Report

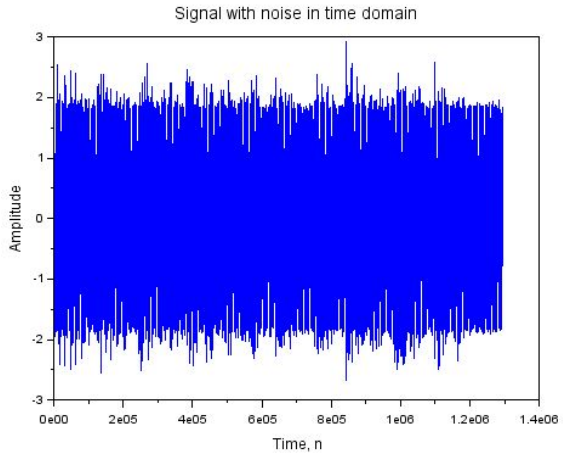
Lab 4 FIR Filter Design

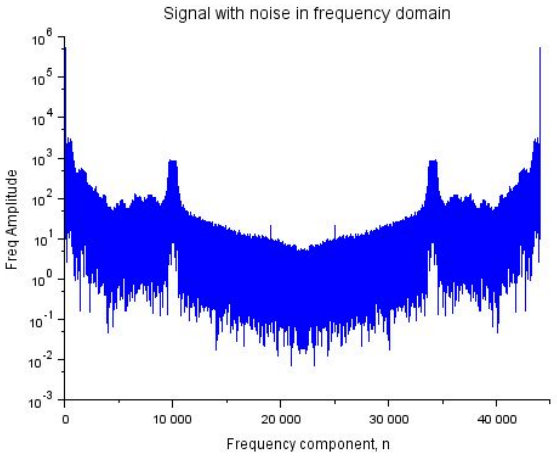
Ilnur Mamedbakov B17-DS-02

Task 1

Abstract

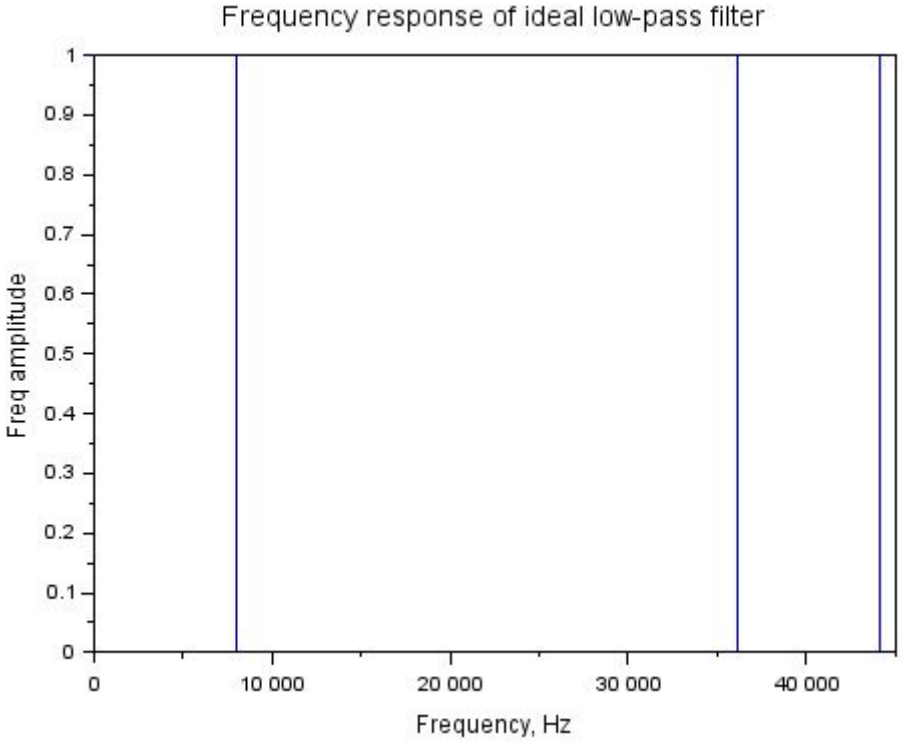
For first task we had to design low-pass and high-pass filters to prevent noise from 10 kHz and higher, and noise on 10 and 20 Hz. 10kHz is max possible frequency for human voice, but average frequency is around ~4-8 kHz, so we did filter, accepting frequencies up to 8kHz, except 10 and 20 Hz. Below are signal in time domain and frequency domain:

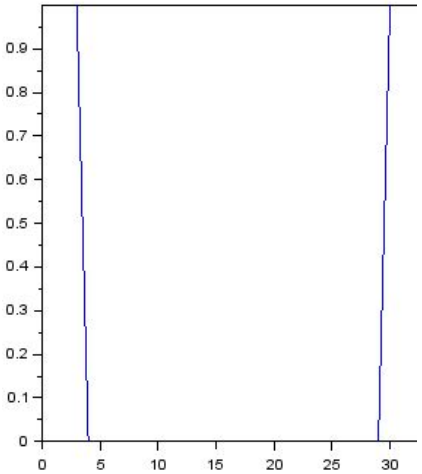




Filter

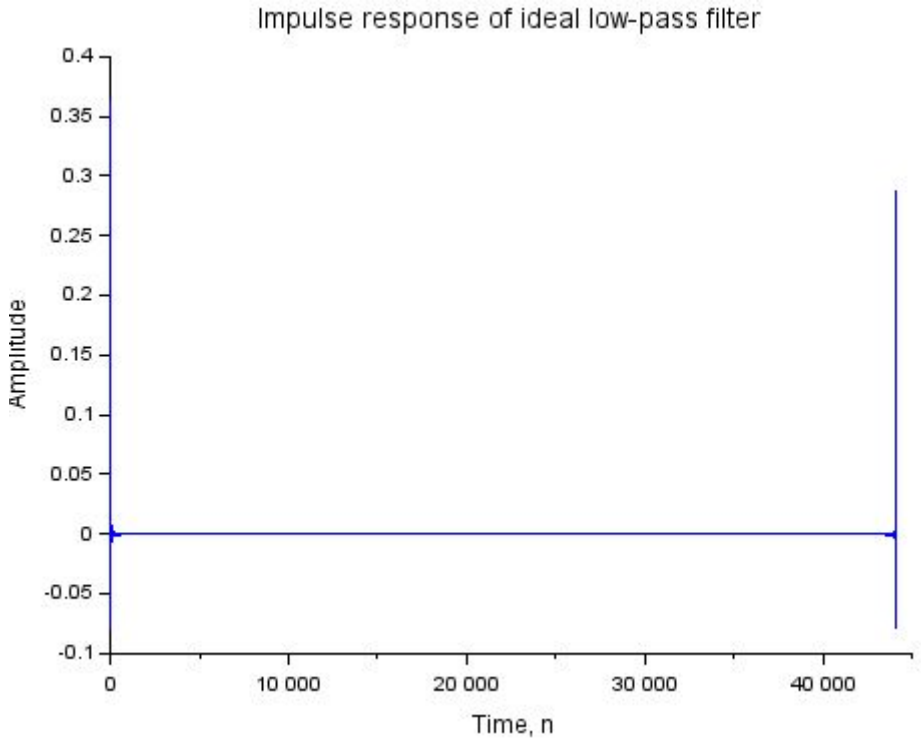
Our filter are not accepting: from 30 Hz to 30 Hz, and everything after 8000 Hz.





I made this by just making vector with zeros and ones, and in places where I wanted to mute noises I putted 0.

