122140012

April 28, 2025

Hands On 2 DSP

NIM: 122140012

Tanggal Lahir: 29 Februari 2004 (040229)

0.0.1 Soal 1

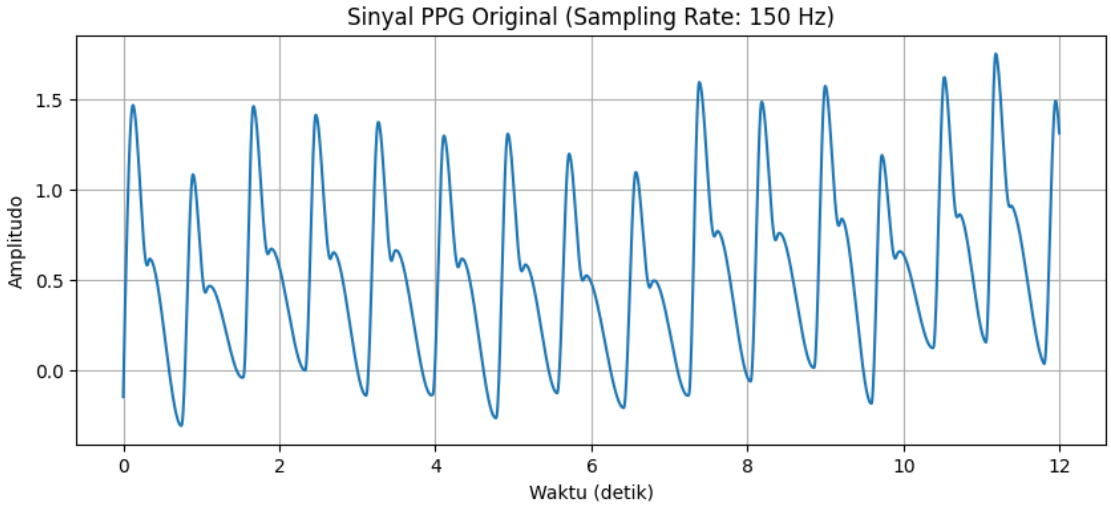
[22]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from scipy.fft import fft, fftfreq
from scipy.io import wavfile
import neurokit2 as nk
```

[3]:
```
# Parameter
duration = 12  # 3 digit terakhir NIM (012)
fs_original = 150  # Sampling rate awal (Hz)
noise_level = 0.12  # 2 digit terakhir NIM (12)
heart_rate = 80  # BPM (Beat Per Minute)
random_state = 40229  # Format YYMMDD tanggal lahir
```

[4]:
```
# Simulasi sinyal PPG
ppg_signal = nk.ppg_simulate(duration=duration,
                             sampling_rate=fs_original,
                             heart_rate=heart_rate,
                             drift=noise_level,
                             random_state=random_state)

# Waktu untuk plot original
t_original = np.linspace(0, duration, len(ppg_signal), endpoint=False)
```

[5]:
```
# Visualisasi sinyal PPG original
plt.figure(figsize=(10, 4))
plt.plot(t_original, ppg_signal)
plt.title("Sinyal PPG Original (Sampling Rate: {} Hz)".format(fs_original))
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
```

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



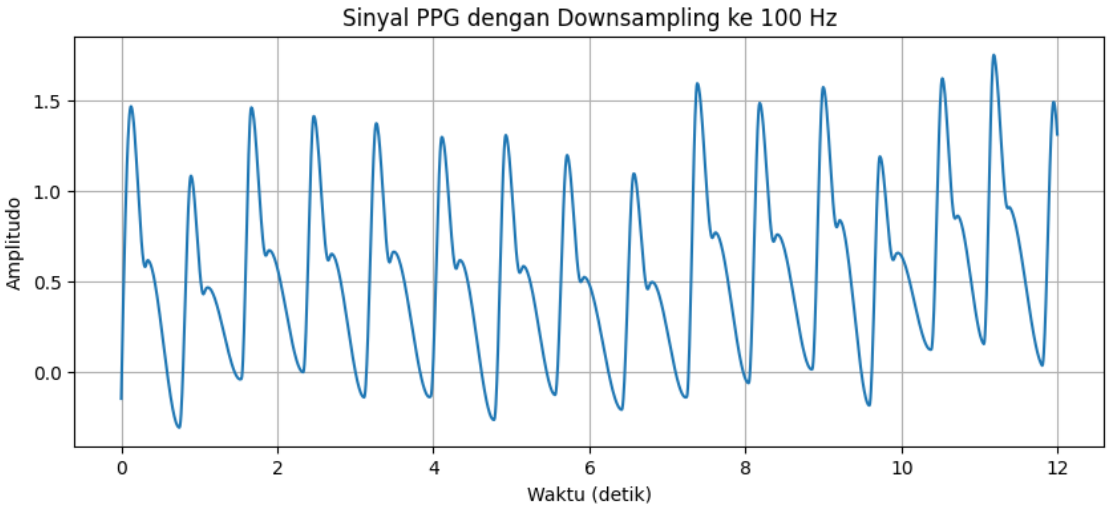
Plot ini menunjukkan sinyal PPG (Photoplethysmography) yang disimulasikan dengan parameter sesuai data pribadi (sampling rate 150 Hz, noise level 0.12, heart rate 80 BPM). Pola gelombang yang berulang menggambarkan detak jantung, dengan puncak-puncak yang jelas dan teratur. Sinyal ini mencerminkan aliran darah dari jantung ke tangan dengan amplitudo yang berkisar antara -0.5 hingga 1.7.

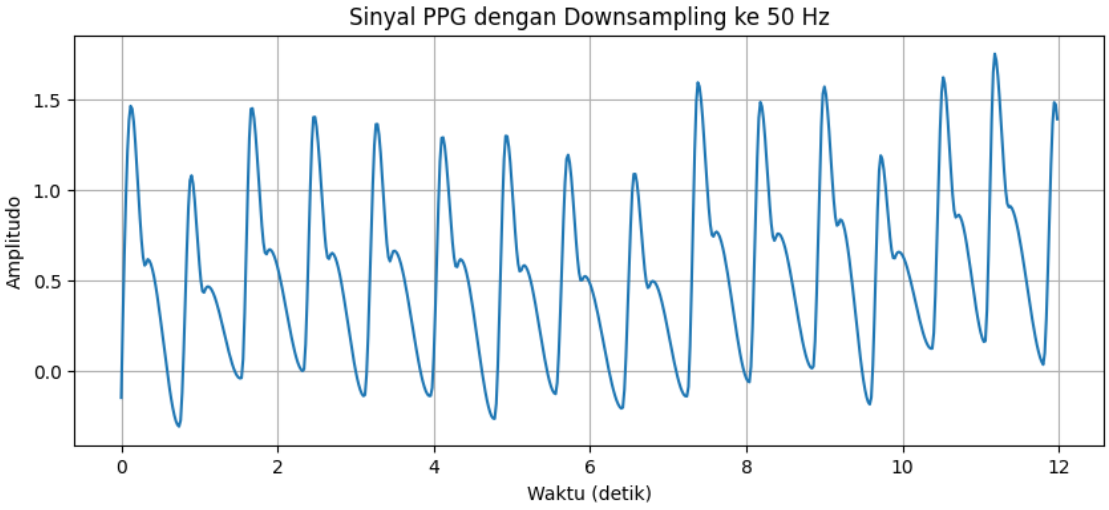
[6]:
```
# List sampling rate yang akan diuji
sampling_rates = [100, 50, 25, 10, 5]
```

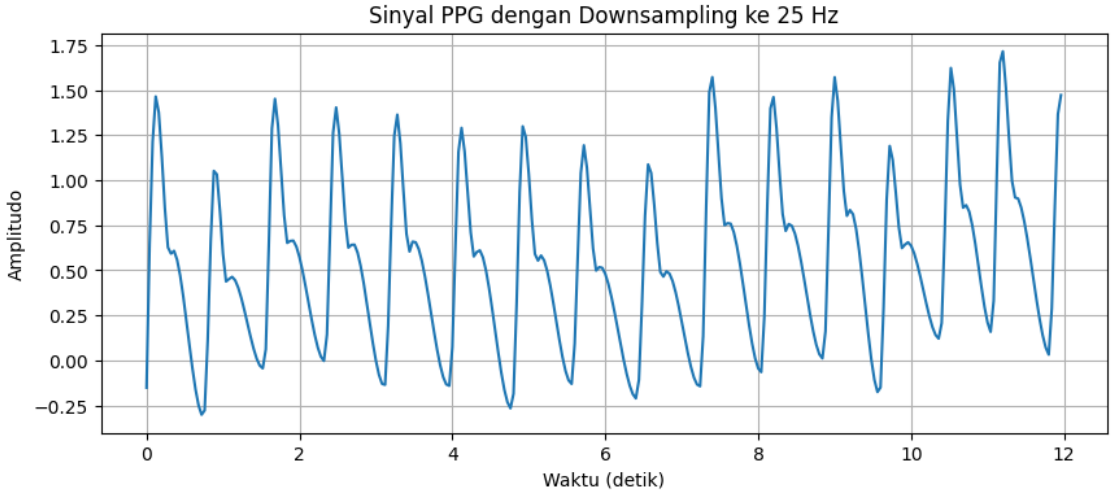
[7]:
```
# Untuk menyimpan sinyal hasil downsampling
downsampled_signals = {}
```

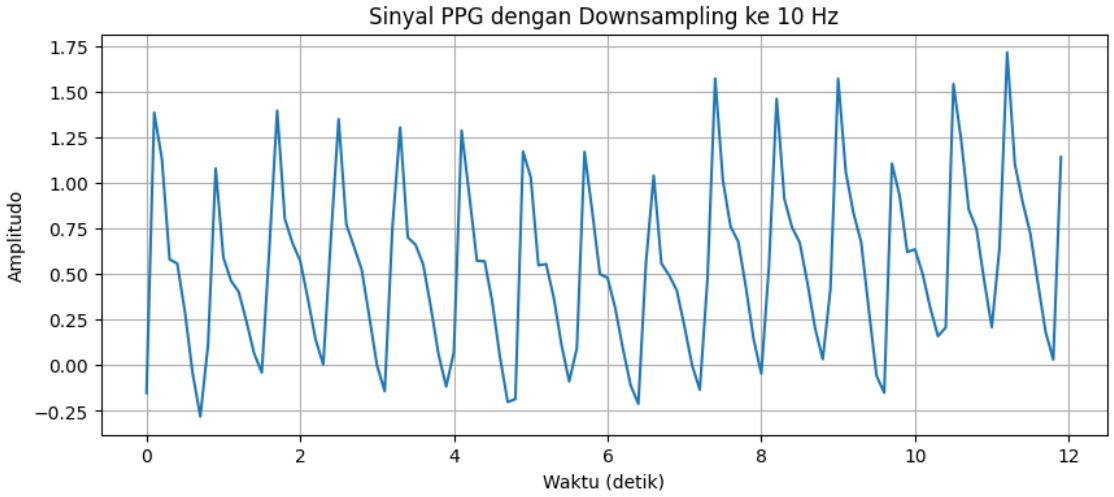
[8]:
```
# Melakukan downsampling untuk setiap sampling rate
for fs_new in sampling_rates:
    # Faktor downsampling
    factor = int(fs_original / fs_new)

