

Vuvuzela Project:

Implementation of an “anti-vuvuzela” filter to attenuate its sound

Tristan Londe
Florian Pasco

Liam Vasseur-Viton
S4A-A



Contents

1	Analysis of the sound file vuvuzela.wav	3
2	Determination of the number L of significant sinusoidal components	3
3	Determination of the frequencies f_l	5
4	Approximate determination of m	6
5	Implementation of the filter	7
5.1	Choice of a notch filter topology	7
5.2	Determination of electronic components	7
5.3	LTSpice Test of the chosen cascade of filters	8
6	Conclusion	10

List of Figures

1	Magnitude spectrum of the chosen part of the recording	3
2	Parameterized magnitude spectrum	4
3	Parameterized magnitude spectrum with value readings	5
4	Excerpt from Chapter 2: Second-order filters - S4 Electronics Course	6
5	Circuit created in LTSpice	8
6	Spectral analysis via FFT	9

Introduction

During the 2010 FIFA World Cup (and the 1995 Rugby World Cup), viewers around the world discovered a musical instrument from South Africa called the **Vuvuzela**. During the broadcasts of the matches, the continuous sound produced by this instrument quickly became bothersome (example: <https://www.youtube.com/watch?v=bKCIFXqhLzo>).

In this project, we will **implement an "anti-vuvuzela" filter** to attenuate the sound of this instrument and improve the audio quality of the sports commentary.

The proposed solution in this document will consist of:

- Cascading L second-order notch filters.
- Ensuring that each notch filter removes a specific frequency component f_l .

1 Analysis of the sound file vuvuzela.wav

We start by playing the audio "vuvuzela.wav." After listening to it, we find that it is indeed difficult to hear the commentators. The amplitude of the vuvuzela sound is greater than that of the commentators' voices, making it hard to hear them.

2 Determination of the number L of significant sinusoidal components

To determine the number L of significant sinusoidal components (which corresponds to the number of sinusoids that form the sound of a Vuvuzela), we begin by isolating a part of the recording "vuvuzela.wav" where this instrument is alone to minimize the noise created by the commentators' voices. We then perform a magnitude spectrum analysis on this recording using the matplotlib library, as shown in the following code:

```
1 from scipy.io import wavfile
2 import matplotlib.pyplot as plt
3 import IPython.display as ipd
4 import numpy as np
5
6 path = "../wav/vuvuzela_only.wav"
7 Fs, data = wavfile.read(path)
8
9 spectrum, freqs, line = plt.magnitude_spectrum(data, Fs=Fs, color='C1')
```

We obtain the following spectrum:

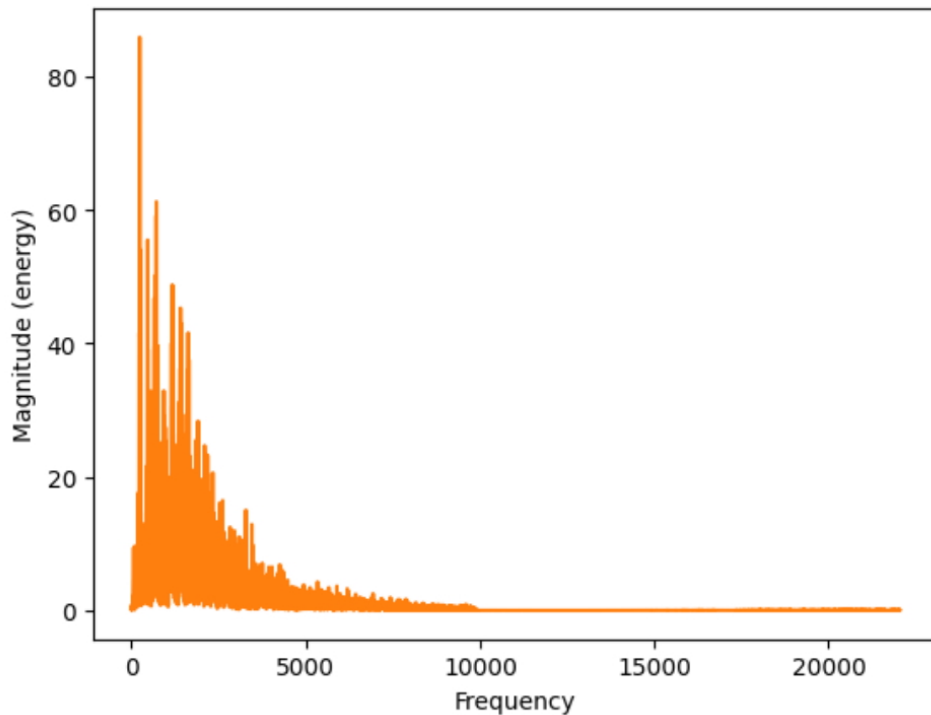


Figure 1: Magnitude spectrum of the chosen part of the recording

To focus on the frequencies with the highest magnitudes, we analyze the frequencies between 0 Hz and 1750 Hz and display only the magnitudes that exceed 35. For this, we add the following to our code:

```
1 plt.xlim(0, 1750)
2 plt.ylim(35, 100)
```

We now obtain this spectrum:

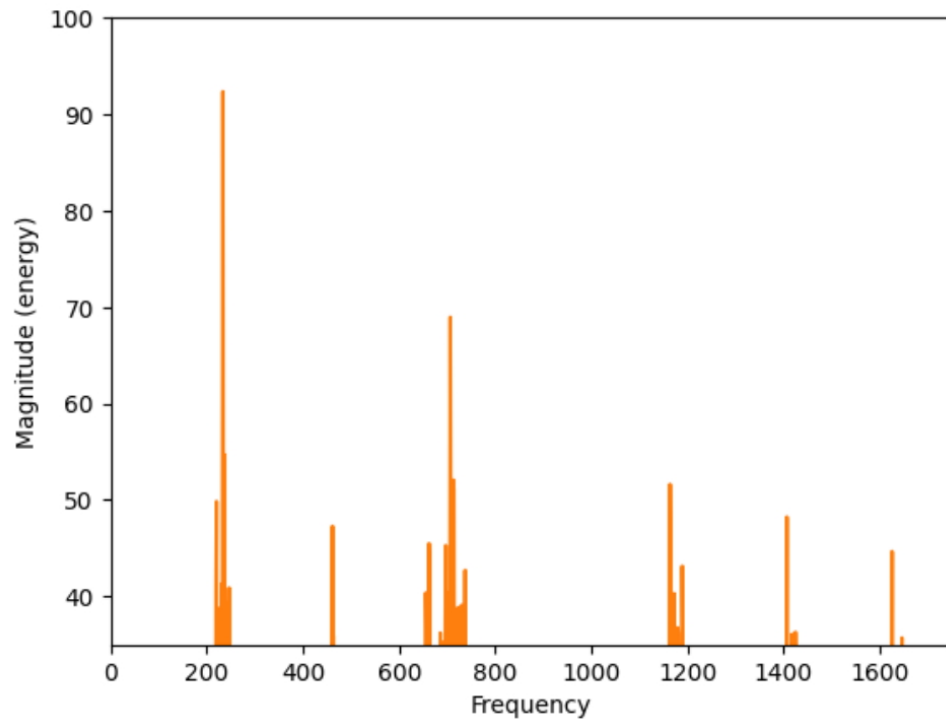


Figure 2: Parameterized magnitude spectrum

Now that the spectrum is properly parameterized, we can determine the number L of significant sinusoidal components without any issues. Indeed, $L=6$ as shown by the 6 peaks above.

3 Determination of the frequencies f_l

Next, we seek to determine the specific frequency components f_l that we need to eliminate in order to hear the commentators.

