HW3: Fourier Transform of Velocity Data

Austin Gardiner

September 12, 2025

1 FFT Code

The following code calculates the Fourier Transform using numpy's FFT function. The .lvm data files are read into Python and all velocity and sample number values are saved in a list. For each file, the velocities are passed to the *plot_fft_magnitudes* function. The *plot_fft_magnitudes* function then uses the *compute_fft* function to compute the FT for each record size and for the whole set. The *compute_fft* function starts by splitting the data into batches of the record size. It then calculates the FFT for each batch. When it has reached the end of the data and the remnants are not enough for another full batch, the remaining data points are ignored. I also tested a version where the remaining data was supplemented with a set of zeros to match the record size and the plots looked very similar. The frequencies for plotting are obtained using the np.fft.fftfreq function and are multiplied by the sampling rate (15,000 samples/s) to obtain the frequency in Hz. *compute_fft* function saves the results from each record and the average is calculated point for point using the np.mean function in *plot_fft_magnitudes*.

```
# Compute the FFT magnitude for the whole set or in windows, and return magnitude and frequencies
def compute_fft(velocities, record_size=None):
    velocities = np.asarray(velocities)
    n = len(velocities) if record_size is None else record_size
    sampling_rate = 15000  # samples per second

    if record_size is None or record_size >= len(velocities):
        # Compute FFT for the whole set
        fft_result = np.fft.fft(velocities, n=n)
        fft_magnitude = np.abs(fft_result)[:n//2]
        fft_freq = np.fft.fftfreq(n)[:n//2] * sampling_rate
        return fft_freq, fft_magnitude

    # Windowed FFT
    num_windows = len(velocities) // record_size
    fft_results = []
    for i in range(num_windows):
        start = i * record_size
        end = start + record_size
        window_data = velocities[start:end]
        fft_result = np.fft.fft(window_data)
        fft_magnitude = np.abs(fft_result)[:record_size//2]
        fft_results.append(fft_magnitude)

    fft_freq = np.fft.fftfreq(record_size)[:record_size//2] * sampling_rate
    return fft_freq, fft_results

# Plot FT magnitude for different record sizes on a log-log plot
def plot_fft_magnitudes(data, record_sizes=[None, 256, 2048, 16384]):
    velocities = data[1]
    plt.figure(figsize=(12, 6))

    # Plot each record size
    for rec_size in record_sizes:
        freq, mag = compute_fft(velocities, record_size=rec_size)
        if rec_size is None:
            plt.plot(freq[1:], mag[1:], label='Whole set', linestyle='--', color='k')
        else:
            # Compute average magnitude across windows
            ave_mag = np.mean(mag, axis=0)
            plt.plot(freq[1:], ave_mag[1:], label=f'Record Size: {rec_size}')  # skip DC component

    # Adjust plot settings
    plt.xscale('log')
    plt.yscale('log')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Magnitude')
```

```
plt.title(f'Average FT Magnitude vs Frequency for Different Record Sizes for {data[2]}')
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
plt.tight_layout()
plt.show()
```

2 Plots

A plot is created for each of the data.lvm files. As the record size changes, so does the resolution of the plot. Larger record sizes allow for more frequencies as the number of frequencies $n_F = \frac{N}{2}$. For the whole data set, N is the number of data points, but for the smaller record sizes, N is the record size. This results in the FT plots looking less smooth for larger record sizes and the FT curves reaching farther to the left in the plots. The data does keep its shape across the different record sizes, as seen by the magnitude spike near $f = 5000$ in Fig. 1. However, the overall magnitude of the FT results decreases significantly with the smaller record sizes.

For all data sets, the maximum frequency computed is the Nyquist frequency $f_N = \frac{SR}{2}$, where SR is the sampling rate. With this data, the sampling rate is 15,000 so the Nyquist frequency is 7,500.

