



Cover

X +

← → Q Sains Data - Universitas Negeri Surabaya - Pengolahan Sinyal Digital

# Remove Seismic Noise

**Kelompok 7 - sains data 2022A**



Cover

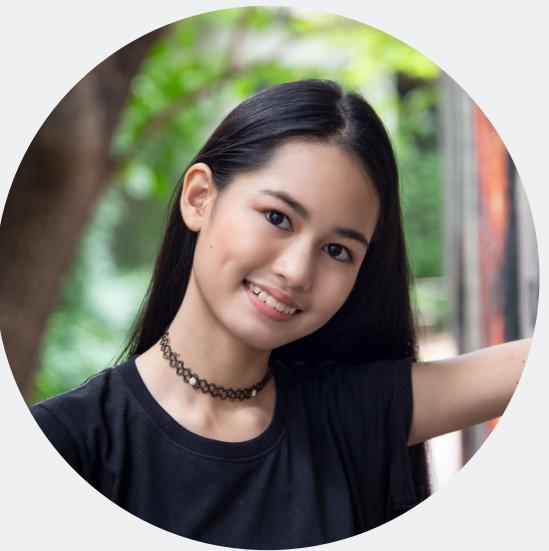
Nama kelompok



← → Q #SayHItogroup7

# Kelompok 7

MEMBERS



Fadhilah Nuria Shinta  
22031554003



Daffa Fazly Rashidhan  
22031554006



Dhani Aditya Putra E.  
22031554038

Cover

Anggota

Latar Belakang

X

+

← → Q Insert your topic here

# Latar Belakang

Dalam konteks proyek penghapusan noise untuk meningkatkan kejernihan suara dan mengurangi gangguan dalam pemutaran suara seismic, penghapusan noise suara laut dapat menjadi langkah yang penting. Suara laut, seperti ombak, burung laut, atau suara hewan laut lainnya, bisa menjadi sumber noise yang signifikan dalam rekaman audio seismik di perairan. Dalam rekaman audio, penghapusan noise membantu meningkatkan kejernihan suara dan mengurangi gangguan dalam pemutaran suara seismic.



← → Q Insert your topic here

# Masalah



1

Metode atau teknik yang digunakan untuk menghapus dan memfilter noise pada sinyal seismic



2

Perbandingan sinyal suara asli dengan sinyal suara yang noisenya dihapus



3

Perbandingan visualisasi sinyal seismic pada beberapa audio (normal audio, noise audio, dan reduced audio)

← → Q Insert your topic here

# Urgensi



Alasan penghapusan noise dari suara seismic memiliki beberapa urgensi:

- Dalam analisis dan interpretasi data seismic, kualitas sinyal yang akurat dan jernih sangat penting. Noise dalam sinyal seismic dapat mengaburkan informasi yang diinginkan, sehingga mengurangi kejelasan dan keakuratan data. Dengan menghapus atau mengurangi noise, kualitas data seismic dapat ditingkatkan, dan memungkinkan interpretasi yang lebih baik dan hasil analisis yang lebih akurat.
- Noise pada sinyal seismic dapat berasal dari sumber yang tidak diinginkan, seperti getaran buatan manusia, lalu lintas, atau suara alam, termasuk suara air laut atau suara angin. Penghapusan noise memungkinkan identifikasi dan isolasi sumber seismic yang sebenarnya

Masalah

Urgensi

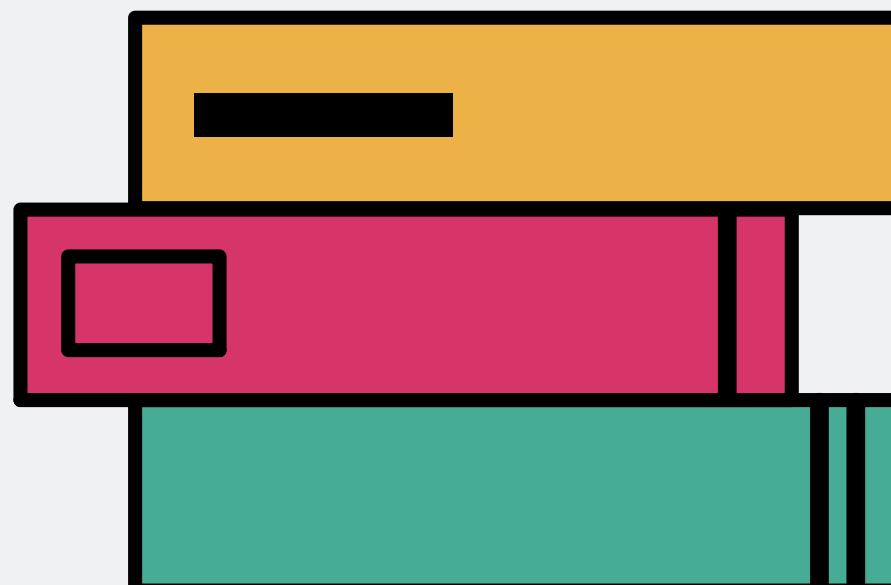
Tujuan

X

+

← → Q Insert your topic here

# Tujuan



**Tujuan penghapusan noise pada sinyal seismic adalah**

- 1.Untuk meningkatkan kualitas suara
- 2.Meningkatkan kualitas dan kejelasan data seismik yang diperoleh

Urgensi

Tujuan

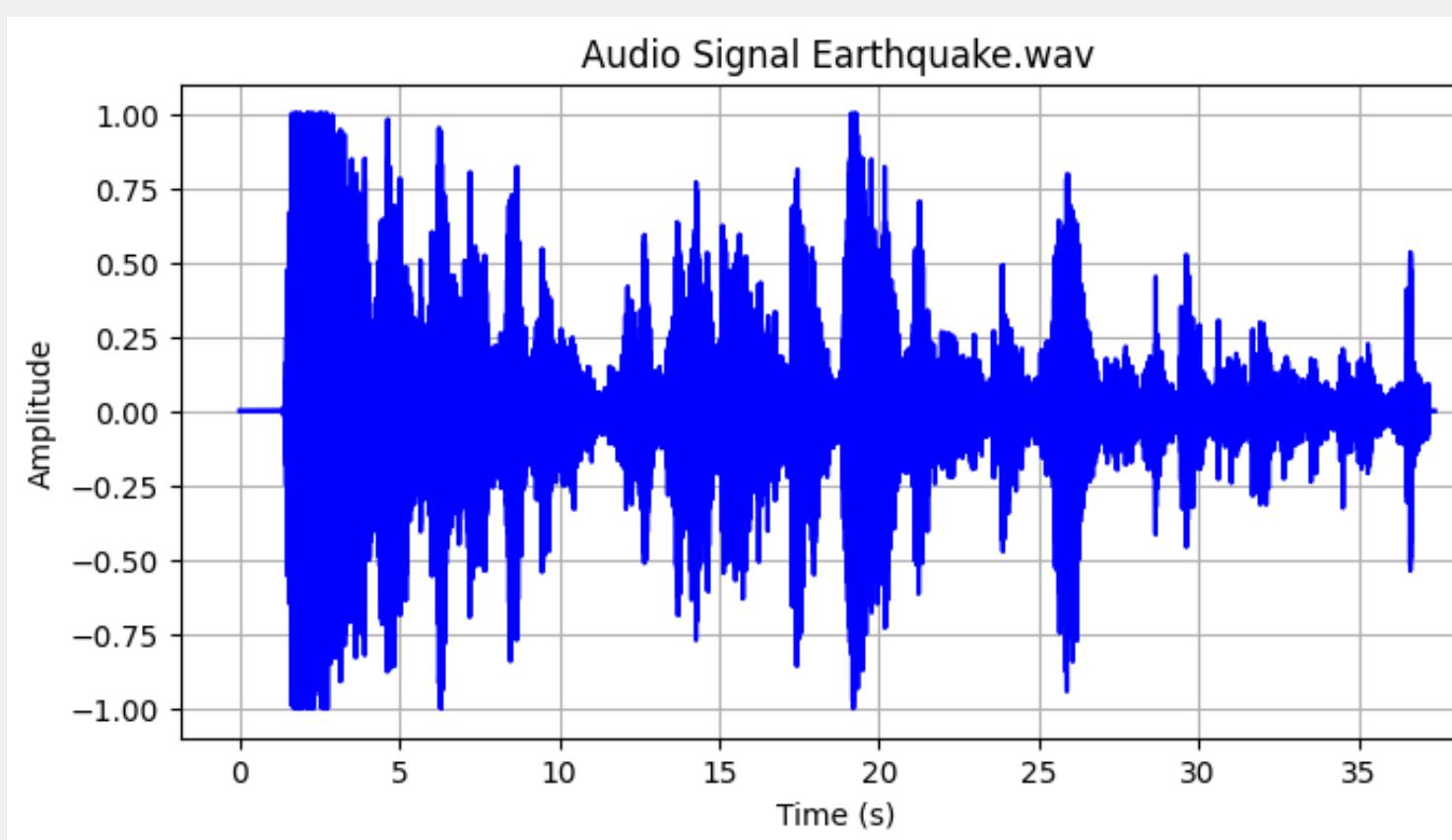
Hasil

X

+

← → Q konversi gelombang suara

# Hasil konversi suara asli mp3 ke wav file



```
import matplotlib.pyplot as plt # Mengimpor modul pyplot dari pustaka Matplotlib untuk membuat plot.
import numpy as np # Mengimpor modul numpy untuk operasi numerik.
from pydub import AudioSegment # Mengimpor kelas AudioSegment dari pustaka PyDub untuk membaca file audio.
import scipy.io.wavfile as wav # Mengimpor modul wavfile dari pustaka SciPy untuk membaca dan menulis file WAV.

# Membaca file audio dengan nama "Earthquake.mp3" menggunakan AudioSegment dari PyDub.
audio = AudioSegment.from_file("Earthquake.mp3", format="mp3")

# Simpan sebagai file WAV sementara
wav_file = "Earthquake.wav" # Menyimpan nama file WAV sementara yang akan dihasilkan.
audio.export(wav_file, format="wav") # Menyimpan file audio dalam format WAV menggunakan export dari PyDub.

# Membaca file audio WAV menggunakan read dari wavfile dari SciPy
sampling_rate, audio_data = wav.read(wav_file)

# Normalisasi sinyal audio
audio_data = audio_data / np.max(np.abs(audio_data))

# Menghitung durasi audio dalam detik dengan membagi jumlah sampel audio (len(audio_data)) dengan tingkat sampel
duration = len(audio_data) / sampling_rate

# Buat sumbu waktu untuk plot dengan rentang waktu dari 0 hingga durasi audio.
time = np.linspace(0., duration, len(audio_data))

# Plot sinyal audio
plt.figure(figsize=(8, 4)) # Membuat objek gambar (figure) dengan ukuran 8x4 inci
plt.plot(time, audio_data, color='b') # Membuat plot dari sinyal audio dengan menggunakan plt.plot(). Sumur waktu
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Audio Signal Earthquake.wav')
plt.grid(True) # Menampilkan garis-garis grid pada plot
plt.show()
```

