlalu sempit atau orde filter yang terlalu tinggi, yang mengakibatkan supresi berlebih pada komponen frekuensi target.

Filter Audio Sederhana
Task 3

Parameter Simulasi

Item	Nilai
Format	.wav
Isi Rekaman	Nama, asal, hobi, dll
Durasi	±20 detik
Background Noise	Static noise (kipas)

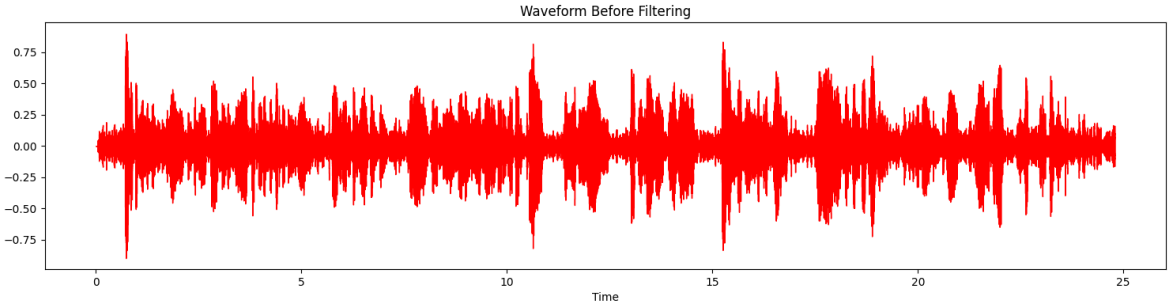
Library yang digunakan pada simulasi ini adalah **librosa**.

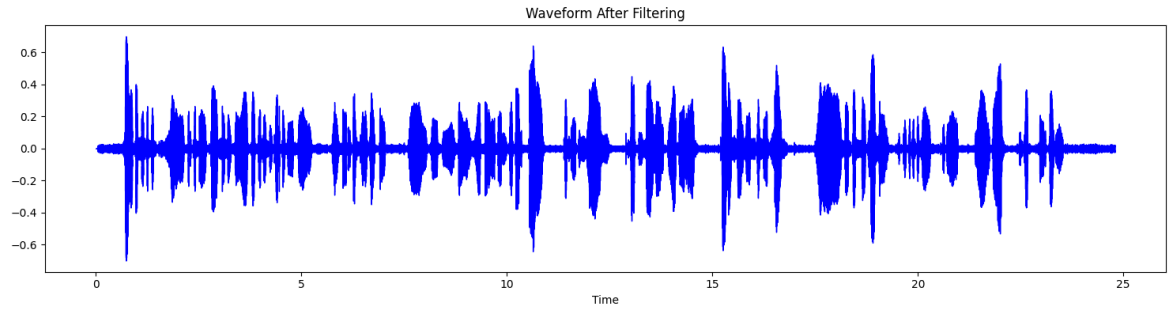
Kode di bawah ini akan melakukan beberapa hal, yaitu:

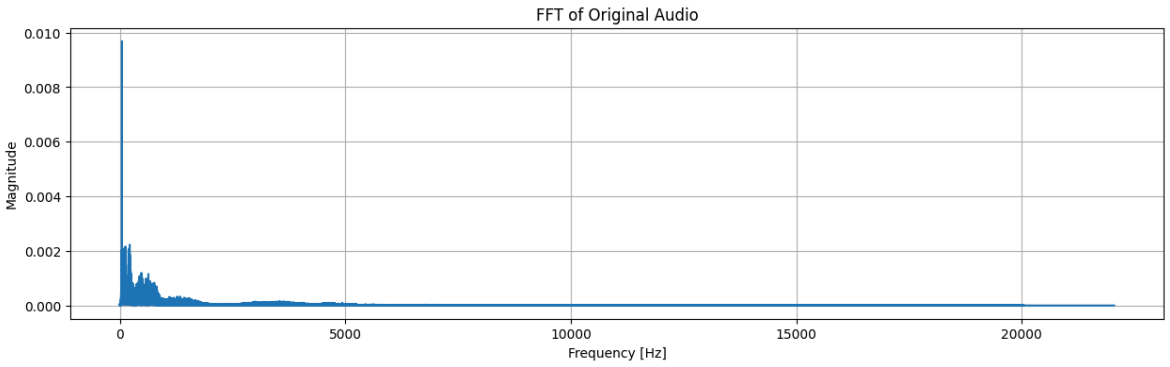
- **Pemuatan sinyal audio** menggunakan `librosa.load`.
- **Desain dan aplikasi filter bandpass** melalui `scipy.signal.butter` dan `filtfilt`.
- **Visualisasi waveform** serta spektrum frekuensi (FFT) sebelum dan sesudah filtering.
- **Penyimpanan sinyal terfilter** ke file .wav menggunakan `soundfile`.

