University *of Ljubljana*
Faculty *of Mathematics and Physics*

Department of Physics

Filtering and Spectral Analysis

10. Task for Model Analysis I, 2023/24

Author: Marko Urbanč
Professor: Prof. Dr. Simon Širca
Advisor: doc. dr. Miha Mihovilovič

Ljubljana, August 2024

Contents

1 Introduction

Today we're taking a look at the filtering and spectral analysis of signals. Filtering is a process of removing unwanted parts of a signal, while spectral analysis is a process of decomposing a signal into its frequency components. Both of these processes are crucial in signal processing and would not be possible without the Fourier transform. The Fourier transform is a mathematical operation that transforms a function of time into a function of frequency. It is used to represent the signal as a sum of sinusoidal functions from which we can extract important frequency information. The equation for the Fourier transform and its inverse are given by:

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \mathrm{d}t \,, \tag{1}$$

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{i\omega t} \mathrm{d}\omega \,. \tag{2}$$

The Fourier transform has various properties that we've discussed in other tasks. What will turn out to be significant in this task is the fact that the Fourier transform *imagines/expects* that the input signal is periodic. This is important because the Fourier transform of a signal that is not periodic will be subject to various effects of aliasing and leakage. These effects can be mitigated by windowing the signal before applying the Fourier transform. Windowing is a process of multiplying the signal by a window function that is zero outside of a certain interval. This effectively makes the signal periodic and reduces the effects of aliasing and leakage. Figure 1 shows some common window functions that are used in signal processing.

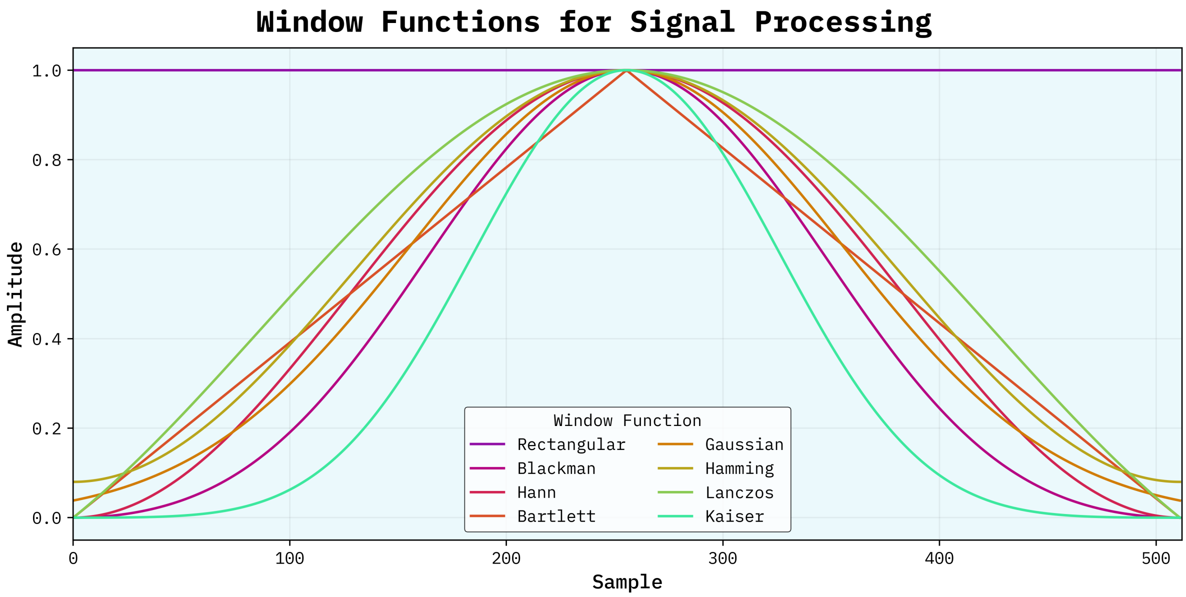

Figure 1: Common window functions used in signal processing.

An important operation in signal processing is the convolution. Convolution is a mathematical operation that combines two signals to produce a third signal. It is used to model the effect of one signal on another signal. The convolution of two signals $f(t)$ and $g(t)$ is given by:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\mathrm{d}\tau \, . \tag{3}$$

The convolution operation is commonly used in filtering. Filtering is a process of removing unwanted parts of a signal. Filters can be divided into roughly two categories: low-pass filters and high-pass filters. Low-pass filters allow (or pass) low-frequency signals and block high-frequency signals, while high-pass filters do the opposite. Filters can be implemented in the time domain or in the frequency domain. In the time domain, filters are implemented as convolution operations, while in the frequency domain, filters are implemented as multiplication operations. For todays task we'll take a look at **Wiener's (Optimal) Filter**. Wiener's filter is an optimal filter that minimizes the mean square error between the estimated random process (noise) and the desired process (signal). Imagine we have a signal $u(t)$ which we measure using a sensor with the transfer function $r(t)$. The signal with the addition of noise $n(t)$ is then given by:

$$c(t) = u(t) * r(t) + n(t) = s(t) + n(t) \, , \tag{4}$$

where $*$ denotes the convolution operation. From the measured quantity $c(t)$ we want to reconstruct the signal $u(t)$, given the fact that we have some information on the noise $n(t)$ and the sensors response $r(t)$. Following analogously to the Least Squares method, Wiener proposed a filter in which we have to multiply the Fourier transform of the measured signal $\hat{c}(\omega)$ with:

$$\Phi(\omega) = \frac{|\hat{s}(\omega)|^2}{|\hat{s}(\omega)|^2 + |\hat{n}(\omega)|^2} \, . \tag{5}$$

We can also perform the so-called Wiener deconvolution using the Wiener filter and a convolution kernel (which is the transfer function of the sensor). So in the case of image restoration we can use the Wiener filter to remove the noise from the image if we know the transfer function of the sensor, which could for example be the point spread function of the camera with leads to blurring of the image. There are many other methods for image restoration, but Wiener's filter is a good starting point and we'll limit ourselves to this method in this task.

2 Task

2.1 Spectra of Signals

In the first subtask, the instructions want us to calculate the spectra of signals, that were provided in `val2.dat` and `val3.dat`. We should try out different windowing functions to see how they affect the spectra. We can also try and see what happens if we only select a part of the signal and calculate the spectrum of that part. Figure 2 shows the signals we've been given and their raw spectra.

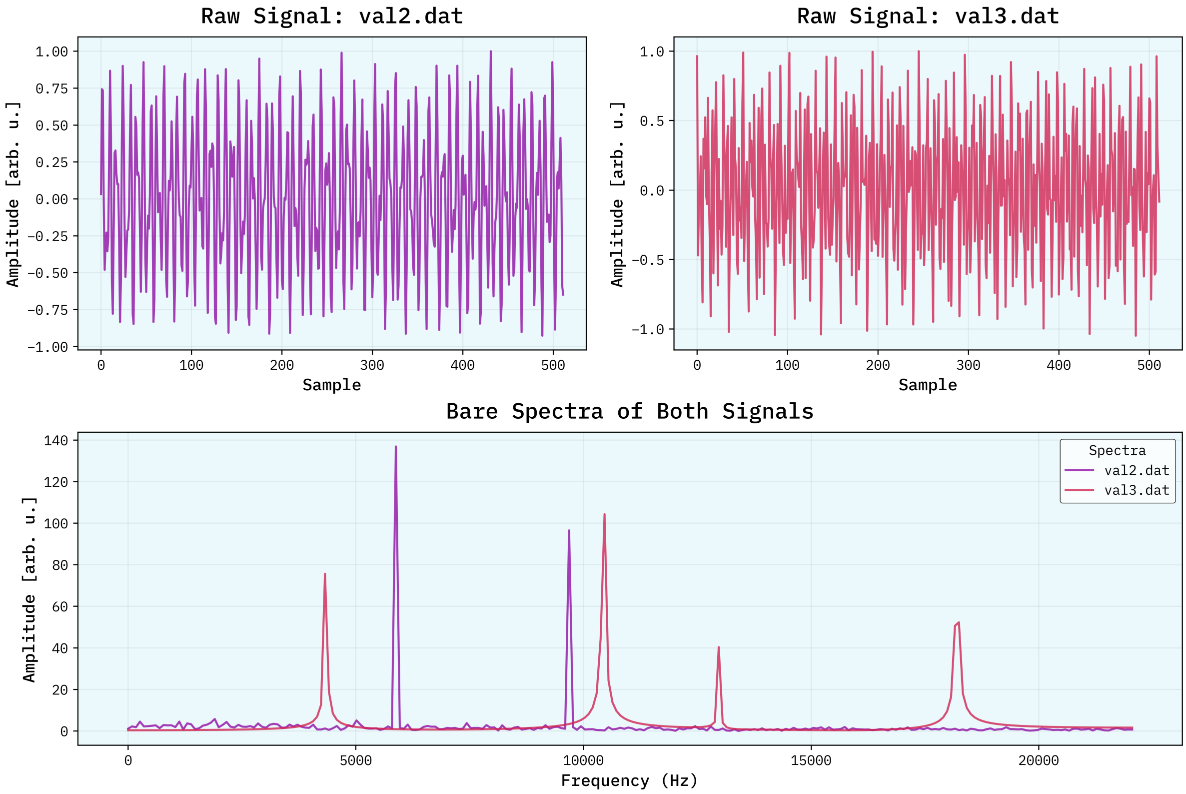

Figure 2: Signals `val2.dat` and `val3.dat` and their raw spectra.

2.2 Wiener Filtering

We have signals `signal{0,1,2,3}.dat` provided for the second task, each 512 samples long. Using Wiener's Filter we should try and remove the noise from the signals. `signal0.dat` represents the noiseless signal while the other signals have increasing levels of noise added to them. The transfer function of the sensor is given by:

$$r(t) = \frac{1}{2\tau} e^{-|t|/\tau} \,, \qquad \text{where} \quad \tau = 16\,. \tag{6}$$

Figure 3 shows the signals we've been given and Figure 4 shows the spectra of the signals.

2.3 Wieners Deconvolution

For the last subtask we've received (cropped) images of Playboy model Lena Forsen (previously Soderberg). Her portrait called `Lenna` has become the standard test for various image processing algorithms and techniques. We've been given images of Lena that have been damaged by the addition of one of three convolution kernels and increasing levels of noise. The instructions want us to use Wiener's deconvolution to restore the images as best we can making sure to take care of artifacting due to a non-periodic signal by using either windowing or zero-padding. For the final challenge we we're also given images that have an additional periodic perturbation to them. We should try and remove the periodic perturbation from the images using some form of frequency domain filtering. Figure 5 shows some of the images we've been given and their matching convolution kernels.

