Assignment IV Report

IU DSP Spring 2020

Leonid Lygin

Contents

1 Task 1. Filter design

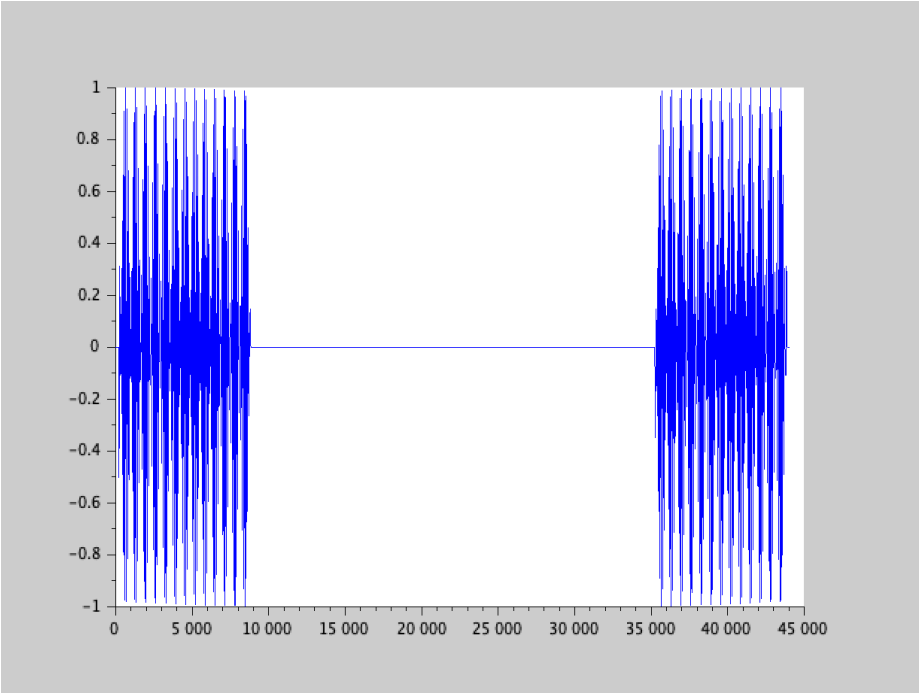
This task involved designing an "ideal" filter, which in turn involved the idea that designing filters in frequency domain is relatively easy, while doing so in time domain is relatively hard.

As the voice is probably located below the high-frequency noise at $10000Hz$, and above the low-frequency noises at $10Hz$ and $20Hz$, I have chosen to go with a band-pass filter, which required minimal modification from the one presented in the task:

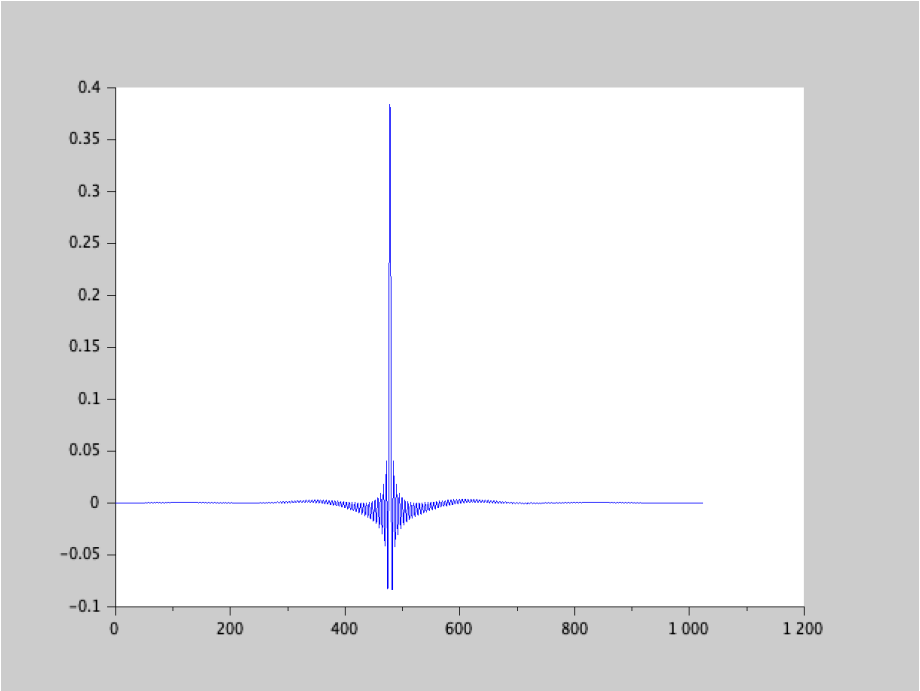
```
function H = ideal_bandpass(N, low_freq, high_freq, stop_value)
  N = (N - modulo(N,2)) / 2;
  low_cutoff = 1 + floor(low_freq * 2 * N);
  high_cutoff = N - floor(high_freq * 2 * N);
  H = ones(1, N) * stop_value;
  H(low_cutoff : high_cutoff) = 1;
  H = [H flipdim(H, 2)];

  // shifting the frequency response
  h_len = length(H);
  frequencies = (0 : h_len - 1) / h_len * Fs;
  shifts = %e ^ (%i * %pi * frequencies);
  H = H .* shifts;
endfunction
```