Urgensi

Tujuan

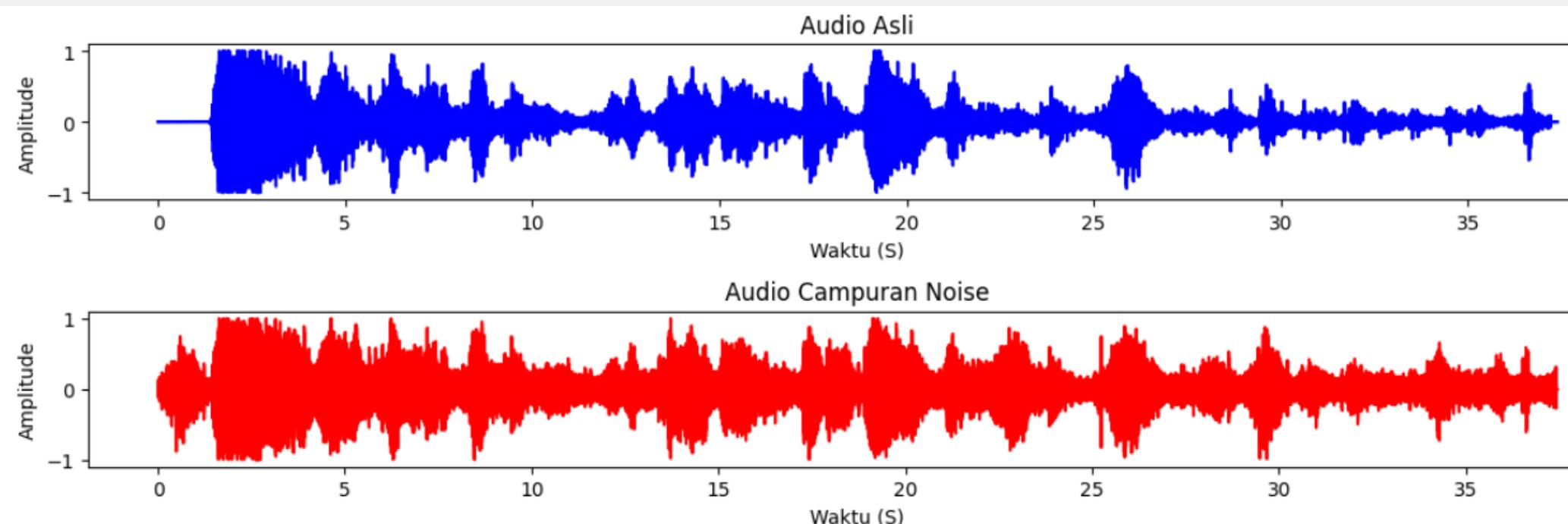
Hasil

X

+

← → Q Gelombang audio asli dengan noise dan tanpa noise

# Visualisasi hasil perbandingan suara asli dengan suara campuran noise



```
import numpy as np
from pydub import AudioSegment # Mengimpor kelas AudioSegment dari pustaka PyDub untuk membaca file audio.
import scipy.io.wavfile as wav # Mengimpor modul wavfile dari pustaka SciPy untuk membaca dan menulis file WAV.
import matplotlib.pyplot as plt # Untuk membuat plot dari data audio.

# Menyimpan nama file WAV sementara yang akan dihasilkan.
wav_file = "campuran.WAV"
# Membaca file audio WAV menggunakan read dari wavfile dari SciPy
sampling_rate, audio_data = wav.read(wav_file)
# Normalisasi sinyal audio
audio_visual = audio_data / np.max(np.abs(audio_data))
# Menghitung durasi audio dalam detik dengan membagi jumlah sampel audio (len(audio_data)) dengan tingkat sampel (sampling_rate).
duration = len(audio_data) / sampling_rate
# Buat sumbu waktu untuk plot dengan rentang waktu dari 0 hingga durasi audio.
time = np.linspace(0., duration, len(audio_data))

# Menampilkan sinyal audio Campuran
plt.figure(figsize=(12, 4))
plt.subplot(2, 1, 2)
plt.title('Audio Campuran Noise')
plt.plot(time, audio_visual, color='r')
plt.xlabel('Waktu (S)')
plt.ylabel('Amplitude')
plt.tight_layout()

# Menyimpan nama file WAV sementara yang akan dihasilkan.
wav_file = "Earthquake.wav"
# Membaca file audio WAV menggunakan read dari wavfile dari SciPy
sampling_rate, audio_data = wav.read(wav_file)
# Normalisasi sinyal audio
audio_visual = audio_data / np.max(np.abs(audio_data))
# Menghitung durasi audio dalam detik dengan membagi jumlah sampel audio (len(audio_data)) dengan tingkat sampel (sampling_rate).
duration = len(audio_data) / sampling_rate
# Buat sumbu waktu untuk plot dengan rentang waktu dari 0 hingga durasi audio.
time = np.linspace(0., duration, len(audio_data))

# Menampilkan sinyal audio Asli
plt.subplot(2, 1, 1)
plt.title('Audio Asli')
plt.plot(time, audio_visual, color='b')
plt.xlabel('Waktu (S)')
plt.ylabel('Amplitude')
plt.tight_layout()

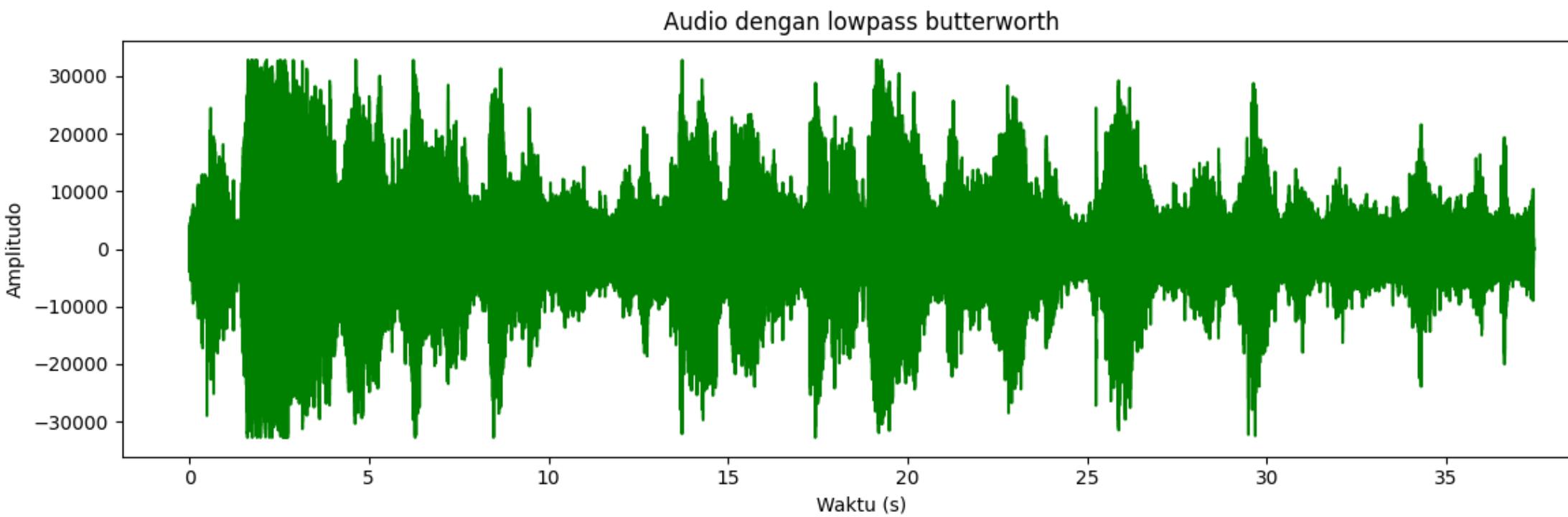
# Menampilkan plot
plt.show()
```

Python

← → Q visualisasi sinyal suara dengan lowpass butterworth

# Visualisasi sinyal audio dengan lowpass butterworth

filter lowpass Butterworth dapat digunakan untuk mengurangi noise atau sinyal gangguan dengan memotong komponen frekuensi tinggi yang tidak diinginkan.



```
import numpy as np # untuk manipulasi data numerik.
import scipy.signal as signal # untuk pemrosesan sinyal.
import matplotlib.pyplot as plt

# Membaca file "campuran"
sample_rate, audio_data = wav.read("campuran.WAV")

# Menentukan frekuensi yang akan dipotong (3000 Hz)
cutoff_freq = 3000

# Menghitung nilai normalisasi cutoff
normalized_cutoff = cutoff_freq / (0.5 * sample_rate)

# Membangun filter lowpass Butterworth dengan parameter b dan a
b, a = signal.butter(4, normalized_cutoff, btype='low', analog=False, output='ba')

# Menerapkan filter ke audio
filtered = signal.lfilter(b, a, audio_data) # Fungsi ini menerapkan filter FIR (Finite Impulse Response)

# Menyimpan file yang sudah difilter
wav.write("reduce_low.wav", sample_rate, np.int16(filtered)) # sample_rate menyimpan nilai sample rate dari file awal. np.int16(filtered) menyimpan hasil pemrosesan filter dari sinyal audio yang telah diolah.

# Menghitung waktu untuk visualisasi
time = np.arange(0, len(audio_data)) / sample_rate

# Menvisualisasikan audio setelah mengurangi noise
plt.figure(figsize=(12, 4))
plt.title('Audio dengan lowpass butterworth')
plt.ylabel('Amplitudo')
plt.xlabel('Waktu (s)')
plt.plot(time, audio_data, color='g')
plt.tight_layout()

# Menampilkan plot
plt.show()
```