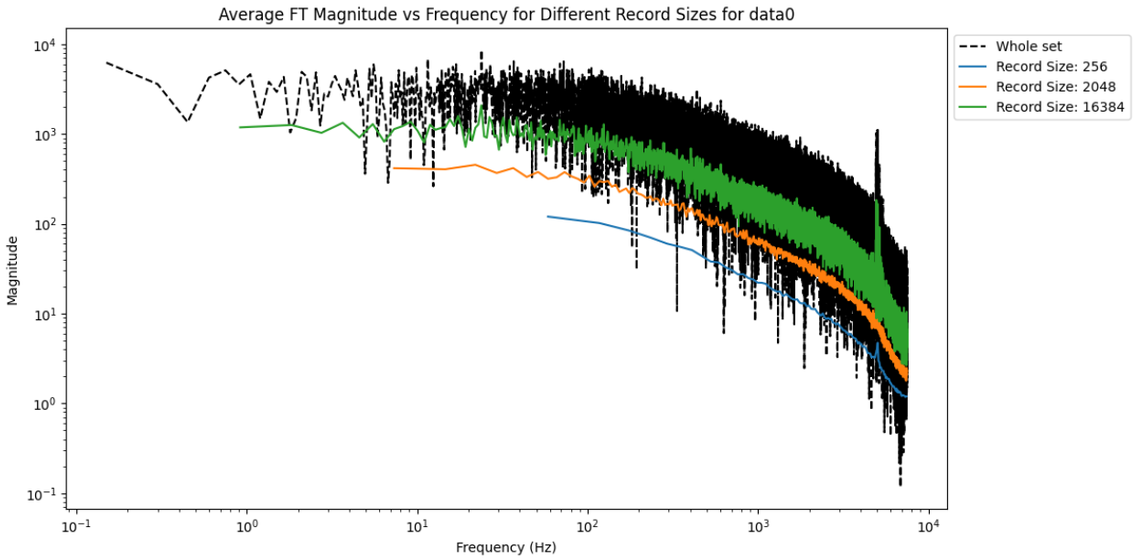

Figure 1: Average FT magnitude vs frequency for record sizes 256, 2048, and 16384

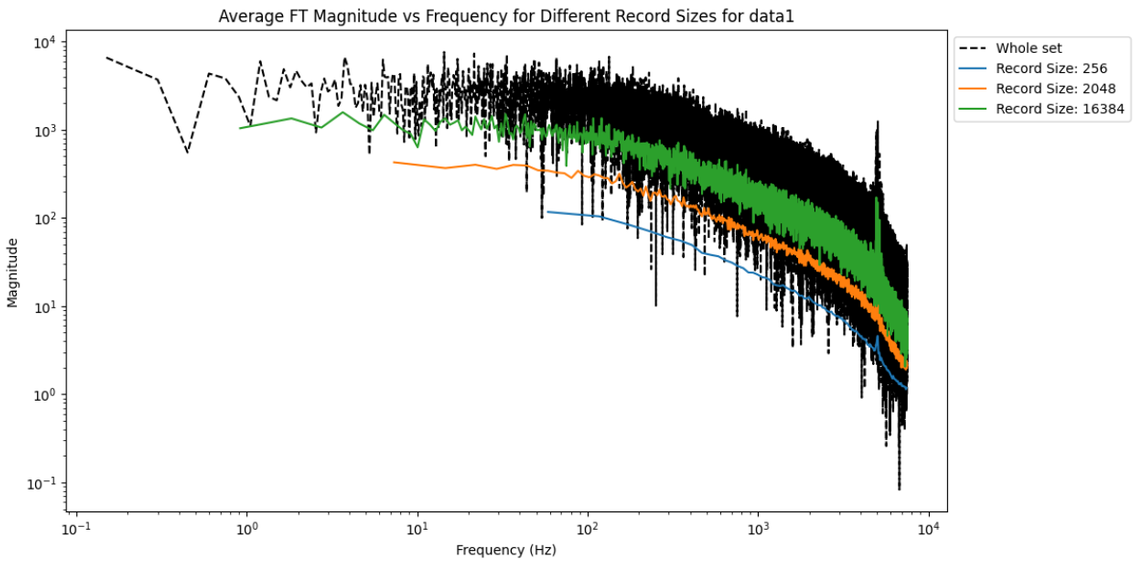

Figure 2: Average FT magnitude vs frequency for record sizes 256, 2048, and 16384

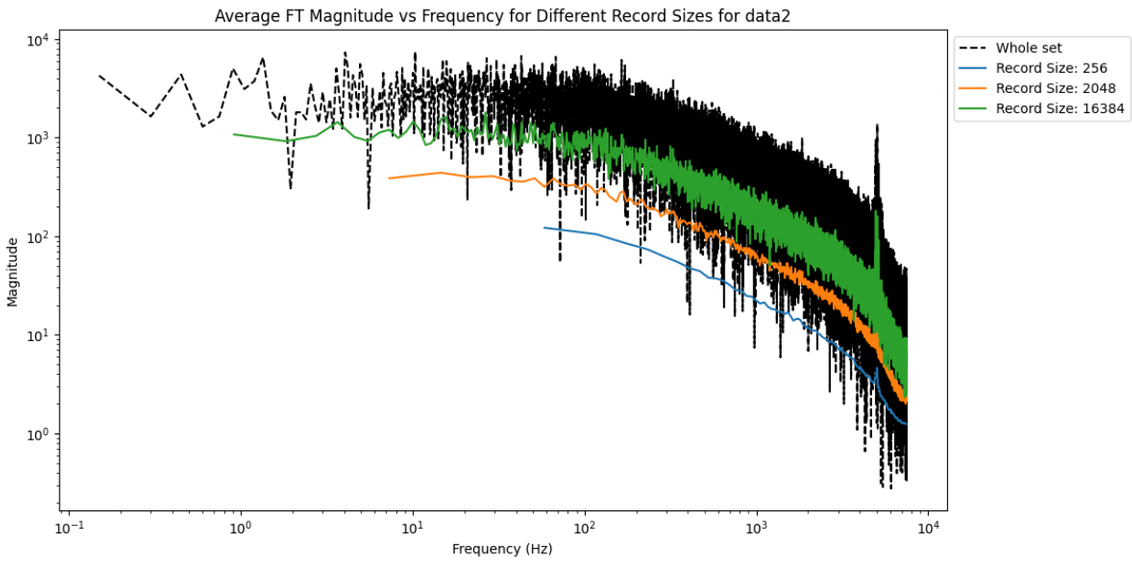

Figure 3: Average FT magnitude vs frequency for record sizes 256, 2048, and 16384

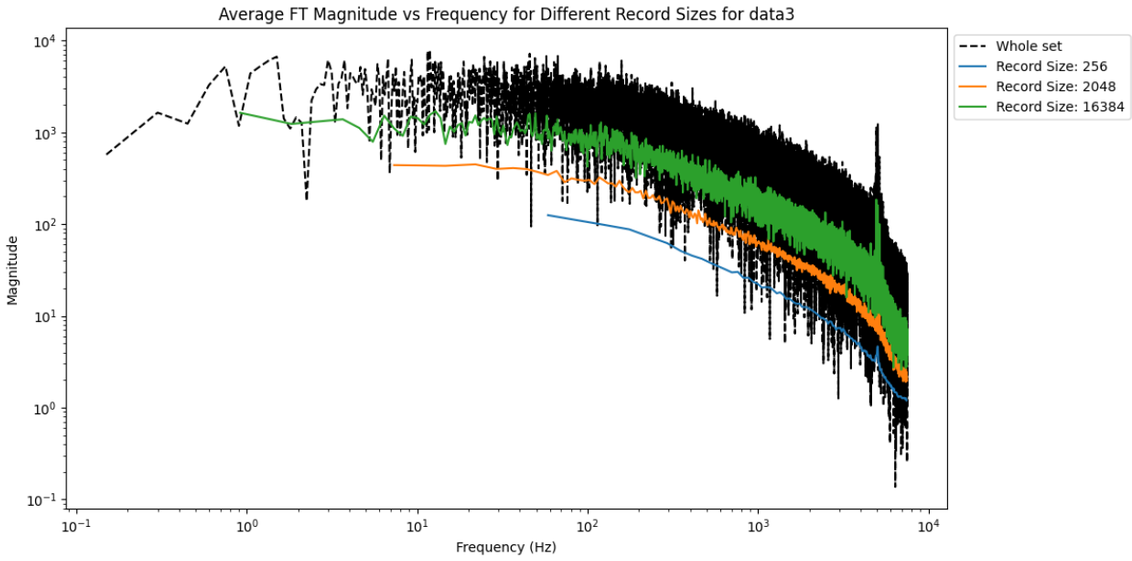

Figure 4: Average FT magnitude vs frequency for record sizes 256, 2048, and 16384