To do this, we will place points on our curve by trial and consider them as value readings. Here is the code that allows us to do this:

```
1 spectrum, freqs, line = plt.magnitude_spectrum(data, Fs=Fs, color='C1')
2 plt.xlim(0, 1700)
3 plt.ylim(35, 100)
4
5 #-----
6
7 plt.scatter(233, 92)
8
9 plt.scatter(2*233, 48)
10
11 plt.scatter(3*233, 70)
12
13 #plt.scatter(4*233,49)
14
15 plt.scatter(5*233,52)
16
17 plt.scatter(6*233, 48)
18
19 plt.scatter(7*233, 45)
20
21 print("Voici la liste des frequences (elles ont ete normalises pour etre des multiples de la
22       fondamentale) :")
23 print(f"[233, {2*233}, {3*233}, {5*233}, {6*233}, {7*233}]")
24 print(f"L'absence de la frequence {4*233}Hz est remarquable.")
```

We now obtain this spectrum:

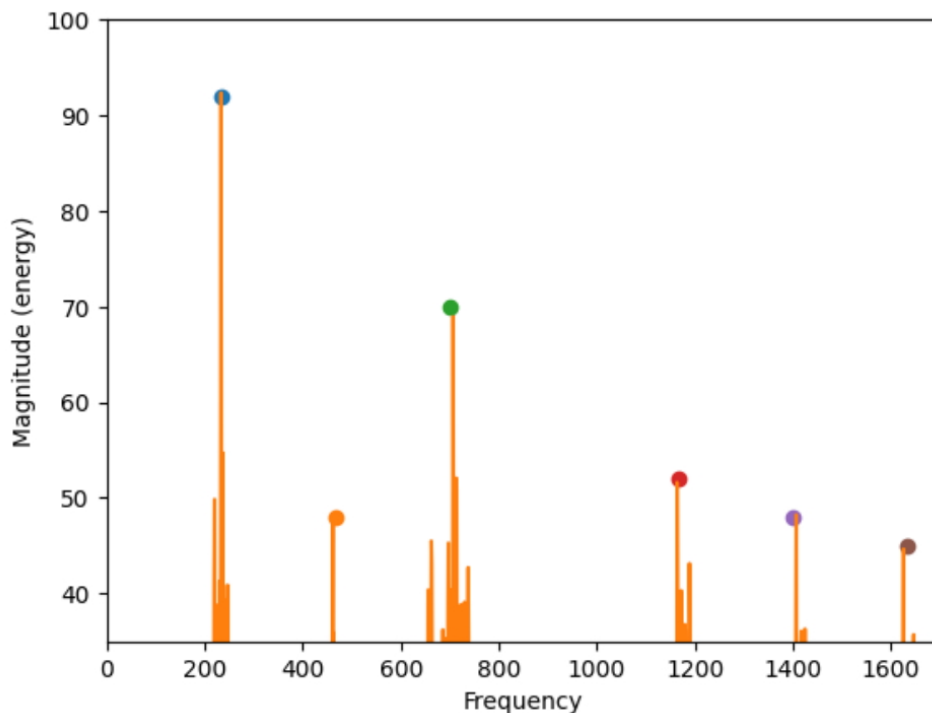


Figure 3: Parameterized magnitude spectrum with value readings

The code displays: "Here is the list of frequencies (they have been normalized to be multiples of the fundamental frequency): [233, 466, 699, 1165, 1398, 1631]
The absence of the frequency 932 Hz is notable."

We now have our specific frequency components f_l : **233 Hz, 466 Hz, 699 Hz, 1165 Hz, 1398 Hz, and 1631 Hz.**

It can be noted, as highlighted by the code, that we have a fundamental frequency of 233 Hz and its harmonics (the frequency being an integer multiple of the fundamental frequency).

4 Approximate determination of m

To determine the value of m, we will start from a rejected bandwidth value that we will use. The formula for the width of the rejected band is presented below:

- Largeur de la bande rejetée à -3dB ($|H(j\omega_c)| = |T_0|/\sqrt{2}$):

$$\left. \begin{array}{l} \omega_{c1} = \omega_0(-m + \sqrt{m^2 + 1}) \\ \omega_{c2} = \omega_0(m + \sqrt{m^2 + 1}) \end{array} \right\} \Rightarrow \Delta\omega = \omega_{c2} - \omega_{c1} = 2m\omega_0$$

Figure 4: Excerpt from Chapter 2: Second-order filters - S4 Electronics Course

We have:

$$\Delta\omega = \omega_{c2} - \omega_{c1} = 2m\omega_0$$

Which allows us to express m as:

$$m = \frac{2\Delta\omega}{\omega_0}$$

From graphical reading, we determine the rejected bandwidths around our specific frequency components f_l as follows:

1. 233 Hz: 80 Hz => $m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.68$
2. 466 Hz: 10 Hz => $m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.04$
3. 699 Hz: 90 Hz => $m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.25$
4. 1165 Hz: 45 Hz => $m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.08$
5. 1398 Hz: 33 Hz => $m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.04$
6. 1631 Hz: 30 Hz => $m \approx \frac{2\Delta\omega}{\omega_0} \approx 0.04$

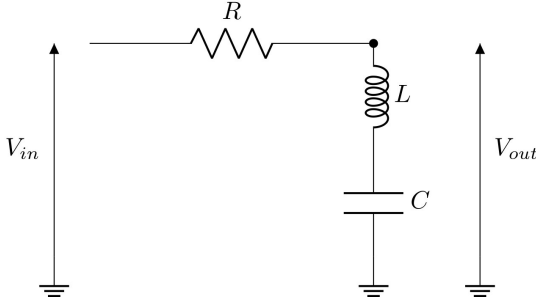
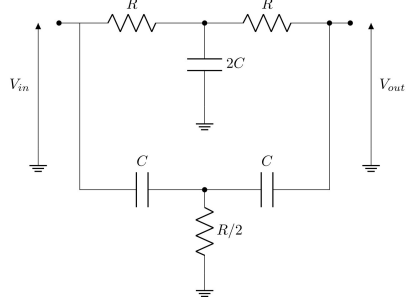
5 Implementation of the filter

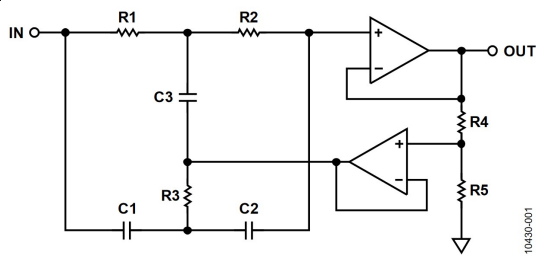
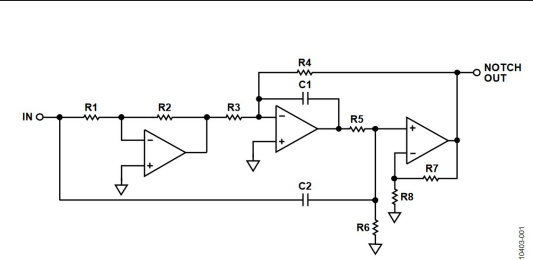
We have now determined the parameters of the different filters. It is noteworthy that we will implement a filter for each value of ω_0 . To summarize:

$$T_0 = 1, \omega_0 = [233, 466, 699, 1165, 1398, 1631], m = \frac{2\Delta\omega}{\omega_0}.$$

5.1 Choice of a notch filter topology

To compare the different filters, we present their main characteristics in a table (All the information presented here comes from the Jupyter Notebooks and LTSpice simulations related to each filter).

Filter	RLC Filter	Passive Twin T Filter
Diagram		
Transfer function	$H(p) = \frac{1+LCp^2}{LCp^2+RCp+1}$	$H(p) = \frac{(RC)^2p^2+1}{(RC)^2p^2+2RCp+1}$
Comment	The filter model is very interesting and easily manipulable, but unfortunately, the internal resistance of the coil disrupts the model and makes it incorrect.	The filter model shows that the damping coefficient m is fixed. This prevents us from choosing the bandwidth, so this filter is not retained. (Here our transfer function is probably incorrect based on our tests, but it seems evident that m will be constant.)

