plotting the voice as a function of time

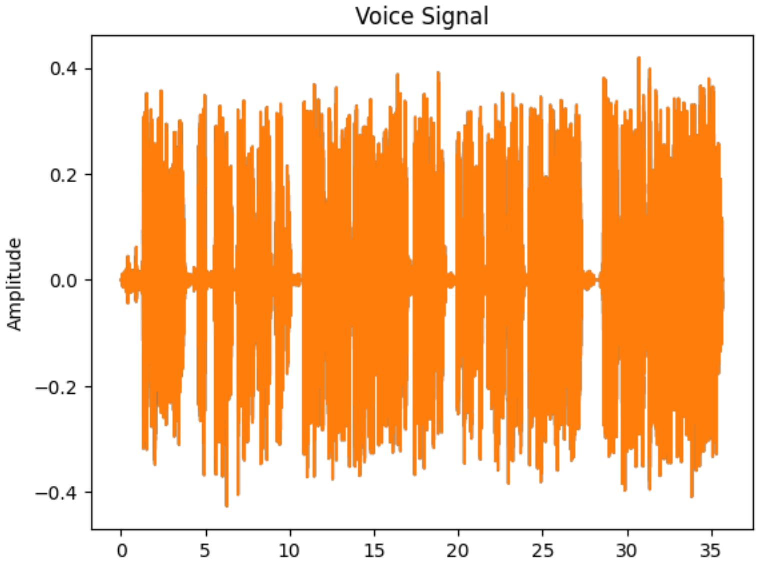
- List item
- List item

```
!pip install pydub
!apt-get install ffmpeg
from google.colab import files
uploaded = files.upload()

from pydub import AudioSegment
import numpy as np
import soundfile as sf
import io
filename = next(iter(uploaded))
file_contents = uploaded[filename]

audio = AudioSegment.from_file(io.BytesIO(file_contents), format='m4a')

audio.export('converted.wav', format='wav')

voice_signal, fs = sf.read('converted.wav')
```

```
Requirement already satisfied: pydub in /usr/local/lib/python3.10/dist-packages (0.25.1)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ffmpeg is already the newest version (7:4.4.2-0ubuntu0.22.04.1).
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to

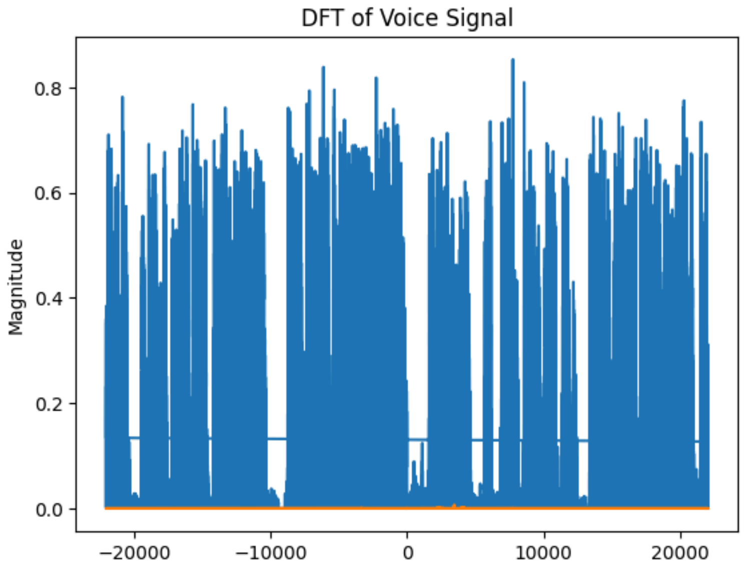
```
import numpy as np
import matplotlib.pyplot as plt


t = np.arange(len(voice_signal)) / fs

plt.plot(t, voice_signal)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Voice Signal')
plt.show()
```



calculating the dft

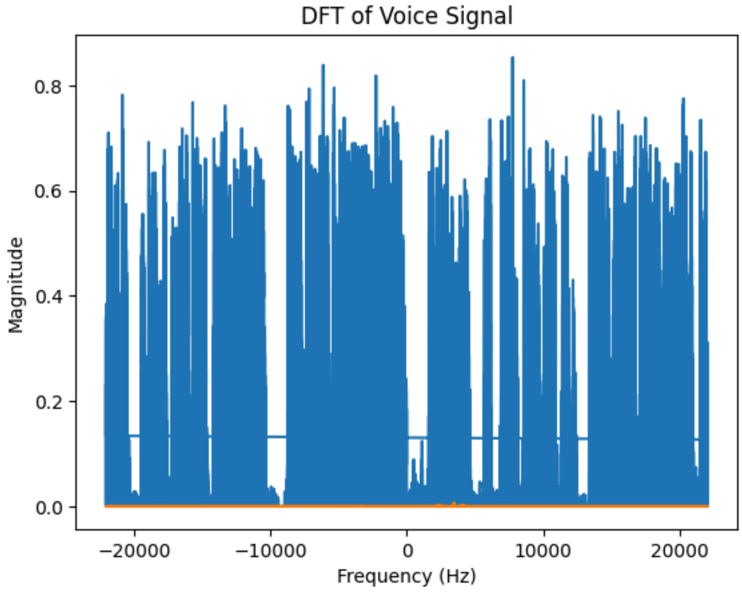
```
import matplotlib.pyplot as plt
import numpy as np

dft = np.fft.fft(voice_signal)
```

```
    dft = np.fft.fft(voice_signal)

    frequencies = np.fft.fftfreq(len(dft), 1/fs)

    # Plot the magnitude spectrum
    plt.plot(frequencies, np.abs(dft))
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Magnitude')
    plt.title('DFT of Voice Signal')
    plt.show()
```



plotting the spectrum

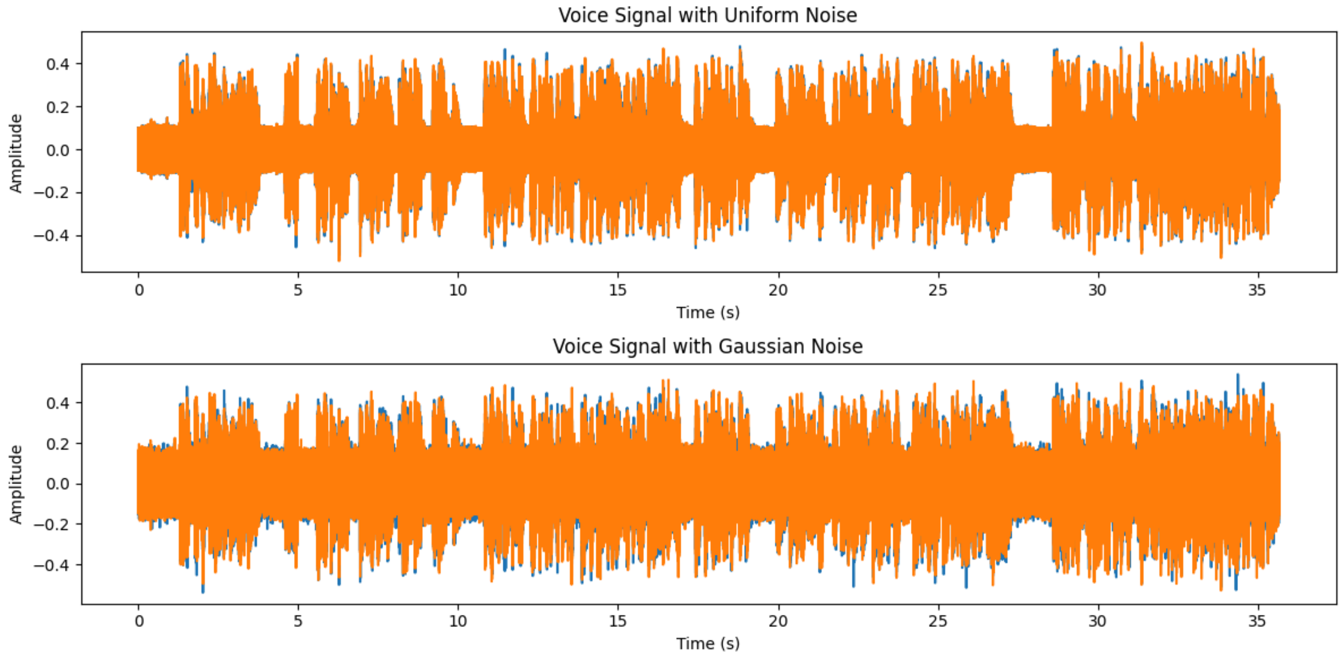
```
# pPlot the spectrum.

