

ECE 420 – Prelab 5: Resampling

```
In [3]: import numpy as np
import matplotlib.pyplot as plt
from scipy.io.wavfile import read
from scipy import signal
from IPython.display import Audio

# Load audio
Fs, data = read('test_audio.wav')
data = data[:, 0]
print("Sample rate:", Fs, "Hz")
print("Data shape:", data.shape)
```

Sample rate: 48000 Hz
Data shape: (981916,)

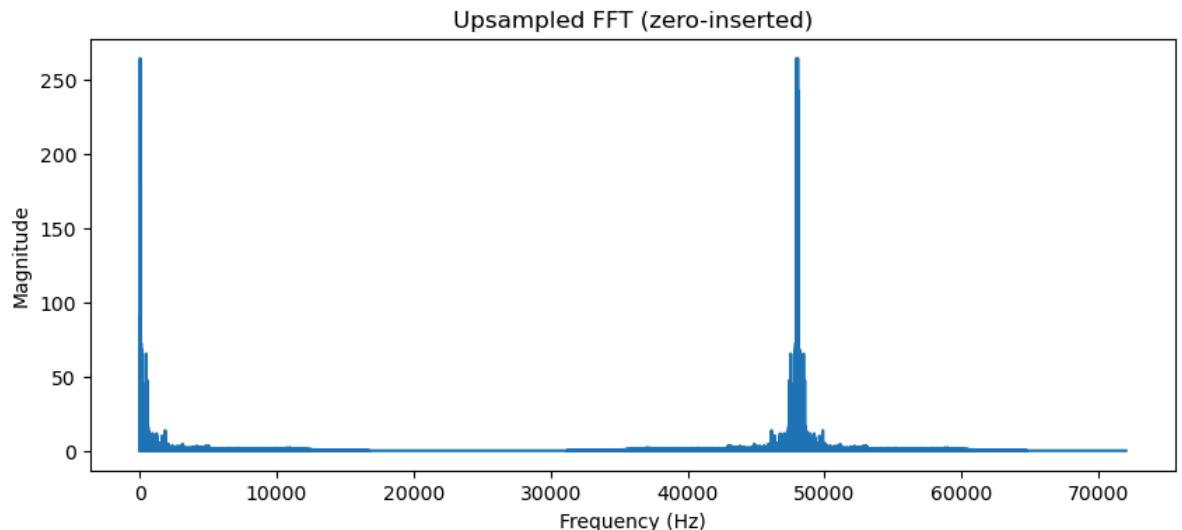
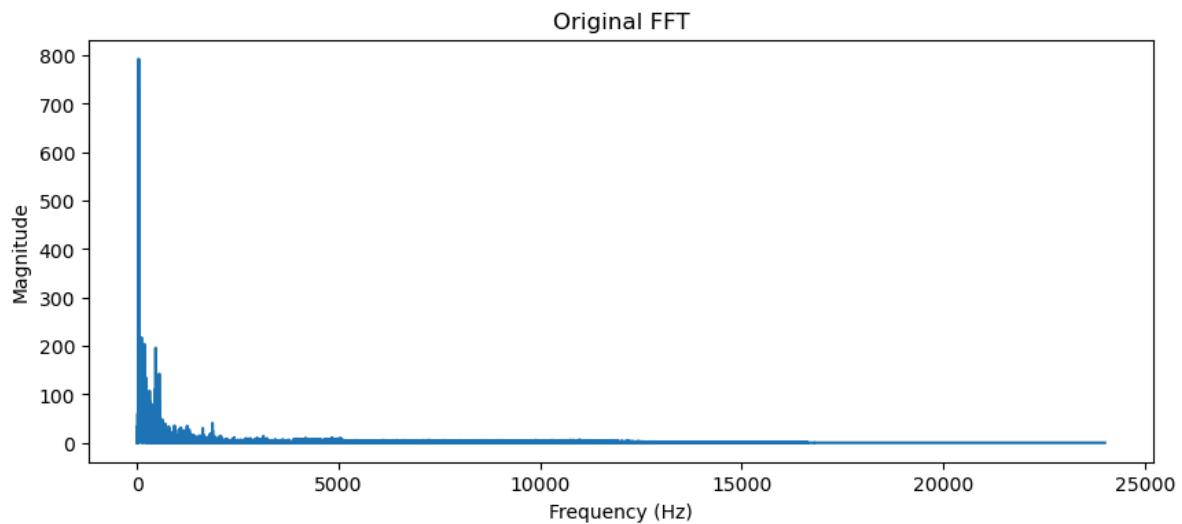
Part 1 – Upsample by 3x

In [6]:

```
# upsampling
upsample_factor = 3
upsampled = np.zeros(len(data) * upsample_factor)
upsampled[::upsample_factor] = data

# Compute FFTs
def plot_fft(sig, Fs, title):
    N = len(sig)
    f = np.fft.fftfreq(N, d=1/Fs)
    fft_mag = np.abs(np.fft.fft(sig)) / N
    plt.figure(figsize=(10, 4))
    plt.plot(f[:N//2], fft_mag[:N//2])
    plt.title(title)
    plt.xlabel("Frequency (Hz)")
    plt.ylabel("Magnitude")
    plt.show()

plot_fft(data, Fs, "Original FFT")
plot_fft(upsampled, Fs*upsample_factor, "Upsampled FFT (zero-inserted)")
```



Q1: What is the relationship between the original signal's FFT and the upsampled signal's FFT?
How do we preserve the original information without introducing aliasing?