Filter	Active Twin T Filter	Active Bainter Filter
Diagram		
Transfer function	$H(p) = \frac{p^2 + (\frac{1}{RC})^2}{p^2 + p(\frac{1}{RC})(\frac{1}{1+\frac{R5}{R4}}) + (\frac{1}{RC})^2}$	$H(p) = H * \frac{p^2 + 2w_z}{p^2 + \frac{w_0}{Q} + w_0^2}$
Comment	This filter seems to be the best compromise; no real complaints can be made about it.	This filter, although very powerful, is hard to tune, and inconsistencies are observable in certain parts of the filter during the simulation.

Following this analysis, the filter that seems most appropriate for our project is the **Active Twin T**.

5.2 Determination of electronic components

To determine the electronic components necessary for the implementation of the filter, we use the code below. (The various parameters are displayed in comments at the bottom of the code).

```

1 def compute_component(f0,C1,R4,df):
2     m=df/(2*f0)
3     R1 = 1/(2*np.pi*C1*f0)
4     R2 = R1
5     R3=R1/2
6     C3=2*C1
7     Q = 1/(2*m)
8     R5=R4*(4*Q-1)
9     return {"C1": "{:.2e}".format(C1), "C2": "{:.2e}".format(C1), "R1": "{:.2e}".format(R1), "R2": "{:.2e}".format(R1), "R3": "{:.2e}".format(R3), "R4": "{:.2e}".format(R4), "R5": "{:.2e}".format(R5)}
10
11
12 # example

```

```

13 C1=1*(10**-9)
14 R4=1*(10**3)
15 freqs= [
16     {"f0": 233, "df": 100},
17     {"f0": 466, "df": 250},
18     {"f0": 699, "df": 175},
19     {"f0": 1165, "df": 290},
20     {"f0": 1398, "df": 250},
21     {"f0": 1631, "df": 300},
22 ]
23
24 for freq in freqs:
25     print(compute_component(freq["f0"],C1,R4,freq["df"]))
26
27 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '6.83e+05', 'R2': '6.83e+05', 'R3': '3.42e+05', 'R4':
28   '1.00e+03', 'R5': '8.32e+03'}
29 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '3.42e+05', 'R2': '3.42e+05', 'R3': '1.71e+05', 'R4':
30   '1.00e+03', 'R5': '6.46e+03'}
31 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '2.28e+05', 'R2': '2.28e+05', 'R3': '1.14e+05', 'R4':
32   '1.00e+03', 'R5': '1.50e+04'}
33 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '1.37e+05', 'R2': '1.37e+05', 'R3': '6.83e+04', 'R4':
34   '1.00e+03', 'R5': '1.51e+04'}
35 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '1.14e+05', 'R2': '1.14e+05', 'R3': '5.69e+04', 'R4':
36   '1.00e+03', 'R5': '2.14e+04'}
37 # {'C1': '1.00e-09', 'C2': '1.00e-09', 'R1': '9.76e+04', 'R2': '9.76e+04', 'R3': '4.88e+04', 'R4':
38   '1.00e+03', 'R5': '2.07e+04'}

```

5.3 LTSpice Test of the chosen cascade of filters

To perform an LTSpice simulation of our technical solution, we create a filter for each specific frequency component f_l and cascade them.