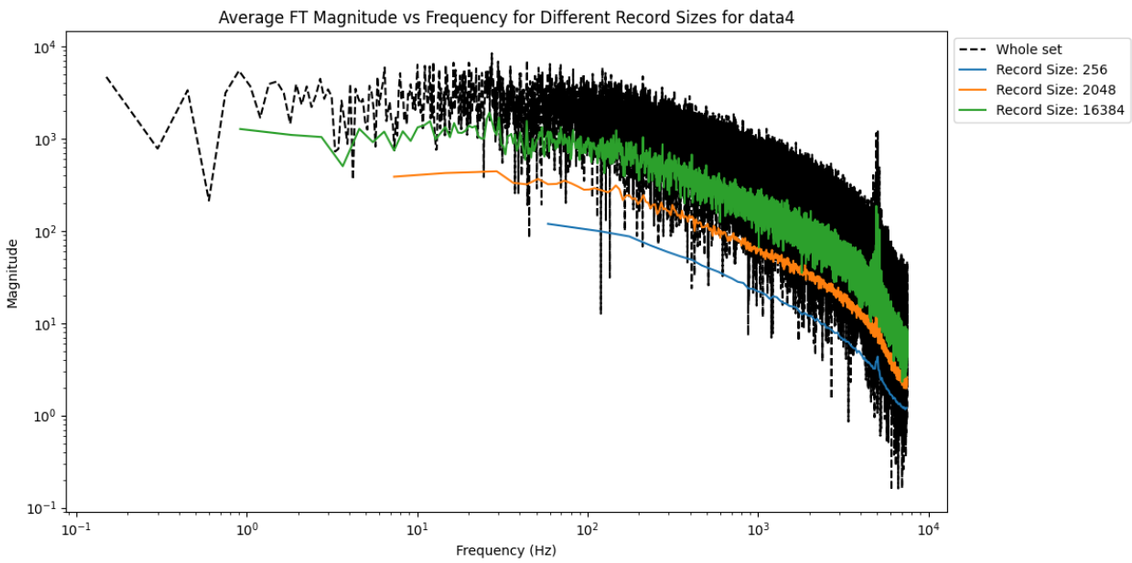

Figure 5: Average FT magnitude vs frequency for record sizes 256, 2048, and 16384