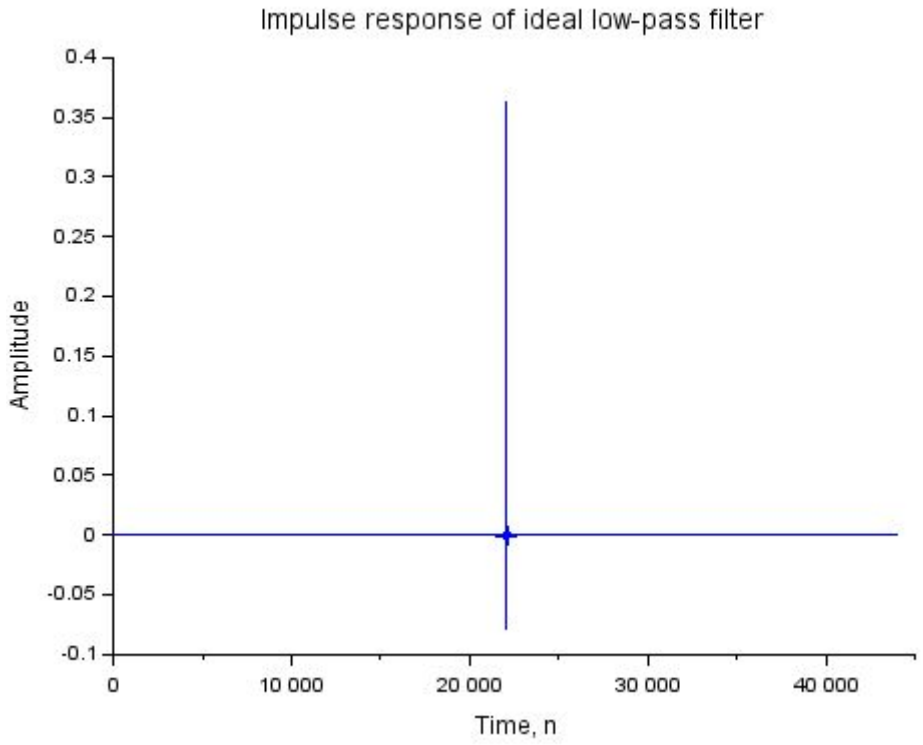
Here Is our filter in time domain:



For converting signal to time domain I did IFFT:

```
h_l = real(ifft(H_l))
```

But we want our filter to be not cyclical, because we need it to be fading to the ends. So we shifted it relative to the middle (because it is symmetric).

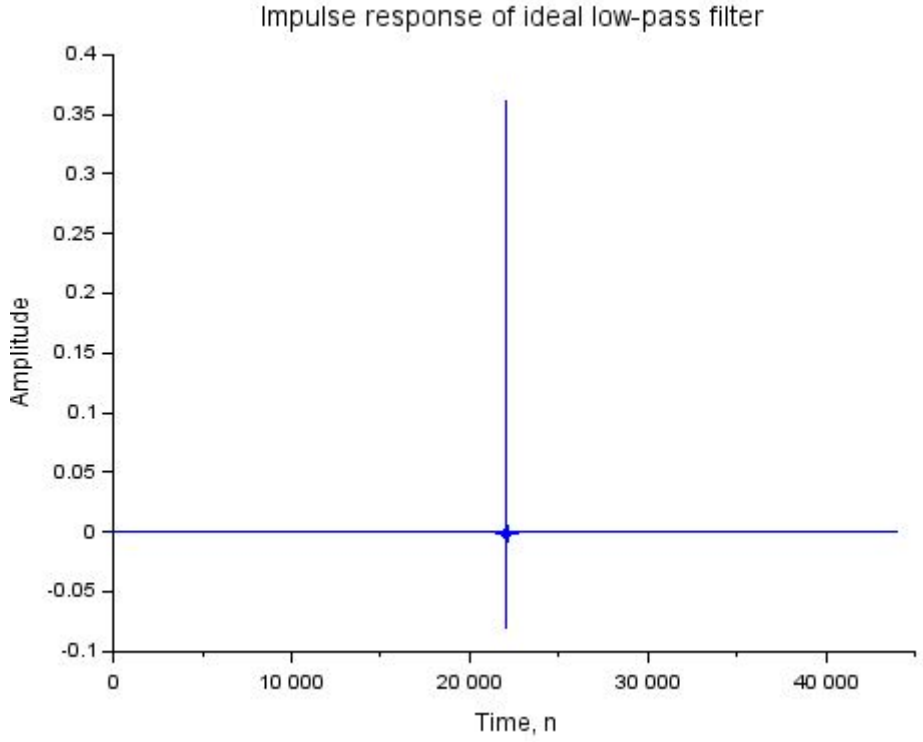


I shifted it just by finding middle of signal:

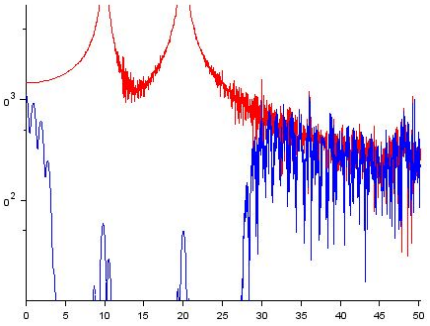
```
h_l = [h_l(ceil(length(h_l)/2):length(h_l)), h_l(1:floor(length(h_l)/2))]
```

But we can notice that filter does not decay to 0 (if zoom in on the graph). So we using window function from scilab.