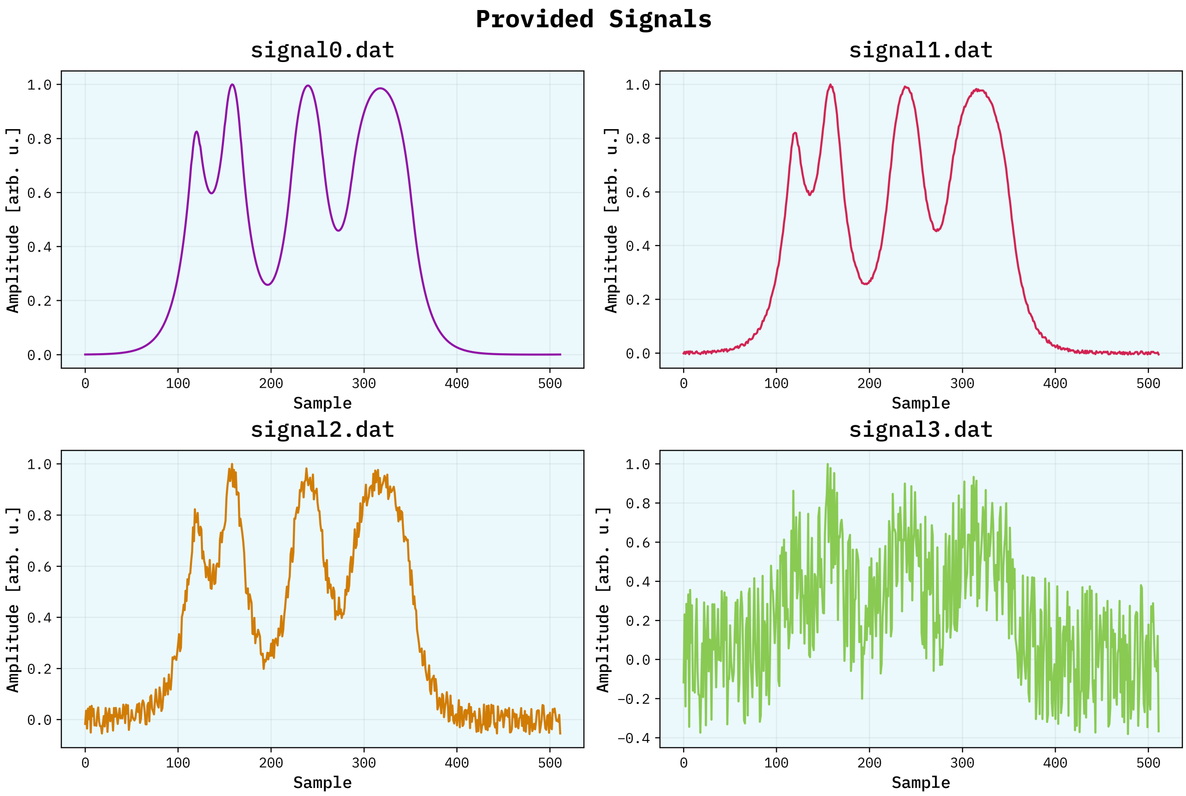

Figure 3: Signals `signal{0,1,2,3}.dat` and their convolution kernel.

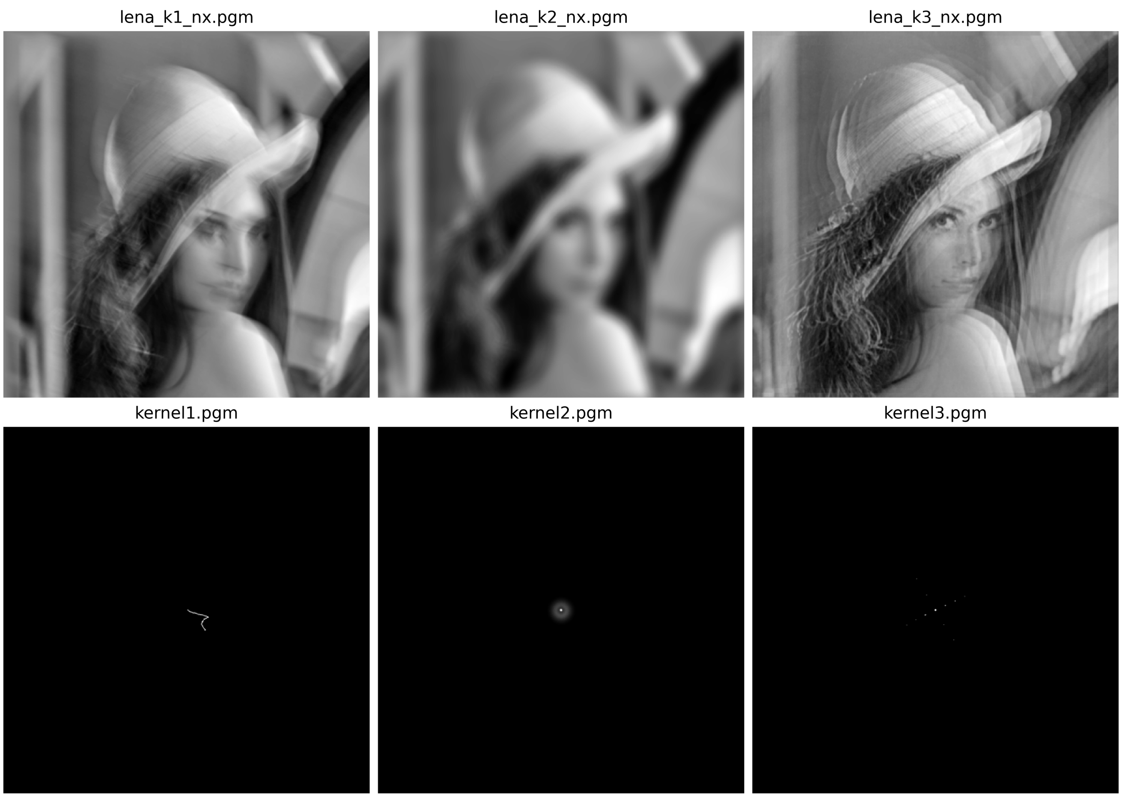

Figure 5: Images of Lena Forsen and their matching convolution kernels.

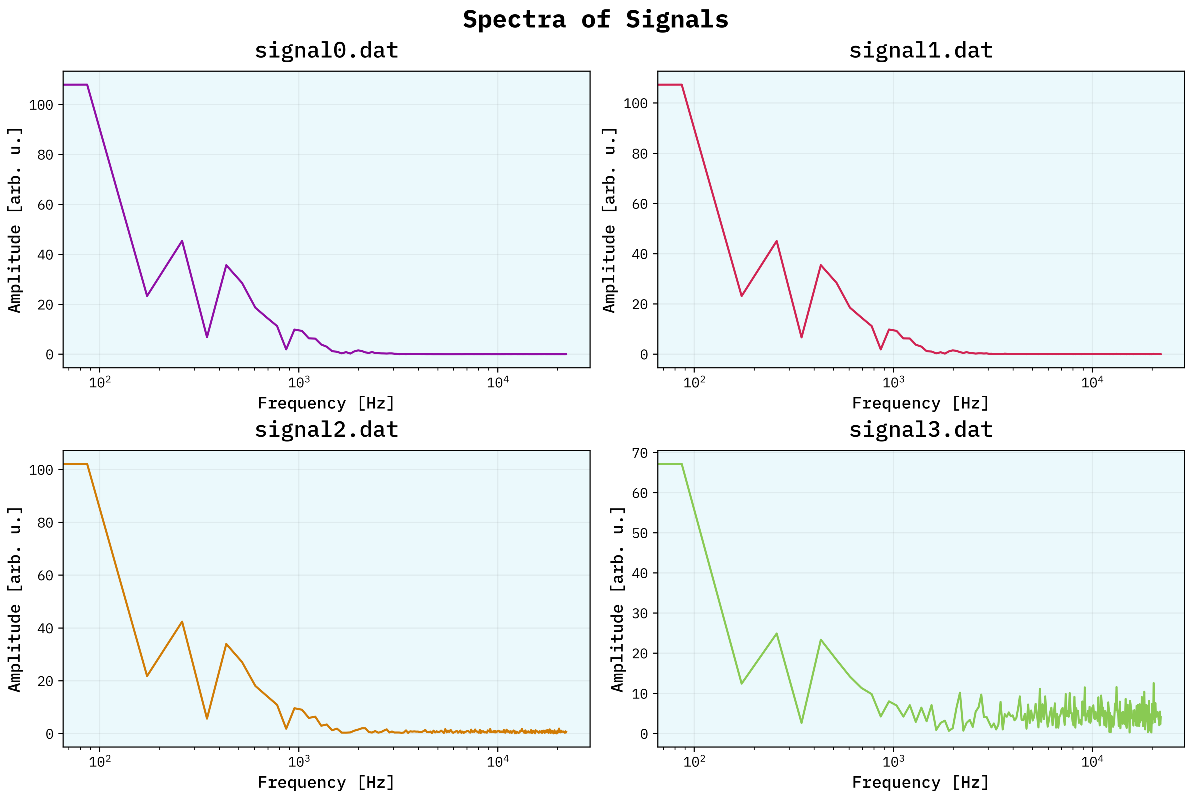

Figure 4: Spectra of signals `signal{0,1,2,3}.dat`.

3 Solution Overview

Another core mantra I want my stubborn brain to learn is to use already existing libraries and tools to solve problems. Sure I think it would be much more educational to write all the presented methods from scratch but that is unfortunately time consuming and thus not very practical. As this task doesn't really include bulk data gathering I also didn't make use of any multiprocessing, threading or distributed computing via say a package like `ray`. Besides the Python data science gold standard `numpy` and `scipy` I also used `scikit-image` for its plethora of already implemented image processing algorithms. Especially the submodule `skimage.restoration` was very useful as it already contains both Wiener's filter and Wiener's deconvolution. The rest of the task was mostly about reading in the data, applying and adjusting the filters etc. and plotting the results using `matplotlib`. As I didn't do any parameter scans I didn't really se the use of taking a class-based approach. Oh I'd also like to mention that I used a sample rate of 44.1 kHz for the spectral analysis of the signals. This is because I wanted to imagine the signals as audio signals which I best understand. I also tried to have some fun with them by reconstructing the signals from their spectra using `Audacity`.

