Filtering Frequency Domain Noise

Introduction

This application filters noise from the frequency domain representation of an experimental data set.

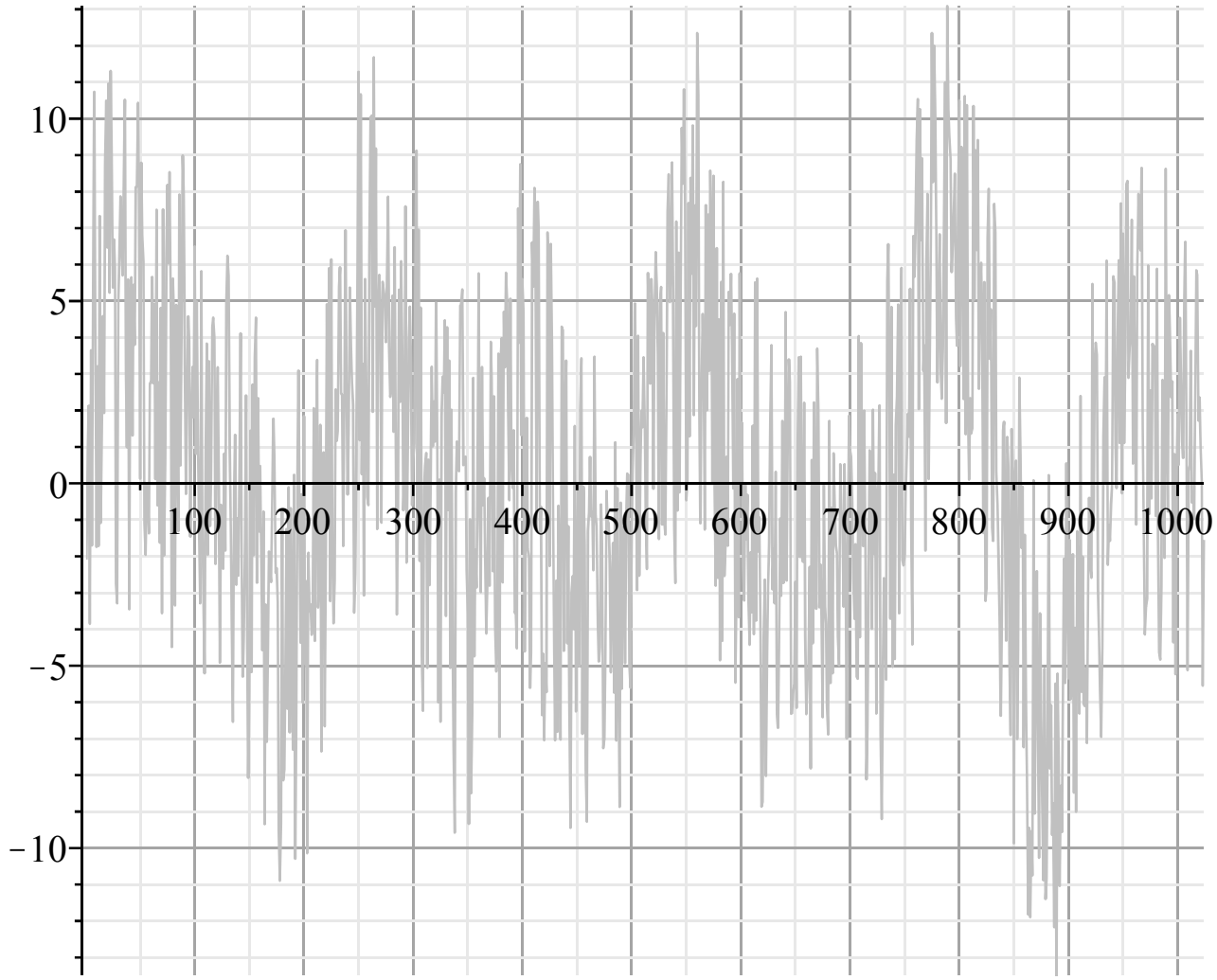
```
> restart
> with(SignalProcessing[Engine]) : with(plots) :
```

Experimental Data

This datatable contains 1024 experimental data points, assigned to the variable 'data'.