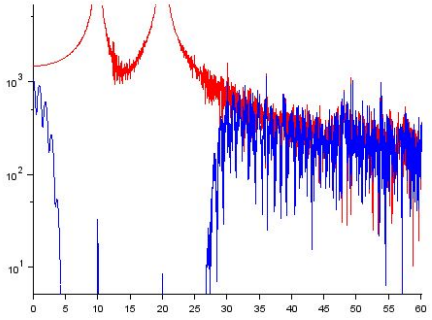
```
h_l = h_l .* window('hn', length(h_l), 8)
```



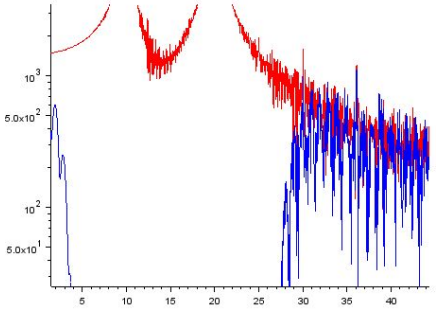
For window function We started with Kaiser window, but I experimented and found that Hann window is best for filter, because after applying this filter we can see that noize decreases more in the comparison with Kaizer and Hamming windows.



Kaizer window



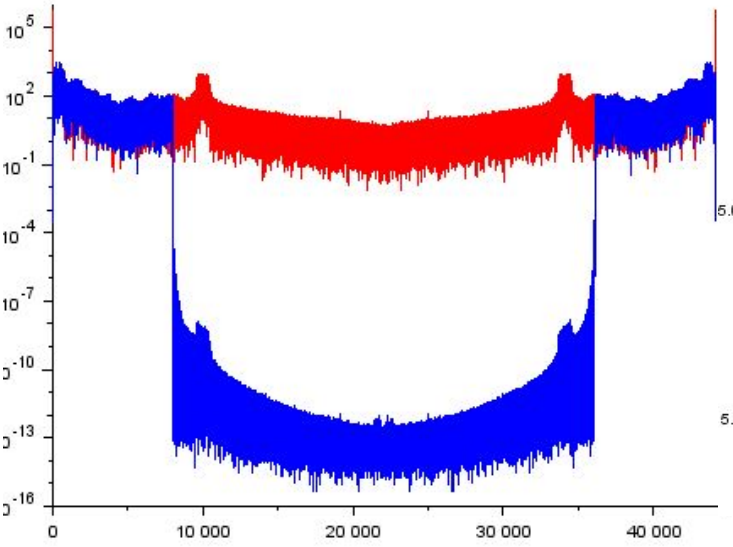
Hamming window

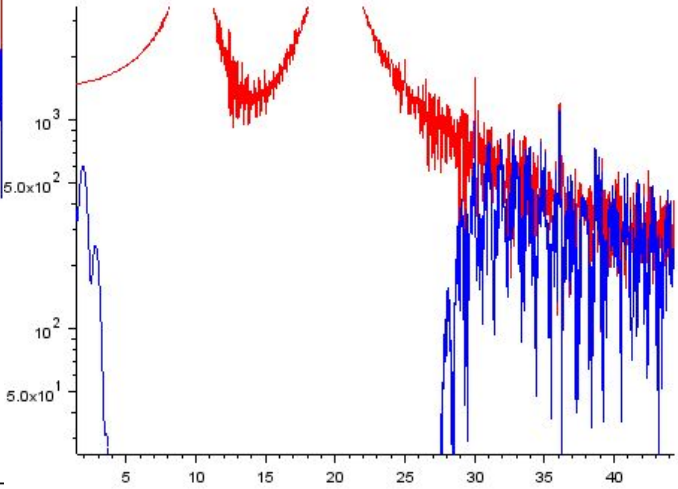


Hann window

<u>Applying filter</u>

Now we need just do convolve original sound and our filter.





Red is our original sound with noise, and blue is sound with applied filter.