4 Results

4.1 Spectra of Signals

The spectra of the signals `val2.dat` and `val3.dat` are shown in Figure 2. We can see very clearly the prominent peaks in the spectra. It is also evident that peaks in `val3.dat` are much wider than in `val2.dat`. I assume they are subject to leakage effects due to the signal end-points not matching this making the signal non-periodic. I tested this theory out by performing a *Ghetto Periodicity Fix™* where I set the last value of the signal to the first value called *1-point Fix™* and the last 10 points to the first value called *10-point Fix™*. The results are shown in Figure 6.

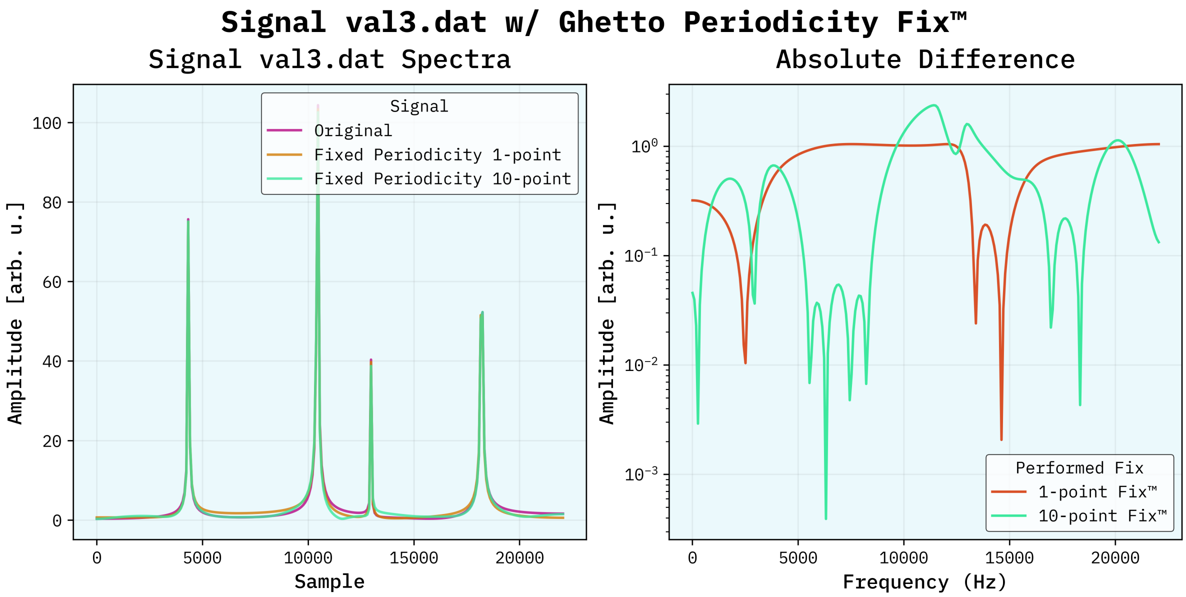

Figure 6: Spectra of signals `val3.dat` with and without the *Ghetto Periodicity Fix™*.

We see that the stupid methods of *1-point Fix™* and *10-point Fix™* actually work and make the spectral lines much sharper. This would however be greatly improved using a proper windowing function and with that we can move on to the next Figures where we do just that. Figures 7 and 8 show the absolute difference between the bare unwindowed spectra and the windowed spectra of the two signals `val2.dat` and `val3.dat` with different windowing functions applied. Do note that all the spectra have been normalized for easier comparison.

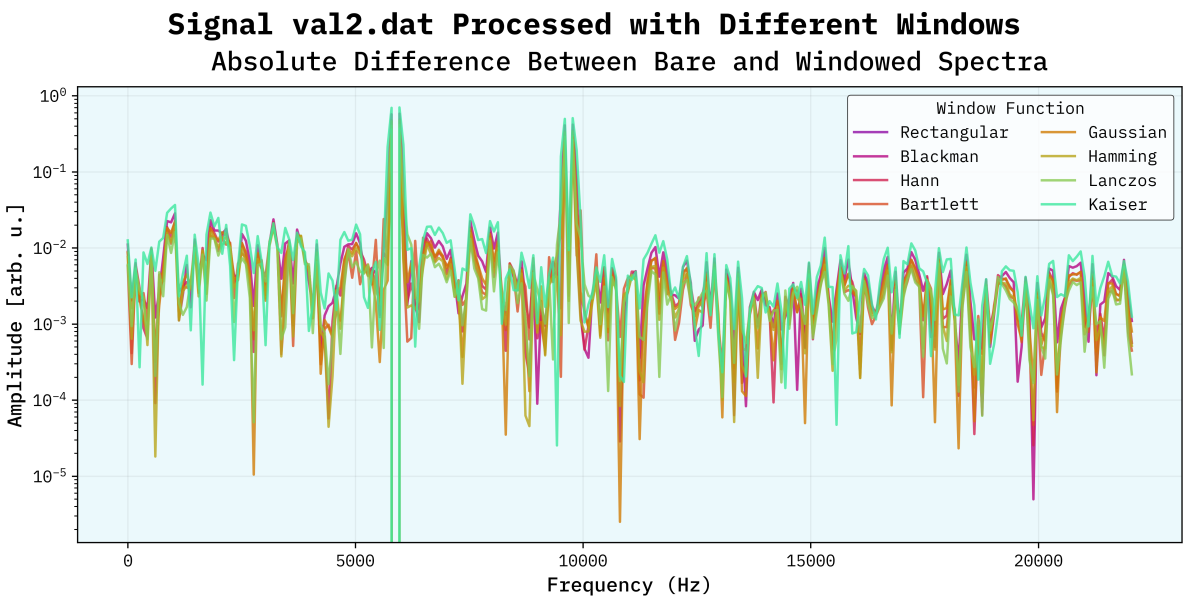

Figure 7: Spectra of signal `val2.dat` with different windowing functions applied.

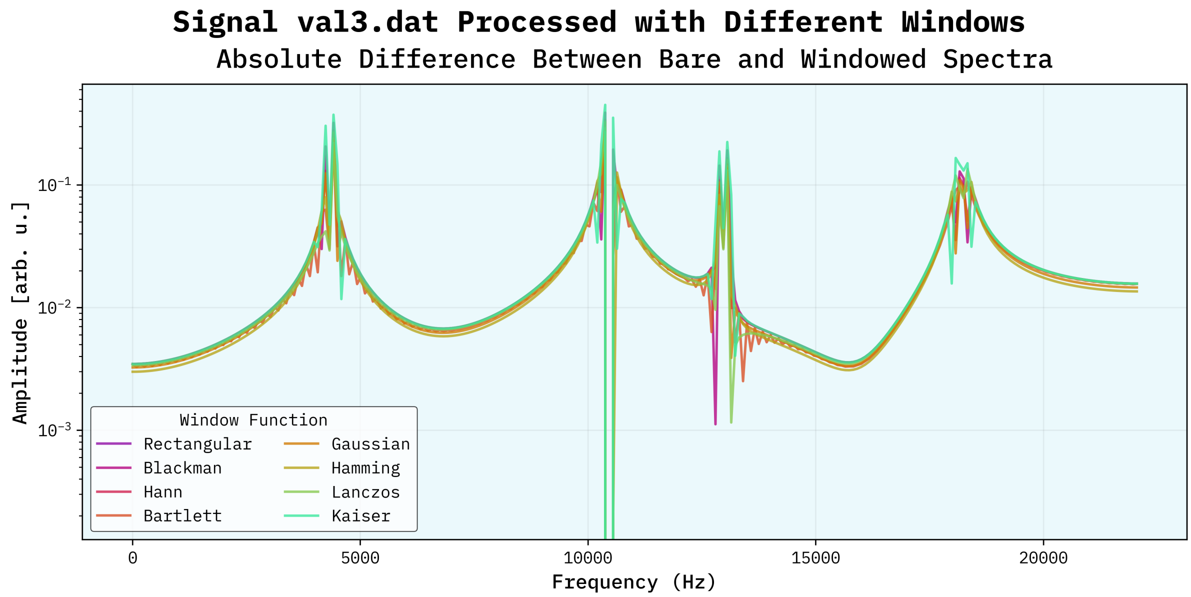

Figure 8: Spectra of signal `val3.dat` with different windowing functions applied.

A larger difference here is not bad since we actually want to remove the leakage effects. For the `val2.dat` signal I think that mostly the choice of window does not matter as much. All windows besides the Rectangular, Kaiser and Gaussian seem to produce similar results. Likewise for the `val3.dat` signal. It is interesting to note that the Hann window seems to cause some weird oscillatory behavior around the peaks in the spectrum. This is probably due to the fact that the Hann window has a sharper cutoff at its edges. Similar behavior can be observed in the Rectangular window, however it is a bit difficult to see due to the way the data is visualized. Since we're mainly interested in peak widths and heights I came up with a better way to visualize the differences. Using calculated peak heights and widths, which were computed at 0.1 relative height of the peaks, I plotted the differences between the peak dimensions for various windows. The results are shown in Figure 9 for the `val2.dat` signal and in Figure 10 for the `val3.dat` signal. Since the Rectangular window used here is essentially the same as no windowing I used it to compare other windows to relatively.

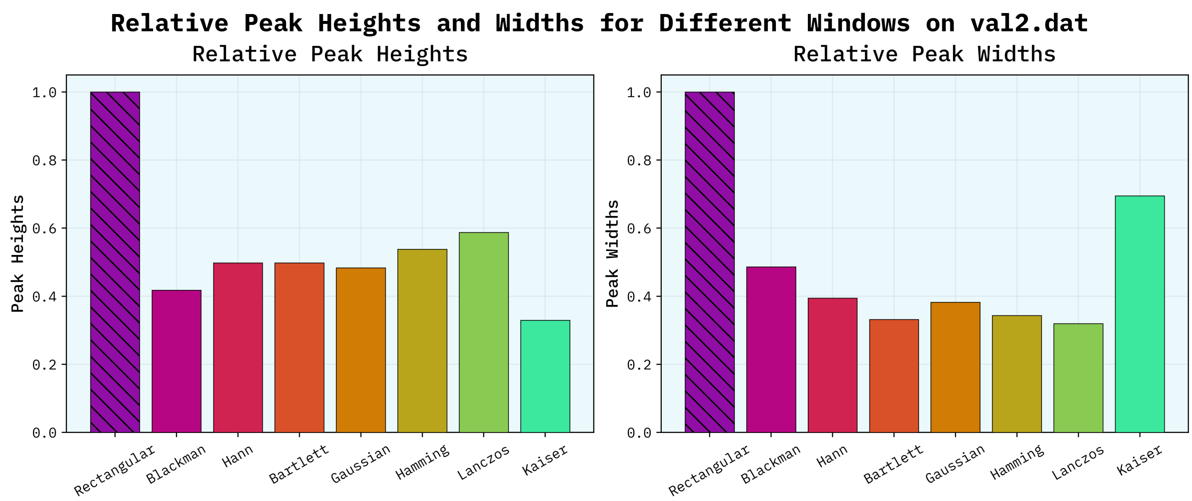

Figure 9: Differences in peak heights and widths for the `val2.dat` signal.