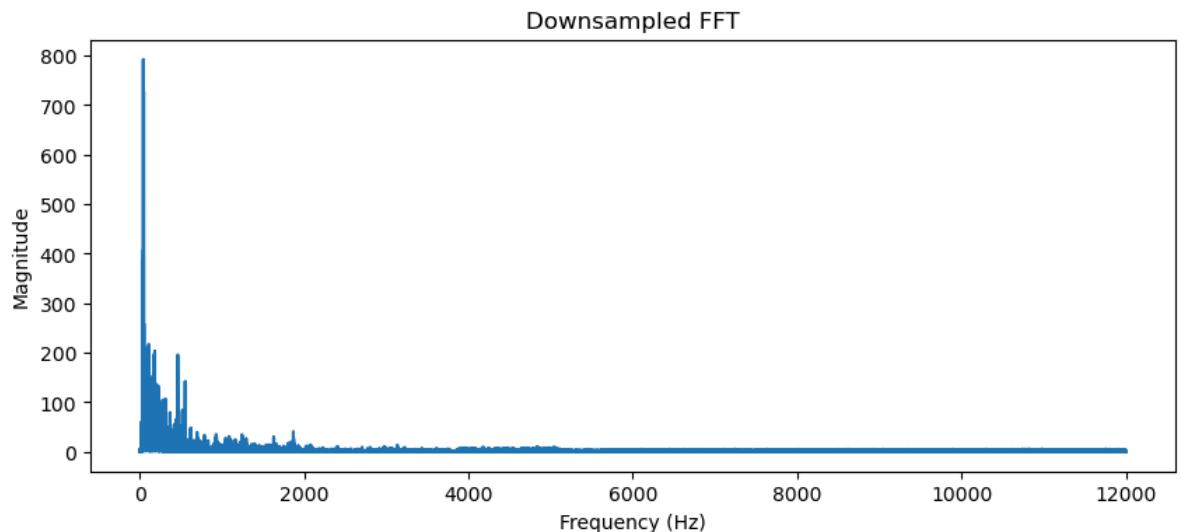
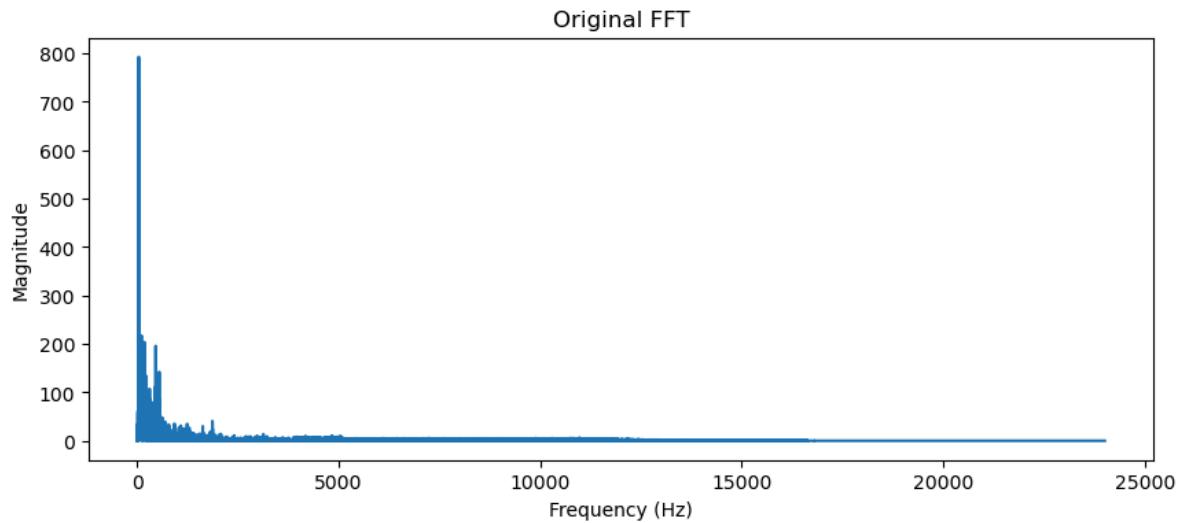
Relationship of FFTs: The upsampled signal's FFT shows the original spectrum repeated (images) at multiples of the new Nyquist frequency.

How to preserve info: Apply a low-pass filter after upsampling to remove the extra spectral images and retain only the original band.

Part 2 – Downsample by 2x

```
In [7]: # Downsampling
downsample_factor = 2
downsampled = data[::downsample_factor]

plot_fft(data, Fs, "Original FFT")
plot_fft(downscaled, Fs/downsample_factor, "Downscaled FFT")
```



Q2: What is the relationship between the original signal's FFT and the downsampled signal's FFT?

How do we avoid aliasing when downsampling?

Relationship of FFTs: The downsampled FFT compresses the frequency axis, causing spectral overlap (aliasing) if no filter is used.

How to preserve info: Apply an anti-aliasing low-pass filter before decimation to ensure the spectrum fits into the new Nyquist limit.

Part 3 – Playback with resample_poly

```
In [11]: from scipy import signal
from scipy.io.wavfile import read
from IPython.display import Audio
import numpy as np

Fs, data = read('test_audio.wav')
data = data[:, 0].astype(np.float32)

# Normalize
data = data / np.max(np.abs(data))

# Pitch up (2x)
output1 = signal.resample_poly(data, 2, 1)
Audio(output1, rate=Fs)

# Pitch down (0.5x)
output2 = signal.resample_poly(data, 1, 2)
Audio(output1, rate=Fs)
```

Out[11]:

0:00 / 0:40

Q3: How does the resampled audio sound?

Be specific: what changes do you notice in pitch and vocal characteristics?

Pitch up (2x): The audio sounds higher, like a “chipmunk” effect. Vocal formants shift upward, making the voice thinner and brighter.

Pitch down (0.5x): The audio sounds deeper and slower, like a “giant” or “demon” voice. The vocal formants shift downward, making the sound darker and heavier.

In []:

