DSP Lab Presentation

EE3025 - Independent Project

S.Abdur Rahman Nawaz - EE18BTECH11052

IIT HYDERABAD

June 12, 2021

- Modify the following code given in problem 2.3 with different input parameters to get the best possible output.

```
import soundfile as sf
from scipy import signal

input_signal,fs = sf.read('Sound_Noise.wav')

sampl_freq=fs
order = 3
cutoff_freq=4000.0
Wn=2*cutoff_freq/sampl_freq

b, a = signal.butter(order,Wn,'low')
output_signal = signal.filtfilt(b, a, input_signal)
sf.write('Sound_With_ReducedNoise.wav', output_signal, fs)
```

Solution

The main parameters for a standard Butterworth filter are,

- Cutoff frequency
- Order

Solution

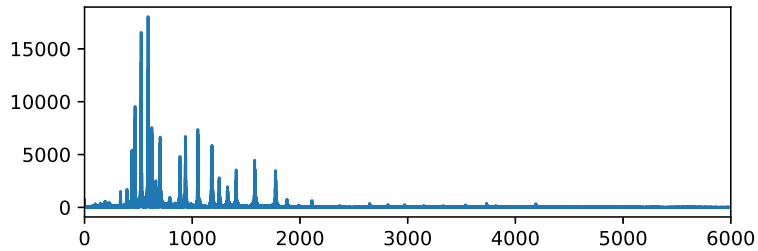
Cutoff Frequency:
Cutoff frequency for a Butterworth filter is the $-3dB$ point after which all the frequency components roll off down to zero.
To select a good cutoff frequency, we observe the spectogram from Problem 2.2 and conclude that most of the frequencies corresponding to piano notes are around mid point of 440Hz and 4700Hz, so let the cutoff frequency be $(440 + 4700)/2 = 2570Hz$

Solution

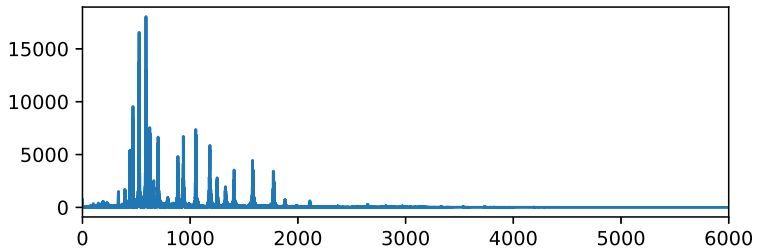
Order:
The main disadvantage of the Butterworth filter is that it achieves the pass band flatness at the expense of a wide transition band as the filter changes from the pass band to the stop band.
We can fix this by increasing the order, but having a very high order would create numerical instabilities while simulating. So we will stick to the general order 4.

Solution

Results: After applying the filter with above parameters we observe that we still have noise. The frequency response of signal before and after applying the filter shows that the transition region is too wide, there are too many non zero frequency components after the cutoff frequency.



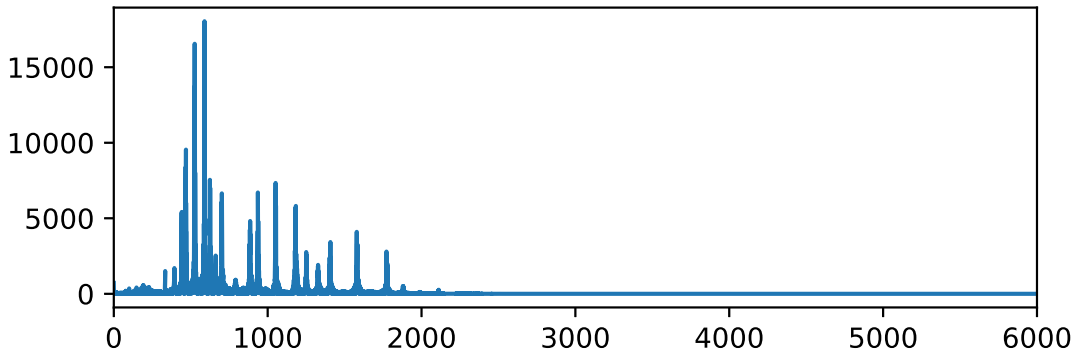
. Before Filtering



After Filtering

Solution

To overcome this we would need a steeper transition. Since we cant increase the order, one solution is to cascade multiple butterworth filters. The transfer function rolls off to zero from pass band to stop band, as we cascade more and more filters the gain values< 1 would get multiplied and reach closer to zero.



. After Cascaded Filtering

Solution

To get a better understanding of the noise present in the signal we sum up all the non zero frequency components before and after the cutoff frequency to define the metric fraction which tells us how much non zero frequency component is present before and after the cutoff.

Parameter	Original	Filtered	Cascaded
Cutoff Frequency	$2570Hz$	$2570Hz$	$2570Hz$
Order	-	4	4
No. of Cascades	-	0	19
Fraction of signal before cutoff	82.35%	97.38%	99.51%
Fraction of signal after cutoff	17.64%	2.61%	0.485%

Appendix

- C code for generating the plots with FFT and IFFT implementations can be found here.
- Python codes for generating the plots can be found here.
- Input and Output Sound files can be found here.