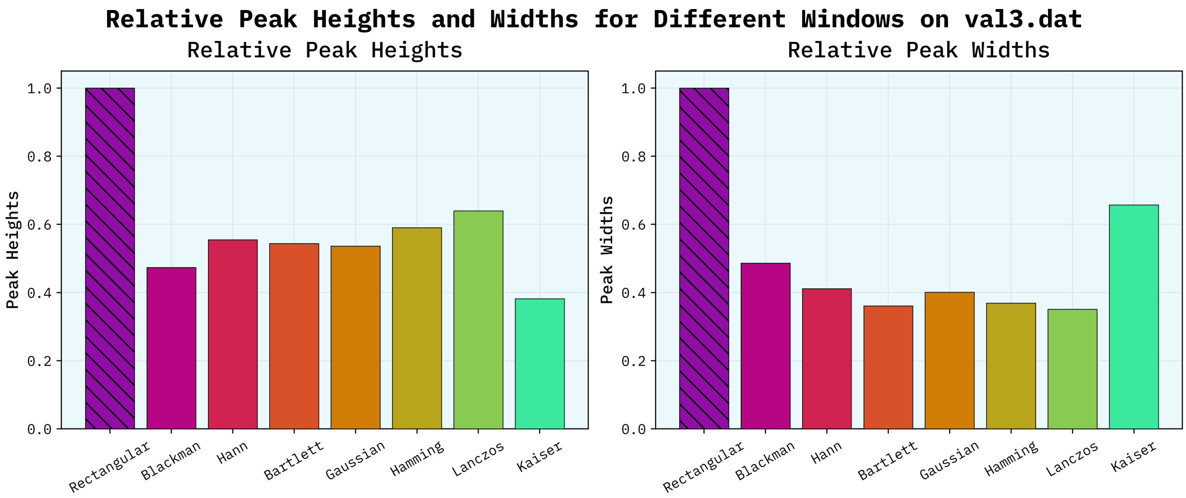

Figure 10: Differences in peak heights and widths for the `val3.dat` signal.

We can see that using windows makes sense in both cases. While we do sacrifice some peak intensity we gain in peak sharpness. The Rectangular window is the worst choice in both cases (as expected as it is essentially no windowing). For signal `val2.dat` the Lanczos window seems to be the best choice as it has the sharpest peak with the least intensity loss. The same holds true for the `val3.dat` signal. We can see however that the performance across windows is quite comparable. Even the Kaiser window could be a good choice if its width parameter β is chosen correctly.

We can also take a look at how the signal's spectra change when we only take a part of the signal. Figures 11 and 12 show the spectra of the signals `val2.dat` and `val3.dat` when only the first $256, 128, 64, 32, 16$ samples are taken. The plots present the spectra as well as the absolute and average relative differences between the full signal and the partial signals.

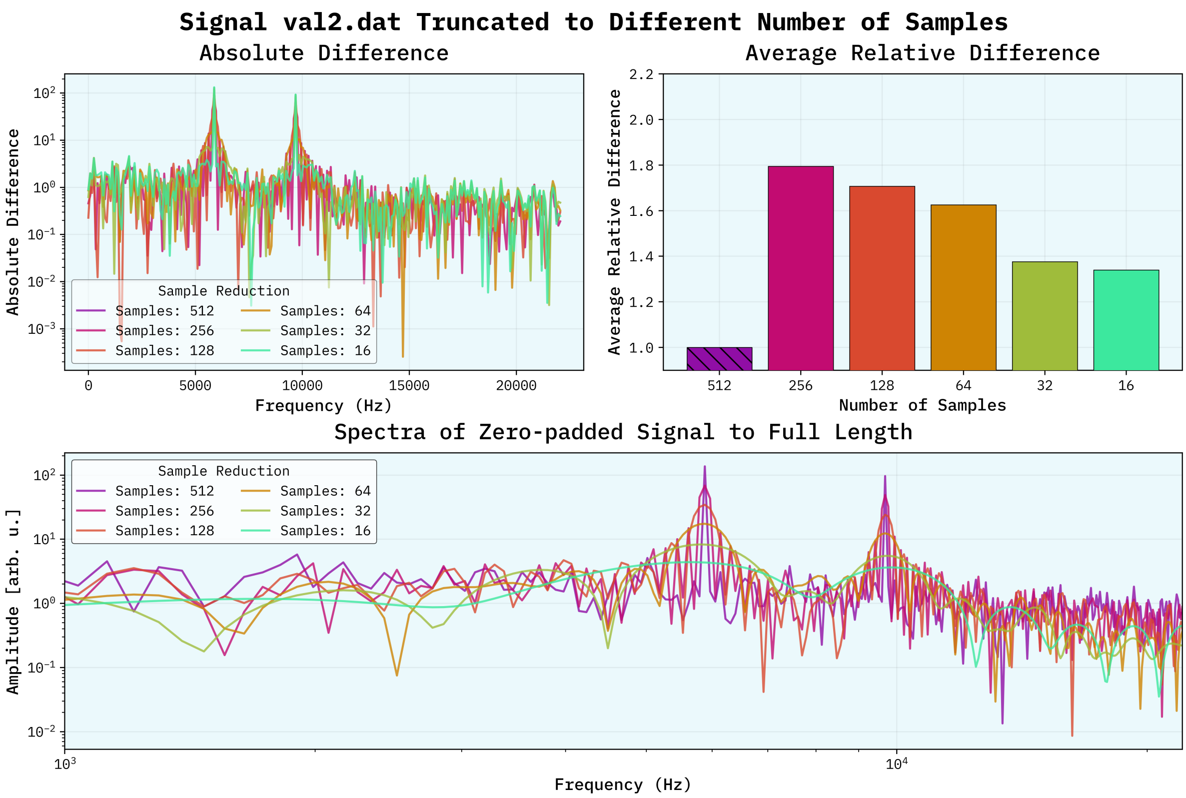

Figure 11: Spectra of signal `val2.dat` when only a part of the signal is taken.

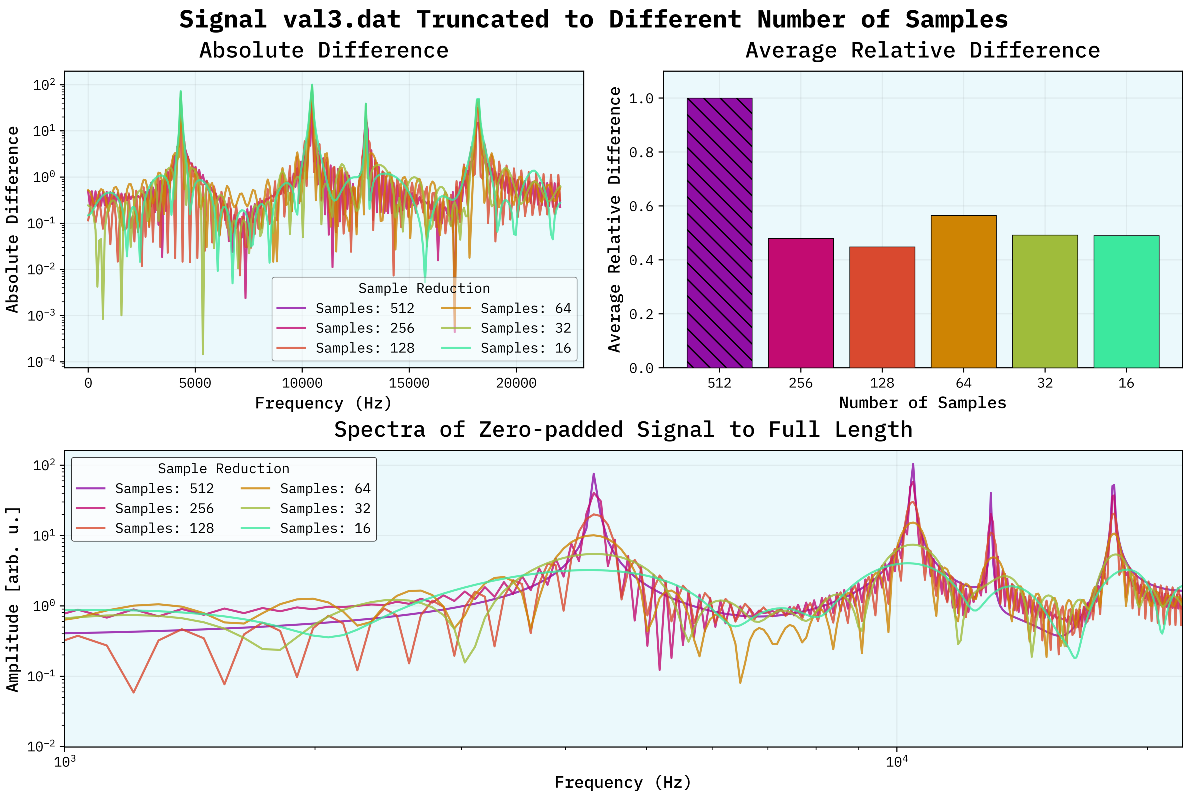

Figure 12: Spectra of signal `val3.dat` when only a part of the signal is taken.

I think it's safe to say that the spectra at 16 samples are essentially useless, however doubling the number of samples already gives us some noticable humps where the peaks should be. From this one could at least estimate the frequency content of a signal with very few samples. Its interesting to see how some of the spectra are broken due to non-periodicity of the samples used. This is especially evident in the `val3.dat` spectra for 128 samples and in the `val2.dat` spectra for 256 samples. I think that it is due to this effect that the matching average relative differences are comparably so high. All this could be greatly improved by using a proper windowing function but I wanted to demonstrate the raw effect of taking only a part of the signal.

For a little fun I also tried reconstructing the signals `val2.dat` and `val3.dat` from their spectra using `Audacity`. This was done by reading the peaks and their relative intensities from the spectra and then generating a signal from that. The results are shown in Figure 13 for the `val2.dat` signal and in Figure 14 for the `val3.dat` signal.

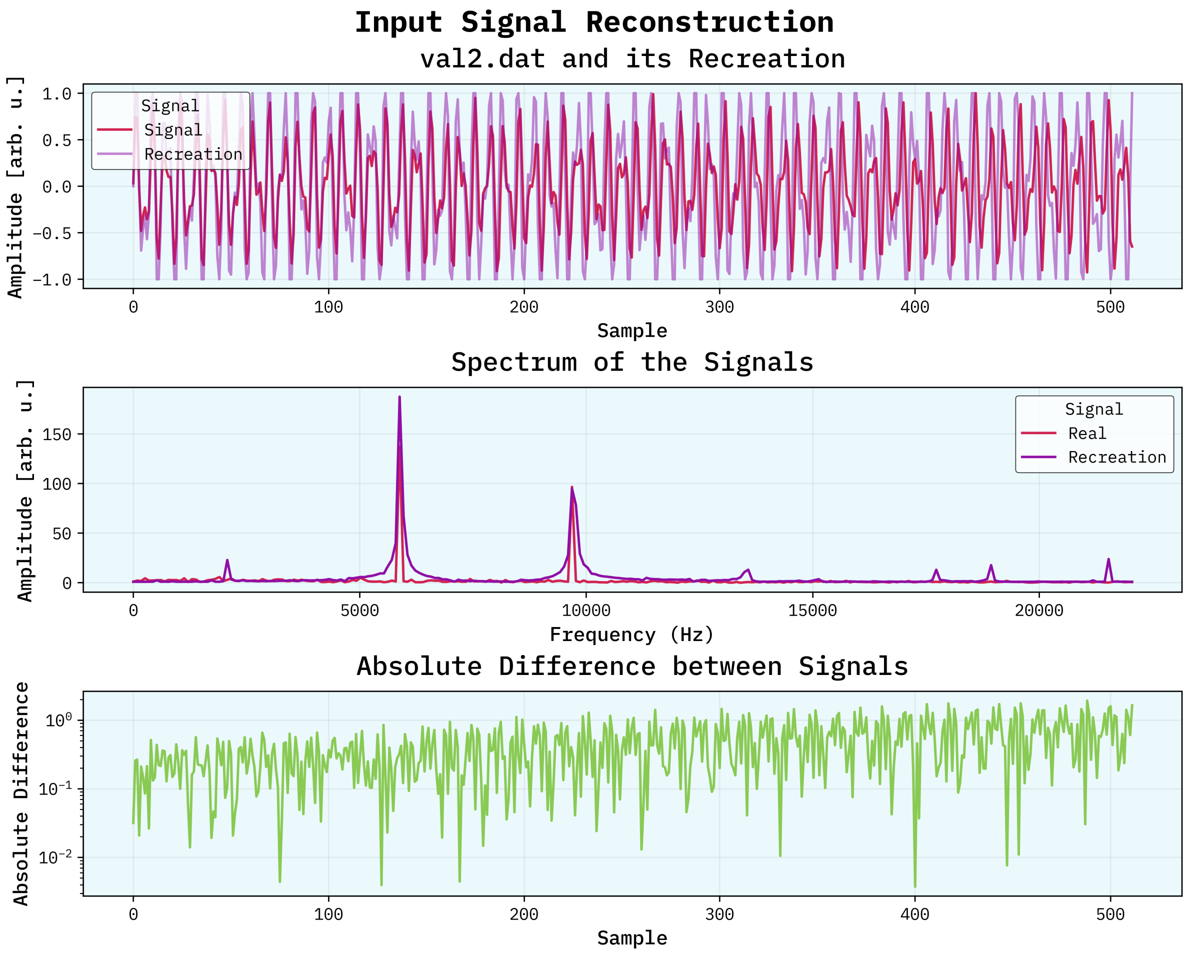

Figure 13: Reconstructed signal `val2.dat` from its spectrum.

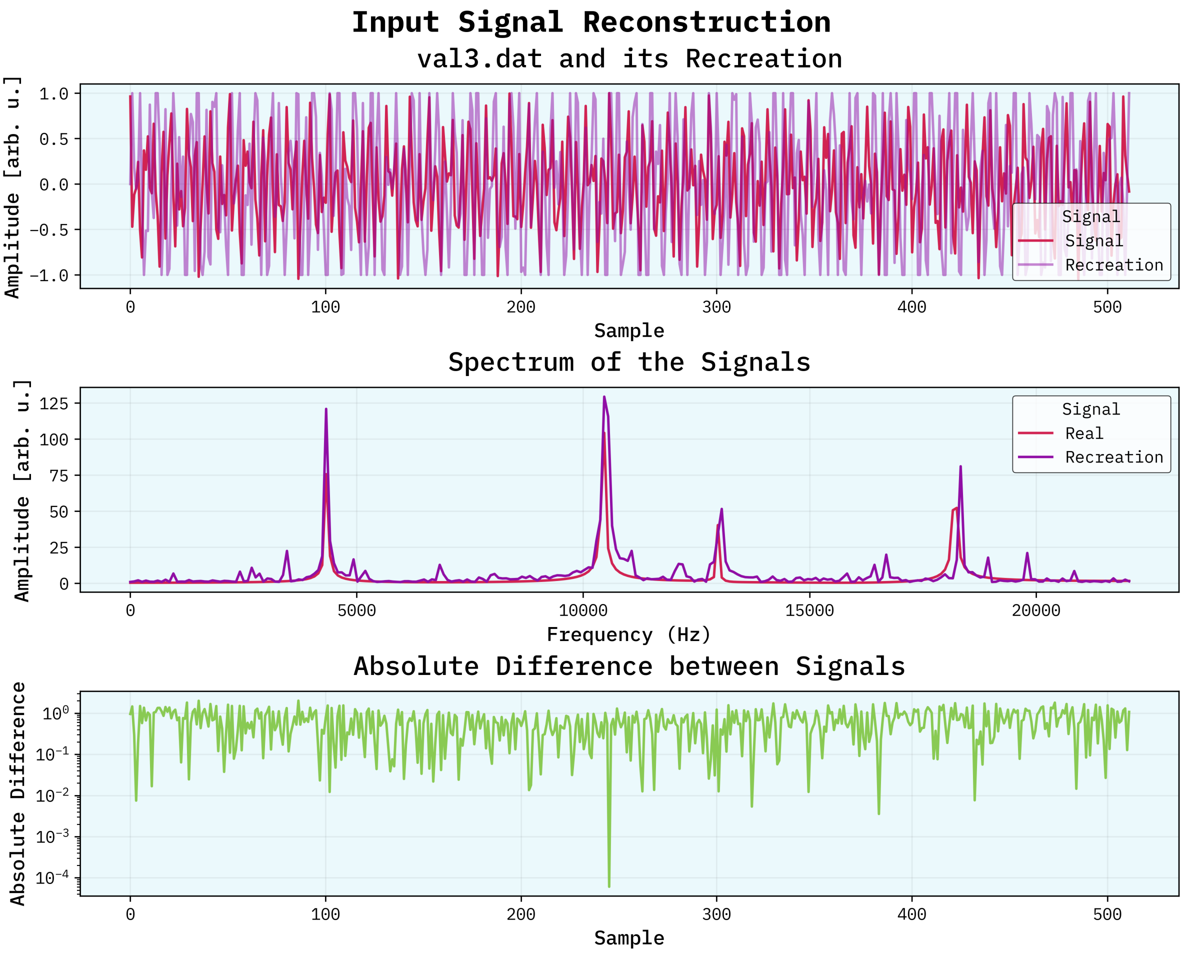

Figure 14: Reconstructed signal `val3.dat` from its spectrum.

I think I managed to reconstruct the first signal well despite some obvious leakage but the second signal is a bit of a mess with many additional peaks appearing in the spectrum. I was careful not to mess with the normalization of the signal or anything that could maybe lead to resonant effects. Not sure what happened here.

4.2 Wiener Filtering

I applied Wiener's filter to the signals `signal{0,1,2,3}.dat` as instructed. I've plotted the results of the filter applications with different window sizes in Figures 15, 16 and 17 for the signals `signal{1,2,3}.dat` respectively. The results I think are quite good. Do note that the figures have been placed at the end of the report due to their large size. From these plots we can see that the Wiener filter does a good job at removing the noise from the signals at least in the first two cases. The third signal is heavily distorted by the noise and the filter does its best to remove it, however the damage is already done. For `signal1.dat` its evident that a small window size is the way to go, as a larger window size starts to essentially flatten the signal which means it looses all its small scale features. The same holds for `signal2.dat`. For `signal3.dat` however a slightly larger window size seems to be the best choice. This is probably due to the fact that the noise is much more prominent in this signal and a larger window size allows the filter to better estimate the noise and extract the meaningful signal.

I wondered if multiple consecutive applications of the filter could improve the results. The results are shown in Figures 18, 19 and 20 as before. I was surprised by how well consecutive applications of the filter work on the first two signals. The noise is lowered consistently with each application. The third signal however is so damaged that even multiple applications of the filter can't really save it, however the base shape it does manage to extract does get cleaner and cleaner.