Title Page

Reporters

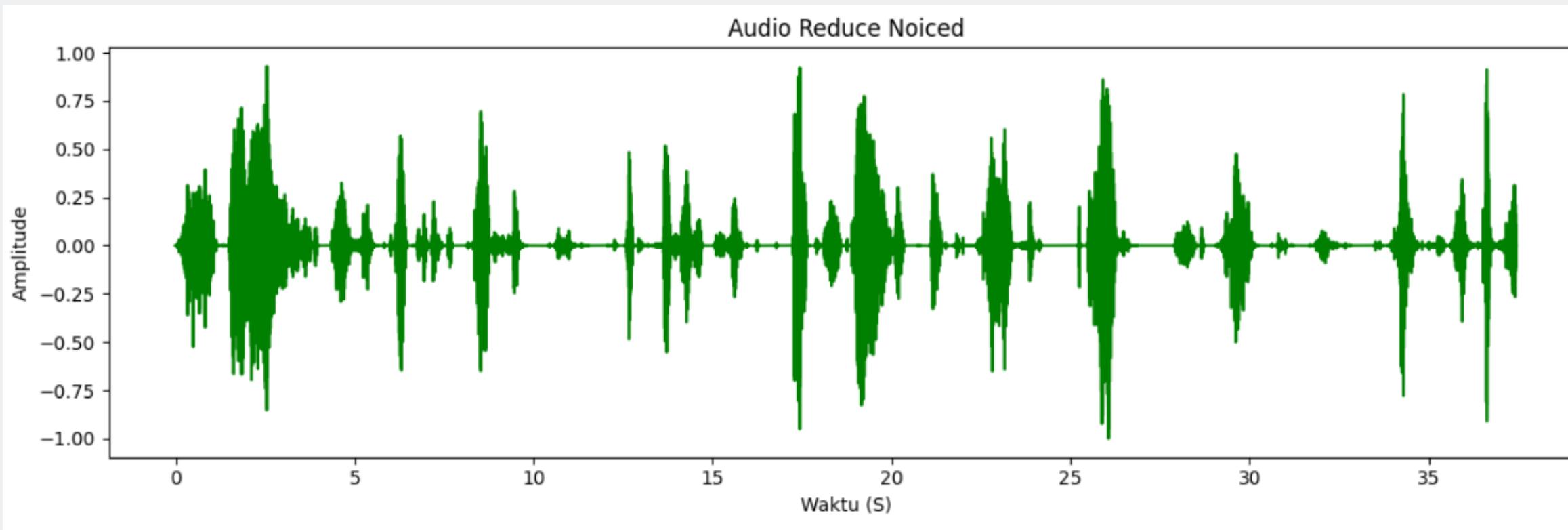
Hasil

X

+

← → Q visualisasi sinyal suara yang noisenya sudah dihapus

# Visualisasi sinyal suara reduce noiced



```
import os #untuk berinteraksi dengan sistem operasi.
import librosa #untuk memuat dan memproses file audio.
import noisereduce #untuk memuat dan memproses file audio.
import matplotlib.pyplot as plt #untuk membuat plot dari data audio.
import soundfile as sf #untuk menyimpan file audio yang sudah diolah.
import numpy as np
from pydub import AudioSegment # Mengimpor kelas AudioSegment dari pustaka PyDub untuk membaca file audio.
import scipy.io.wavfile as wav # Mengimpor modul wavfile dari pustaka SciPy untuk membaca dan menulis file WAV.

# Mendapatkan jalur Lengkap ke file audio untuk dioprasikan
audio_filename = 'campuran.WAV'
audio_path = os.path.join(os.getcwd(), audio_filename)
# Memuat file audio
audio_data, sr = librosa.load(audio_path)
# Ekstraksi noise dari audio
# Menghitung spektrum frekuensi
noise_clip = audio_data[:80000]
# Mengurangi noise dari audio
reduced_noise = noisereduce.reduce_noise(y=audio_data,y_noise=noise_clip, sr=sr) #y=audio_data adalah data audio
# Menyimpan audio yang sudah tereduksi
output_path = 'audio_reduced.wav' # Ganti dengan path untuk menyimpan audio hasil
sf.write(output_path, reduced_noise, sr, format='wav')

# Menyimpan nama file WAV sementara yang akan dihasilkan.
wav_file = "audio_reduced.wav"
# Membaca file audio WAV menggunakan read dari wavfile dari SciPy
sampling_rate, audio_data = wav.read(wav_file)
# Normalisasi sinyal audio
audio_visual = audio_data / np.max(np.abs(audio_data))
# Menghitung durasi audio dalam detik dengan membagi jumlah sampel audio (len(audio_data)) dengan tingkat sampel
duration = len(audio_data) / sampling_rate
# Buat sumbu waktu untuk plot dengan rentang waktu dari 0 hingga durasi audio.
time = np.linspace(0., duration, len(audio_data))

# Menvisualisasikan audio-reduce noise
plt.figure(figsize=(12, 4))
plt.title('Audio Reduce Noiced')
plt.ylabel('Amplitude')
plt.xlabel('Waktu (S)')
plt.plot(time, audio_visual,color='g')
plt.tight_layout()

# Menampilkan plot
plt.show()
```



Title Page

Reporters

Hasil

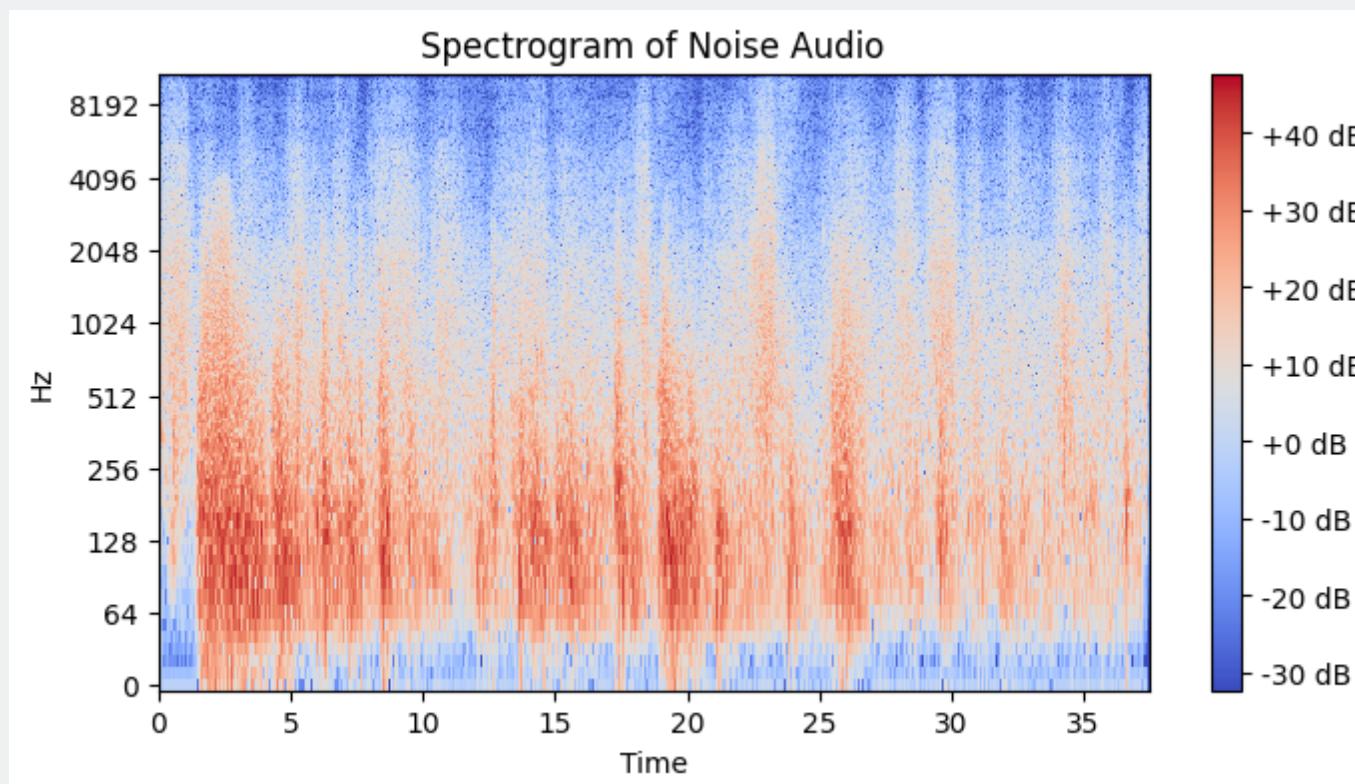
X

+

← → Q Spektogram dari noise audio

# Visualisasi spectogram dengan noise audio

input : sinyal suara 'campuran.wav'



```
import librosa # untuk analisis audio.  
import librosa.display # Mengimpor modul display dari librosa untuk menampilkan spektrogram.  
import matplotlib.pyplot as plt # Mengimpor modul pyplot dari Matplotlib untuk membuat plot.  
  
audio_file = 'campuran.WAV' # Menyimpan nama file audio yang akan dimuat.  
y, sr = librosa.load(audio_file) # Memuat file audio menggunakan librosa.load(). Hasilnya adalah sinyal audi  
  
hop_length = 512 # hop length = jumlah sampel yang digunakan untuk pergeseran jendela saat menghitung spektr  
  
spectrogram = librosa.stft(y, hop_length=hop_length) # Spektrogram dihitung dengan menggunakan Short-Time Fou  
spectrogram_db = librosa.amplitude_to_db(abs(spectrogram)) # Mengubah amplitudo spektrogram menjadi skala Log  
  
plt.figure(figsize=(8, 4)) # Mengubah amplitudo spektrogram menjadi skala logaritmik dalam desibel  