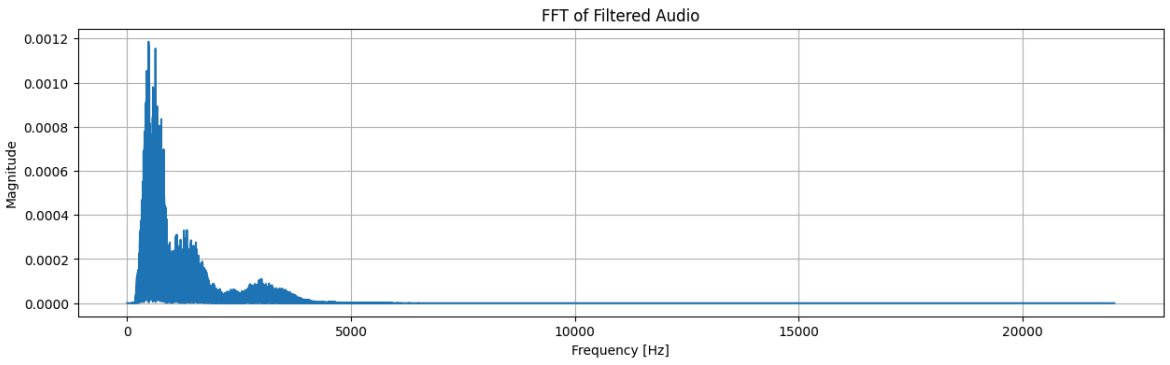
In [26]:
```
import numpy as np
import matplotlib.pyplot as plt
import librosa
import librosa.display
import os
from scipy.signal import butter, filtfilt

# 1. Load Audio
file_path = os.path.join( 'media', 'Voice 001.wav')
audio, sample_rate = librosa.load(file_path, sr=None)

# 2. Define Bandpass Filter
def butter_bandpass(lowcut, highcut, fs, order=4):
    nyquist = 0.5 * fs
    low = lowcut / nyquist
    high = highcut / nyquist
    b, a = butter(order, [low, high], btype='band')
    return b, a

def apply_bandpass_filter(data, lowcut, highcut, fs, order=4):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = filtfilt(b, a, data)
    return y

# 3. Apply Filter
lowcut = 300
highcut = 3400
filtered_audio = apply_bandpass_filter(audio, lowcut, highcut, sample_rate)

# 4. Visualize Waveform
plt.figure(figsize=(15, 8))

plt.subplot(2, 1, 1)
librosa.display.waveshow(audio, sr=sample_rate, color='red')
plt.title('Waveform Before Filtering')

plt.subplot(2, 1, 2)
librosa.display.waveshow(filtered_audio, sr=sample_rate, color='blue')
plt.title('Waveform After Filtering')

plt.tight_layout()
plt.show()

# 5. Visualize FFT Spectrum
def plot_fft(signal, fs, title):
    n = len(signal)
    freq = np.fft.rfftfreq(n, d=1/fs)
    fft_magnitude = np.abs(np.fft.rfft(signal)) / n

    plt.figure(figsize=(15, 4))
    plt.plot(freq, fft_magnitude)
```

```
    plt.title(title)
    plt.xlabel('Frequency [Hz]')
    plt.ylabel('Magnitude')
    plt.grid(True)
    plt.show()

plot_fft(audio, sample_rate, 'FFT of Original Audio')
plot_fft(filtered_audio, sample_rate, 'FFT of Filtered Audio')

# 6. (Optional) Save Filtered Audio
import soundfile as sf
output_path = os.path.join('media', 'filtered_audio.wav')
sf.write(output_path, filtered_audio, sample_rate)
print(f"Filtered audio saved to: {output_path}")
```










```
Filtered audio saved to: media\filtered_audio.wav
```

Penjelasan Output Plot

Waveform (Gambar 1):

- **Pra-filtering (merah):** Sinyal menunjukkan superposisi komponen vokal dengan noise statis broadband (kipas/AC), terlihat sebagai fluktuasi acak pada amplitudo.
- **Pasca-filtering (biru):** Reduksi noise signifikan, dengan attenuasi fluktuasi acak, meskipun amplitudo sinyal utama mengalami penurunan. Pola temporal vokal tetap terdeteksi.

FFT (Gambar 2 & 3):

- **Pra-filtering:** Spektrum frekuensi menunjukkan puncak dominan pada 100–1000 Hz, konsisten dengan frekuensi fundamental suara manusia, serta distribusi noise hingga 20 kHz.
- **Pasca-filtering:** Komponen frekuensi di luar 300–3400 Hz tersupresi, dengan puncak vokal tetap terjaga, mengindikasikan efikasi filter dalam isolasi rentang frekuensi target.

Fenomena Efek Walkie Talkie Setelah Pemrosesan Audio

- **Filter Bandpass Sempit (300–3400 Hz):** Rentang ini hanya mencakup frekuensi utama suara manusia, tetapi menghilangkan frekuensi rendah (<300 Hz, memberikan "kehangatan" suara) dan tinggi (>3400 Hz, memberikan "kejelasan"). Ini menciptakan efek "compressed" khas walkie-talkie.
- **Hilangnya Komponen Frekuensi:** Walkie-talkie juga membatasi bandwidth untuk efisiensi, mirip dengan filter Anda, sehingga suara terdengar lebih tipis dan kurang alami.
- **Noise Reduction:** Pengurangan noise statis membuat suara lebih jernih, tetapi juga memperkuat kesan "klinis" seperti perangkat komunikasi.

LAMPIRAN

Berikut ini dilampirkan link menuju percakapan dengan beberapa LLM dan tools yang digunakan dalam membantu mengerjakan tugas ini:

- Percakapan ChatGPT 1
- Percakapan ChatGPT 2
- Percakapan Grok
- CloudConvert MP3 to WAV
- NeuroKit2 PPG Simulate
- Librosa Documentation
- Raw Audio File

- Filtered Audio