Since Wiener's filter is hardly the only one out there I wanted to present how a few other filters

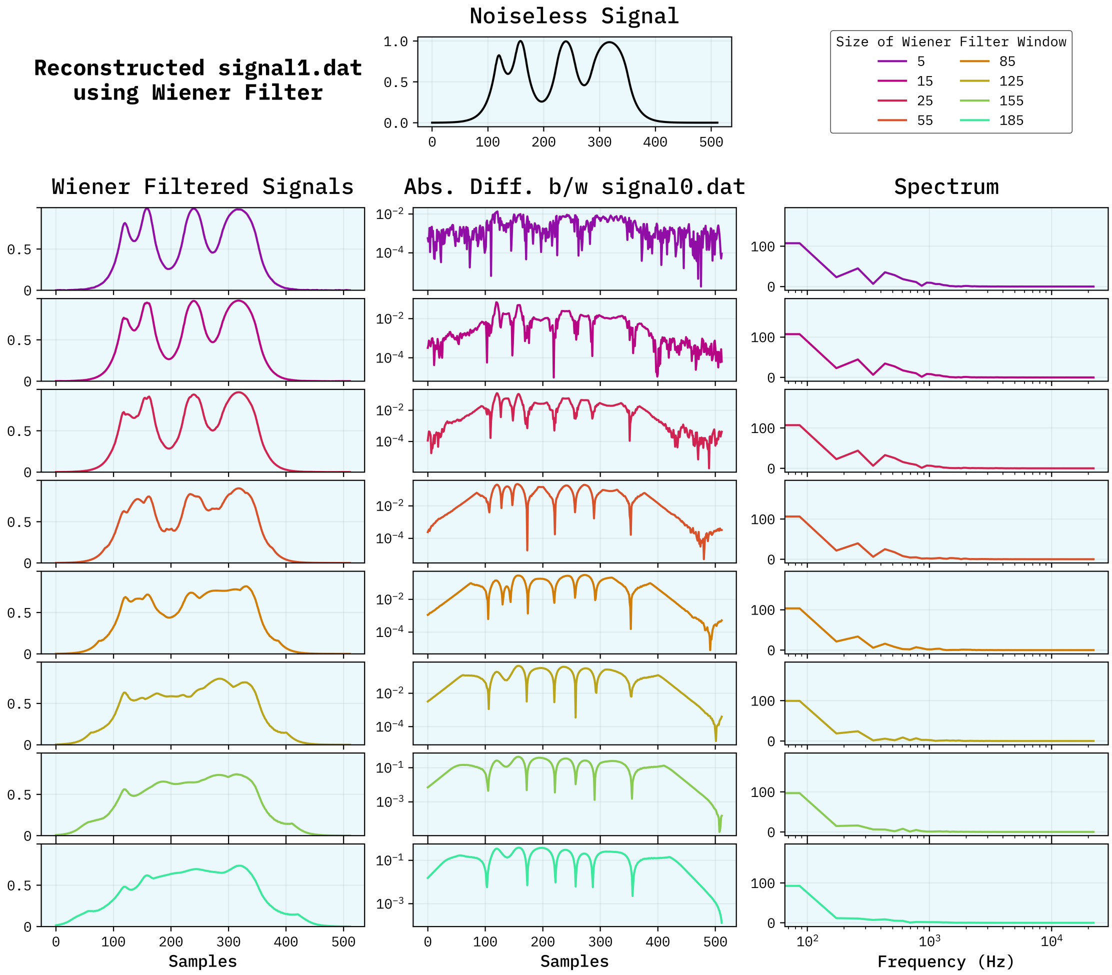

Figure 15: Wiener filter applied to the signal `signal1.dat`.

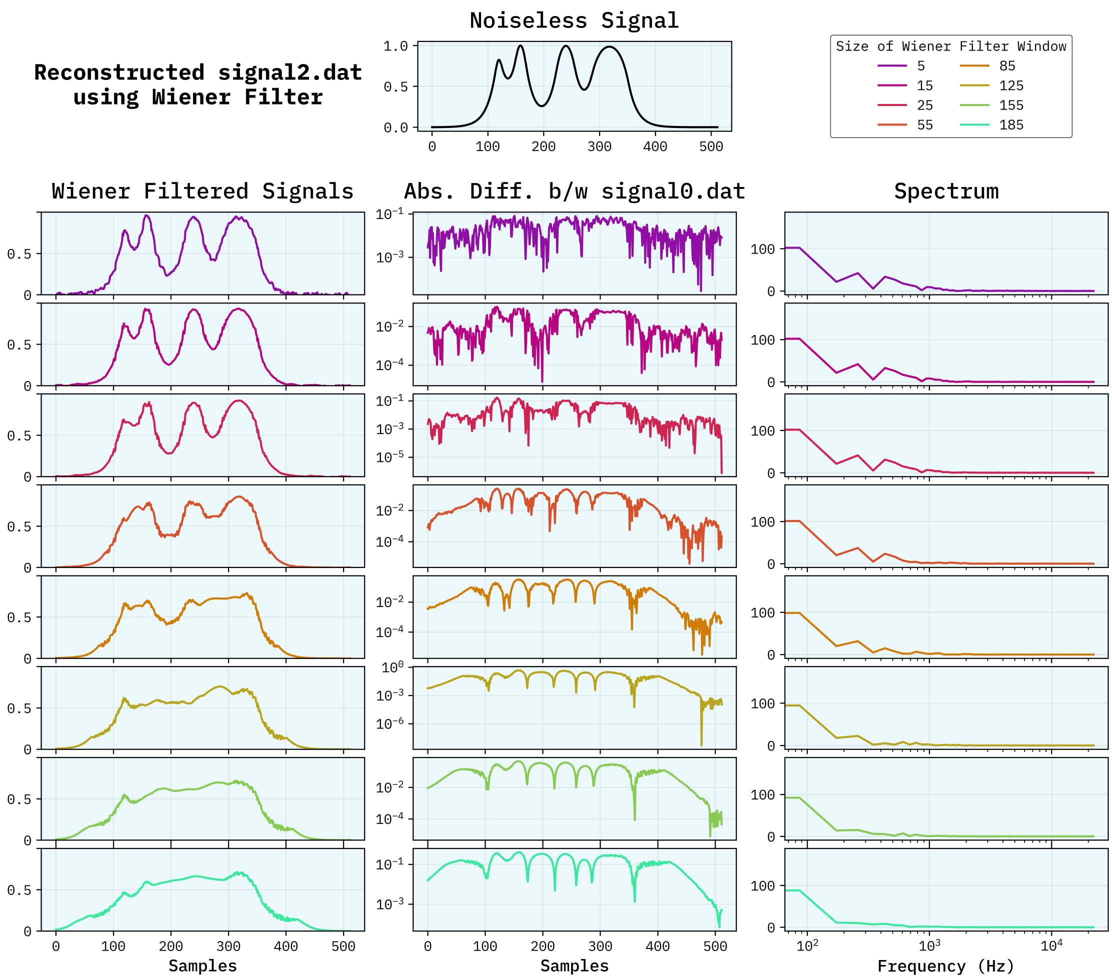

Figure 16: Wiener filter applied to the signal `signal2.dat`.

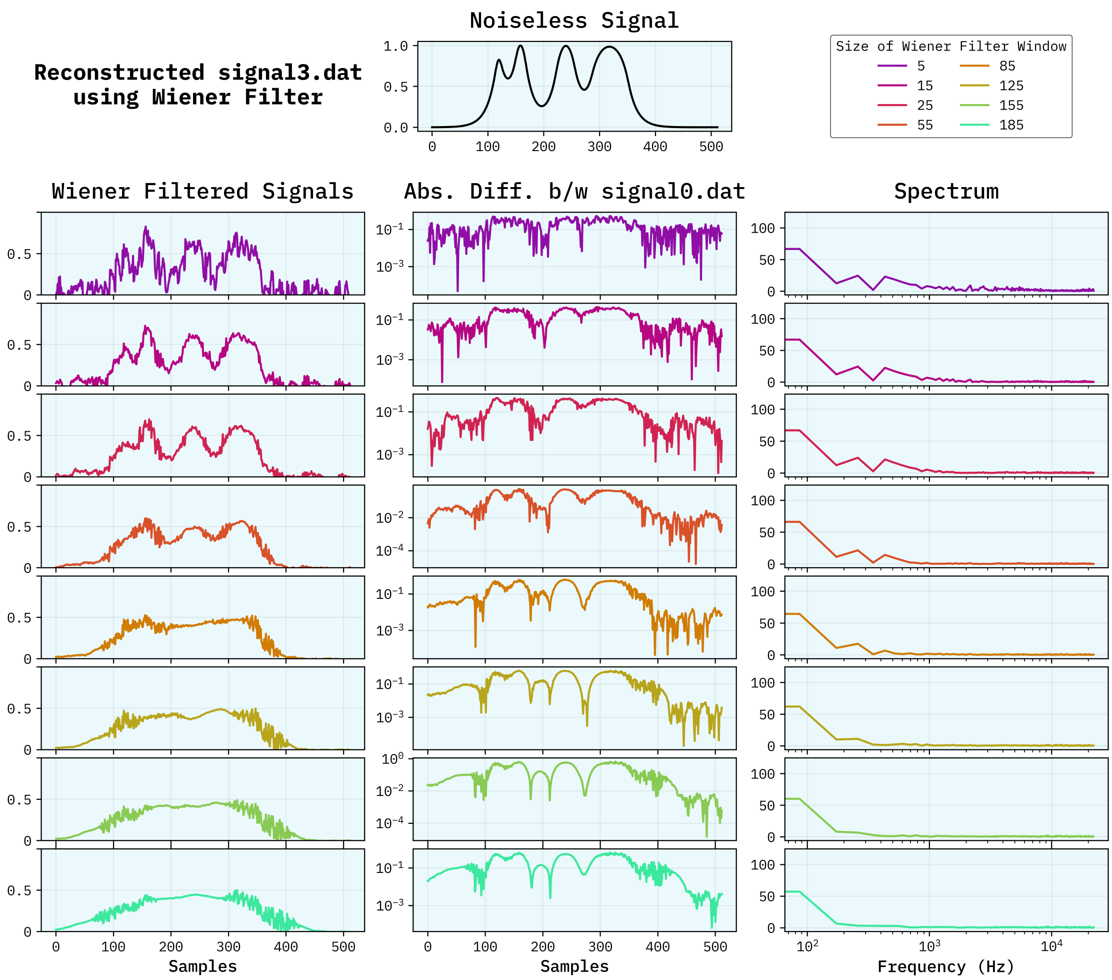

Figure 17: Wiener filter applied to the signal `signal3.dat`.