```

Python



Title Page

Reporters

Hasil

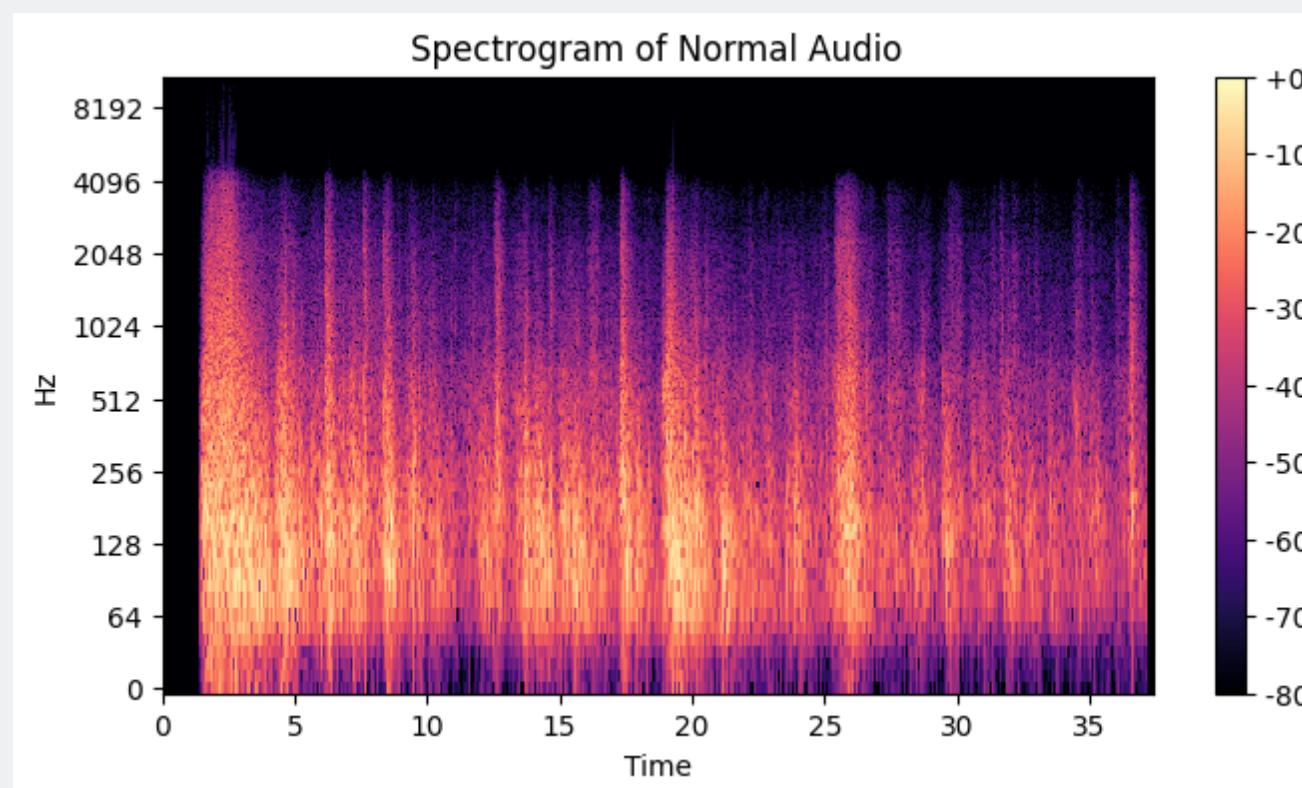
X

+

← → Q Spektogram audio normal

# Visualisasi spectrogram suara asli (normal audio)

input : sinyal suara 'Earthquake.wav'



```
import librosa # untuk analisis audio.
import numpy as np # untuk operasi numerik.
import matplotlib.pyplot as plt # untuk membuat plot.

audio_file = 'Earthquake.wav'
audio, sr = librosa.load(audio_file) # Memuat file audio menggunakan librosa.load(). Hasilnya adalah sinyal aud

n_fft = 2048 # Menentukan jumlah titik FFT yang digunakan dalam perhitungan spektrogram.
hop_length = 512 # Menentukan jarak pergeseran antara frame dalam perhitungan spektrogram.
win_length = 2048 # Menentukan ukuran jendela yang digunakan dalam perhitungan spektrogram.
spectrogram = librosa.stft(audio, n_fft=n_fft, hop_length=hop_length, win_length=win_length) # Argumen n_fft, h
spectrogram_db = librosa.amplitude_to_db(np.abs(spectrogram), ref=np.max) # Mengubah amplitudo spektrogram menj

plt.figure(figsize=(8, 4)) # Membuat objek gambar (figure) dengan ukuran 8x4 inci
librosa.display.specshow(spectrogram_db, sr=sr, hop_length=hop_length, x_axis='time', y_axis='log')
plt.colorbar(format='%.2f dB') # Menampilkan colorbar pada plot spektrogram
plt.title('Spectrogram of Normal Audio')
plt.show()
```



Title Page

Reporters

Hasil

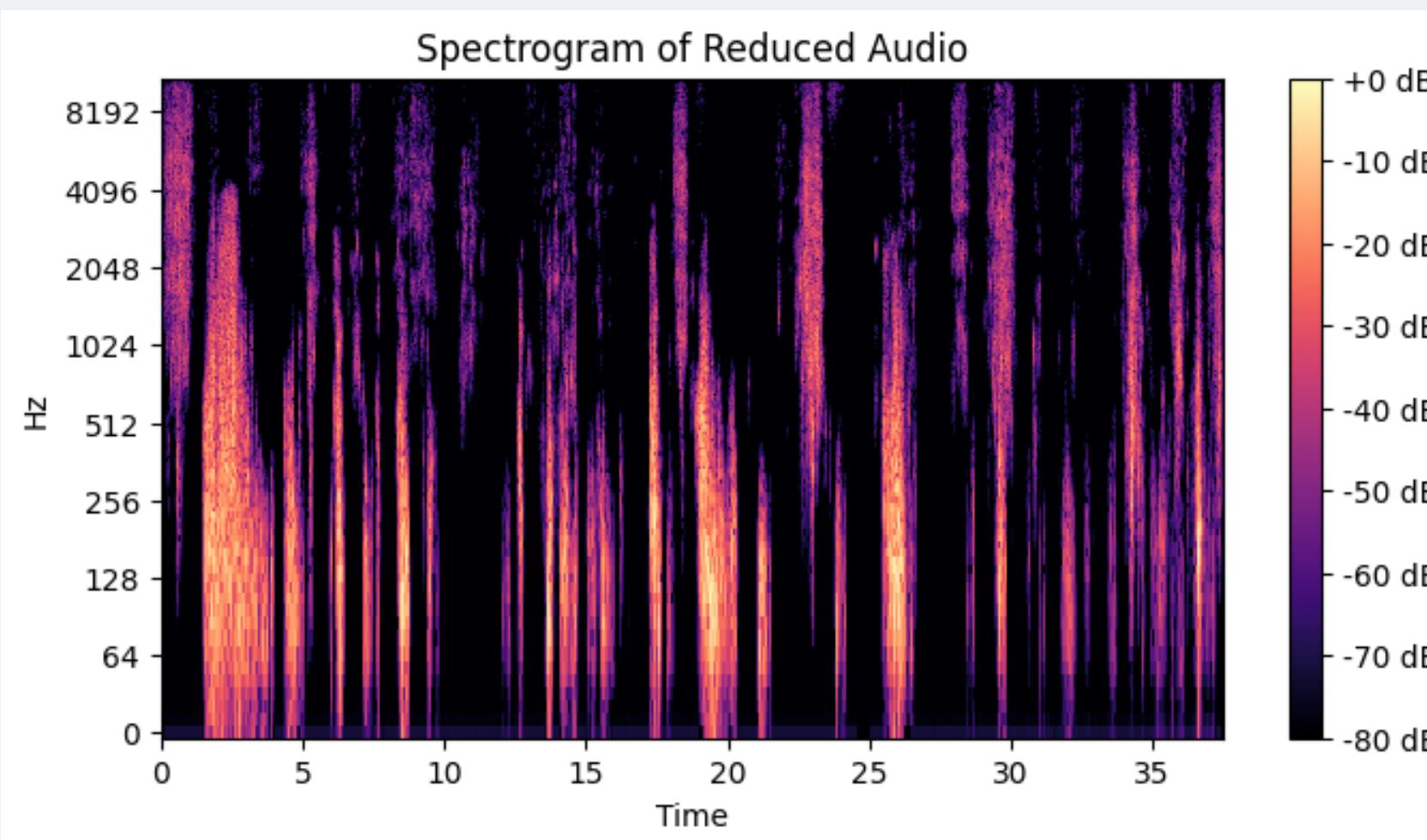
X

+

← → Q Spektogram dari audio yang telah di reduce

# Visualisasi spectogram suara yang noisenya sudah dihilangkan (reduced audio)

input : sinyal suara 'audio\_reduced.wav'



```
import librosa # untuk analisis audio.
import numpy as np # untuk operasi numerik.
import matplotlib.pyplot as plt # untuk membuat plot.

audio_file = 'audio_reduced.wav'
audio, sr = librosa.load(audio_file) # Memuat file audio menggunakan librosa.load(). Hasilnya adalah sinyal aud

n_fft = 2048 # Menentukan jumlah titik FFT yang digunakan dalam perhitungan spektrogram.
hop_length = 512 # Menentukan jarak pergeseran antara frame dalam perhitungan spektrogram.
win_length = 2048 # Menentukan ukuran jendela yang digunakan dalam perhitungan spektrogram.
spectrogram = librosa.stft(audio, n_fft=n_fft, hop_length=hop_length, win_length=win_length) # Argumen n_fft, h
spectrogram_db = librosa.amplitude_to_db(np.abs(spectrogram), ref=np.max) # Mengubah amplitudo spektrogram menj

plt.figure(figsize=(8, 4)) # Membuat objek gambar (figure) dengan ukuran 8x4 inci
librosa.display.specshow(spectrogram_db, sr=sr, hop_length=hop_length, x_axis='time', y_axis='log')
plt.colorbar(format='%+2.0f dB') # Menampilkan colorbar pada plot spektrogram
plt.title('Spectrogram of Reduced Audio')
plt.show()
```