import matplotlib.pyplot as plt
import numpy as np
# Assuming 'voice_signal' and 'fs' are available from the preceding code

# Calculate the DFT
dft = np.fft.fft(voice_signal)

# Calculate the frequencies corresponding to the DFT values
frequencies = np.fft.fftfreq(len(dft), 1/fs)

# Plot the magnitude spectrum
plt.plot(frequencies, np.abs(dft))
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.title('DFT of Voice Signal')
plt.show()

# Find the dominant frequencies
dominant_frequencies = frequencies[np.argsort(np.abs(dft))[::-1]][:5]  # Top 5 frequencies

print("Dominant frequencies:", dominant_frequencies)
```



```
Dominant frequencies: [[0.02803802 0.        ]
 [0.02803802 0.        ]
 [0.02803802 0.        ]
 [0.02803802 0.        ]
```

adding uniform or gaussian noise

```
# Add Uniform/ Gaussian noise to the voice

import matplotlib.pyplot as plt
import numpy as np
# Assuming 'voice_signal' and 'fs' are available from the preceding code

# Add uniform noise
noise_uniform = np.random.uniform(-0.1, 0.1, size=voice_signal.shape) # Create noise with the same shape as voice_signal
voice_with_uniform_noise = voice_signal + noise_uniform

# Add Gaussian noise
noise_gaussian = np.random.normal(0, 0.05, size=voice_signal.shape) # Create noise with the same shape as voice_signal
voice_with_gaussian_noise = voice_signal + noise_gaussian

# Plot the signals with noise
plt.figure(figsize=(12, 6))

plt.subplot(2, 1, 1)
plt.plot(t, voice_with_uniform_noise)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Voice Signal with Uniform Noise')

plt.subplot(2, 1, 2)
plt.plot(t, voice_with_gaussian_noise)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Voice Signal with Gaussian Noise')

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



hearing the noisy voice

```
# Step 1: Install necessary libraries
!pip install pydub
!apt-get install ffmpeg
!pip install soundfile

# Step 2: Upload the audio file
from google.colab import files
uploaded = files.upload()

from pydub import AudioSegment
import soundfile as sf
import numpy as np
import io # Import the io module

filename = list(uploaded.keys())[0]
file_content = uploaded[filename]

audio = AudioSegment.from_file(io.BytesIO(file_content), format=filename.split('.')[-1])
audio.export('converted.wav', format='wav')

voice_signal, fs = sf.read('converted.wav')

# Step 3: Add noise to the audio signal

# Add uniform noise
uniform_noise = np.random.uniform(-0.05, 0.05, voice_signal.shape)
voice_with_uniform_noise = voice_signal + uniform_noise

# Add Gaussian noise
gaussian_noise = np.random.normal(0, 0.05, voice_signal.shape)
voice_with_gaussian_noise = voice_signal + gaussian_noise

# Step 4: Play the original and noisy audio signals
from IPython.display import Audio

# Play the original audio
print("Original Audio:")
display(Audio(voice_signal[:, 0], rate=fs)) # Select the first channel for mono audio

# Play the audio with uniform noise
print("Audio with Uniform Noise:")
display(Audio(voice_with_uniform_noise[:, 0], rate=fs)) # Select the first channel for mono audio

# Play the audio with Gaussian noise
print("Audio with Gaussian Noise:")
display(Audio(voice_with_gaussian_noise[:, 0], rate=fs)) # Select the first channel for mono audio
```

```
Requirement already satisfied: pydub in /usr/local/lib/python3.10/dist-packages (0.25.1)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ffmpeg is already the newest version (7:4.4.2-0ubuntu0.22.04.1).
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
Requirement already satisfied: soundfile in /usr/local/lib/python3.10/dist-packages (0.12.1)
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.10/dist-packages (from soundfile) (1.16.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.0->soundfile) (2.22)
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving meghaat2.m4a.m4a to meghaat2.m4a (3).m4a
Original Audio:
```