Conclusion

We solved problem using our filter. During the solution I encountered a problem with low frequency noise (10 and 20 Hz), because in the beginning I removed noise only on 10 and 20 Hz, but on 44090 and 44080 Hz. But later I understand that and removed noise on 44090 and 44080 Hz. I think my solution maybe can be improved by choosing better hyperparameters for window function, and it can by optimized, because I understand that some parts of my code are not really efficient.

Task 2

Abstract

We need to add echo to original sound by convoling IRC and sound, and after it remove this effect of echo by creating reversed signal of IRC.

Steps

1. Moving to the frequency domain our signal by FFT.

```
ss_fft = fft(ss)
```

2. $$\tilde{H}(e^{-j\omega}) = \frac{H^*(e^{-j\omega})}{|H(e^{-j\omega})|}$$ Divide the Delta function by our IRC

```
ss_inv = 1./ss_fft
```

3. Shifting our signal.

```
h_l = [h_l(ceil(length(h_l)/2):length(h_l)), h_l(1:floor(length(h_l)/2))]
```

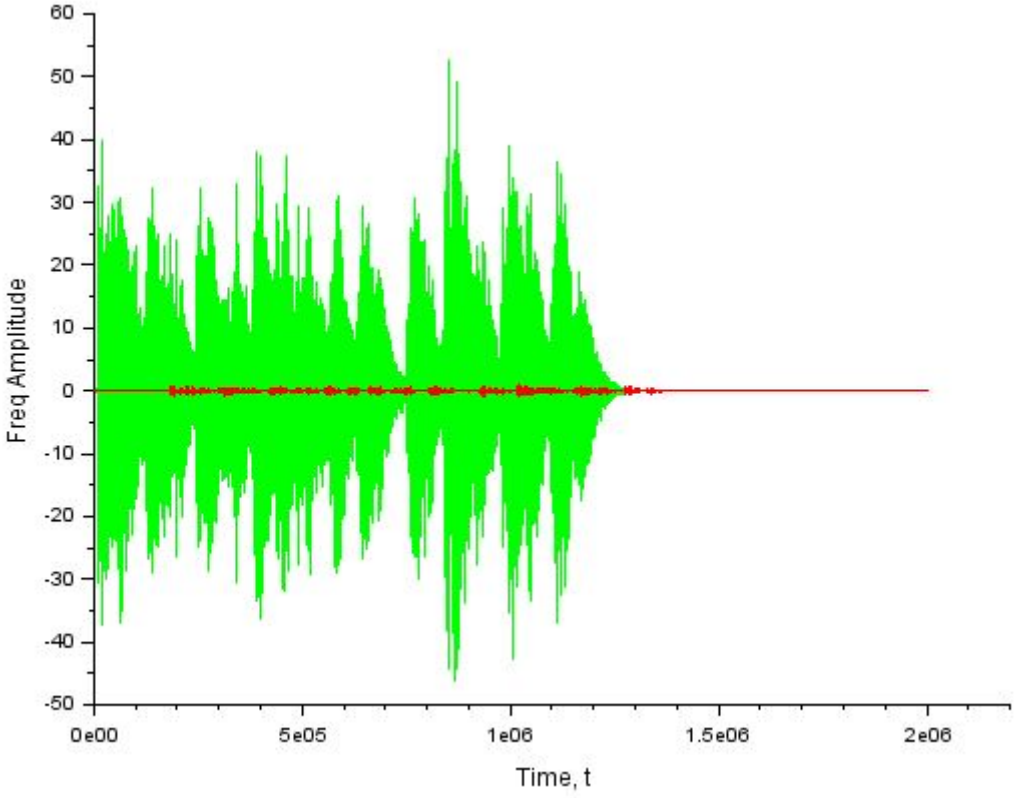
4. Using Kaizer window function from Scilab.

```
h_l = h_l .* window('kr', length(h_l), 8)
```

5. Applying this signal to our original sound with echo

```
sss = convol(ss, s)//clear with irc
final = convol(sss, h_l) //echo with reversed echo
```

Result



Here we can see: red is our signal after convolving our inverted IRC, and green is signal after convolving original voice with IRC.

Conclusion

We solved task, but we can still hear some noise on background. But I think we can't made our sound perfect, because the record is not perfect. I think we still can change parameters in window function and maybe it will improve our quality.