After iterating, somewhat good parameters for this were $N = 1024$, $low_freq = 0.005$, and $high_freq = 0.3$ (at least these resulted in the lowest distortion).

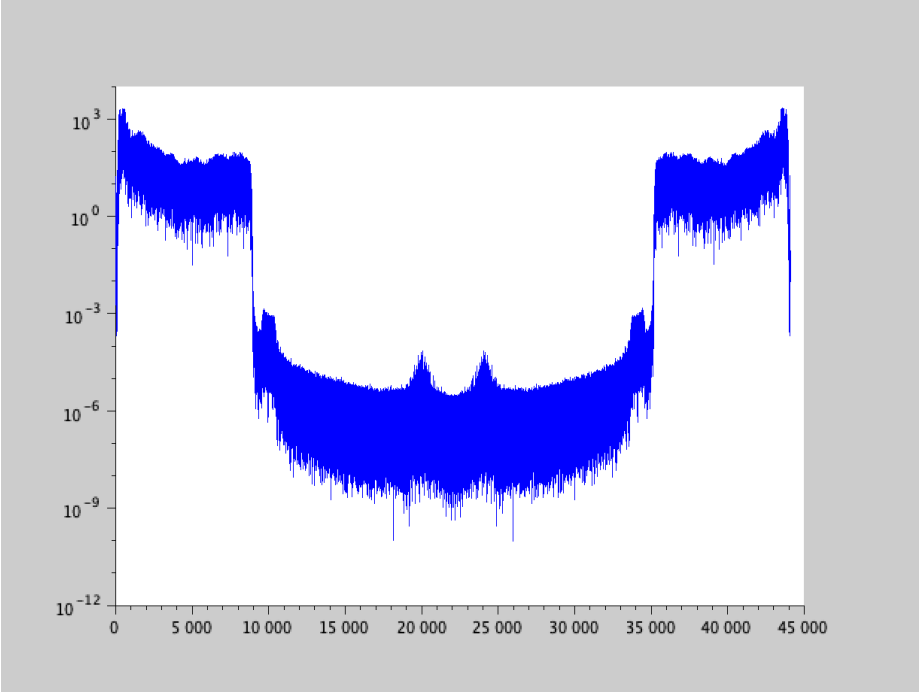
The filter frequencies look like this (the edges are not straight due to the shift):



The filter in plain time-view looks like:



The final filtered frequencies look like this:



2 Task 2. Inverse filter design

This task involved designing a filter that would be an "inverse" to some other original filter.

The original formula, presented in the task PDF, looked awfully like a formula for inverting a complex number ($\frac{1}{z}$), so I figured that the $|_|$ in the denominator probably meant taking the absolute value of *each* element, and the fraction itself was meant to be computed element-wise. With that in mind, the final code for this task is:

```
irc_fft = fft(irc);
inv_irc_fft = 1 ./ irc_fft;
inv_irc = real(ifft(inv_irc_fft));
inv_irc = inv_irc / max(inv_irc);  // normalization
```

By design, when convolving this filter with the original, this should produce a Kronecker delta (or something similar). Thus the output of the following code:

```
kron = convol(inv_irc, irc);
```

is this plot: