Signal Processing of a Voice Signal

ECE 303 01: Final Project

Ethan Bolt, Ryan Jones, and Connor Prikkel

University of Dayton

Contents

1 Introduction

As a culmination of the techniques learned throughout the semester, students are tasked to remove added noise from a voice sample using different filtering methods. To that end, the students develop the transfer function for a 3rd-order, low-pass Sallen-Key filter to remove the noise from the generated voice clip. By varying the configuration, the general Sallen-Key topology is specialized into a Butterworth or Chebyshev filter; both approaches are used to filter the noisy sample. Various graphs are generated to compare the behavior of the two filters and the signal at different stages of processing.

2 Theory

2.1 Process

Using MATLAB, each member of the group records a short clip of them reciting the famous Mao quote, "Political power grows from the barrel of a gun." Each signal is processed, adding a random amount of Gaussian noise and a chirp sweep[1]. The nosy signal is ran through both a student-parameterized Butterworth and Chebyshev low-pass filter to remove the Gaussian noise and frequency sweep. Time domain, spectrograms, and fast-Fourier transfer graphs are generated, included in **Results**. The general behavior of the two filters is explored in **Parameterized Sallen-Key Filters**.

2.2 Derivation of the Transfer Function of a Sallen-Key Amplifier

Analyzing the stage prior to the 1st-order LPF, since the operational amplifier is noninverting,

$$v_{out} = \left(1 + \frac{R_{f2}}{R_{f1}}\right) v_-$$
$$\Rightarrow v_- = \left(\frac{R_{f1}}{R_{f1} + R_{f2}}\right) v_{out}$$

Letting $K = 1 + \frac{R_{f2}}{R_{f1}}$,

$$v_- = \frac{v_{out}}{K}$$

Using voltage divider at the node in between R_1 and R_2 (y),

$$v_+ = \frac{v_y}{R_2 + \frac{1}{sC_2}} \cdot \frac{1}{sC_2} = \left(\frac{1}{R_2C_2s + 1}\right) v_y$$
$$\Rightarrow v_y = (R_2C_2s + 1)v_+$$

Applying node voltage at y,

$$\frac{v_y - v_{in}}{R_1} + \frac{v_y - v_+}{R_2} + \frac{v_y - v_{out}}{1/sC_1} = 0$$

[1] Gaussian noise has a generated SNR $\in (0.01, 1)$. The chirp sweep starts from 200 Hz, reaching 16 kHz over course of the signal.

Substituting $(R_2C_2+1)v_+ \to v_y$,

$$\frac{(R_2C_2s+1)v_+ - v_{in}}{R_1} + \frac{(R_2C_2s+1)v_+ - v_+}{R_2} + \frac{(R_2C_2s+1)v_+ - v_{out}}{1/sC_1} = 0$$
$$\frac{(R_2C_2s+1)v_+}{R_1} - \frac{v_{in}}{R_1} + \frac{(R_2C_2s+1)v_+}{R_2} - \frac{v_+}{R_2} + \frac{(R_2C_2s+1)v_+}{1/sC_1} - \frac{v_{out}}{1/sC_1} = 0$$

Since $v_- = v_+$, by substituting $v_+ = \frac{v_{out}}{K}$,

$$\frac{(R_2C_2s+1)v_{out}}{KR_1} - \frac{v_{in}}{R_1} + \frac{(R_2C_2s+1)v_{out}}{KR_2} - \frac{v_{out}}{KR_2} + \frac{(R_2C_2s+1)v_{out}}{1/sC_1} - \frac{v_{out}}{1/sC_1} = 0$$
$$\left[\left(\frac{R_2C_2s+1}{K}\right)\left(\frac{1}{R_1}+\frac{1}{R_2}+sC_1\right) - sC_1 - \frac{1}{KR_2}\right]v_{out} = \frac{1}{R_1}v_{in}$$
$$\left[(R_2C_2s+1)\left(\frac{1}{R_1}+\frac{1}{R_2}+sC_1\right) - sC_1K - \frac{1}{R_2}\right]\frac{1}{K}v_{out} = \frac{1}{R_1}v_{in}$$
$$\left[\frac{R_2C_2s}{R_1} + \frac{1}{R_1} + \frac{R_1C_2s}{R_1} + \frac{R_1R_2C_1C_2s^2}{R_2} + sC_1K - \frac{sC_1}{K}\right]\frac{R_1}{K}v_{out} = v_{in}$$
$$\left[\frac{R_1R_2C_1C_2s^2 + (R_1C_1 + R_1C_2 + R_2C_2 - R_1C_1K)s + 1}{R_1}\right]\frac{R_1}{K}v_{out} = v_{in}$$
$$\therefore H_{amp}(s) = \frac{K}{R_1R_2C_1C_2s^2 + [(1-K)R_1C_1 + R_1C_2 + R_2C_2]s + 1}$$