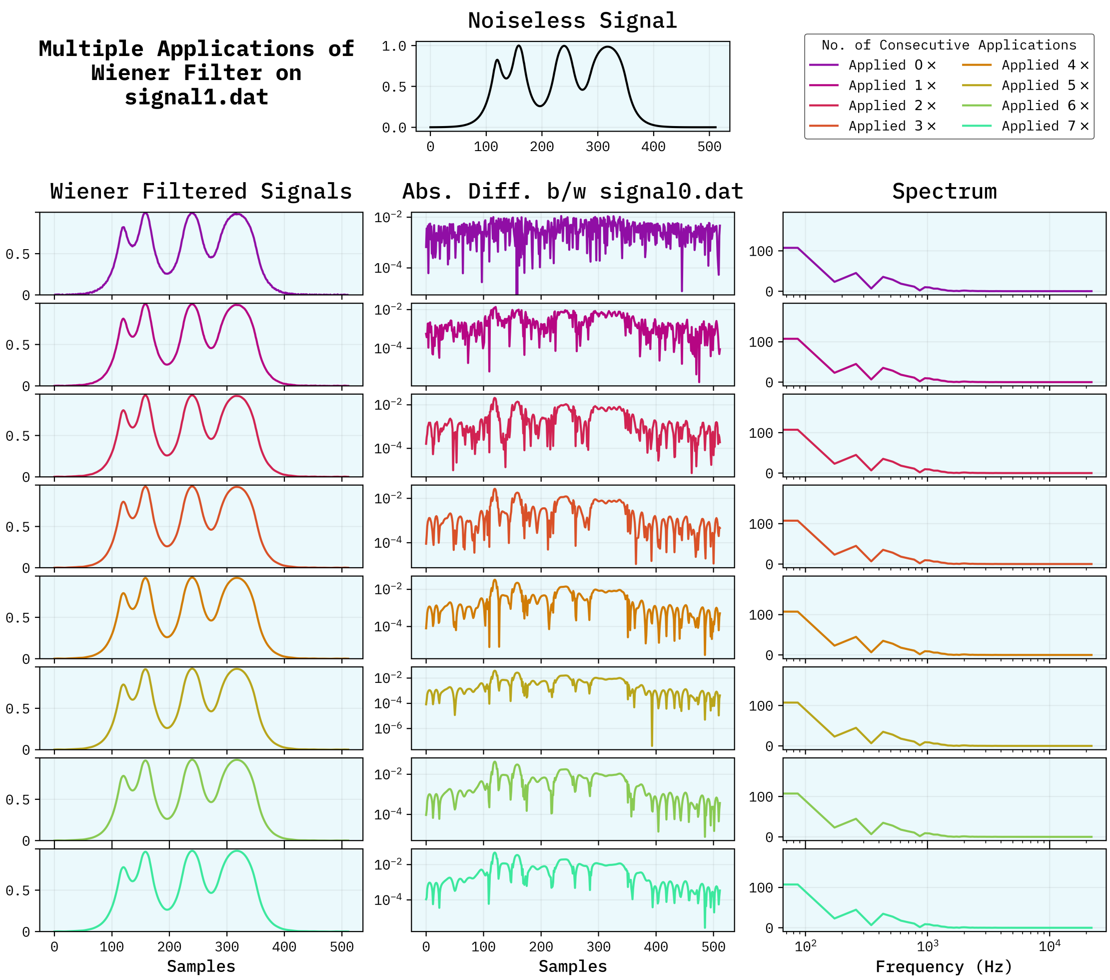

Figure 18: Wiener filter applied consecutively to the signal `signal1.dat`.

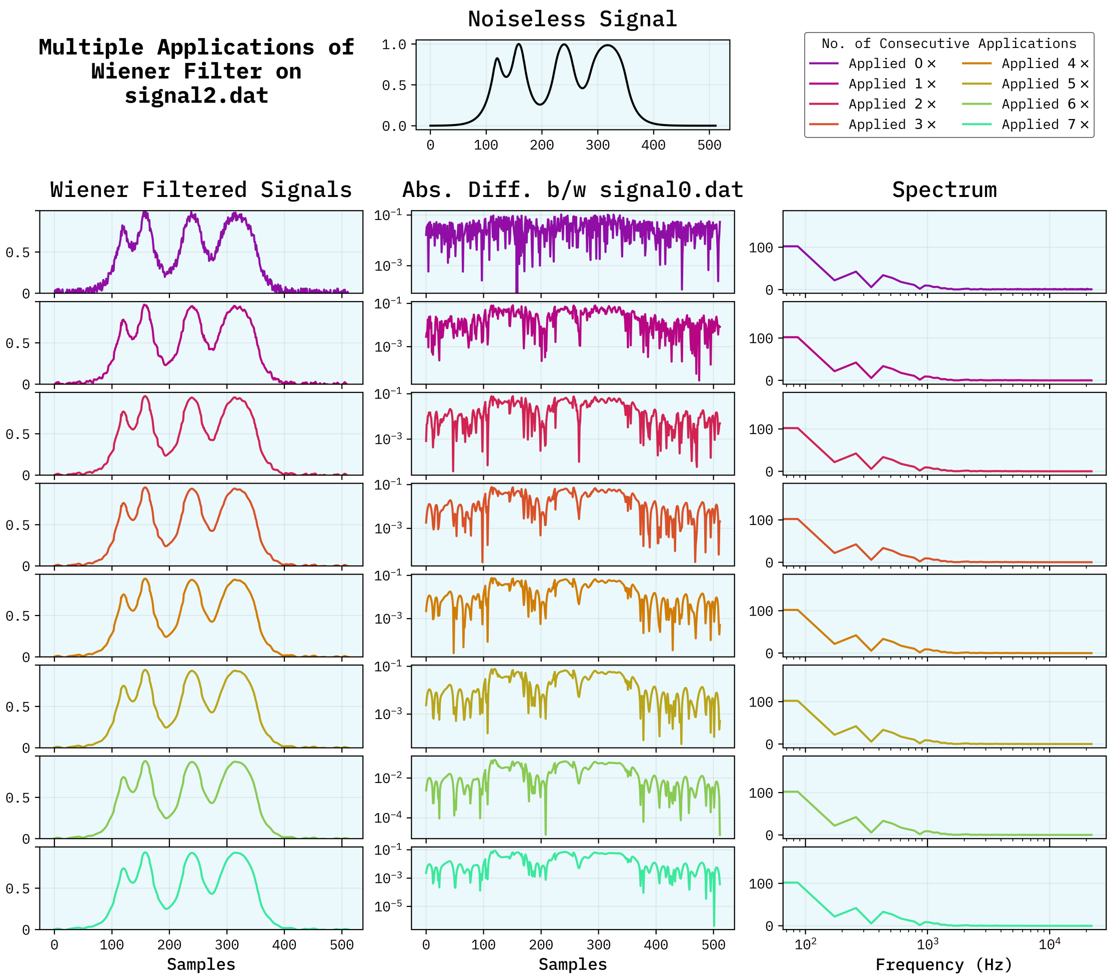

Figure 19: Wiener filter applied consecutively to the signal `signal2.dat`.

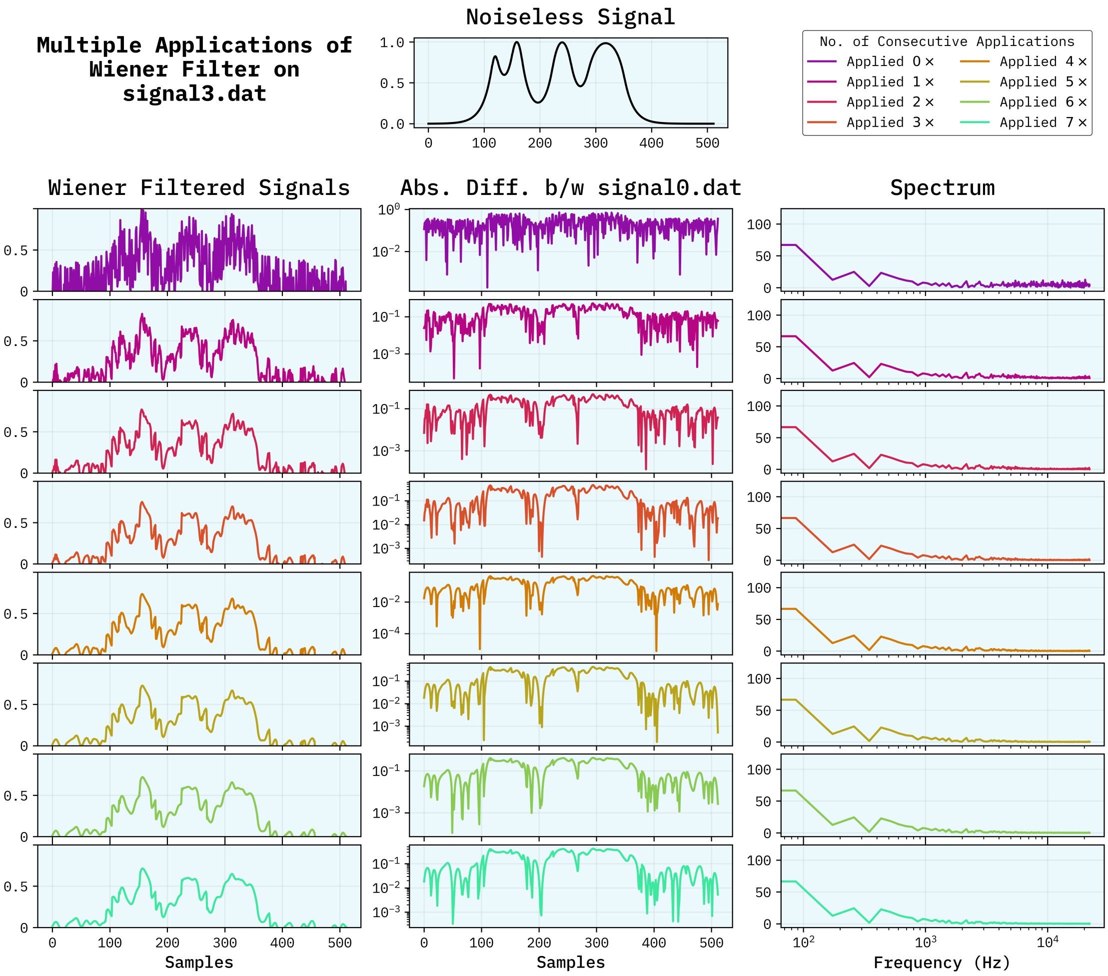

Figure 20: Wiener filter applied consecutively to the signal `signal3.dat`.

perform on the signals. The results are shown in Figures 21, 22 and 23 again at the end of the report. It's worth explaining that Convolve with Uniform means convolution with constant signal and Downsample after AA means downsampling after anti-aliasing which is also known as decimation in signal processing. The AA filter used was an 8th order type I Chebyshev filter. Performance across filters is quite comparable for the first signal. I'm pleased to say that my favourite filter, the Savitzky-Golay filter seems to always do the trick well. This is because it is designed to remove high frequency noise of a base signal. This is really evident for the second signal. Surprisingly convolution with uniform values also works exceptionally well, arguably yielding the best result for the last signal. Wiener's Filter however still gives good results across all signals.

