SATFD lab 04

Dawid Karpiński, 25.04.2024 r.

1 Noisy sine signal

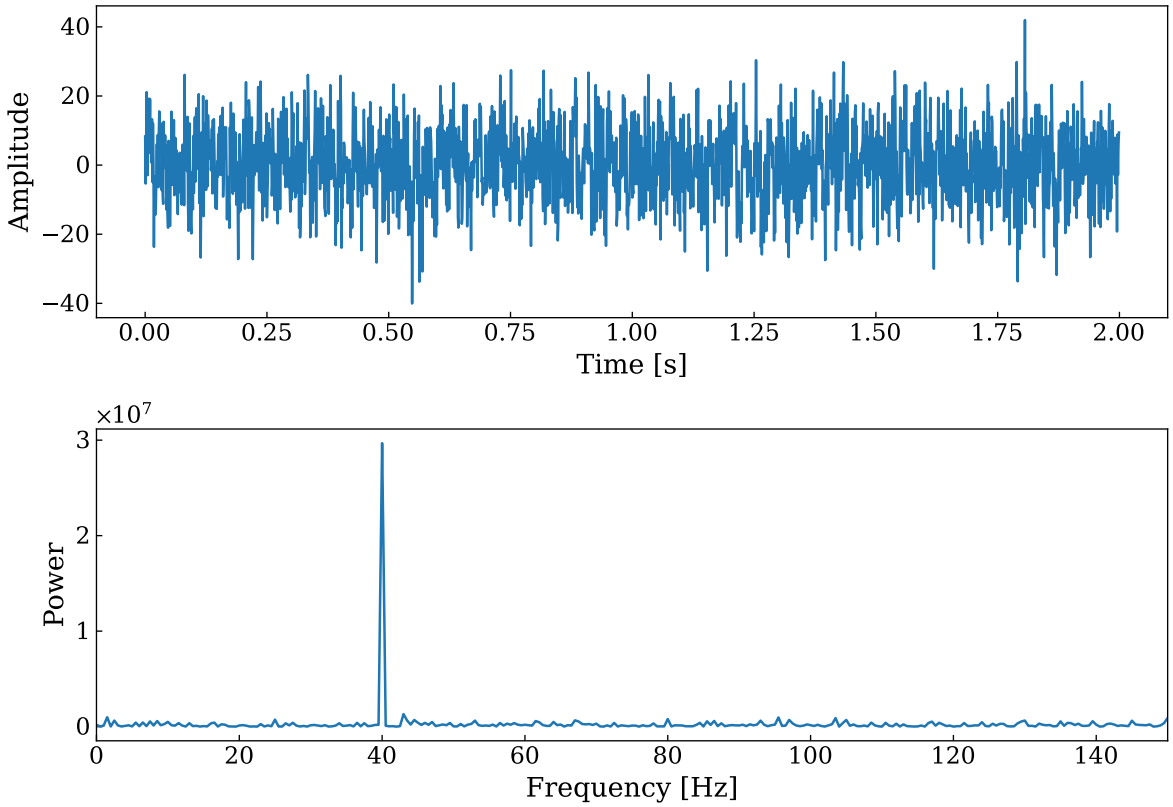
In this section, a noisy sine wave has been generated and different types of filters have been applied to it. The goal is to compare the working of Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters in removing noise from a signal.

Listing 1: **Code snippet for generating the noisy sine wave.**

```
N = 2000
A = 5
f = 40
fs = 1000

dt = 1 / fs
times = dt * np.arange(N)
wave = A * np.sin(2 * np.pi * f * times) + A * 2 * np.random.randn(N)


def power_spectrum(
    *, signal: npt.NDArray, fs: int
) -> Tuple[npt.NDArray, npt.NDArray]:
    spectrum = np.fft.fft(signal)

    spectrum = np.abs(spectrum) ** 2
    spectrum = np.roll(spectrum, spectrum.size // 2)

    freqs = np.fft.fftfreq(signal.size, 1 / fs)
    freqs = np.roll(freqs, freqs.size // 2)

    return freqs, spectrum

freqs, spect = power_spectrum(signal=wave, fs=fs)
```

This code generates a sine wave with a frequency of 40 [Hz] and an amplitude of 5, sampled at a rate of 1000 [Hz]. The sine wave is then corrupted with random noise.

Figure 1: **Noisy sine wave.**



As can be seen from the figure 1, the noise makes it difficult to distinguish the underlying sine wave. In the next sections, different types of filters have been applied to remove the noise and recover the original signal.