Including the cascaded LPF,

$$\begin{aligned} H(s) &= H_{amp}(s)H_{LPF}(s) = H_{amp}\left(\frac{1/R_3C_3}{s+1/R_3C_3}\right) \\ &= \left(\frac{K}{s^2 + \frac{(1-K)R_1C_2+R_1C_2+R_2C_2}{R_1R_2C_1C_2}s + \frac{1}{R_1R_2C_1C_2}}\right)\left(\frac{1/R_3C_3}{s+1/R_3C_3}\right) \\ &= \left(\frac{(R_{f1}+R_{f2})/R_{f1}}{s^2 + \frac{(R_{f2}/R_{f1})R_1C_2+R_1C_2+R_2C_2}{R_1R_2C_1C_2}s + \frac{1}{R_1R_2C_1C_2}}\right)\left(\frac{1/R_3C_3}{s+1/R_3C_3}\right) \end{aligned} \tag{1}$$

3 Results

For all MATLAB-generated plots, please find them exported in the PDF attached in conjunction with this report.

3.1 Comparative Analysis of Signal-to-Noise Ratios

Signal	P_{org} **(W)**	P_{noise} **(W)**	P_{butter} **(W)**	P_{cheby} **(W)**
Connor's Voice	0.0599	1.5624	0.6823	0.2548
Ethan's Voice	0.0004	1.3855	0.5792	0.2224
Ryan's Voice	0.0928	1.6096	0.7191	0.2653

Figure 1: Tabulation of signal powers

Signal	**SNR**$_{noise}$	**SNR**$_{butter}$	**SNR**$_{cheby}$
Connor's Voice	0.03834	0.08779	0.2351
Ethan's Voice	0.0002887	0.0006906	0.001799
Ryan's Voice	0.05765	0.1291	0.3498

Figure 2: Tabulation of signal-to-noise ratios

3.2 Parameterized Sallen-Key Filters

3.2.1 Calculation of Filter Components

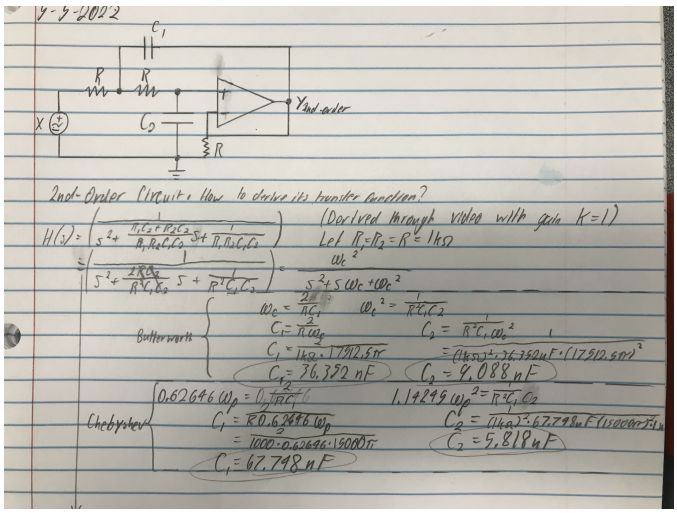
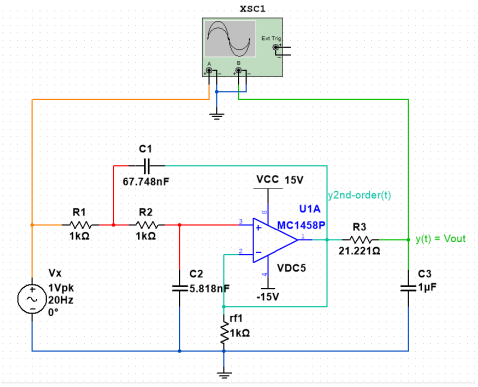


