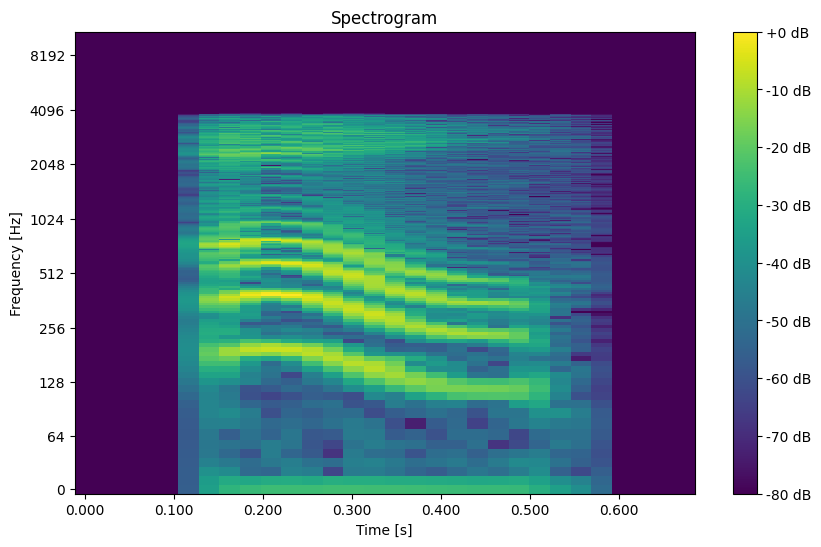
**SPECTROGRAM**

* Spectrogram when an audio file of sound “a” is given as input is

**CODE 1**

import numpy as np

import librosa

import librosa.display

import matplotlib.pyplot as plt

from scipy.io.wavfile import write

# Load the audio file

filename = 'your\_audio\_file.wav'  # Replace with the path to your audio file

y, sr = librosa.load(filename)

# Generate the spectrogram

S = librosa.stft(y)  # Short-time Fourier transform

S\_db = librosa.amplitude\_to\_db(np.abs(S), ref=np.max)  # Convert to decibel scale

# Display the spectrogram (optional)

plt.figure(figsize=(10, 6))

librosa.display.specshow(S\_db, sr=sr, x\_axis='time', y\_axis='log', cmap='viridis')

plt.colorbar(format='%+2.0f dB')

plt.title('Spectrogram')

plt.xlabel('Time [s]')

plt.ylabel('Frequency [Hz]')

plt.show()

# Convert the spectrogram back to audio

S\_inverse = librosa.db\_to\_amplitude(S\_db)  # Convert from dB back to amplitude

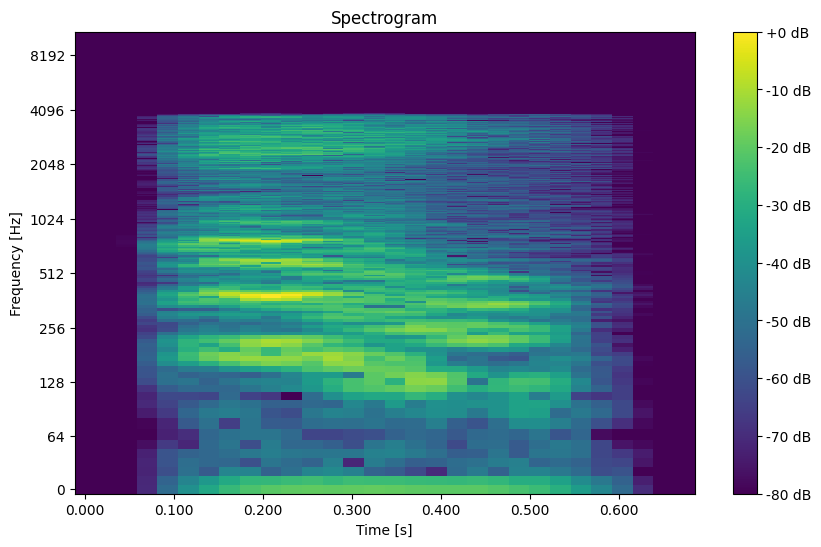
y\_reconstructed = librosa.istft(S\_inverse)  # Inverse Short-time Fourier transform

# Save the reconstructed audio to a file

output\_filename = 'reconstructed\_audio.wav'

write(output\_filename, sr, y\_reconstructed.astype(np.float32))

print(f'Reconstructed audio saved as {output\_filename}')

* Spectrogram when the output of audio file of the above mentioned spectrogram is

**CODE 2**

import numpy as np

import librosa

import librosa.display

import matplotlib.pyplot as plt

from scipy.io.wavfile import write

# Load the audio file

filename = '/content/reconstructed\_audio.wav'  # Replace with the path to your audio file

y, sr = librosa.load(filename)

# Generate the spectrogram

S = librosa.stft(y)  # Short-time Fourier transform

S\_db = librosa.amplitude\_to\_db(np.abs(S), ref=np.max)  # Convert to decibel scale

# Display the spectrogram (optional)

plt.figure(figsize=(10, 6))

librosa.display.specshow(S\_db, sr=sr, x\_axis='time', y\_axis='log', cmap='viridis')

plt.colorbar(format='%+2.0f dB')

plt.title('Spectrogram')

plt.xlabel('Time [s]')

plt.ylabel('Frequency [Hz]')

plt.show()

# Convert the spectrogram back to audio

S\_inverse = librosa.db\_to\_amplitude(S\_db)  # Convert from dB back to amplitude

y\_reconstructed = librosa.istft(S\_inverse)  # Inverse Short-time Fourier transform

# Save the reconstructed audio to a file

output\_filename = 'reconstructed\_audio.wav'

write(output\_filename, sr, y\_reconstructed.astype(np.float32))

print(f'Reconstructed audio saved as {output\_filename}')