Python



Title Page

Reporters

Hasil

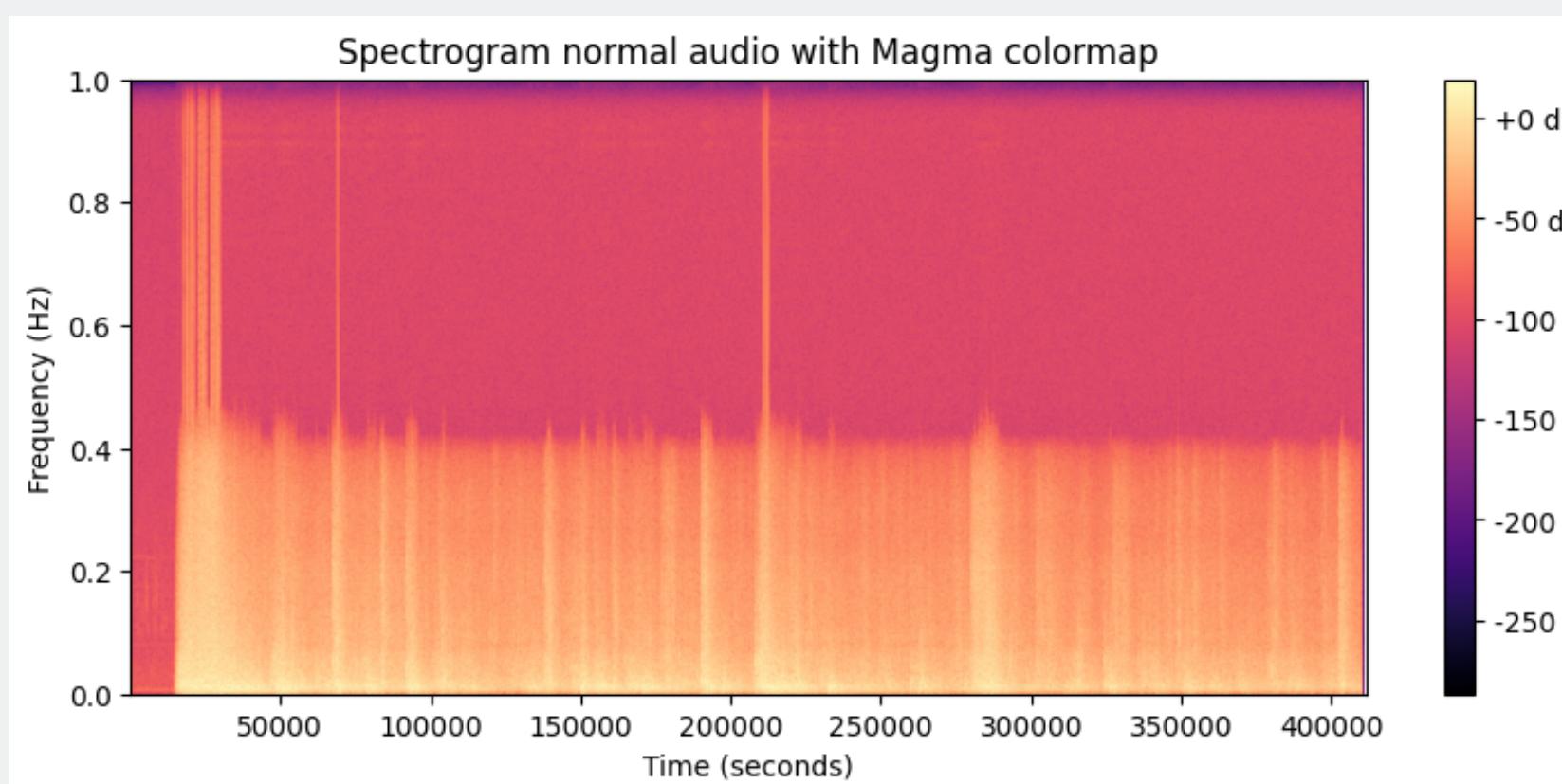
X

+

← → Q spectogram normal audio dengan peta magma

# visualisasi spectogram normal audio (suara asli) menggunakan peta warna magma

input : sinyal suara 'Earthquake.wav'



```
# Membaca file WAV
audio_path = 'Earthquake.wav'
waveform, sample_rate = librosa.load(audio_path)

# Menghitung waktu dari indeks sampel
duration = len(waveform) / sample_rate
time = np.linspace(0, duration, len(waveform))

# Menampilkan gelombang suara dengan peta warna Magma
plt.figure(figsize=(10, 4))
plt.specgram(waveform, NFFT=2048, Fs=2, noverlap=1024, cmap='magma')
plt.colorbar(format='%.2f dB')
plt.title('Spectrogram normal audio with Magma colormap')
plt.xlabel('Time (seconds)')
plt.ylabel('Frequency (Hz)')
plt.show()
```

usr/local/lib/python3.10/dist-packages/matplotlib/axes/\_axes.py:7773: RuntimeWarning: divide by zero encountered in log10  
Z = 10. \* np.log10(spec)

[Title Page](#)[Reporters](#)[Hasil](#)

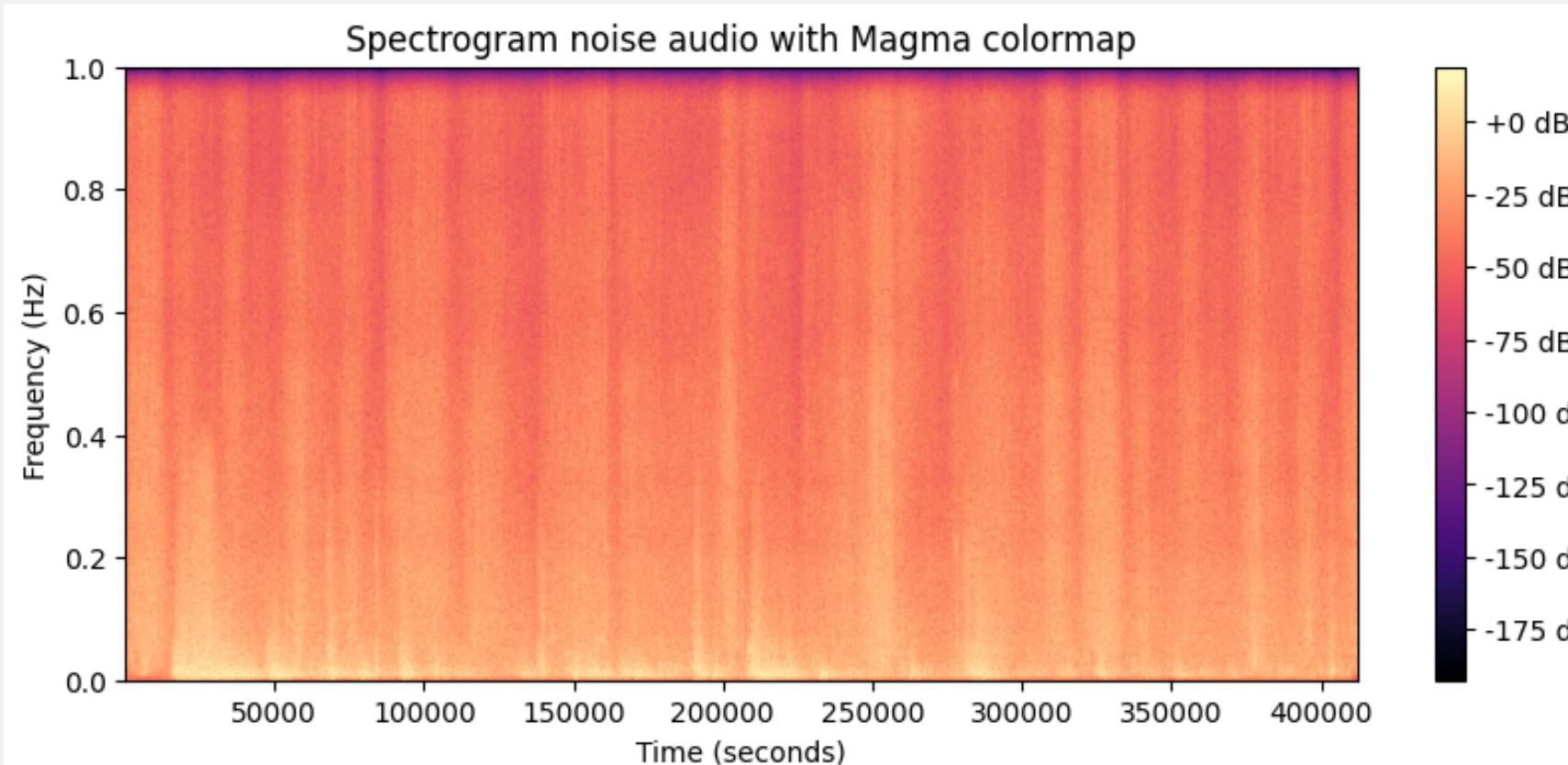
X

+

← → Q spectogram noise audio dengan peta magma

# visualisasi spectogram noise audio menggunakan peta warna magma

input : sinyal suara 'campuran.wav'



```
# Membaca file WAV
audio_path = 'campuran.WAV'
waveform, sample_rate = librosa.load(audio_path)

# Menghitung waktu dari indeks sampel
duration = len(waveform) / sample_rate
time = np.linspace(0, duration, len(waveform))

# Menampilkan gelombang suara dengan peta warna Magma
plt.figure(figsize=(10, 4))
plt.specgram(waveform, NFFT=2048, Fs=2, noverlap=1024, cmap='magma')
plt.colorbar(format='%.2f dB')
plt.title('Spectrogram noise audio with Magma colormap')
plt.xlabel('Time (seconds)')
plt.ylabel('Frequency (Hz)')
plt.show()
```