    # Downsampling sederhana dengan mengambil setiap 'factor' sampel
    downsampled = ppg_signal[::factor]

    # Simpan sinyal hasil downsampling
    downsampled_signals[fs_new] = downsampled

    # Waktu untuk plot downsampled
    t_downsampled = np.linspace(0, duration, len(downsampled), endpoint=False)

    # Visualisasi sinyal hasil downsampling
```

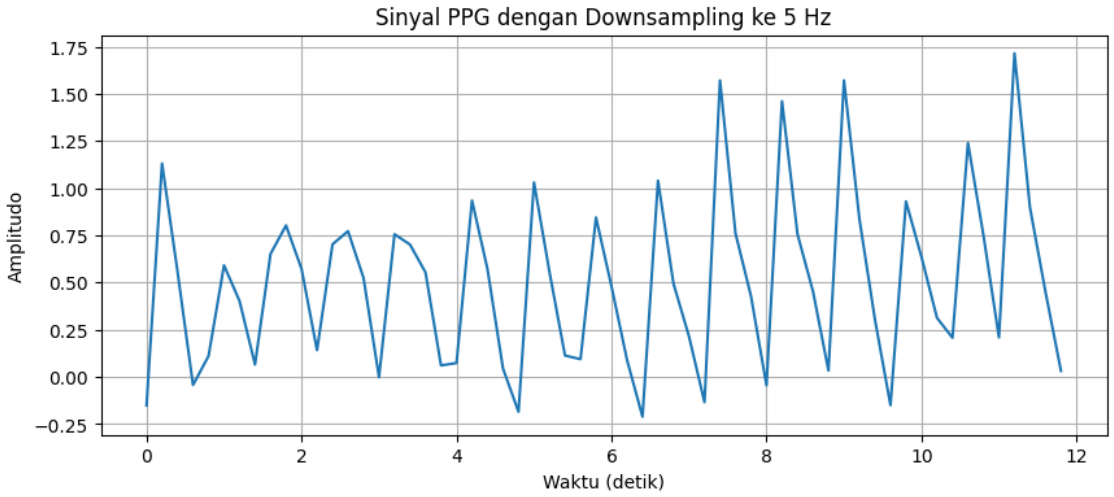
```
    plt.figure(figsize=(10, 4))
    plt.plot(t_downsampled, downsampled)
    plt.title("Sinyal PPG dengan Downsampling ke {} Hz".format(fs_new))
    plt.xlabel("Waktu (detik)")
    plt.ylabel("Amplitudo")
    plt.grid(True)
    plt.show()
```





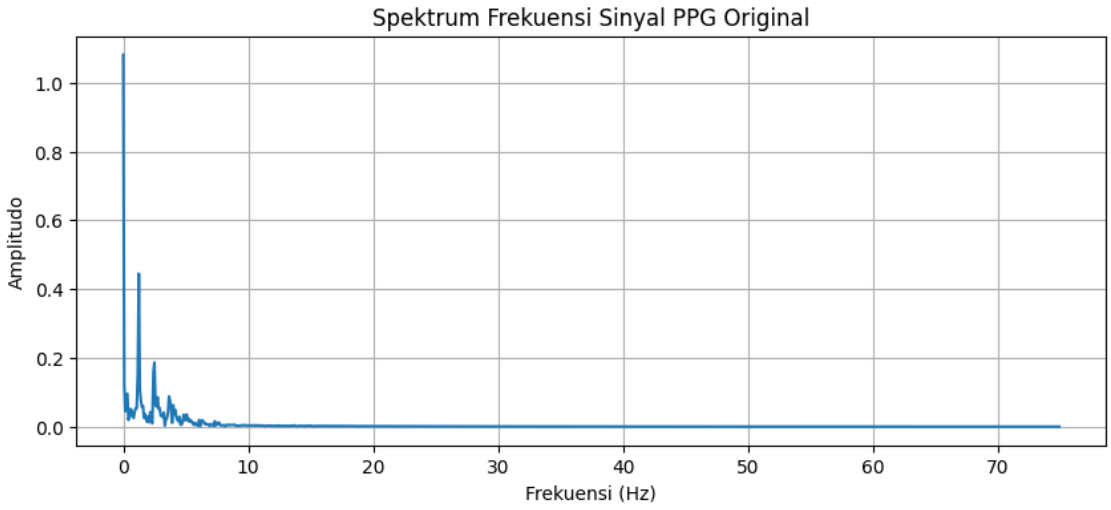

Sinyal PPG dengan Downsampling ke 25 Hz
Amplitudo
Waktu (detik)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
0
2
4
6
8
10
12


Sinyal PPG dengan Downsampling ke 10 Hz
Amplitudo
Waktu (detik)
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
0
2
4
6
8
10
12



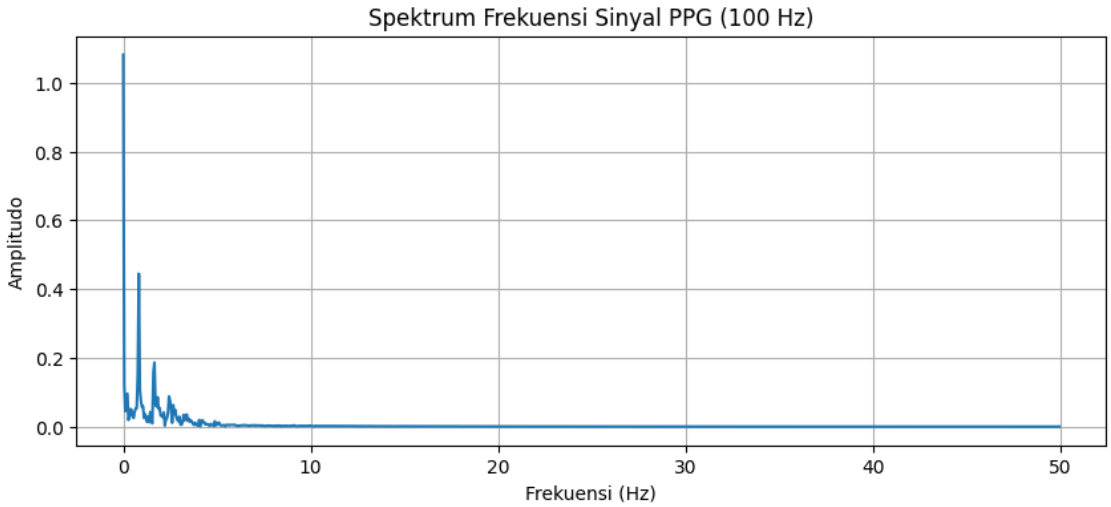
Downsampling ke 100 Hz Pada sampling rate 100 Hz, sinyal masih mempertahankan bentuk aslinya dengan sangat baik. Tidak ada distorsi yang terlihat karena frekuensi sampling masih jauh di atas frekuensi Nyquist untuk sinyal PPG. Downsampling ke 50 Hz Sinyal pada 50 Hz juga masih sangat mirip dengan sinyal asli. Informasi penting tentang detak jantung masih terjaga dengan baik. Downsampling ke 25 Hz Pada 25 Hz, mulai terlihat beberapa perubahan kecil pada detail sinyal, tetapi pola utama dan informasi detak jantung masih dapat diidentifikasi dengan jelas. Downsampling ke 10 Hz Dengan sampling rate 10 Hz, distorsi mulai terlihat lebih jelas. Beberapa informasi detail dari sinyal mulai hilang, meskipun pola detak jantung masih dapat dikenali. Downsampling ke 5 Hz Pada 5 Hz, sinyal mengalami distorsi yang signifikan. Ini adalah contoh nyata dari fenomena aliasing, di mana sampling rate menjadi terlalu rendah untuk merepresentasikan sinyal dengan benar. Bentuk sinyal telah berubah secara substansial, menunjukkan bahwa 5 Hz berada di bawah frekuensi Nyquist yang diperlukan.

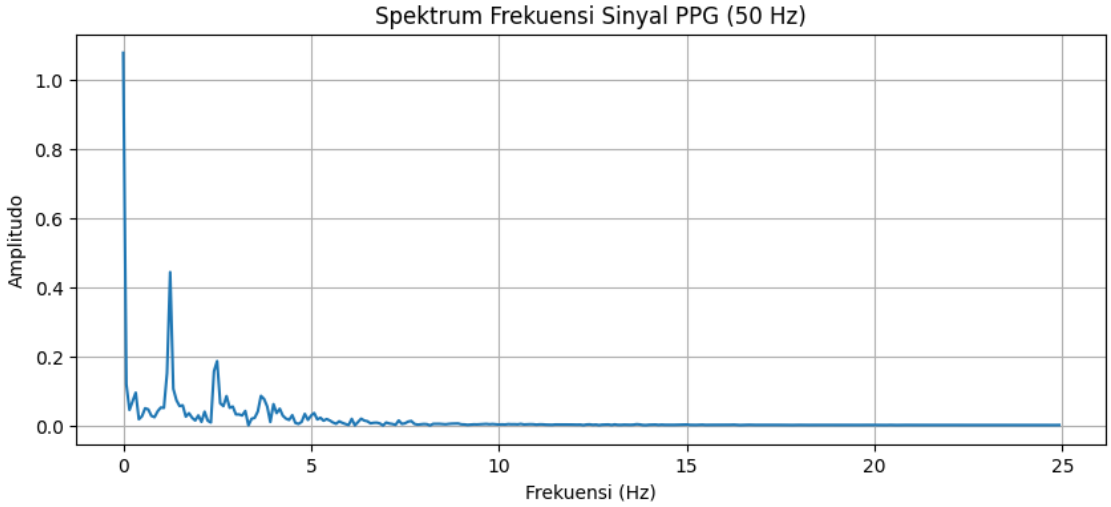
[9]:
```
# Analisis Spektrum Frekuensi untuk sinyal original
N_orig = len(ppg_signal)
yf_orig = fft(ppg_signal)
xf_orig = fftfreq(N_orig, 1/fs_original)