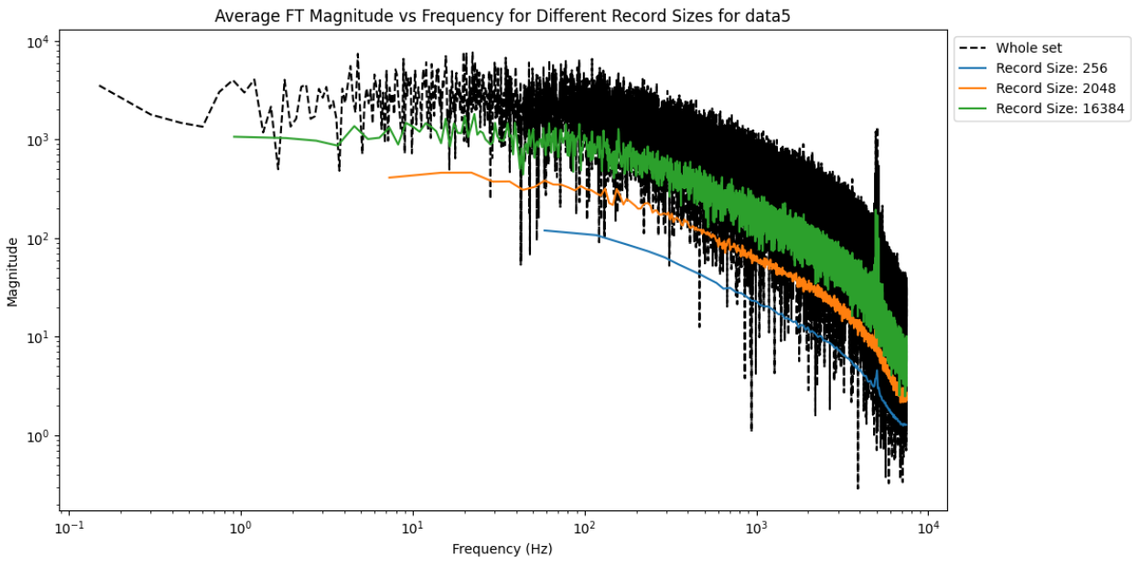

Figure 6: Average FT magnitude vs frequency for record sizes 256, 2048, and 16384

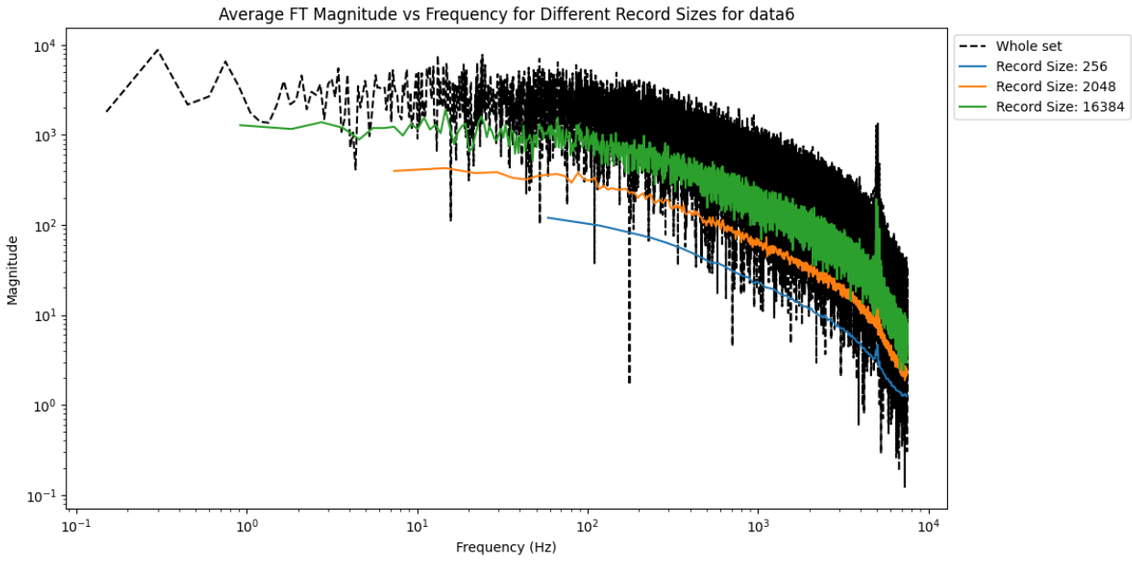

Figure 7: Average FT magnitude vs frequency for record sizes 256, 2048, and 16384