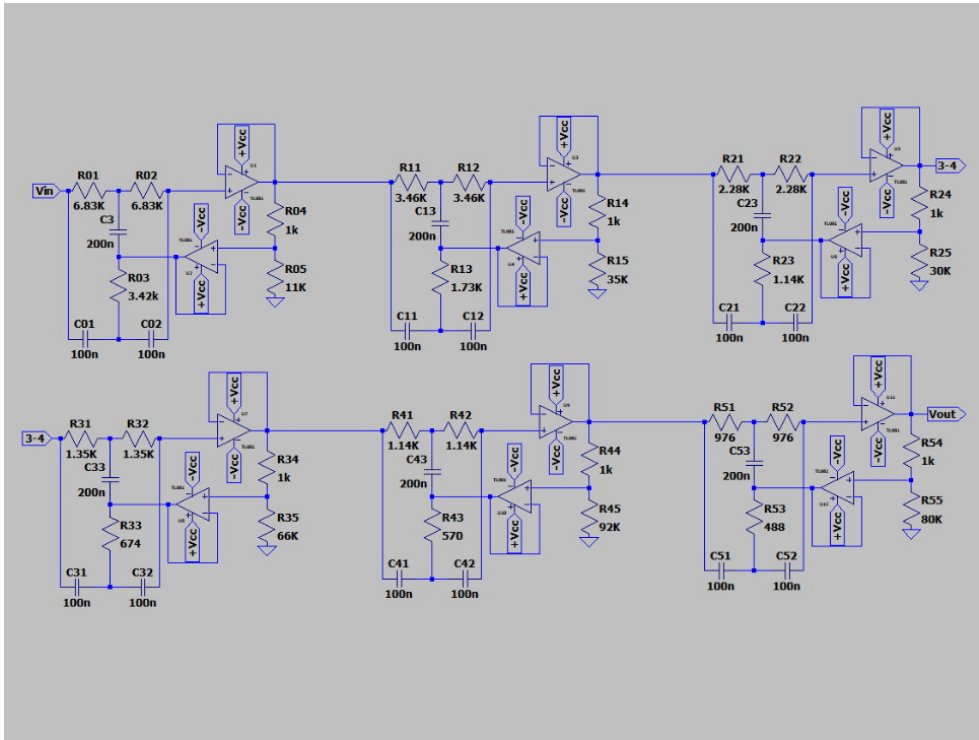


Figure 5: Circuit created in LTSpice

To verify the correct operation of the circuit, we perform a spectral analysis via FFT.

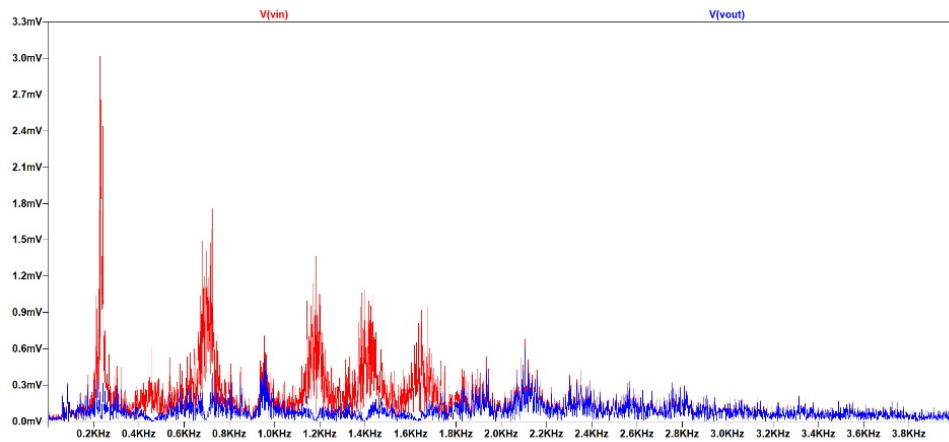


Figure 6: Spectral analysis via FFT

When we look at the analysis, we see that we have successfully removed the magnitude peaks specific to our "vuvuzela-only" recording. The FFT analysis allows us to conclude that the filter works well!

6 Conclusion

Through this project, we have successfully implemented a filter that removes the "Vuvuzela" noise present in the audio signal, significantly improving the intelligibility of the commentators' voices. We also demonstrated how to approach a problem involving sound filtering by analyzing the audio signal, determining significant frequency components, and finally implementing a cascading series of notch filters to target these frequencies. Our solution is a practical demonstration of the principles of signal processing in audio engineering.