plt.figure(figsize=(10, 4))
plt.plot(xf_orig[:N_orig//2], 2.0/N_orig * np.abs(yf_orig[:N_orig//2]))
plt.title("Spektrum Frekuensi Sinyal PPG Original")
plt.xlabel("Frekuensi (Hz)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.show()
```

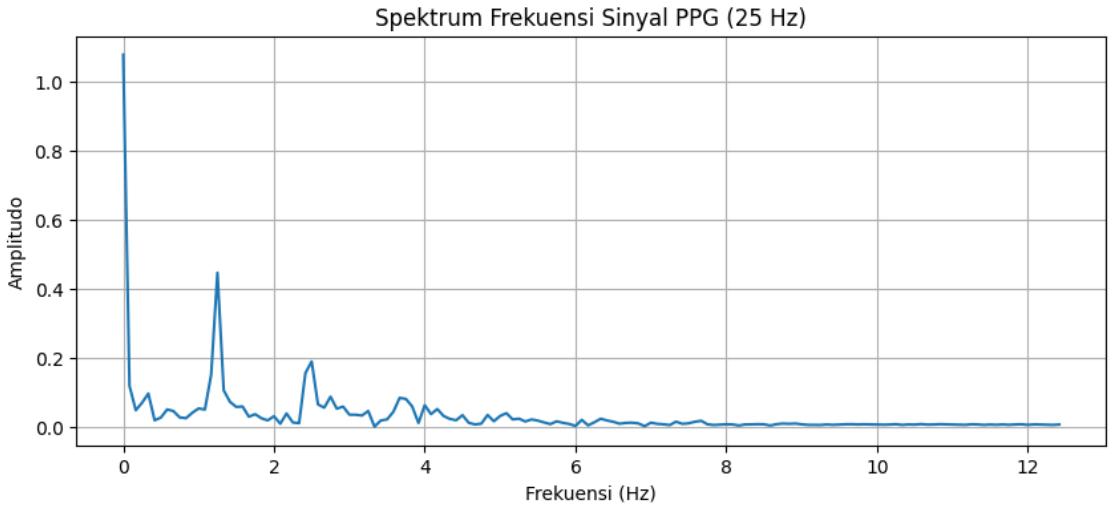


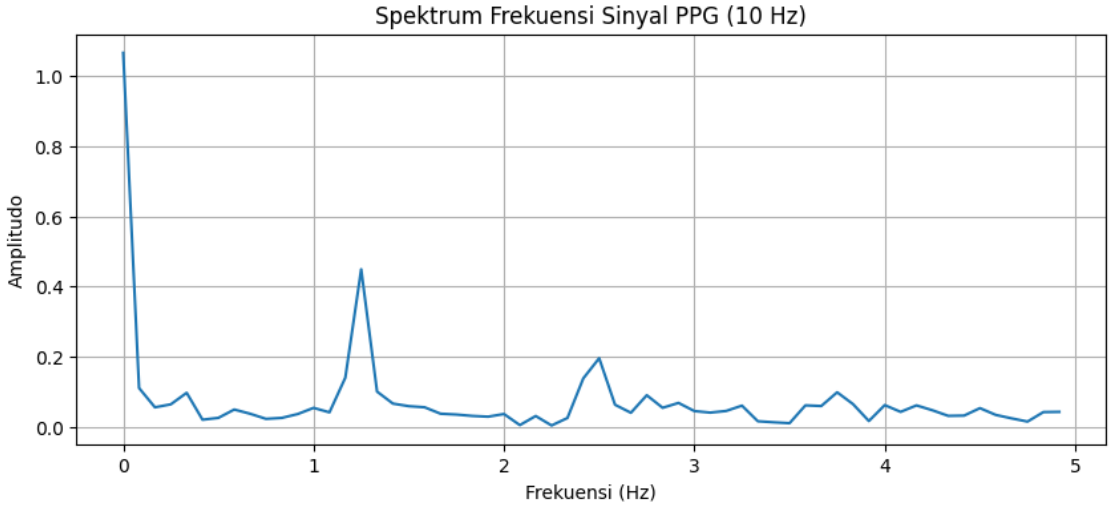
Spektrum ini menunjukkan komponen frekuensi dari sinyal PPG asli. Puncak tertinggi berada di sekitar 1-2 Hz, yang sesuai dengan heart rate 80 BPM (1.33 Hz). Juga terlihat beberapa harmonik di frekuensi yang lebih tinggi. Ini adalah representasi frekuensi yang akan kita bandingkan dengan spektrum sinyal setelah downsampling.

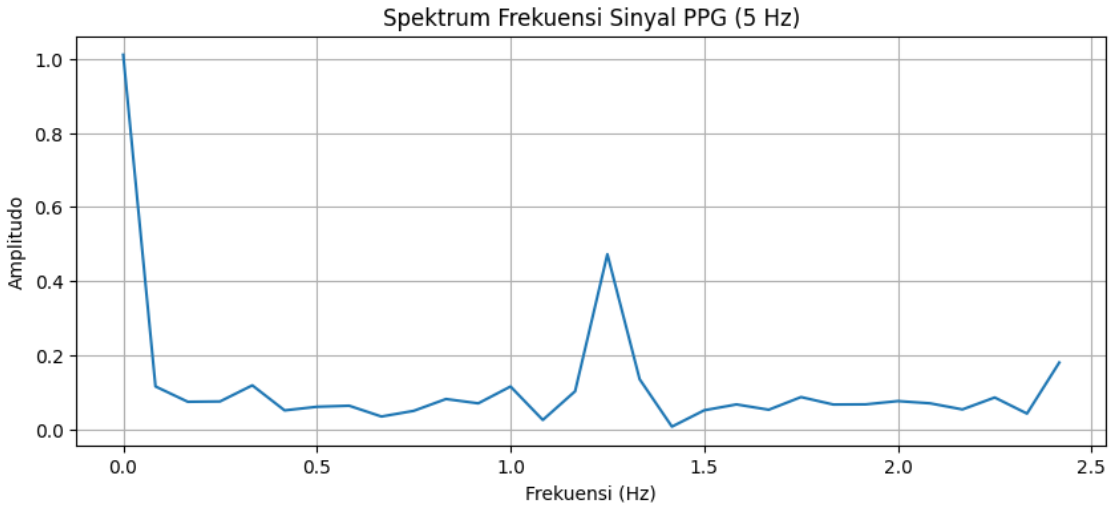
[10]:
```
# Analisis Spektrum Frekuensi untuk setiap sinyal hasil downsampling
for fs_new, downsampled in downsampled_signals.items():
    N = len(downsampled)
    yf = fft(downsampled)
    xf = fftfreq(N, 1/fs_new)