Python



Title Page

Reporters

Hasil

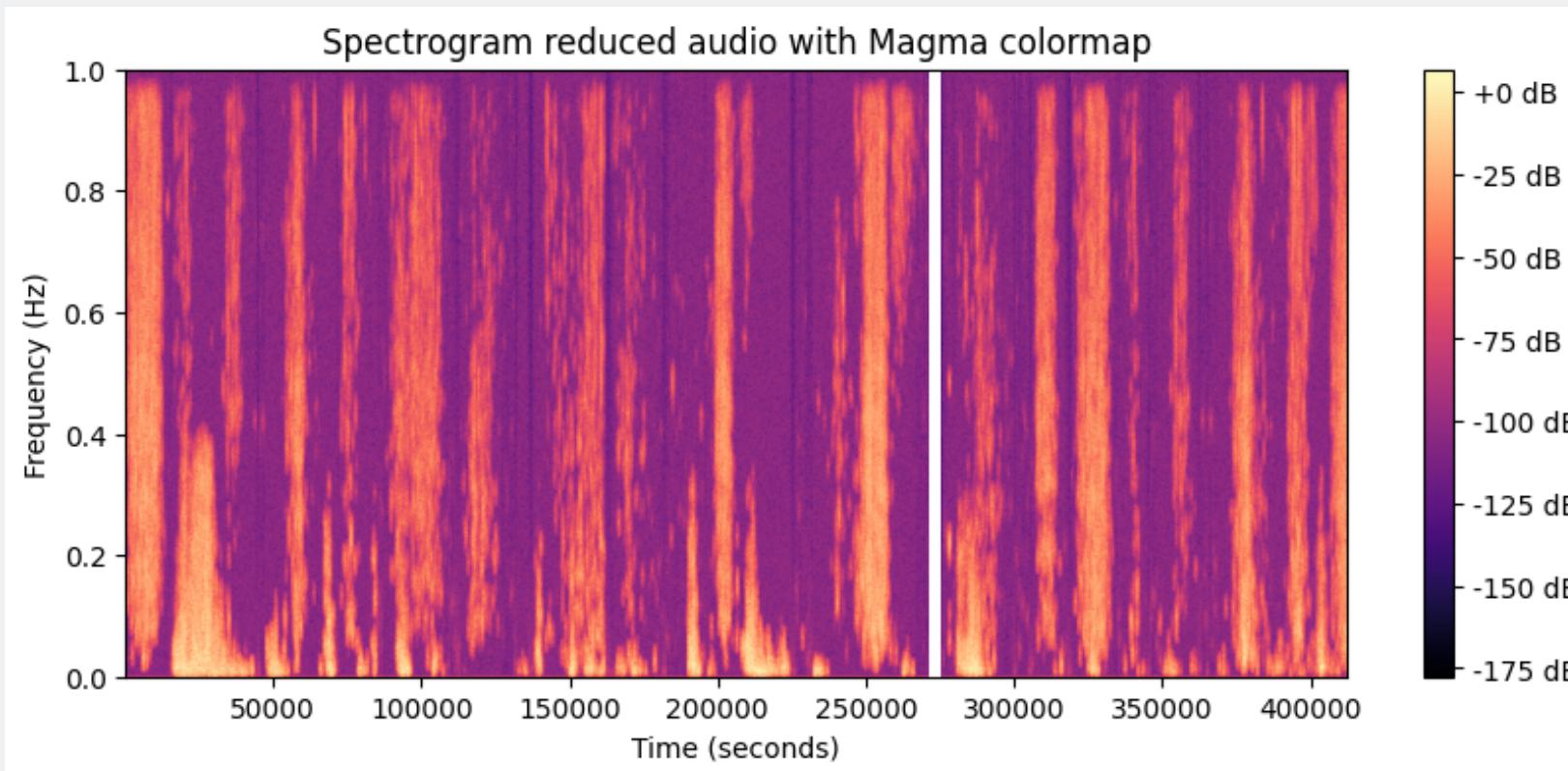
X

+

← → Q spectogram reduced audio dengan peta magma

# visualisasi spectogram reduced audio menggunakan peta warna magma

input : sinyal suara 'audio\_reduced.wav'



```
# Membaca file WAV
audio_path = 'audio_reduced.wav'
waveform, sample_rate = librosa.load(audio_path)

# Menghitung waktu dari indeks sampel
duration = len(waveform) / sample_rate
time = np.linspace(0, duration, len(waveform))

# Menampilkan gelombang suara dengan peta warna Magma
plt.figure(figsize=(10, 4))
plt.specgram(waveform, NFFT=2048, Fs=2, noverlap=1024, cmap='magma')
plt.colorbar(format='%.2f dB')
plt.title('Spectrogram reduced audio with Magma colormap')
plt.xlabel('Time (seconds)')
plt.ylabel('Frequency (Hz)')
plt.show()

/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_axes.py:7773: RuntimeWarning: divide by zero encountered in
Z = 10. * np.log10(spec)
```

Python



Title Page

Reporters

Hasil

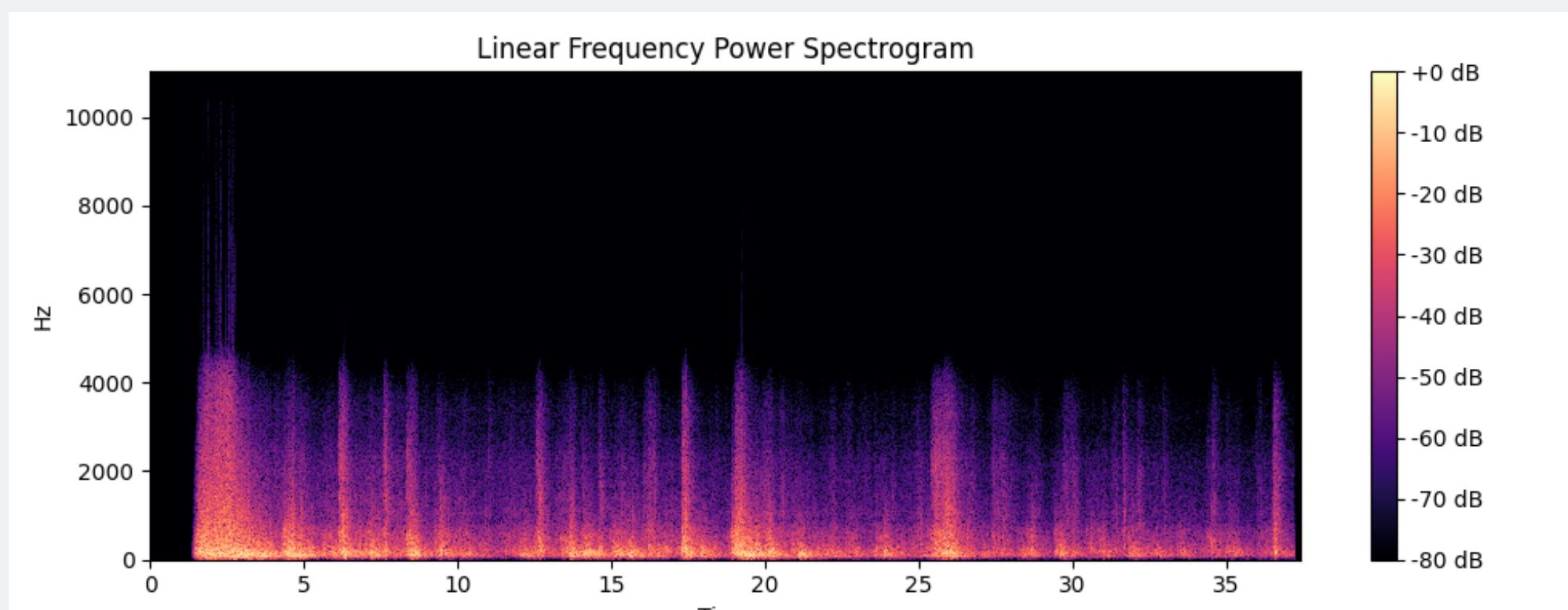
X

+

← → Q linear frequency power spectrogram pada normal audio

# visualisasi linear frequency power spectrogram pada normal audio

input : sinyal suara 'Earthquake.wav'



```
### FILE ASLI
import librosa
import librosa.display
import matplotlib.pyplot as plt

# Memuat file audio
file_audio = 'Earthquake.wav'
audio, sr = librosa.load(file_audio)

# Menghitung linear frequency power spectrogram
n_fft = 2048 # Jumlah titik FFT
hop_length = 512 # Jarak pergeseran antara frame
linear_spectrogram = np.abs(librosa.stft(audio, n_fft=n_fft, hop_length=hop_length))

# Menampilkan linear frequency power spectrogram dengan skala frekuensi linear
plt.figure(figsize=(10, 4))
librosa.display.specshow(librosa.amplitude_to_db(linear_spectrogram, ref=np.max), x_axis='time', y_axis='linear')
plt.colorbar(format='%+2.0f dB')
plt.title('Linear Frequency Power Spectrogram')
plt.tight_layout()

# Menyimpan gambar linear frequency power spectrogram secara otomatis
output_file = 'asli_linear_spectrogram.png'
plt.savefig(output_file)
plt.close()

print(f"Gambar asli linear frequency power spectrogram telah disimpan sebagai '{output_file}'")
```