Three types of filters have been tested: high-pass, low-pass, and band-pass, for FIR and IIR.

1.1 FIR filters

The filters are designed using the `scipy.signal` library, with a filter order of 101. The high-pass filter has a cutoff frequency of 42 [Hz], the low-pass filter has a cutoff frequency of 38 [Hz], and the band-pass filter has a passband of 38-42 [Hz].

The frequency responses are shown in Figures 2, 3, and 4.

Listing 2: **Code snippet for applying the FIR filters.**

```
nyq = fs / 2
order = 101

filters = {
    "highpass": signal.firwin(order, 42 / nyq),
    "lowpass": signal.firwin(order, 38 / nyq, pass_zero=False),
    "bandpass": signal.firwin(
        order, [38 / nyq, 42 / nyq], pass_zero=False
    ),
```

```
}

for filter_name, coefs in filters.items():
    fig, [top, mid, bot] = plt.subplots(3, 1)

    result = signal.lfilter(coefs, 1, wave)

    w, h = signal.freqz(b, a, worN=2000)
    top.plot(w * nyq / np.pi, np.abs(h))
    ...
    SNR = 10 * np.log10(signaltonoise(np.abs(result)))
    ...
```

Figure 2: **High-pass FIR filter.**

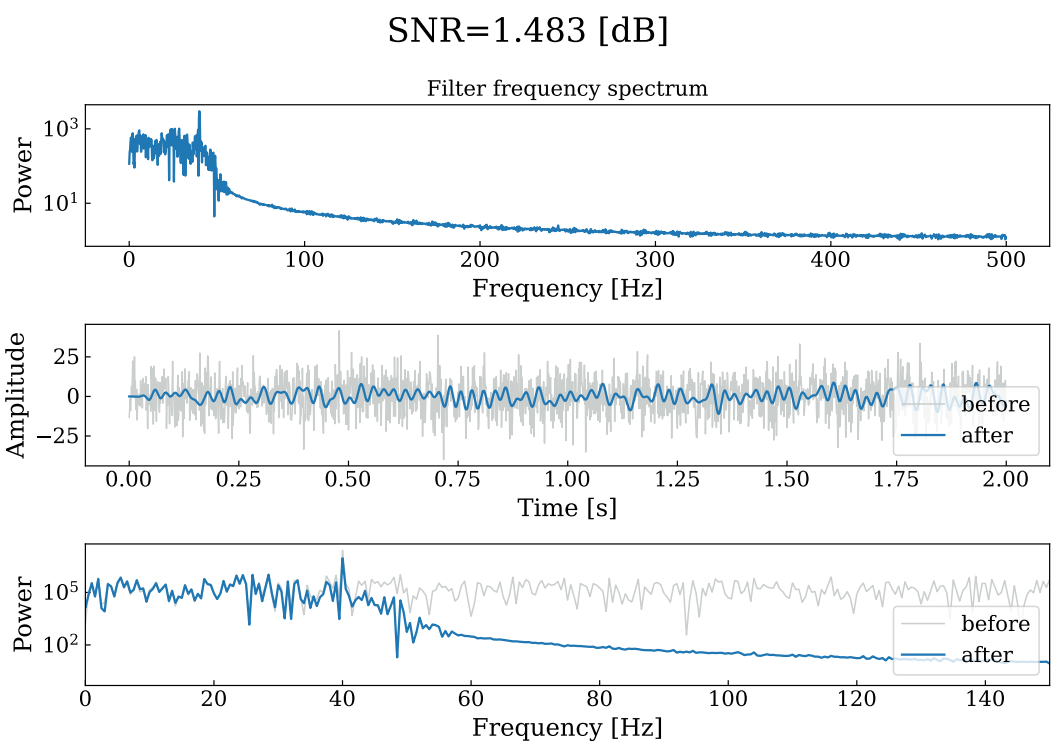


Figure 3: **Low-pass FIR filter.**

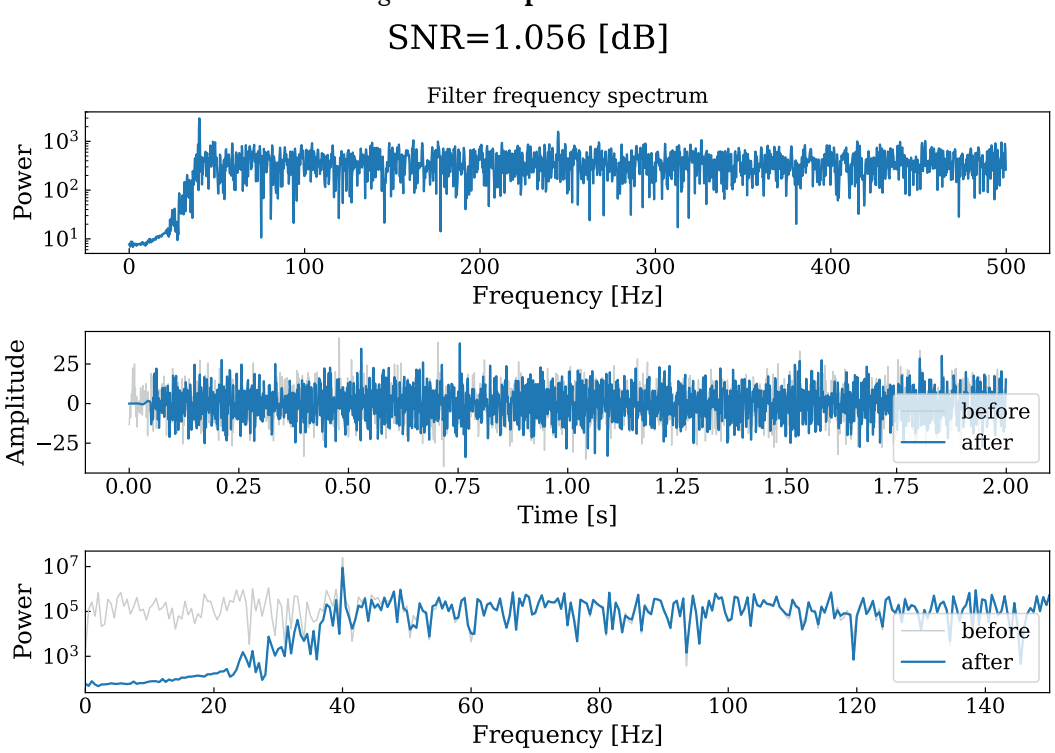
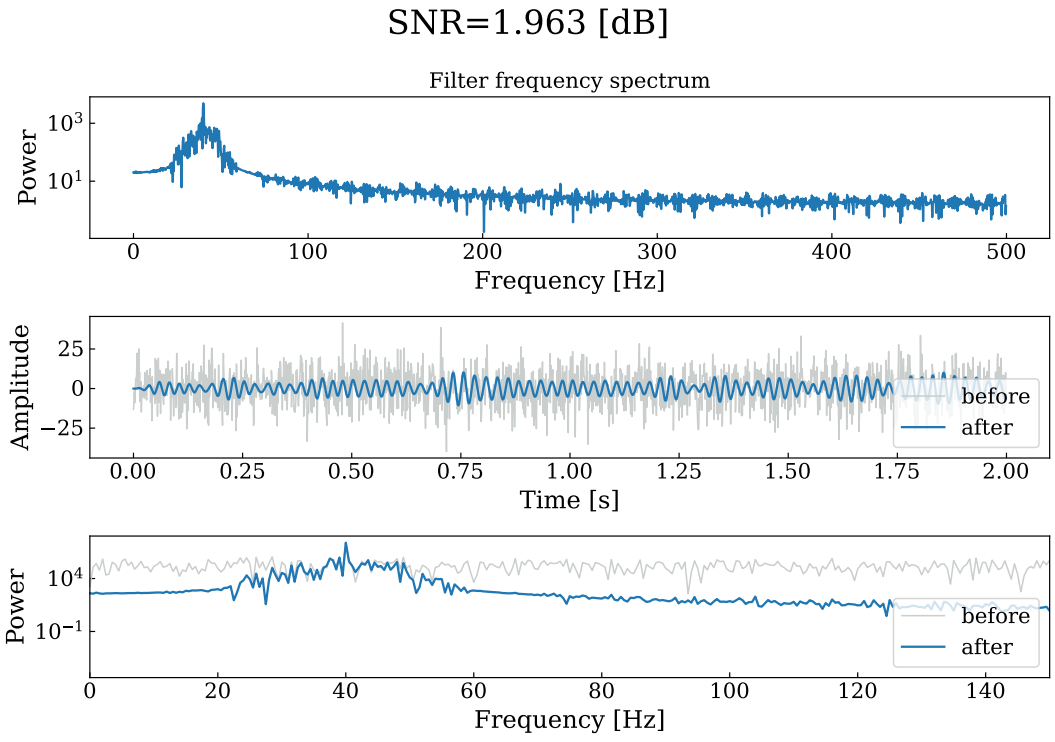


Figure 4: **Band-pass FIR filter.**



1.2 IIR filters

The IIR filters have been designed also by using the `scipy.signal` library, with a filter order of 4. Both the high-pass and low-pass filters have a cutoff frequency of 42 [Hz], and the band-pass filter has a passband of 38-42 [Hz].

Their frequency responses are shown in Figures 5, 6, and 7.

Listing 3: **Code snippet for applying the IIR filters**.

```
nyq = fs / 2
order = 4

filters = {
    "highpass": signal.butter(order, 42 / nyq, btype="highpass"),
    "lowpass": signal.butter(order, 42 / nyq, btype="lowpass"),
    "bandpass": signal.butter(
        order, [38 / nyq, 42 / nyq], btype="bandpass"
    ),
}

for filter_name, (b, a) in filters.items():
    fig, [top, mid, bot] = plt.subplots(3, 1)

    result = signal.lfilter(b, a, wave0)

    w, h = signal.freqz(b, a, worN=2000)
    top.plot(w * nyq / np.pi, np.abs(h))
    ...
    SNR = 10 * np.log10(signaltonoise(np.abs(result)))
    ...
```

Figure 5: **High-pass IIR filter.**

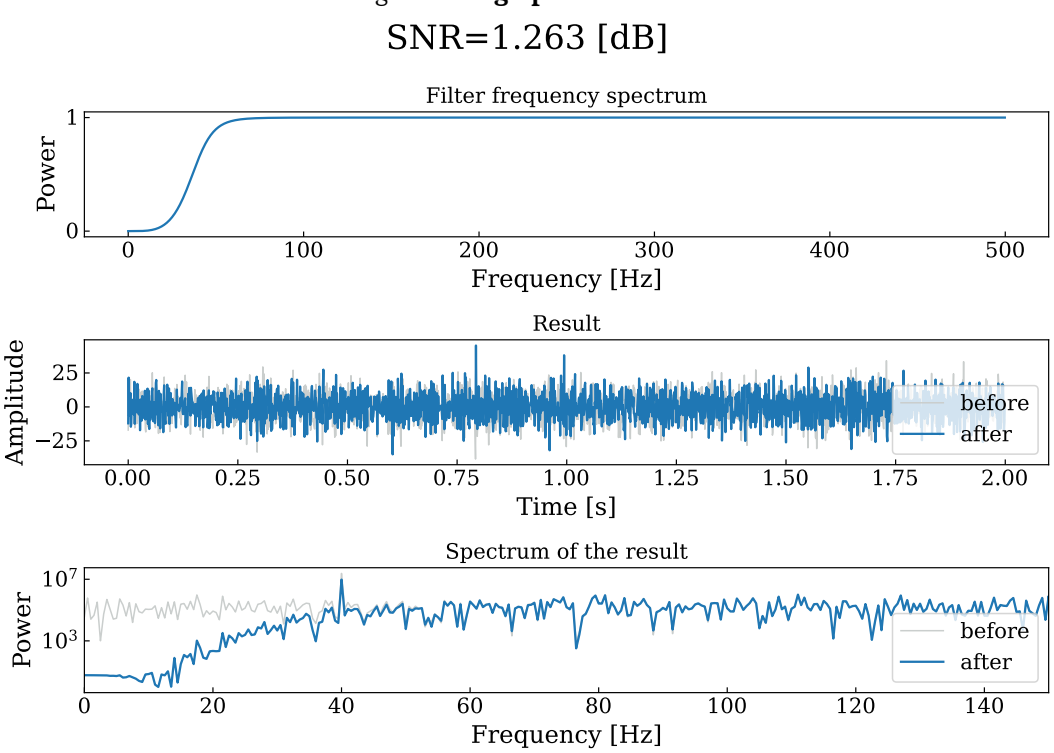


Figure 6: **Low-pass IIR filter.**

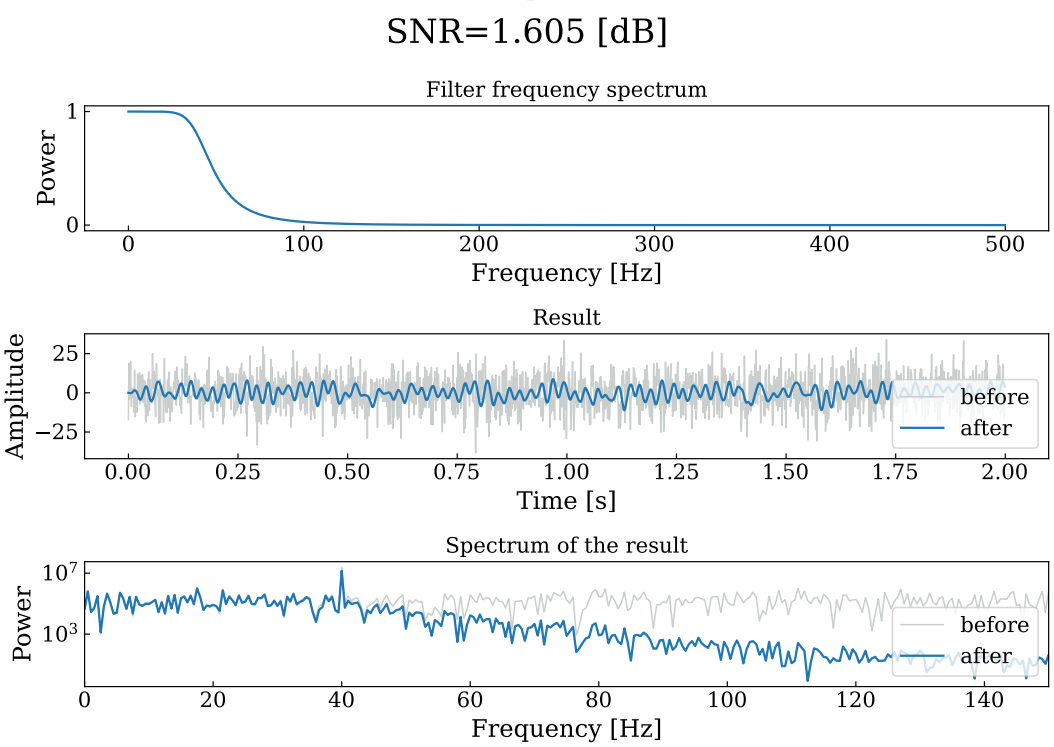
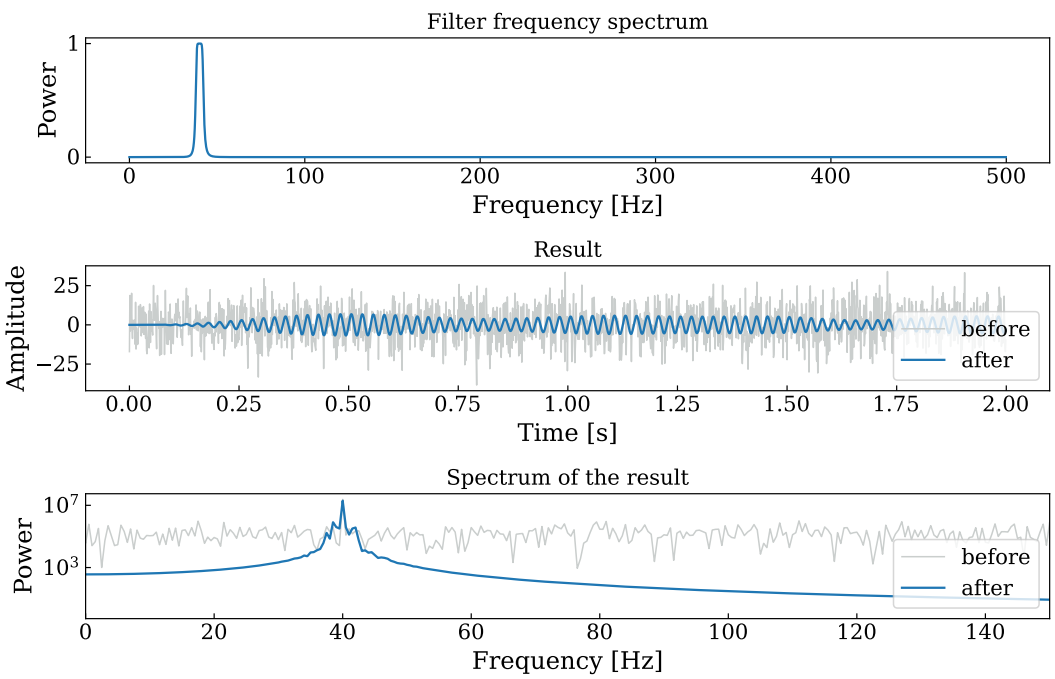


Figure 7: **Band-pass IIR filter.**



As can be seen from the figures, based on the calculated SNR for each filter, the IIR filters are more effective in removing the noise and recovering the original signal.

2 Two sine waves

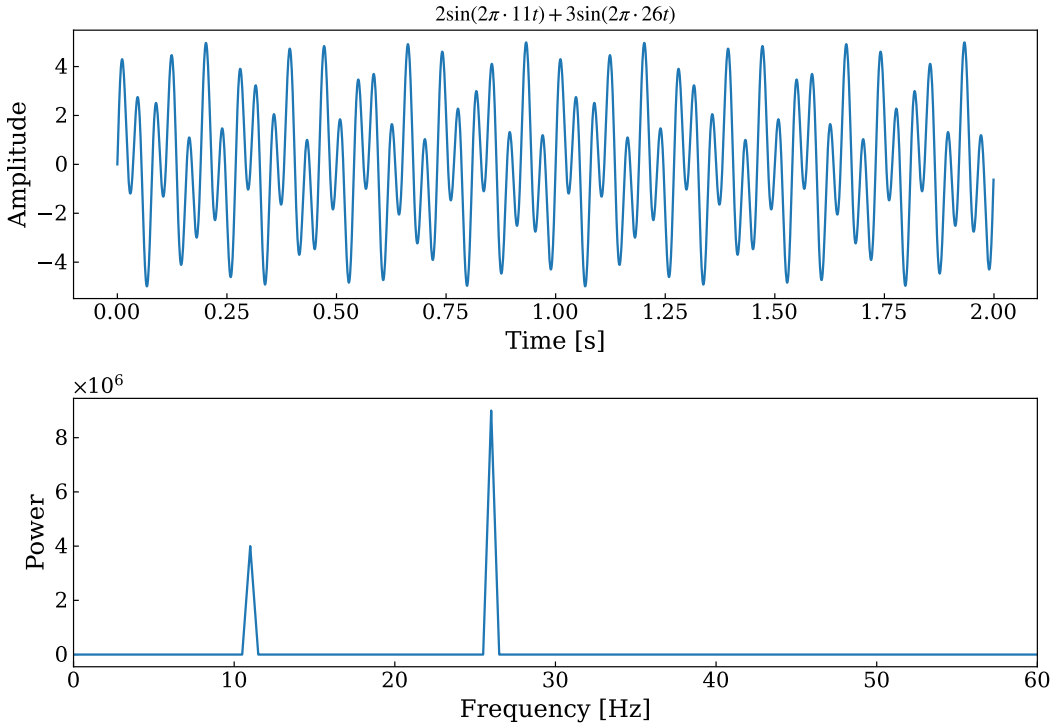
In this section, a sum of two sine waves has been generated. Then, the working of high-pass and low-pass filters have been tested on it. The goal is to demonstrate the ability of these filters to separate two signals with different frequencies.

The sum of two sine waves is generated using the following code:

Listing 4: **Code snippet for generating the two sine waves.**

```
N = 2000
fs = 1000

f1 = 11
f2 = 26
A1 = 2
A2 = 3

times = np.arange(N) / fs
wave = A1 * np.sin(2 * np.pi * f1 * times) + \
       A2 * np.sin(2 * np.pi * f2 * times)

freqs, spect = power_spectrum(signal=wave, fs=fs)
```

Figure 8: **Sum of two sine waves.**



A high-pass filter and a low-pass filter have been applied to the sum of two sine waves, with cutoff frequencies of 1 [Hz] and 36 [Hz], respectively. The frequency responses of these filters are shown in Figures 9 and 10.

Listing 5: **Code snippet for generating the two sine waves.**

```
nyq = fs / 2
order = 5

filters = {
    "highpass": signal.butter(order, (f1 + 10) / nyq, btype="highpass"),
    "lowpass": signal.butter(order, (f2 - 10) / nyq, btype="lowpass"),
}

for filter_name, (b, a) in filters.items():
    fig, [top, mid, bot] = plt.subplots(3, 1)

    result = signal.lfilter(b, a, wave0)

    w, h = signal.freqz(b, a, worN=2000)
    top.plot(w * nyq / np.pi, np.abs(h))
    ...
    SNR = 10 * np.log10(signaltonoise(np.abs(result)))
    ...
```

As can be seen from the figures, the high-pass filter effectively removes the 11 [Hz] sine wave. The low-pass filter also removes the 26 [Hz] sine wave, however with some distortion left in the resulting signal.