    plt.figure(figsize=(10, 4))
    plt.plot(xf[:N//2], 2.0/N * np.abs(yf[:N//2]))
    plt.title("Spektrum Frekuensi Sinyal PPG ({} Hz)".format(fs_new))
    plt.xlabel("Frekuensi (Hz)")
    plt.ylabel("Amplitudo")
    plt.grid(True)
    plt.show()
```


Spektrum Frekuensi Sinyal PPG (100 Hz)
Amplitudo
Frekuensi (Hz)
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50


Spektrum Frekuensi Sinyal PPG (50 Hz)
Amplitudo
Frekuensi (Hz)
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25


Spektrum Frekuensi Sinyal PPG (25 Hz)
Amplitudo
Frekuensi (Hz)
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10
12


Spektrum Frekuensi Sinyal PPG (10 Hz)
Amplitudo
Frekuensi (Hz)
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5

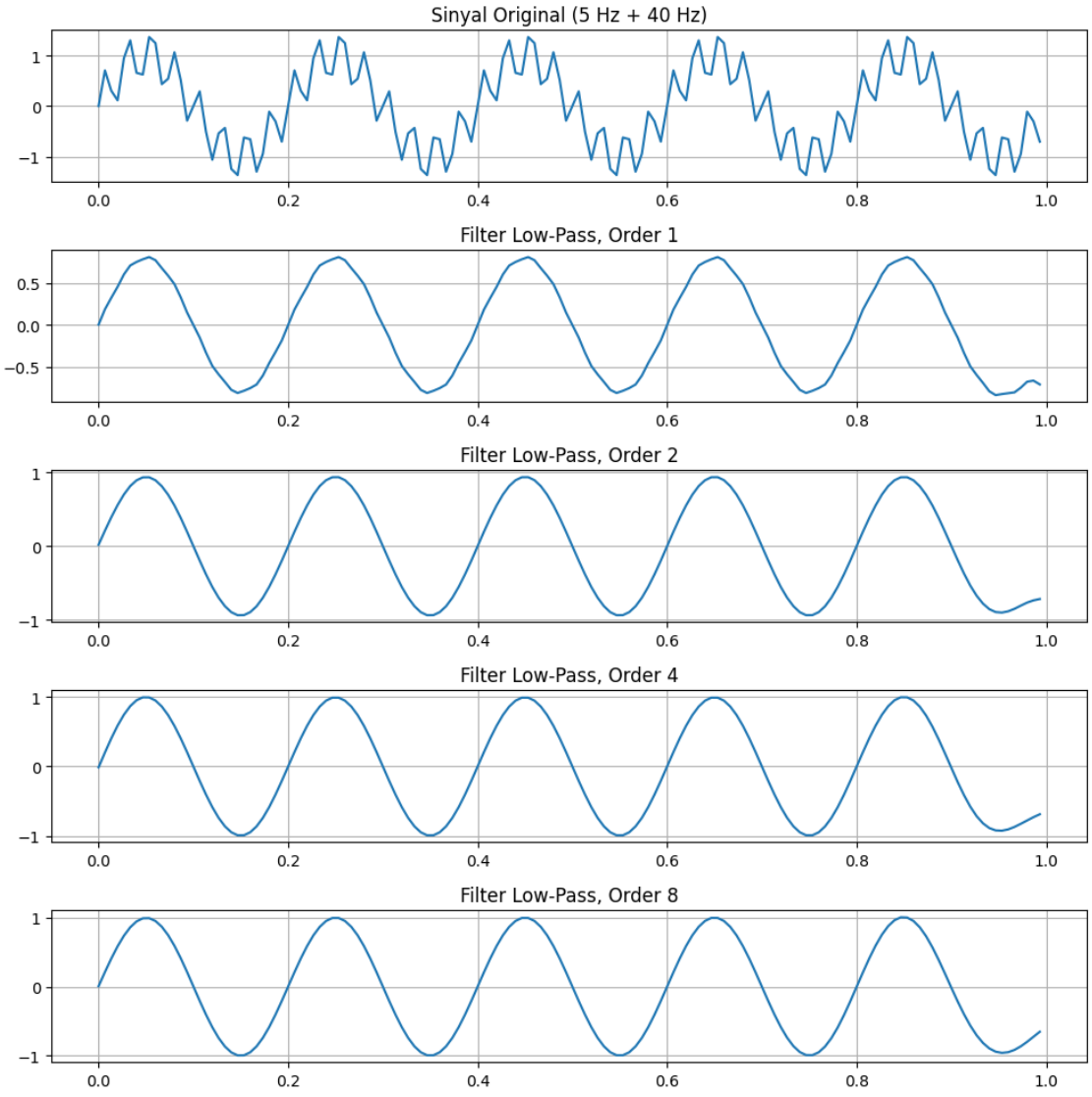


Spektrum Frekuensi Sinyal PPG (100 Hz) Spektrum pada 100 Hz sangat mirip dengan spektrum asli. Semua komponen frekuensi penting masih terjaga dengan baik. Spektrum Frekuensi Sinyal PPG (50 Hz) Pada 50 Hz, spektrum masih sangat mirip dengan aslinya, menunjukkan bahwa tidak ada aliasing yang signifikan. Spektrum Frekuensi Sinyal PPG (25 Hz) Spektrum pada 25 Hz mulai menunjukkan beberapa perubahan kecil, meskipun komponen frekuensi utama masih dipertahankan. Spektrum Frekuensi Sinyal PPG (10 Hz) Pada 10 Hz, spektrum mulai menunjukkan distorsi yang lebih jelas. Beberapa frekuensi palsu (alias) mulai muncul dan mengubah distribusi energi frekuensi. Spektrum Frekuensi Sinyal PPG (5 Hz) Spektrum pada 5 Hz sangat berbeda dari aslinya, menunjukkan aliasing yang signifikan. Frekuensi palsu muncul dan mendominasi spektrum, mengonfirmasi bahwa 5 Hz terlalu rendah untuk sampling sinyal PPG.

[11]:
```
# Membuat sinyal uji (gabungan frekuensi rendah dan tinggi)
t = np.linspace(0, 1, fs_original, endpoint=False)
signal_low = np.sin(2 * np.pi * 5 * t)  # Sinyal 5 Hz
signal_high = 0.5 * np.sin(2 * np.pi * 40 * t)  # Sinyal 40 Hz
combined_signal = signal_low + signal_high
```

[12]:
```
# Membuat filter low-pass dengan berbagai order
cutoff = 10  # Frekuensi cutoff (Hz)
orders = [1, 2, 4, 8]

fig, axs = plt.subplots(len(orders) + 1, 1, figsize=(10, 10))

axs[0].plot(t, combined_signal)
axs[0].set_title("Sinyal Original (5 Hz + 40 Hz)")
axs[0].grid(True)

for i, order in enumerate(orders):
    # Desain filter
    b, a = signal.butter(order, cutoff, fs=fs_original, btype='low')
```

```
    # Aplikasikan filter
    filtered = signal.filtfilt(b, a, combined_signal)

    # Plot
    axs[i + 1].plot(t, filtered)
    axs[i + 1].set_title("Filter Low-Pass, Order {}".format(order))
    axs[i + 1].grid(True)

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



Sinyal Original (5 Hz + 40 Hz) Plot ini menunjukkan sinyal gabungan dari sinyal 5 Hz dan 40

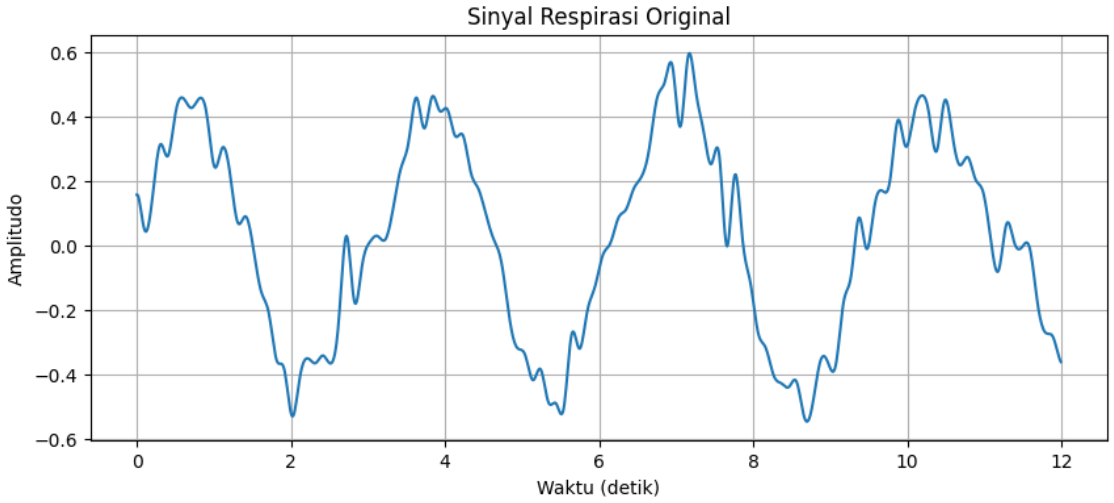
Hz. Pola gelombang kompleks ini akan difilter dengan low-pass filter berbagai order. Filter Low-Pass, Order 1 Dengan order 1, filter memiliki transisi yang sangat gradual antara passband dan stopband. Hasilnya masih mengandung beberapa komponen frekuensi tinggi, menunjukkan bahwa filter belum efektif sepenuhnya. Filter Low-Pass, Order 2 Pada order 2, transisi menjadi lebih tajam. Filter mulai lebih efektif menekan komponen 40 Hz, tetapi masih menyisakan beberapa efek residu. Filter Low-Pass, Order 4 Filter order 4 menunjukkan transisi yang lebih tajam lagi, menghasilkan sinyal yang hampir murni 5 Hz tanpa komponen 40 Hz. Filter Low-Pass, Order 8 Pada order 8, filter sangat efektif dengan transisi yang sangat tajam. Sinyal hasil hampir sempurna hanya terdiri dari komponen 5 Hz tanpa kontaminasi dari frekuensi tinggi.

0.0.2 Soal 2

[13]:
```
# Parameter
duration = 12  # 3 digit terakhir NIM (012)
fs_resp = 100  # Sampling rate (Hz)
noise_level = 0.12  # 2 digit terakhir NIM (12)
resp_rate = 18  # BPM (Breath Per Minute)
random_state = 40229  # Format YYMMDD tanggal lahir
```

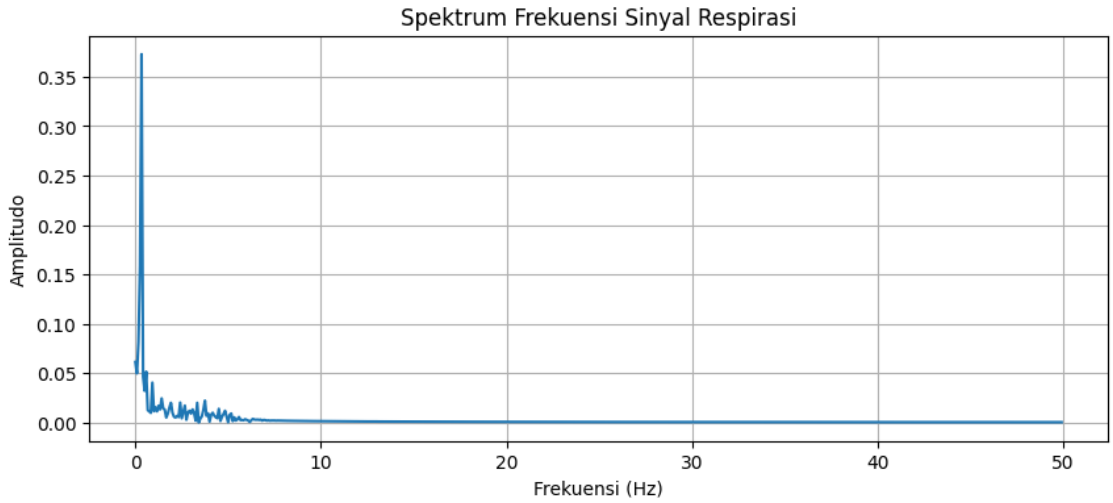
[14]:
```
# Simulasi sinyal respirasi
resp_signal = nk.rsp_simulate(duration=duration,
                              sampling_rate=fs_resp,
                              respiratory_rate=resp_rate,
                              noise=noise_level,
                              random_state=random_state)

# Waktu untuk plot
t_resp = np.linspace(0, duration, len(resp_signal), endpoint=False)
```

[15]:
```
# Visualisasi sinyal respirasi original
plt.figure(figsize=(10, 4))
plt.plot(t_resp, resp_signal)
plt.title("Sinyal Respirasi Original")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.show()
```



Plot ini menampilkan simulasi sinyal pernapasan dengan rate 18 BPM (0.3 Hz). Terlihat pola naik-turun yang mencerminkan siklus inspirasi dan ekspirasi, dengan beberapa noise yang tercampur dalam sinyal.

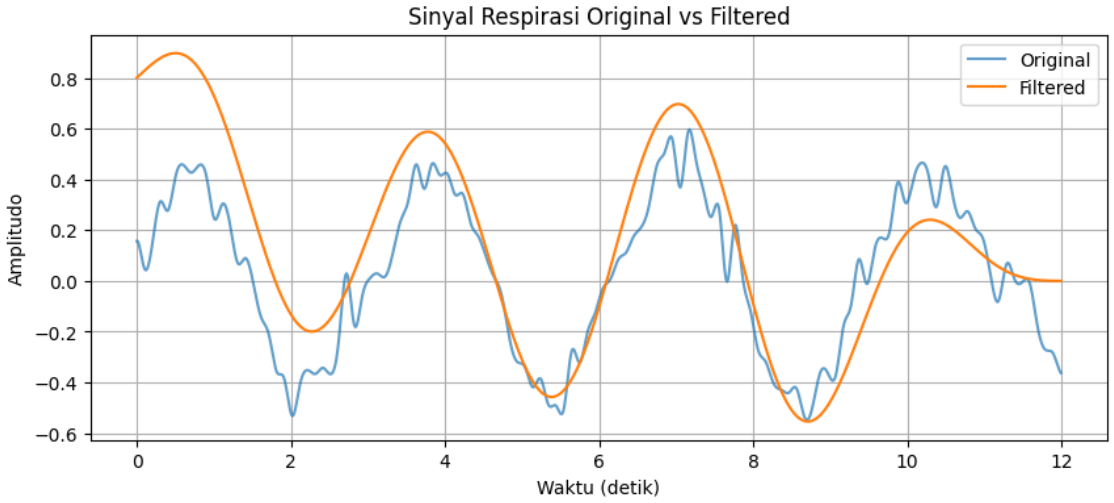
[16]:
```
# Analisis spektrum frekuensi sinyal respirasi untuk menentukan cutoff yang␣
↪tepat
N_resp = len(resp_signal)
yf_resp = fft(resp_signal)
xf_resp = fftfreq(N_resp, 1/fs_resp)

plt.figure(figsize=(10, 4))
plt.plot(xf_resp[:N_resp//2], 2.0/N_resp * np.abs(yf_resp[:N_resp//2]))
plt.title("Spektrum Frekuensi Sinyal Respirasi")
plt.xlabel("Frekuensi (Hz)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.show()
```



Spektrum ini menunjukkan komponen frekuensi dari sinyal respirasi. Puncak utama berada di sekitar 0.3 Hz yang sesuai dengan respiratory rate 18 BPM. Terlihat juga beberapa noise di frekuensi yang lebih tinggi.

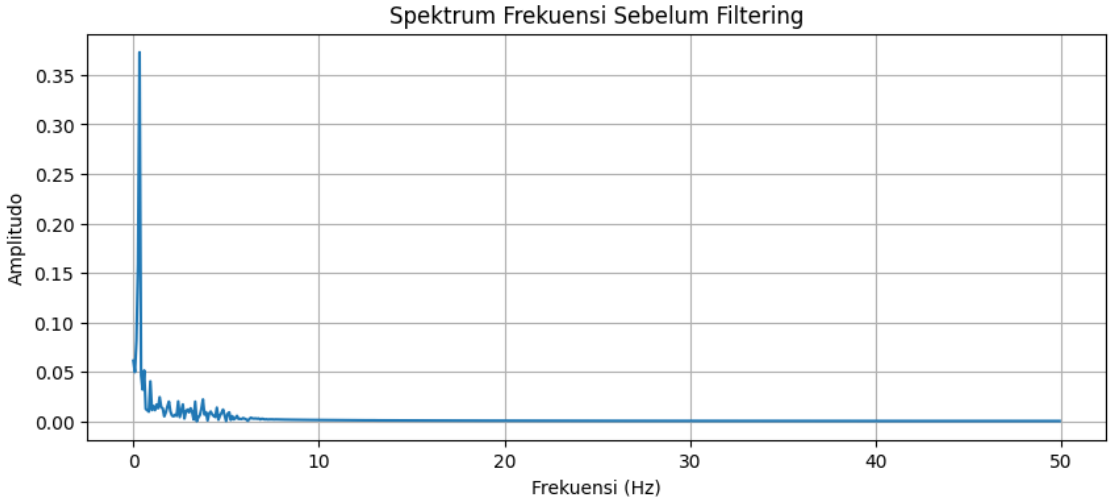
[17]:
```
# Batas bawah dan atas frekuensi untuk band-pass filter (dalam Hz)
low_cutoff = 0.1  # 6 BPM
high_cutoff = 0.5  # 30 BPM
order = 4  # Order filter

# Desain filter band-pass
b, a = signal.butter(order, [low_cutoff, high_cutoff], fs=fs_resp, btype='band')

# Aplikasikan filter
filtered_resp = signal.filtfilt(b, a, resp_signal)
```

[18]:
```
# Visualisasi hasil filter
plt.figure(figsize=(10, 4))
plt.plot(t_resp, resp_signal, alpha=0.7, label="Original")
plt.plot(t_resp, filtered_resp, label="Filtered")
plt.title("Sinyal Respirasi Original vs Filtered")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.legend()
plt.grid(True)
plt.show()
```



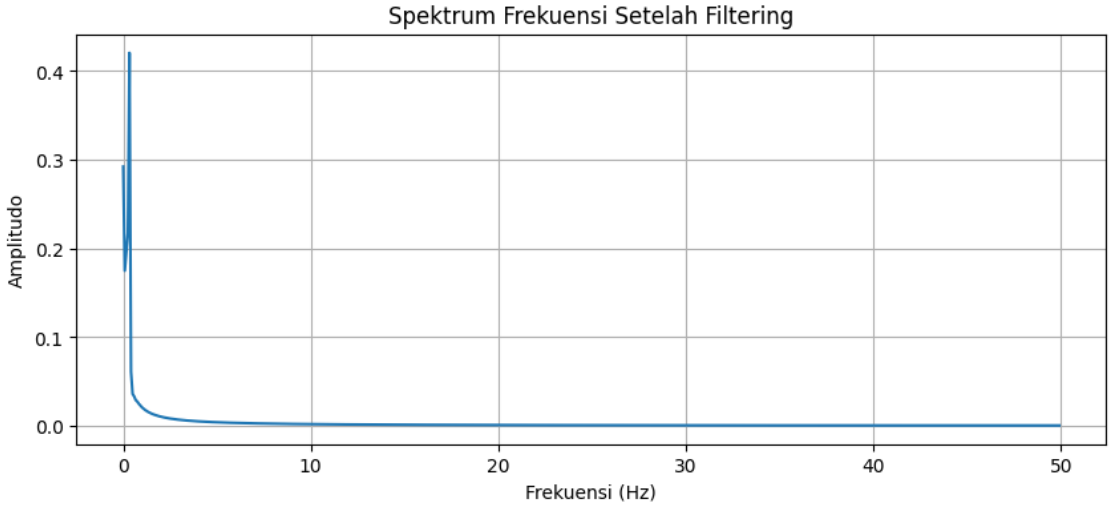
Plot ini membandingkan sinyal respirasi sebelum dan sesudah filtering dengan band-pass 0.1-0.5 Hz. Sinyal hasil filtering (warna oranye) terlihat lebih halus dan bebas dari fluktuasi frekuensi tinggi, menunjukkan bahwa filter berhasil menghilangkan noise sambil mempertahankan pola pernapasan dasar.

[19]:
```
# Bandingkan spektrum frekuensi sebelum dan sesudah filtering
# Spektrum sebelum filtering
plt.figure(figsize=(10, 4))
plt.plot(xf_resp[:N_resp//2], 2.0/N_resp * np.abs(yf_resp[:N_resp//2]))
plt.title("Spektrum Frekuensi Sebelum Filtering")
plt.xlabel("Frekuensi (Hz)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.show()
```



Spektrum ini menunjukkan distribusi frekuensi sinyal respirasi sebelum filtering, dengan komponen frekuensi yang tersebar di berbagai range.

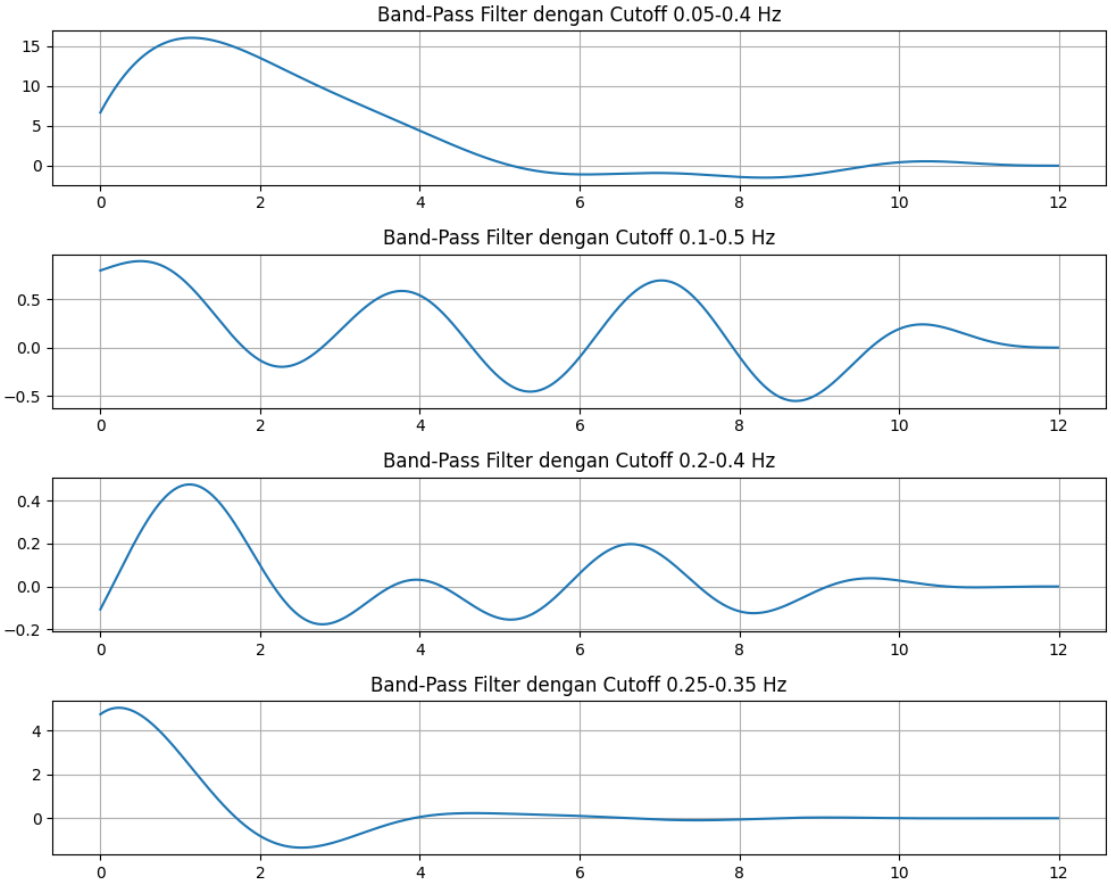
[20]:
```
# Spektrum setelah filtering
N_filtered = len(filtered_resp)
yf_filtered = fft(filtered_resp)
xf_filtered = fftfreq(N_filtered, 1/fs_resp)

plt.figure(figsize=(10, 4))
plt.plot(xf_filtered[:N_filtered//2], 2.0/N_filtered * np.abs(yf_filtered[:
 ↪N_filtered//2]))
plt.title("Spektrum Frekuensi Setelah Filtering")
plt.xlabel("Frekuensi (Hz)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.show()
```



Spektrum setelah filtering menunjukkan bagaimana band-pass filter telah memotong frekuensi di luar range 0.1-0.5 Hz. Puncak di sekitar 0.3 Hz (frekuensi pernapasan) dipertahankan, sementara komponen frekuensi lainnya ditekan.

[21]:
```
# Kita akan mencoba beberapa pasangan nilai cutoff dan membandingkan hasilnya
cutoff_pairs = [
    (0.05, 0.4),  # Lebih lebar
    (0.1, 0.5),   # Standard (yang sudah kita gunakan)
    (0.2, 0.4),   # Lebih sempit
    (0.25, 0.35)  # Sangat sempit
]
```

```
fig, axs = plt.subplots(len(cutoff_pairs), 1, figsize=(10, 8))

for i, (low, high) in enumerate(cutoff_pairs):
    # Desain filter
    b, a = signal.butter(order, [low, high], fs=fs_resp, btype='band')

    # Aplikasikan filter
    filtered = signal.filtfilt(b, a, resp_signal)

    # Plot
    axs[i].plot(t_resp, filtered)
    axs[i].set_title("Band-Pass Filter dengan Cutoff {}-{} Hz".format(low,␣
 ↪high))
    axs[i].grid(True)

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



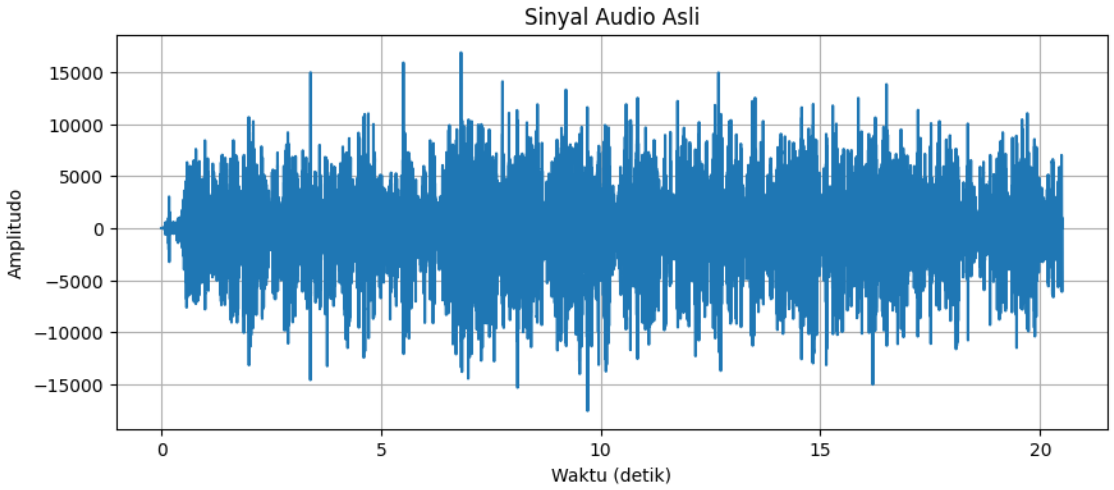
Band-Pass Filter dengan Cutoff 0.05-0.4 Hz Filter ini memiliki rentang yang lebih lebar, memungkinkan lebih banyak komponen frekuensi rendah. Hasil sinyal lebih halus tetapi mungkin menyertakan beberapa noise frekuensi rendah. Band-Pass Filter dengan Cutoff 0.1-0.5 Hz Filter standar ini memberikan keseimbangan yang baik, menangkap frekuensi pernapasan (0.3 Hz) sambil menekan noise frekuensi rendah dan tinggi. Band-Pass Filter dengan Cutoff 0.2-0.4 Hz Filter yang lebih sempit ini lebih agresif dalam menekan noise, tetapi masih mampu menangkap frekuensi pernapasan dengan baik. Band-Pass Filter dengan Cutoff 0.25-0.35 Hz Filter yang sangat sempit ini berfokus sangat spesifik pada frekuensi pernapasan sekitar 0.3 Hz, sangat efektif menekan noise tetapi berisiko menghilangkan beberapa variasi alami dalam pernapasan.

0.1 Soal 3

[23]:
```
# Membaca file audio rekaman
filename = "122140012.wav"
sample_rate, audio_data = wavfile.read(filename)
```

[24]:
```
# Buat array waktu untuk plotting
time = np.linspace(0, len(audio_data) / sample_rate, len(audio_data))
```

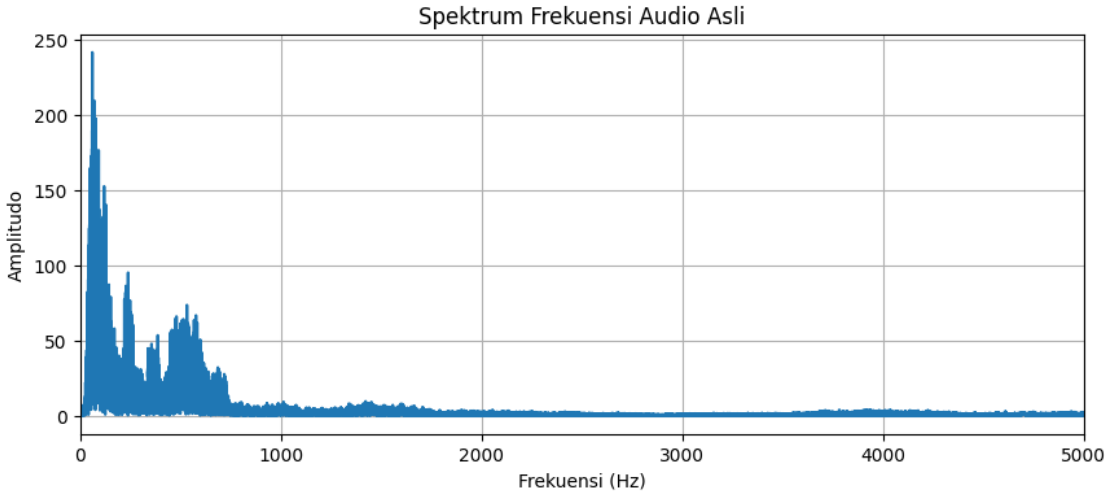
[25]:
```
# Plot sinyal audio asli
plt.figure(figsize=(10, 4))
plt.plot(time, audio_data)
plt.title("Sinyal Audio Asli")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.show()
```



[27]:
```
# Analisis spektrum frekuensi menggunakan FFT
N = len(audio_data)
```

```
yf = fft(audio_data)
xf = fftfreq(N, 1/sample_rate)
```

[ ]:
```
# Plot spektrum frekuensi (sebelum filtering)
plt.figure(figsize=(10, 4))
plt.plot(xf[:N//2], 2.0/N * np.abs(yf[:N//2]))
plt.title("Spektrum Frekuensi Audio Asli")
plt.xlabel("Frekuensi (Hz)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.xlim(0, 5000)
plt.show()
```



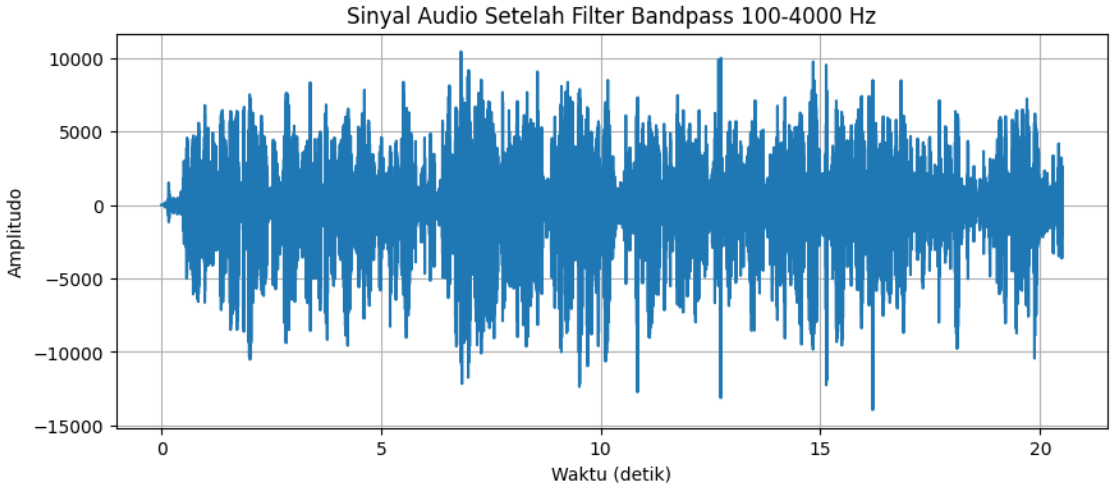
[38]:
```
# Buat tiga filter bandpass dengan frekuensi cutoff berbeda
order = 4  # Order filter
```

[39]:
```
# Filter 1: Bandpass dengan cutoff lebar (100-4000 Hz)
low_cutoff1 = 100
high_cutoff1 = 4000
b1, a1 = signal.butter(order, [low_cutoff1, high_cutoff1], fs=sample_rate,␣
 ↪btype='band')
filtered_audio1 = signal.filtfilt(b1, a1, audio_data)
```

[40]:
```
# Filter 2: Bandpass dengan cutoff standar telepon (300-3400 Hz)
low_cutoff2 = 300
high_cutoff2 = 3400
b2, a2 = signal.butter(order, [low_cutoff2, high_cutoff2], fs=sample_rate,␣
 ↪btype='band')
filtered_audio2 = signal.filtfilt(b2, a2, audio_data)
```

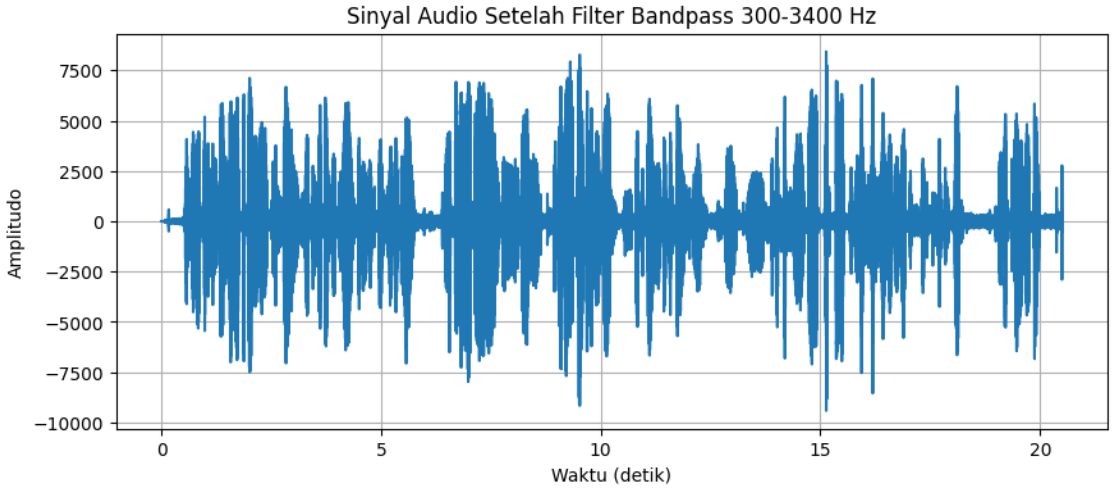
[41]:
```
# Filter 3: Bandpass dengan cutoff sempit (500-2500 Hz)
low_cutoff3 = 500
high_cutoff3 = 2500
b3, a3 = signal.butter(order, [low_cutoff3, high_cutoff3], fs=sample_rate,␣
↪btype='band')
filtered_audio3 = signal.filtfilt(b3, a3, audio_data)
```

[42]:
```
# Plot sinyal hasil filtering
plt.figure(figsize=(10, 4))
plt.plot(time, filtered_audio1)
plt.title(f"Sinyal Audio Setelah Filter Bandpass {low_cutoff1}-{high_cutoff1}␣
↪Hz")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.show()
```



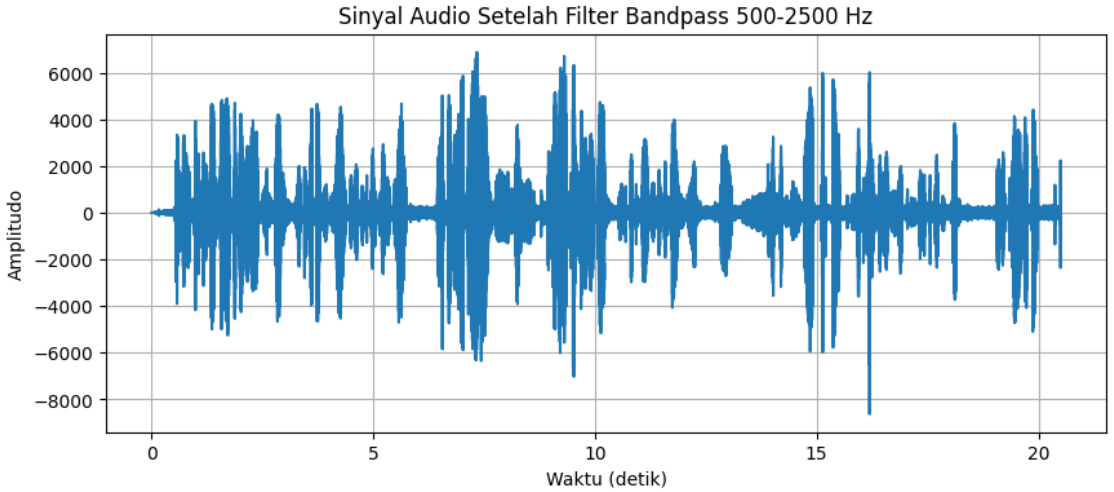
Filter lebar ini mempertahankan sebagian besar komponen suara manusia, tetapi juga masih menyisakan beberapa noise. Suara terdengar natural tetapi mungkin masih mengandung noise latar belakang.

[43]:
```
plt.figure(figsize=(10, 4))
plt.plot(time, filtered_audio2)
plt.title(f"Sinyal Audio Setelah Filter Bandpass {low_cutoff2}-{high_cutoff2}␣
↪Hz")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.show()
```



Filter ini, yang merupakan standar untuk sistem telepon, memberikan keseimbangan yang baik antara kejernihan suara dan penekanan noise. Komponen suara utama dipertahankan dengan baik.

[44]:
```
plt.figure(figsize=(10, 4))
plt.plot(time, filtered_audio3)
plt.title(f"Sinyal Audio Setelah Filter Bandpass {low_cutoff3}-{high_cutoff3}␣
↪Hz")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.show()
```

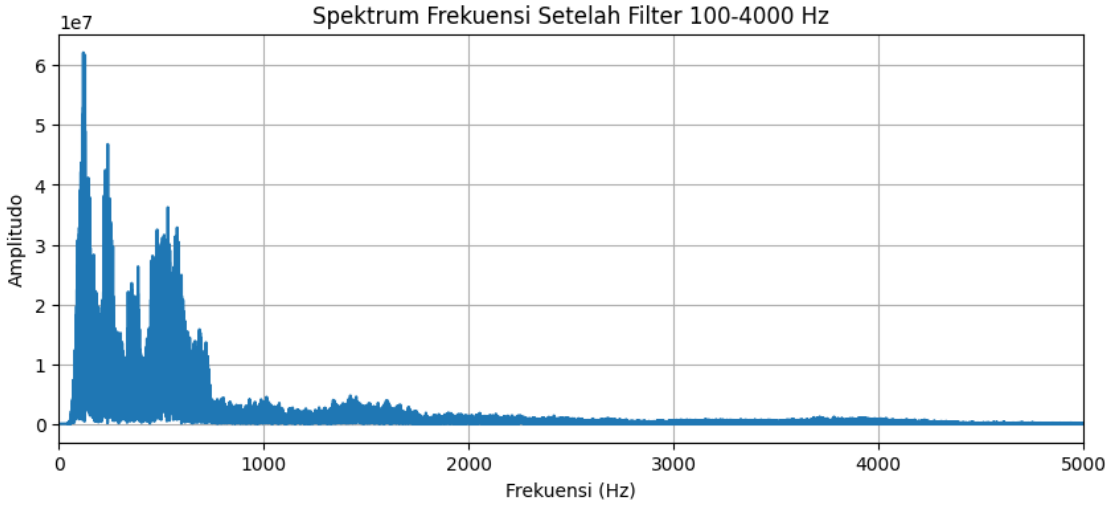


Filter sempit ini lebih agresif dalam menghilangkan noise, tetapi juga menghilangkan beberapa

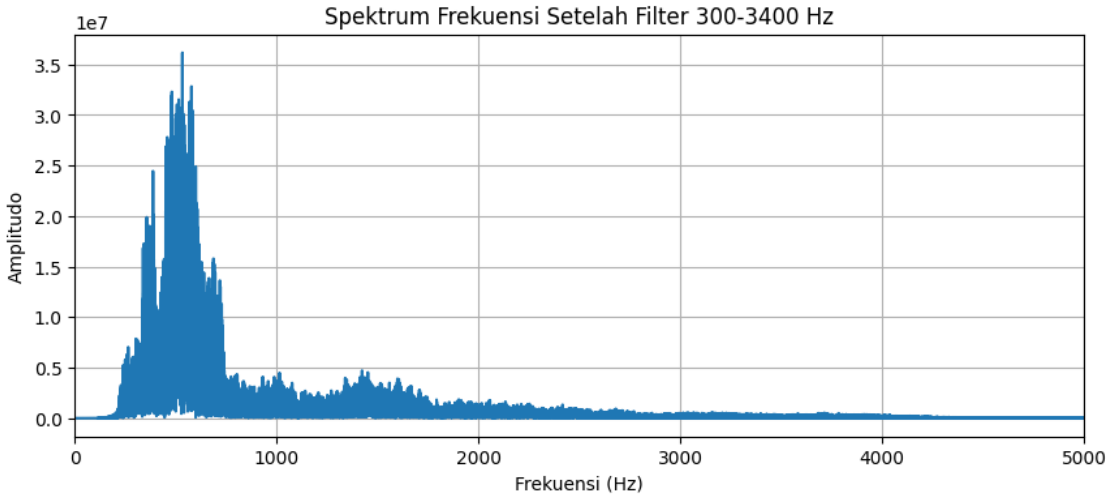
detail suara, khususnya frekuensi rendah dan tinggi yang berkontribusi pada kekayaan suara.

[45]:
```
# Analisis spektrum frekuensi untuk ketiga filter
yf1 = fft(filtered_audio1)
yf2 = fft(filtered_audio2)
yf3 = fft(filtered_audio3)
```

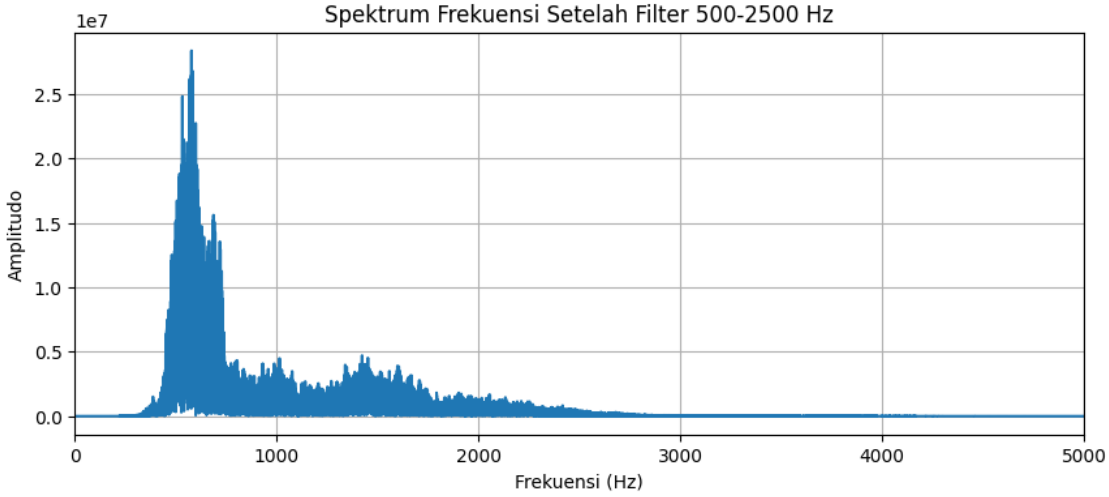
[46]:
```
# Plot spektrum frekuensi setelah filter 1
plt.figure(figsize=(10, 4))
plt.plot(xf[:N//2], np.abs(yf1[:N//2]))
plt.title(f"Spektrum Frekuensi Setelah Filter {low_cutoff1}-{high_cutoff1} Hz")
plt.xlabel("Frekuensi (Hz)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.xlim(0, 5000)
plt.show()
```



[47]:
```
# Plot spektrum frekuensi setelah filter 2
plt.figure(figsize=(10, 4))
plt.plot(xf[:N//2], np.abs(yf2[:N//2]))
plt.title(f"Spektrum Frekuensi Setelah Filter {low_cutoff2}-{high_cutoff2} Hz")
plt.xlabel("Frekuensi (Hz)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.xlim(0, 5000)
plt.show()
```



[48]:
```
# Plot spektrum frekuensi setelah filter 3
plt.figure(figsize=(10, 4))
plt.plot(xf[:N//2], np.abs(yf3[:N//2]))
plt.title(f"Spektrum Frekuensi Setelah Filter {low_cutoff3}-{high_cutoff3} Hz")
plt.xlabel("Frekuensi (Hz)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.xlim(0, 5000)
plt.show()
```



Spektrum-spektrum ini dengan jelas menunjukkan bagaimana masing-masing filter memotong komponen frekuensi di luar range yang ditentukan. Filter 300-3400 Hz tampak memberikan hasil terbaik karena mempertahankan mayoritas energi suara percakapan manusia (yang umumnya berada di

range ini) sambil menghilangkan noise frekuensi rendah dan tinggi.

0.2 Lampiran

LLM