Title Page

Reporters

Hasil

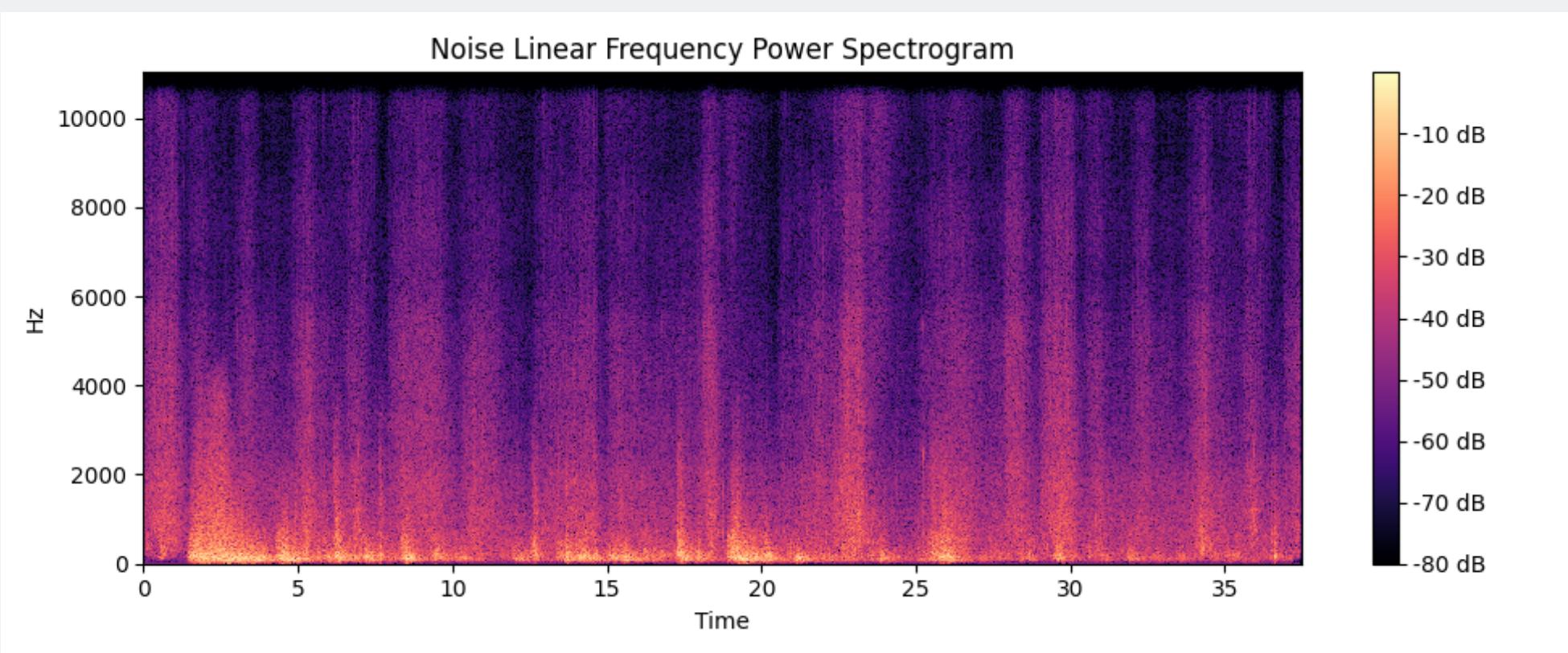
X

+

← → Q linear frequency power spectrogram pada noise audio

# visualisasi linear frequency power spectrogram pada noise audio

input : sinyal suara 'campuran.wav'



```
### FILE noise
import librosa
import librosa.display
import matplotlib.pyplot as plt

# Memuat file audio
file_audio = 'campuran.WAV'
audio, sr = librosa.load(file_audio)

# Menghitung Linear frequency power spectrogram
n_fft = 2048 # Jumlah titik FFT
hop_length = 512 # Jarak pergeseran antara frame
linear_spectrogram = np.abs(librosa.stft(audio, n_fft=n_fft, hop_length=hop_length))

# Menampilkan Linear frequency power spectrogram dengan skala frekuensi Linear
plt.figure(figsize=(10, 4))
librosa.display.specshow(librosa.amplitude_to_db(linear_spectrogram, ref=np.max), x_axis='time', y_axis='linear')
plt.colorbar(format='%.2f dB')
plt.title('Noise Linear Frequency Power Spectrogram')
plt.tight_layout()

# Menyimpan gambar Linear frequency power spectrogram secara otomatis
output_file = 'noise_linear_spectrogram.png'
plt.savefig(output_file)
plt.close()

print(f"Gambar noise linear frequency power spectrogram telah disimpan sebagai '{output_file}'")
```