0:00 / 0:35

```
Audio with Uniform Noise:
```

0:00 / 0:35

```
Audio with Gaussian Noise:
```

designing suitable lpf

Start coding or generate with AI.

```
# Step 1: Install necessary libraries
!pip install pydub
!apt-get install ffmpeg
!pip install soundfile

# Step 2: Upload the audio file
from google.colab import files
uploaded = files.upload()

from pydub import AudioSegment
import soundfile as sf
import numpy as np
import io # Import the io module

# Get the filename and content from the uploaded files
filename = list(uploaded.keys())[0] # Get the first uploaded filename
file_content = uploaded[filename]

# Convert the in-memory file content to an AudioSegment object
audio = AudioSegment.from_file(io.BytesIO(file_content), format=filename.split('.')[-1]) # Use io.BytesIO to read from memory

# Export the audio to a .wav file
audio.export('converted.wav', format='wav')

# Now read the converted .wav file using soundfile
voice_signal, fs = sf.read('converted.wav')

# Step 3: Add noise to the audio signal

# Add uniform noise
uniform_noise = np.random.uniform(-0.05, 0.05, voice_signal.shape)
voice_with_uniform_noise = voice_signal + uniform_noise

# Add Gaussian noise
gaussian_noise = np.random.normal(0, 0.05, voice_signal.shape)
voice_with_gaussian_noise = voice_signal + gaussian_noise

# Step 4: Play the original and noisy audio signals
from IPython.display import Audio

# Play the original audio
print("Original Audio:")
display(Audio(voice_signal[:, 0], rate=fs)) # Select the first channel for mono audio

# Play the audio with uniform noise
print("Audio with Uniform Noise:")
display(Audio(voice_with_uniform_noise[:, 0], rate=fs)) # Select the first channel for mono audio

# Play the audio with Gaussian noise
print("Audio with Gaussian Noise:")
display(Audio(voice_with_gaussian_noise[:, 0], rate=fs)) # Select the first channel for mono audio
# Step 4: Apply a low-pass filter
```

```
def butter_lowpass_filter(data, cutoff, fs, order=5):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    y = lfilter(b, a, data, axis=0)  # Ensure the filter is applied along the correct axis
    return y

# Choose a cutoff frequency for the LPF (adjust as needed)
cutoff_frequency = 3000

# Apply the LPF to the noisy signals
filtered_uniform = butter_lowpass_filter(voice_with_uniform_noise, cutoff_frequency, fs)
filtered_gaussian = butter_lowpass_filter(voice_with_gaussian_noise, cutoff_frequency, fs)

# Step 5: Play the original, noisy, and filtered audio signals

# Play the original audio
print("Original Audio:")
display(Audio(voice_signal[:, 0], rate=fs))  # Select the first channel for mono audio

# Play the audio with uniform noise
print("Audio with Uniform Noise:")
display(Audio(voice_with_uniform_noise[:, 0], rate=fs))  # Select the first channel for mono audio

# Play the audio with Gaussian noise
print("Audio with Gaussian Noise:")
display(Audio(voice_with_gaussian_noise[:, 0], rate=fs))  # Select the first channel for mono audio

# Play the filtered audio with uniform noise
print("Filtered Audio with Uniform Noise:")
display(Audio(filtered_uniform[:, 0], rate=fs))  # Select the first channel for mono audio

# Play the filtered audio with Gaussian noise
print("Filtered Audio with Gaussian Noise:")
display(Audio(filtered_gaussian[:, 0], rate=fs))  # Select the first channel for mono audio
```

```
Requirement already satisfied: pydub in /usr/local/lib/python3.10/dist-packages (0.25.1)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ffmpeg is already the newest version (7:4.4.2-0ubuntu0.22.04.1).
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
Requirement already satisfied: soundfile in /usr/local/lib/python3.10/dist-packages (0.12.1)
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.10/dist-packages (from soundfile) (1.16.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.0->soundfile) (2.22)
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving meghaat2.m4a.m4a to meghaat2.m4a (5).m4a
Original Audio:
```