	1
1	-2.087945...
2	0.880143...
3	2.126859...
4	-3.847789...
5	3.648326...

```
> originalDataPlot := pointplot([seq([i, data[i]], i = 1 ..numelems(data))], connect = true, gridlines
      = true, thickness = 0, color = gray, gridlines) :
  display(originalDataPlot, size = [800, 400])
```

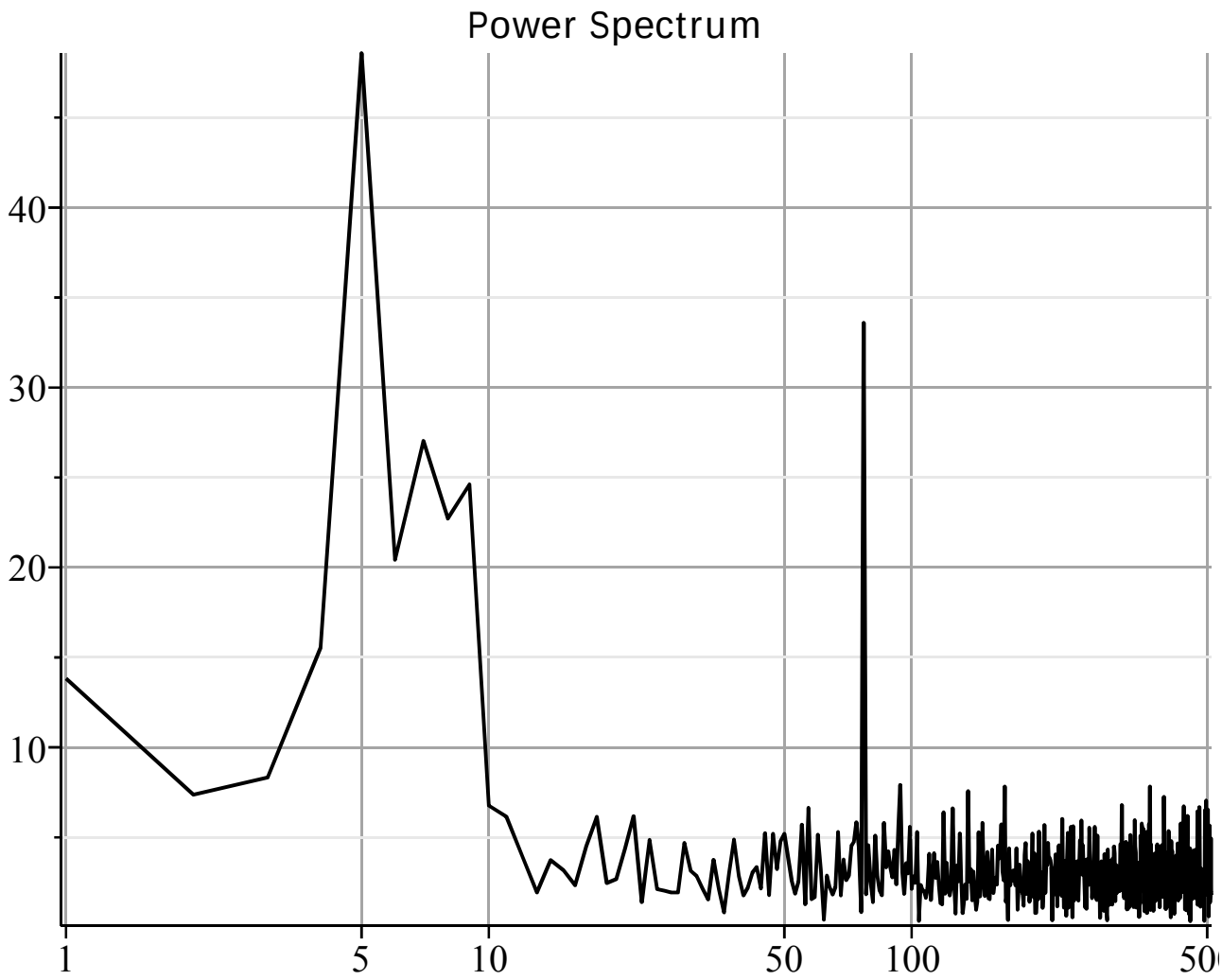


There's an underlying trend, but it's obscured by noise.

Frequency Domain Representation

Let's examine this data in the frequency domain and view the power spectrum

```
> data_fft := FFT( data ) :
> data_fft_ps := sqrt~( PowerSpectrum( data_fft ) ) :
> pointplot( [ seq( [ i, data_fft_ps_i ], i = 1 .. numelems( data_fft_ps ) / 2 ) ], connect = true, thickness = 1,
    axis_1 = [ mode = log ], title = "Power Spectrum", titlefont = [ HELVETICA, 12 ], gridlines, size = [ 800,
    400 ] )
```



There's two dominant frequencies, polluted by low power noise. Let's remove the noise below a specific threshold with a customer filter.

Filtering Noise in the Frequency Domain

> threshold := 8 :

data_filtered_fft := map(data → data·Heaviside(abs(data) − threshold), data_fft) :

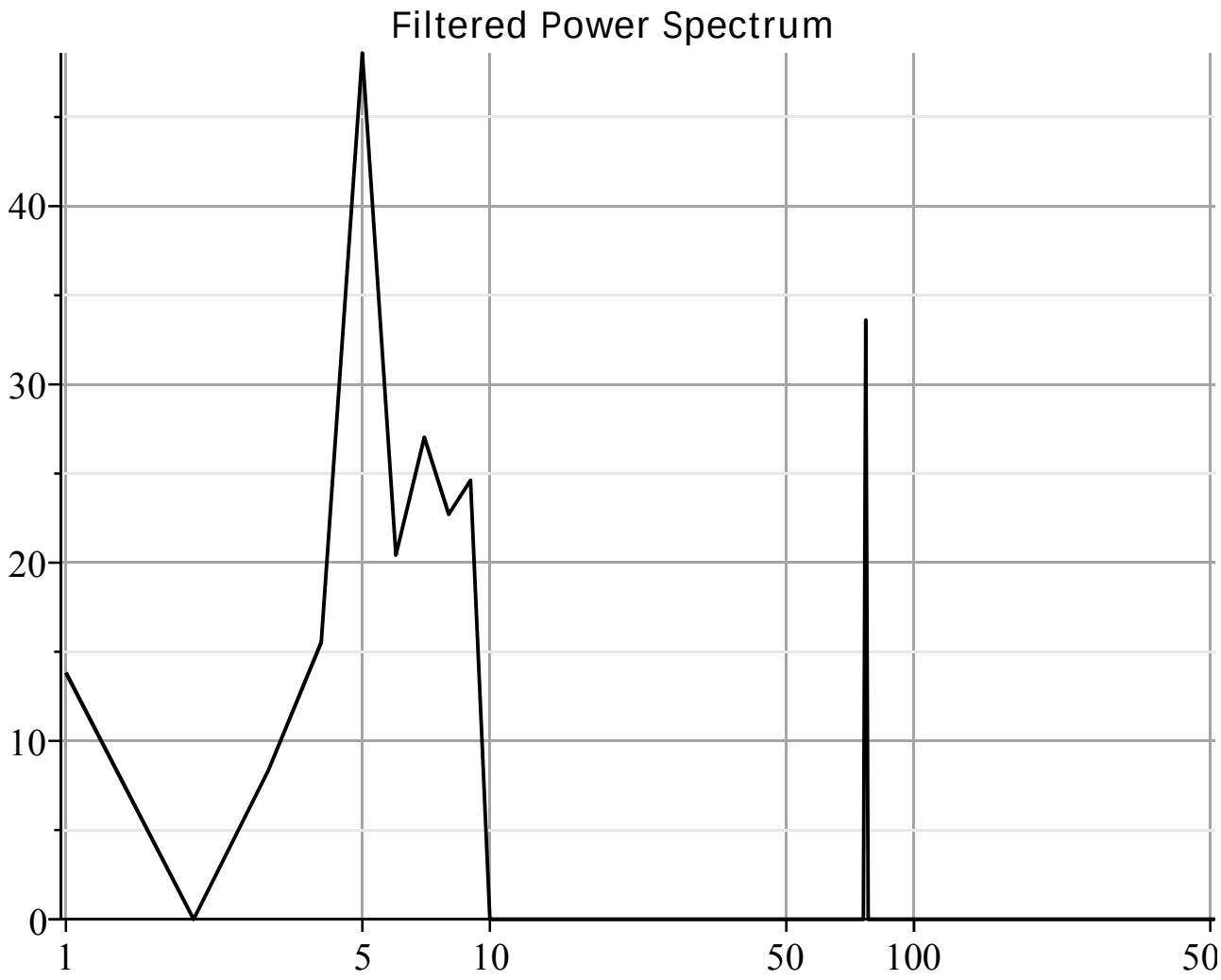
This is the filtered power spectrum

> data_filtered_ps := sqrt~(PowerSpectrum(data_filtered_fft)) :

pointplot([seq([i, data_filtered_ps$_i$], i = 1 .. numelems(data_filtered_ps) / 2)], connect = true,

thickness = 1, axis$_1$ = [mode = log], title = "Filtered Power Spectrum", titlefont = [HELVETICA, 12],

gridlines, size = [800, 400])