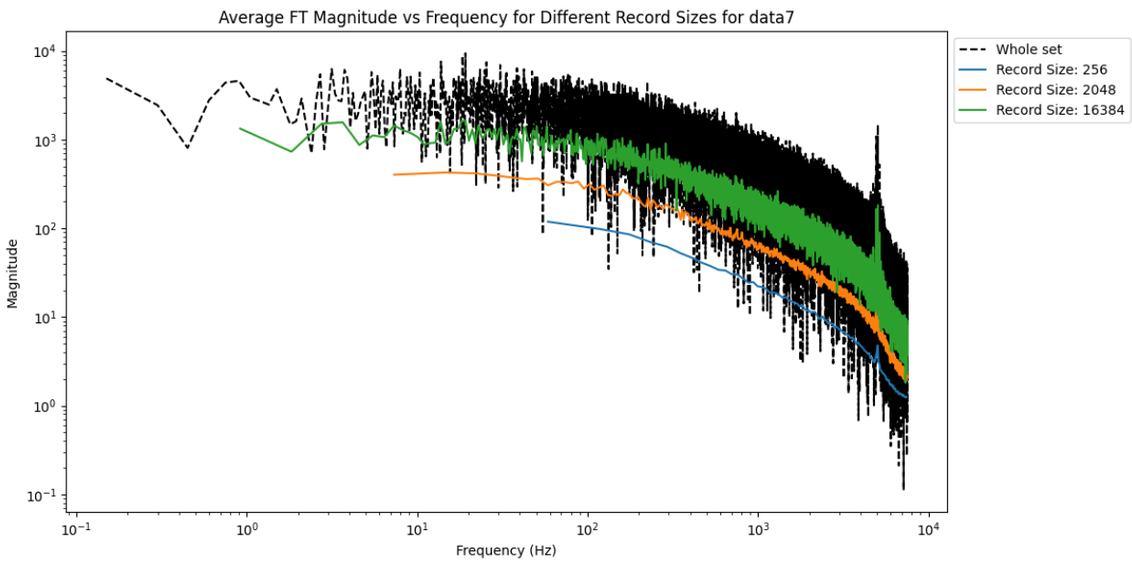

Figure 8: Average FT magnitude vs frequency for record sizes 256, 2048, and 16384

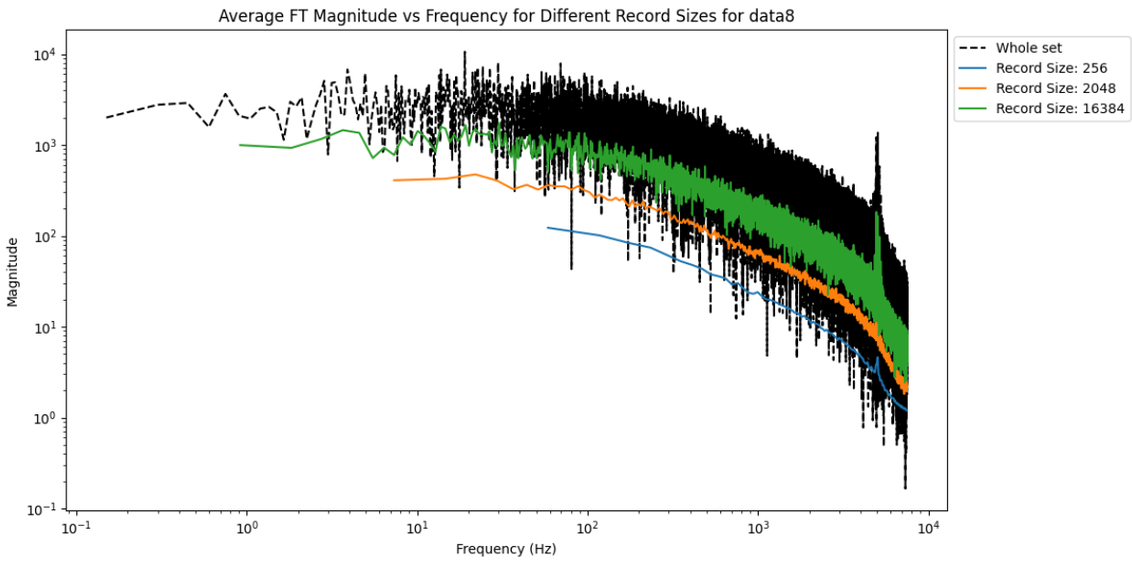

Figure 9: Average FT magnitude vs frequency for record sizes 256, 2048, and 16384

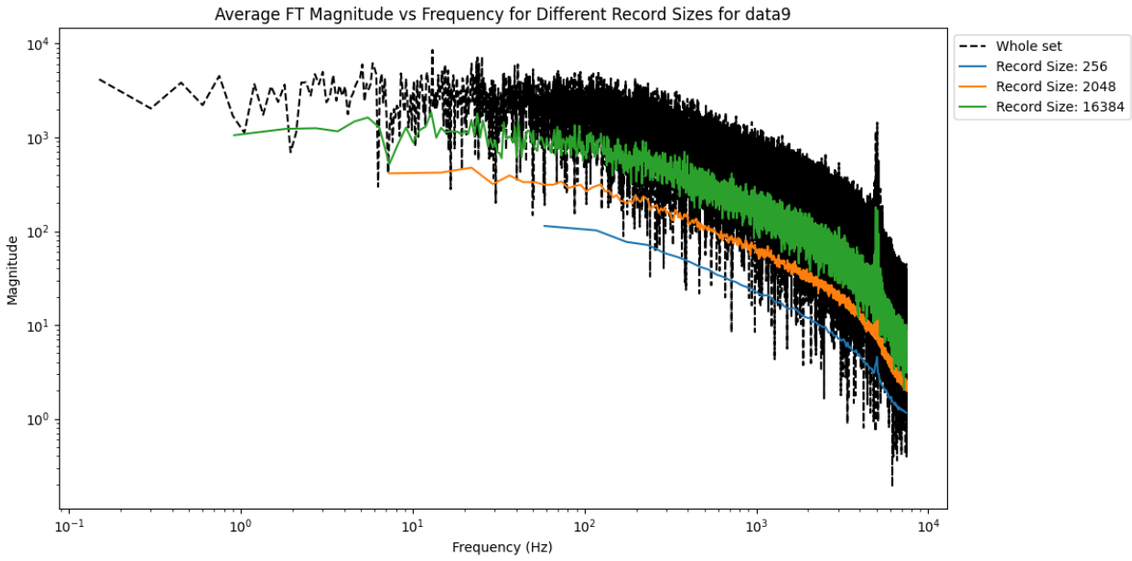

Figure 10: Average FT magnitude vs frequency for record sizes 256, 2048, and 16384

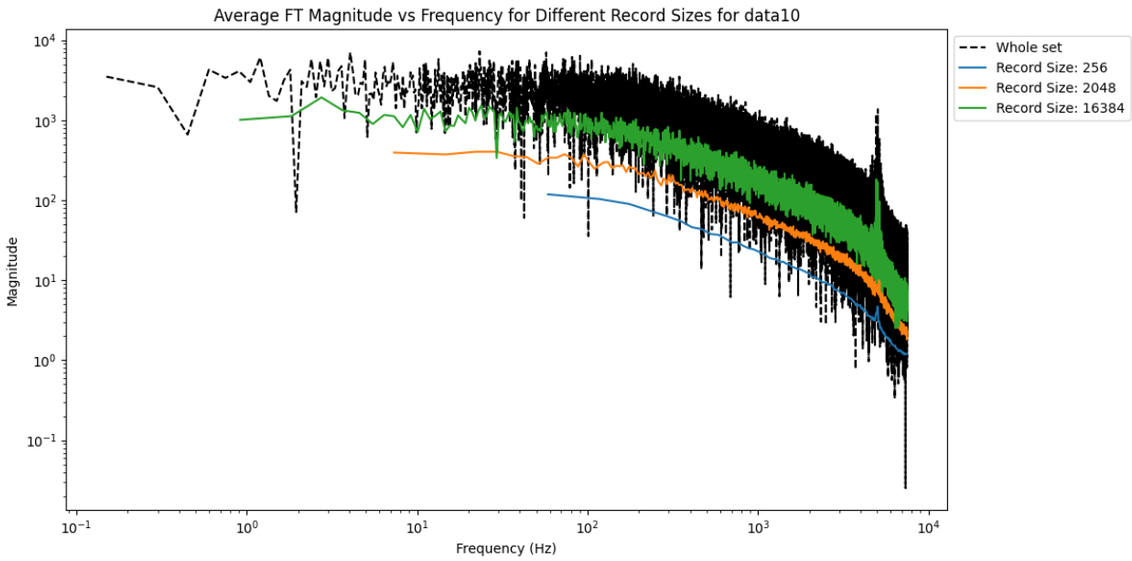

Figure 11: Average FT magnitude vs frequency for record sizes 256, 2048, and 16384