Figure 9: **Result of high-pass filter on two sines.**

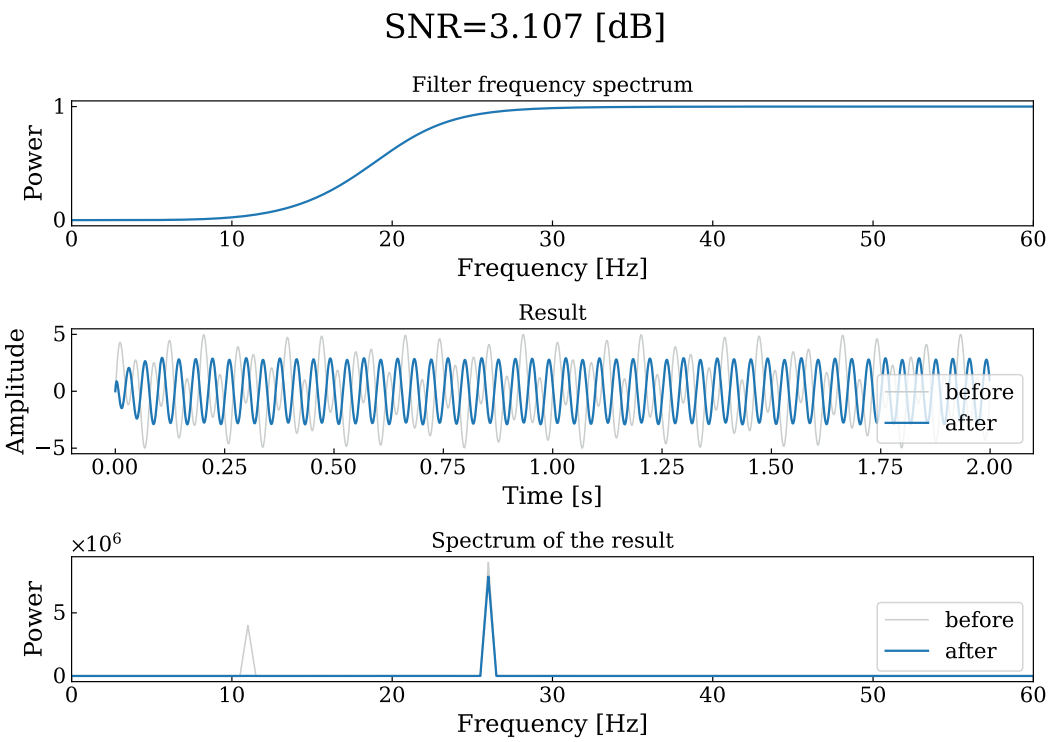
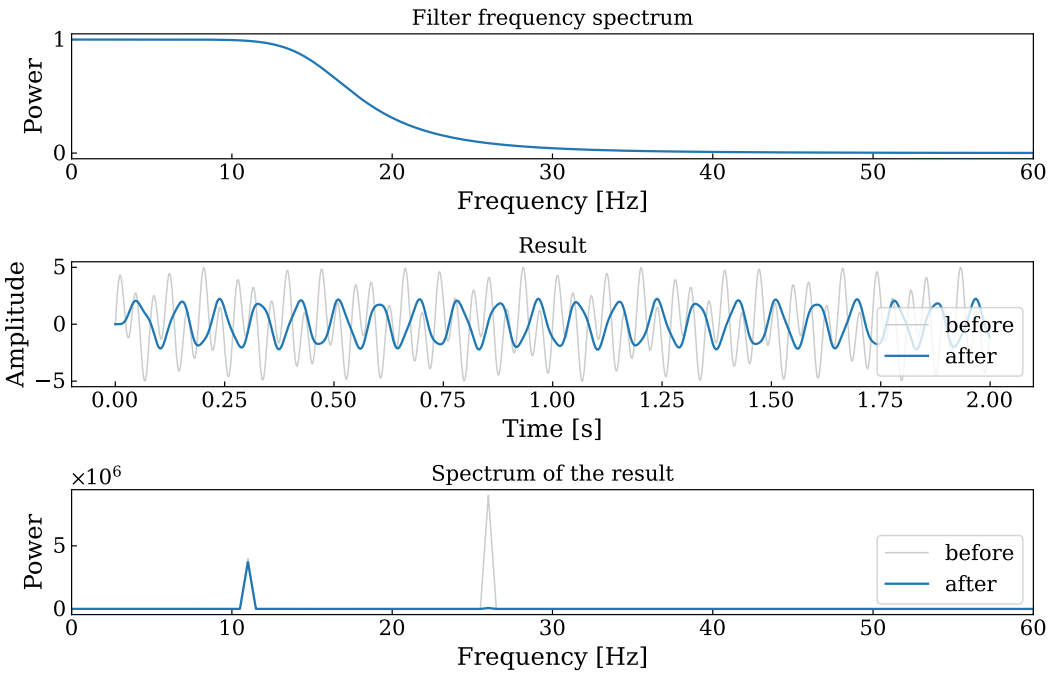


Figure 10: **Result of low-pass filter on two sines.**

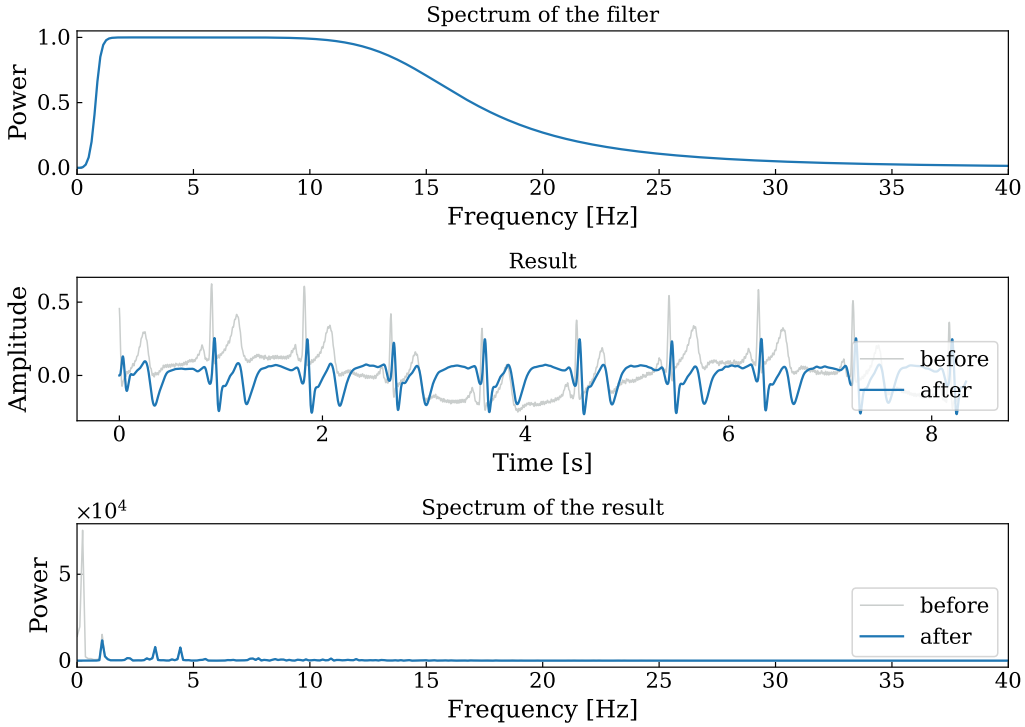


3 Noisy ECG signal

In this last section, the filters have been tried on a real-world signal, specifically an electrocardiogram (ECG) signal. The ECG signal is a biomedical signal that measures the electrical activity of the heart. It has been defined by the provided file `ecg.mat`.

To successfully clean the polluted signal, a band-pass filter has been used, with a passband of 0.9-15 [Hz].

Figure 11: **ECG before and after applying the filter**.



As can be seen from the figure 11, the band-pass filter is enough to largely remove the noise and artifacts from the ECG signal.

4 Conclusion

In this report, I have demonstrated the use of FIR and IIR filters to remove noise and artifacts from signals. I have applied these filters to a noisy sine wave, a sum of two sine waves, and a real-world ECG signal. The results show that the filters are effective in removing noise and recovering the original signal.

The entire code for generating the data and plots can be found at:

https://github.com/davkk/signal-analysis/tree/main/sat/lab04