0:00 / 0:35

```
Audio with Uniform Noise:
```

0:00 / 0:35

```
Audio with Gaussian Noise:
```

0:00 / 0:35

```
Original Audio:
```

0:00 / 0:35

```
Audio with Uniform Noise:
```

0:00 / 0:35

```
Audio with Gaussian Noise:
```

0:00 / 0:35

```
Filtered Audio with Uniform Noise:
```

0:00 / 0:35

```
Filtered Audio with Gaussian Noise:
```

designing suitable hpf

```
!pip install pydub
!apt-get install ffmpeg
!pip install soundfile


from google.colab import files
uploaded = files.upload()

from pydub import AudioSegment
import soundfile as sf
import numpy as np
import io # Import the io module

# Get the filename and content from the uploaded files
filename = list(uploaded.keys())[0] # Get the first uploaded filename
file_content = uploaded[filename]

# Convert the in-memory file content to an AudioSegment object
audio = AudioSegment.from_file(io.BytesIO(file_content), format=filename.split('.')[-1]) # Use io.BytesIO to read from memory

# Export the audio to a .wav file
audio.export('converted.wav', format='wav')

# Now read the converted .wav file using soundfile
voice_signal, fs = sf.read('converted.wav')

# Step 3: Add noise to the audio signal

# Add uniform noise
uniform_noise = np.random.uniform(-0.05, 0.05, voice_signal.shape)
voice_with_uniform_noise = voice_signal + uniform_noise

# Add Gaussian noise
gaussian_noise = np.random.normal(0, 0.05, voice_signal.shape)
voice_with_gaussian_noise = voice_signal + gaussian_noise

# Step 4: Define high-pass filter function

def butter_highpass_filter(data, cutoff, fs, order=5):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='high', analog=False)
    y = lfilter(b, a, data, axis=0)  # Ensure the filter is applied along the correct axis
    return y

# Choose a cutoff frequency for the HPF (adjust as needed)
highpass_cutoff = 300  # Example cutoff frequency, adjust as needed

# Apply the HPF to the noisy signals
filtered_uniform_highpass = butter_highpass_filter(voice_with_uniform_noise, highpass_cutoff, fs)
filtered_gaussian_highpass = butter_highpass_filter(voice_with_gaussian_noise, highpass_cutoff, fs)

# Step 5: Play the original, noisy, and high-pass filtered audio signals

# Play the original audio
print("Original Audio:")
display(Audio(voice_signal[:, 0], rate=fs))  # Select the first channel for mono audio

# Play the audio with uniform noise
print("Audio with Uniform Noise:")
display(Audio(voice_with_uniform_noise[:, 0], rate=fs))  # Select the first channel for mono audio

# Play the audio with Gaussian noise
print("Audio with Gaussian Noise:")
display(Audio(voice_with_gaussian_noise[:, 0], rate=fs))  # Select the first channel for mono audio

# Play the high-pass filtered audio with uniform noise
print("High-Pass Filtered Audio with Uniform Noise:")
display(Audio(filtered_uniform_highpass[:, 0], rate=fs))  # Select the first channel for mono audio

# Play the high-pass filtered audio with Gaussian noise
print("High-Pass Filtered Audio with Gaussian Noise:")
display(Audio(filtered_gaussian_highpass[:, 0], rate=fs))  # Select the first channel for mono audio
```

```
Requirement already satisfied: pydub in /usr/local/lib/python3.10/dist-packages (0.25.1)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ffmpeg is already the newest version (7:4.4.2-0ubuntu0.22.04.1).
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
Requirement already satisfied: soundfile in /usr/local/lib/python3.10/dist-packages (0.12.1)
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.10/dist-packages (from soundfile) (1.16.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.0->soundfile) (2.22)
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.