Title Page

Reporters

Hasil

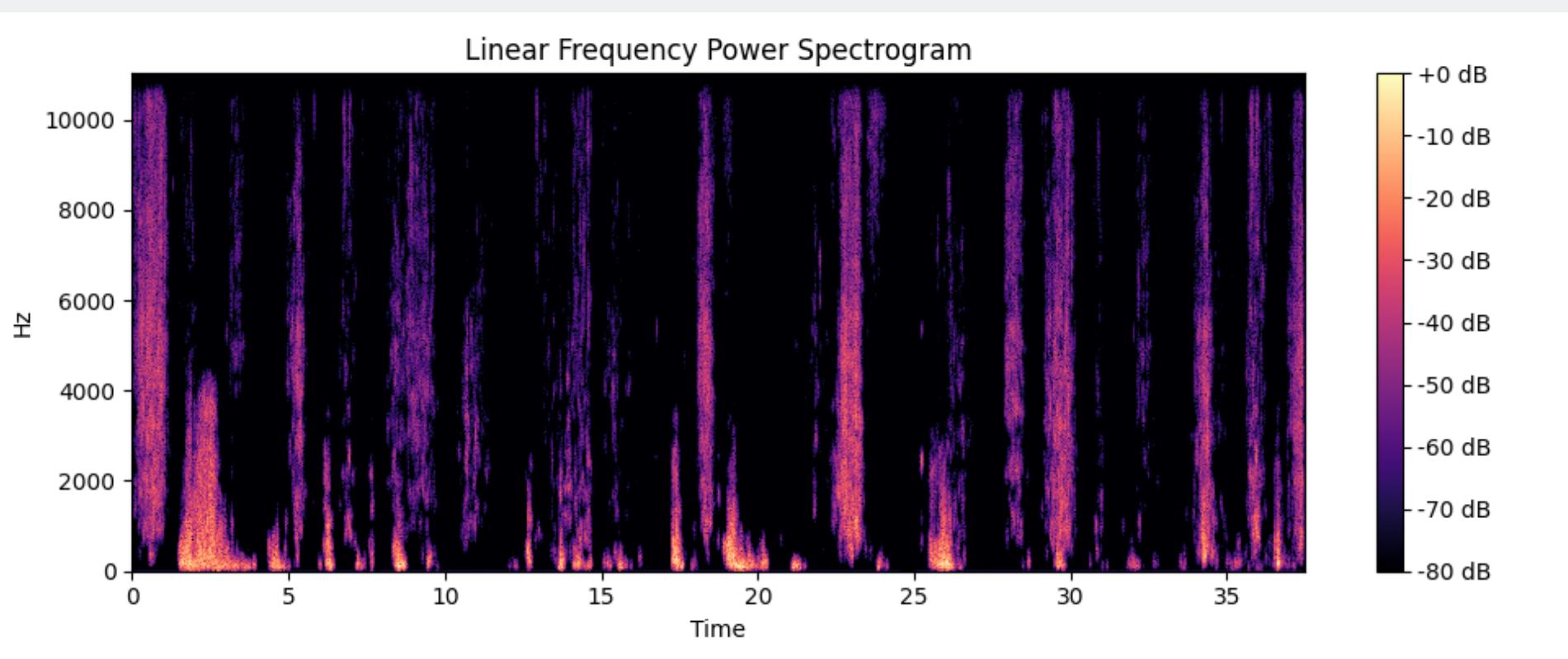
X

+

← → Q linear frequency power spectrogram pada reduced audio

# visualisasi linear frequency power spectrogram pada reduced audio

input : sinyal suara 'audio\_reduced.wav'



```
### FILE REDUCED
import librosa
import librosa.display
import matplotlib.pyplot as plt

# Memuat file audio
file_audio = 'audio_reduced.wav'
audio, sr = librosa.load(file_audio)

# Menghitung Linear frequency power spectrogram
n_fft = 2048 # Jumlah titik FFT
hop_length = 512 # Jarak pergeseran antara frame
linear_spectrogram = np.abs(librosa.stft(audio, n_fft=n_fft, hop_length=hop_length))

# Menampilkan Linear frequency power spectrogram dengan skala frekuensi linear
plt.figure(figsize=(10, 4))
librosa.display.specshow(librosa.amplitude_to_db(linear_spectrogram, ref=np.max), x_axis='time', y_axis='linear')
plt.colorbar(format='%.2f dB')
plt.title('Linear Frequency Power Spectrogram')
plt.tight_layout()

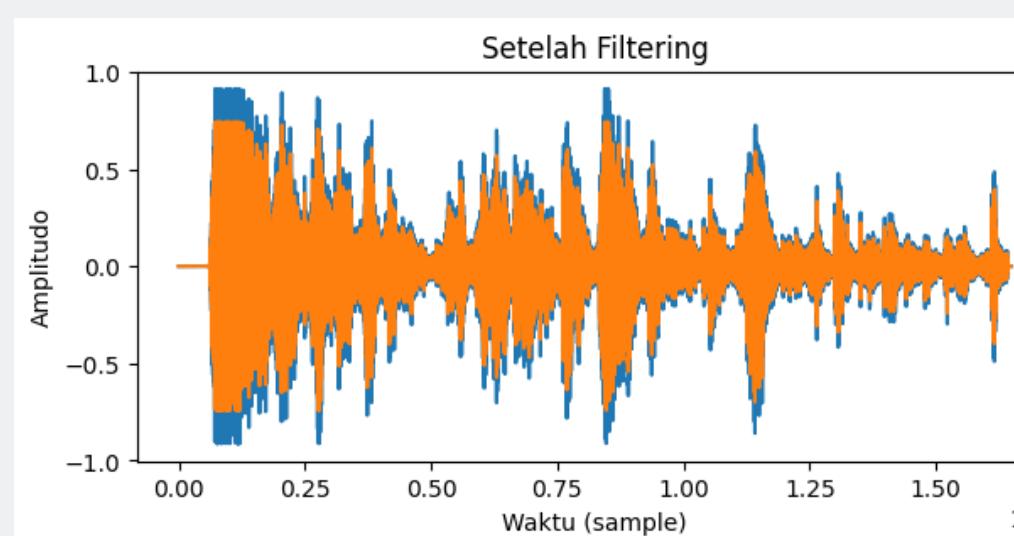
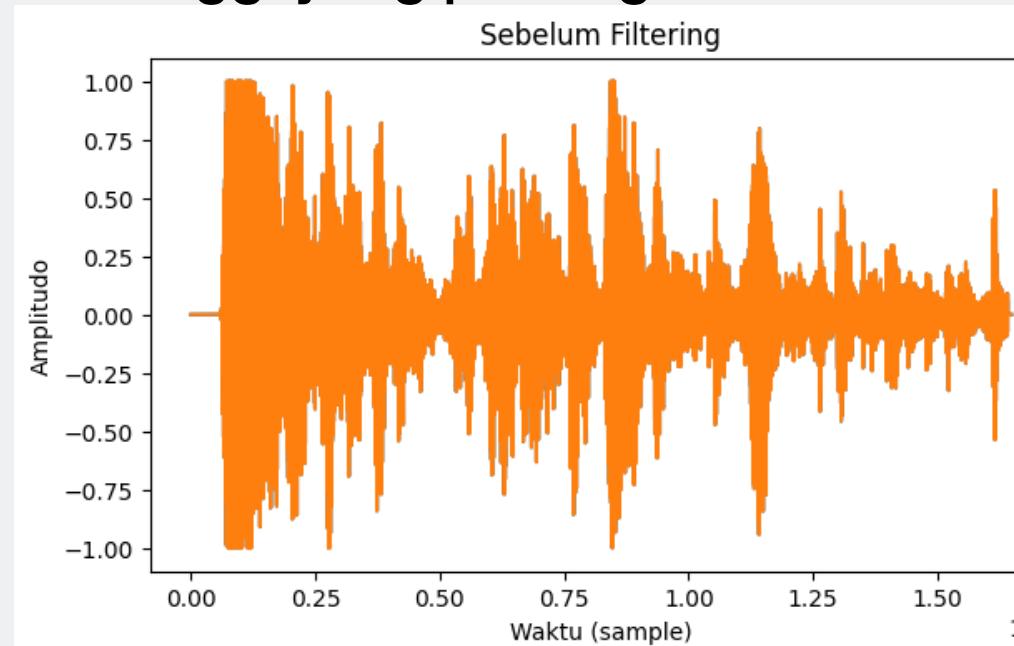
# Menyimpan gambar Linear frequency power spectrogram secara otomatis
output_file = 'reduced_Linear_spectrogram.png'
plt.savefig(output_file)
plt.close()

print(f"Gambar reduced linear frequency power spectrogram telah disimpan sebagai '{output_file}'")
```

← → Q gelombang suara bandpass pada noise audio

## visualisasi gelombang suara highpass pada noise audio

Filter ini digunakan untuk memblokir komponen frekuensi rendah dan mempertahankan komponen frekuensi tinggi pada sinyal seismic. Filter pita tinggi membantu dalam menyorot fitur-fitur frekuensi tinggi yang penting dalam data seismic.



```
import numpy as np
import scipy.signal as signal
import soundfile as sf
import matplotlib.pyplot as plt

# Load audio file
audio_path = 'Earthquake.wav'
audio, sr = sf.read(audio_path)

# Design high-pass filter parameters
cutoff_freq = 500 # Frekuensi cutoff (Hz)

# Normalize cutoff frequency to Nyquist frequency
normalized_cutoff = cutoff_freq / (sr / 2)

# Design and apply high-pass filter
b, a = signal.butter(4, normalized_cutoff, btype='high')
filtered_audio = signal.lfilter(b, a, audio)

# Save filtered audio to a new file
filtered_audio_path = 'filtered_highpass_earthquake.wav'
sf.write(filtered_audio_path, filtered_audio, sr)

# Plot waveform
plt.figure(figsize= (7,4))
plt.plot(audio)
plt.title('Waveform Sebelum Filtering')
plt.xlabel('Waktu (sample)')
plt.ylabel('Amplitudo')
plt.show()

# Plot filtered waveform
plt.figure(figsize=(7,3))
plt.plot(filtered_audio)
plt.title('Waveform Setelah Filtering')
plt.xlabel('waktu (sample)')
plt.ylabel('Amplitudo')
plt.show()

print("Audio yang telah difilter telah disimpan dengan nama 'filtered_highpass_earthquake.wav'.")
```

Python



Title Page

Reporters

Hasil

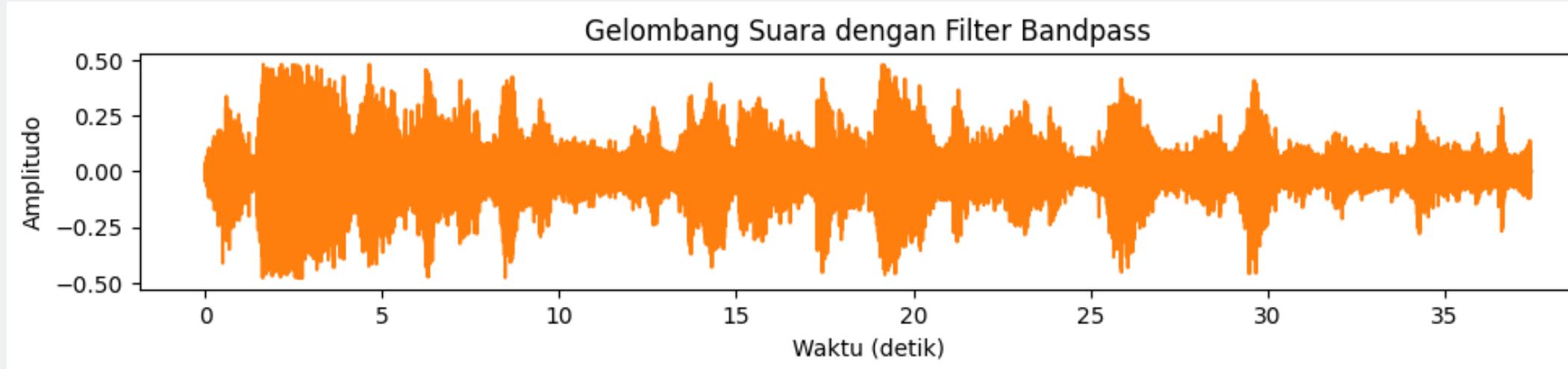
X

+

← → Q gelombang suara bandpass pada noise audio

# visualisasi gelombang suara bandpass pada noise audio

Filter bandpass digunakan untuk menghilangkan frekuensi di luar rentang tertentu dalam sinyal audio atau sinyal lainnya. Rentang frekuensi yang dipertahankan (passband) berada di antara dua frekuensi cutoff yang ditentukan. Frekuensi di luar rentang ini akan ditekan atau diblokir.



```
import numpy as np
from scipy.io import wavfile
from scipy import signal
import matplotlib.pyplot as plt

# Fungsi untuk menampilkan gelombang suara dengan filter bandpass
def plot_wave_with_bandpass(sound_file, low_cutoff, high_cutoff):
    # Membaca file gelombang suara
    sample_rate, data = wavfile.read(sound_file)

    # Mendapatkan frekuensi dan koefisien filter
    nyquist_freq = 0.5 * sample_rate
    normalized_low_cutoff = low_cutoff / nyquist_freq
    normalized_high_cutoff = high_cutoff / nyquist_freq

    # Mendesain filter menggunakan bandpass filter
    b, a = signal.butter(5, [normalized_low_cutoff, normalized_high_cutoff], btype='bandpass')

    # Memfilter data suara menggunakan filter yang telah didesain
    filtered_data = signal.lfilter(b, a, data)

    # Menampilkan gelombang suara dengan filter bandpass
    plt.figure(figsize=(10,4))
    plt.subplot(2, 1, 2)
    plt.plot(np.arange(len(filtered_data)) / sample_rate, filtered_data)
    plt.title('Gelombang Suara dengan Filter Bandpass')
    plt.xlabel('Waktu (detik)')
    plt.ylabel('Amplitudo')

    plt.tight_layout()
    plt.show()

# Memanggil fungsi plot_wave_with_bandpass untuk menampilkan gelombang suara dengan filter bandpass
sound_file = 'campuran.WAV' # Ganti dengan path ke file gelombang suara input
low_cutoff = 2000 # Ganti dengan frekuensi cutoff rendah dalam Hz
high_cutoff = 3000 # Ganti dengan frekuensi cutoff tinggi dalam Hz

plot_wave_with_bandpass(sound_file, low_cutoff, high_cutoff)
```

Python

← → Q gelombang suara bandpass pada noise audio

# Kesimpulan

Dengan tidak adanya file noise asli yang dioprasikan maka kelemahan dari output yang kami proleh adalah sebagai berikut:

1. Kurangnya informasi tentang tingkat kebisingan: Tanpa file noise asli, sulit untuk mengetahui tingkat kebisingan yang spesifik, sehingga mengatur parameter pengurangan kebisingan menjadi sulit.
2. Kehilangan informasi suara asli: Pengurangan kebisingan cenderung menghilangkan komponen frekuensi yang mirip dengan noise, yang dapat menyebabkan kehilangan informasi penting dari suara asli yang diinginkan.
3. Ketidaksempurnaan hasil: Tanpa file noise asli, hasil pengurangan kebisingan mungkin tidak optimal dan distorsi pada suara yang diinginkan, mengurangi kualitas keseluruhan.
4. Dari hasil yang didapat, inyal suara seismic asli tanpa noise dan sinyal reduce noise hasilnya berbeda. Pada reduce noise, masih terdapat noise yang belum sempurna hilang dari sinyal suaranya