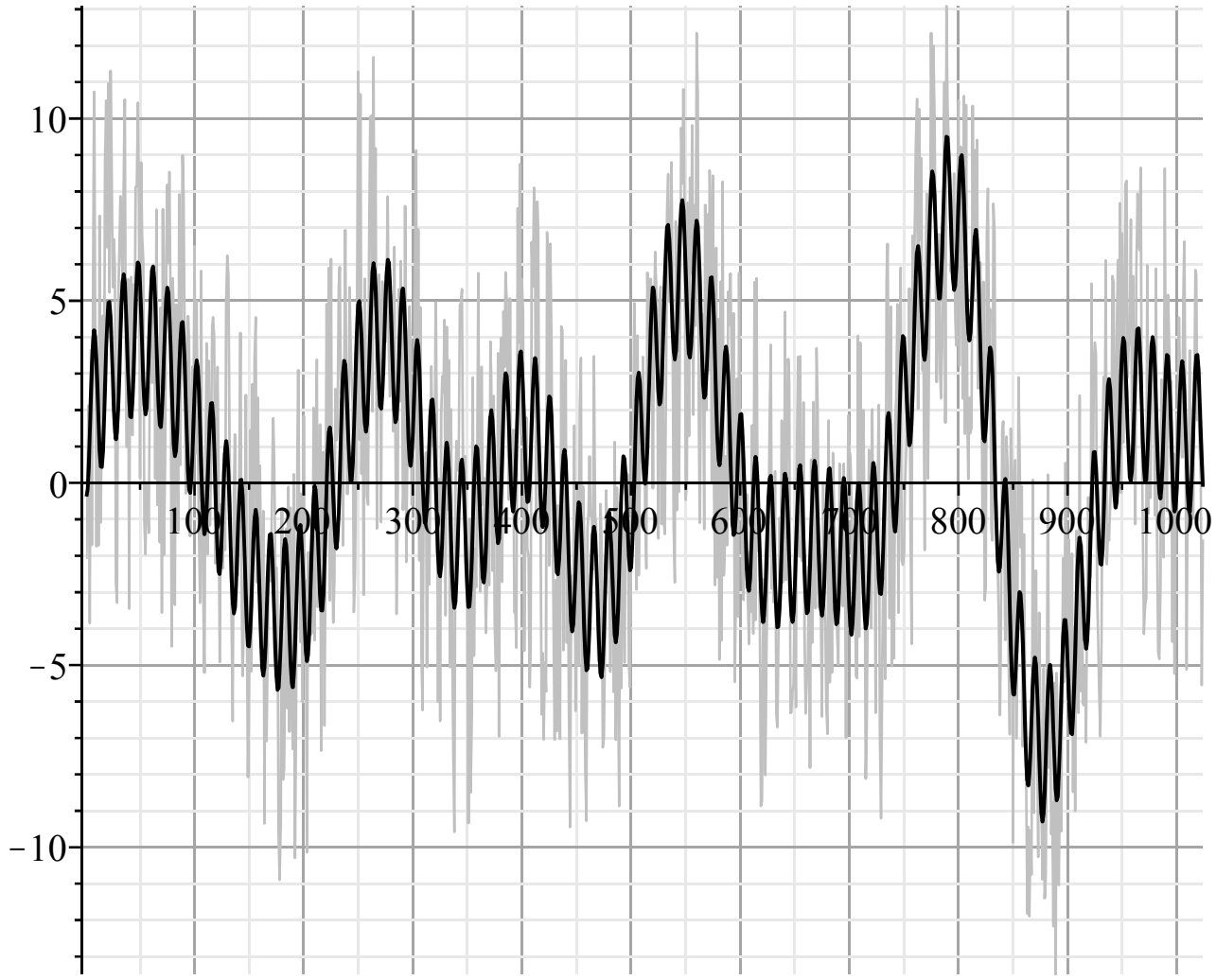
Take the filtered data back to the time domain

> data_filtered := InverseFFT(data_filtered_fft) :

Comparison of Original and Filtered Data Set

> filteredDataPlot := pointplot([seq([i, Re(data_filtered)$_i$], i = 1 ..1024)], connect = true, gridlines = true) :

display(originalDataPlot, filteredDataPlot, size = [800, 400])



As you can see, we have filtered out much of the noise. The underlying trend is clearer