5 Conclusion and Comments

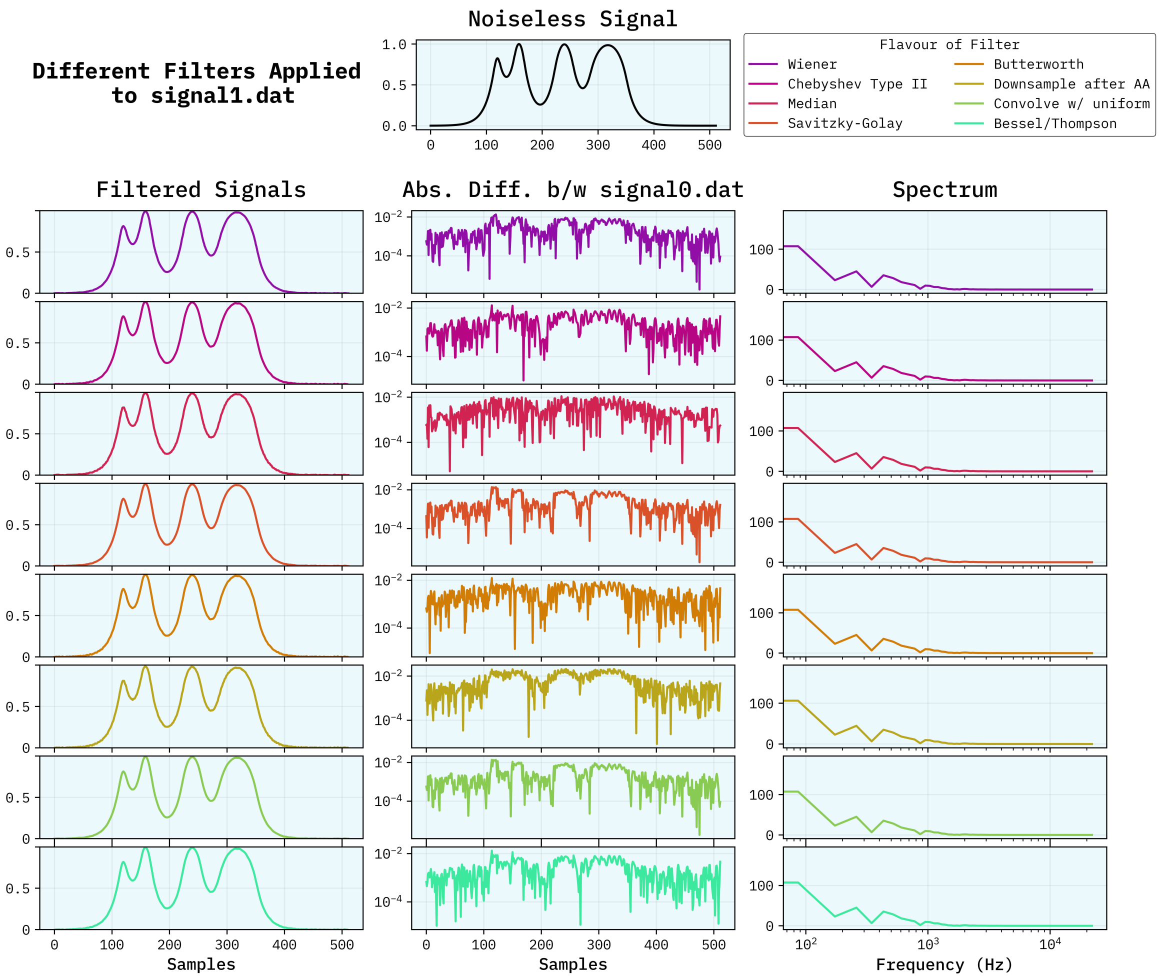

Figure 21: Wiener filter and other filters applied to the signal `signal1.dat`.

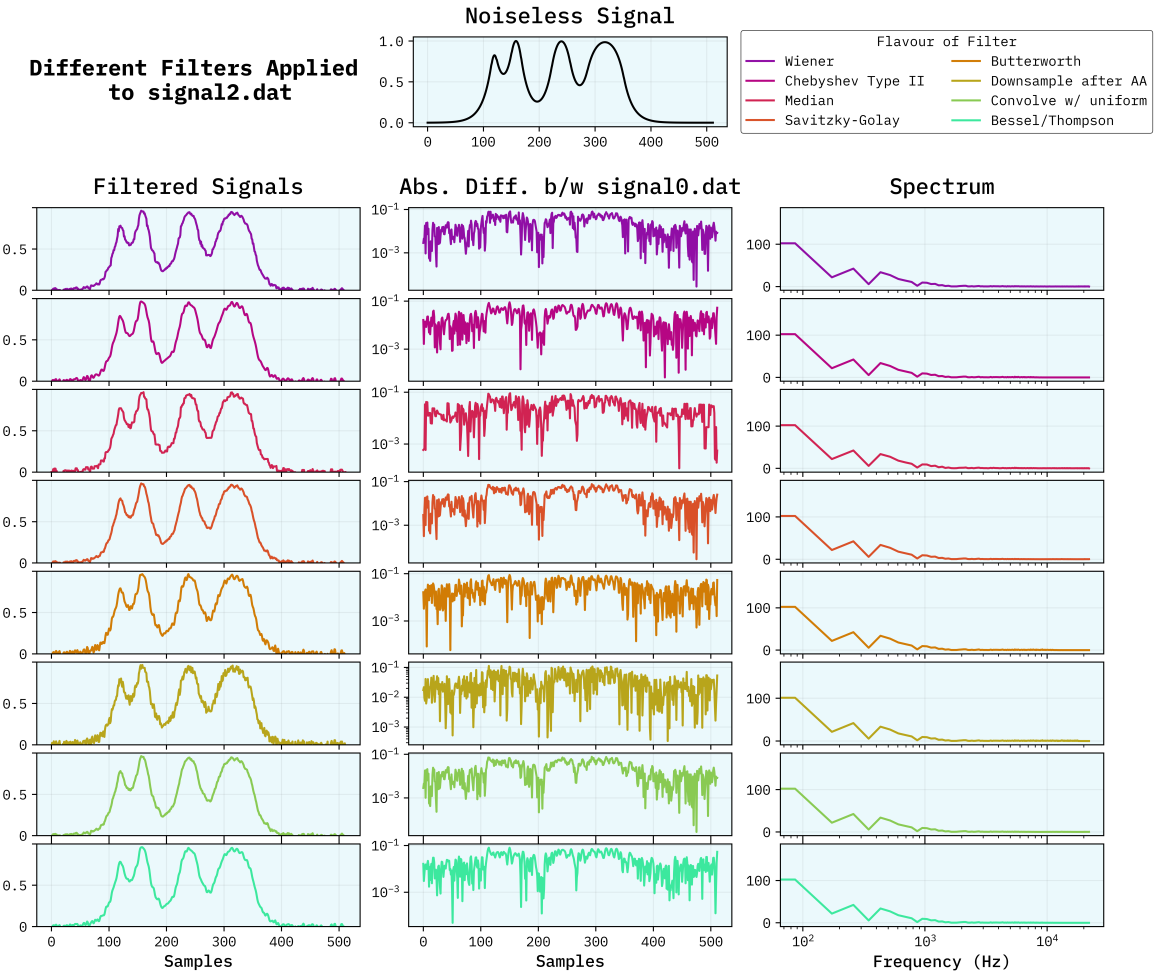

Figure 22: Wiener filter and other filters applied to the signal `signal2.dat`.

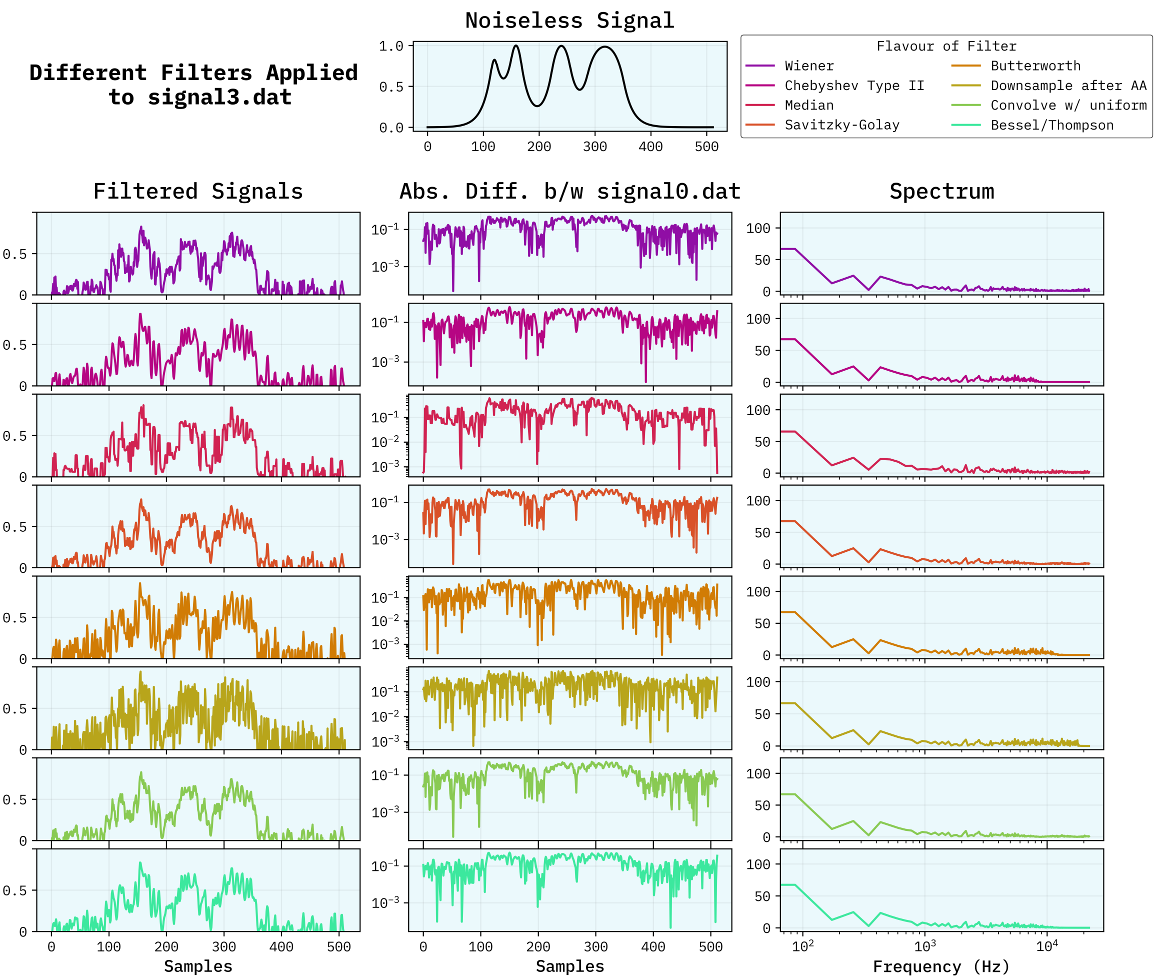

Figure 23: Wiener filter and other filters applied to the signal `signal3.dat`.