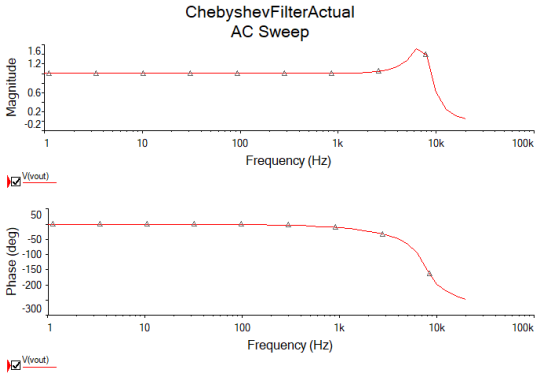
Figure 3: Photograph of hand-written work to determine the values of the resistors and capacitors of the filters

In Figure 3 above, starting with the transfer function, all of the resistors were chosen to be 1 kΩ. Additionally, unity gain was selected, eliminating rf2. The values of the capacitors were then calculated by equating coefficients with the given third-order transfer function.

3.2.2 The Low-Pass Chebyshev



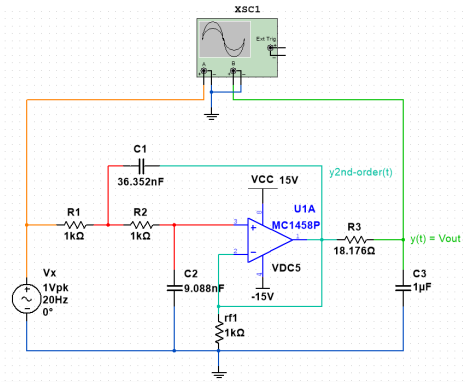
(a) Schematic



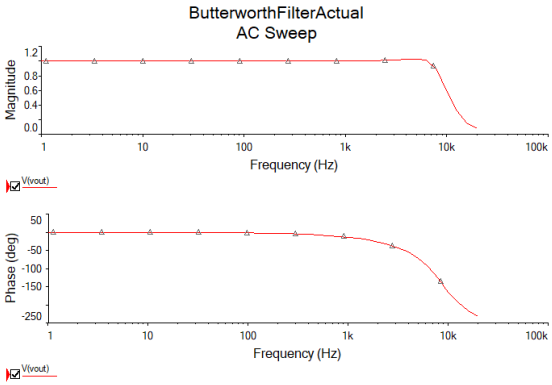
(b) Magnitude and phase plots from AC sweep

Figure 4: 3rd-order, low-pass Chebyshev filter as constructed in Multisim

3.2.3 The Low-Pass Butterworth



(a) Schematic



(b) Magnitude and phase plots from AC sweep

Figure 5: 3rd-order, low-pass Butterworth filter as constructed in Multisim

Please see ***Appendix B*: Filter Implementation** for MATLAB implementations of both filters.

4 Discussion & Conclusion

The Butterworth filter had a very steady gain magnitude of 1 up to the cut-off frequency, after which it dropped off relatively slowly: at 15 kHz, the gain magnitude was 0.195. By comparison, the Chebyshev was relatively flat until about 2 kHz, after which it spiked to a magnitude of 1.51 just before the cut-off frequency. It dropped off more quickly than the Butterworth: at 15 kHz, the gain magnitude was 0.152. The phase response of both filters is very similar: both resemble a negative sigmoid function in shape with the maximum downward slope slightly after the cut-off frequency. The Butterworth begins to drop off sooner but does so smoother and more linearly. By contrast, the Chebyshev stays at a phase angle of 0 for longer but drops more sharply.

Neither filter was effective in obtaining the original voice message. However, putting the noise-modified signal through the Butterworth filter returned a message that was slightly less attenuated in comparison to the Chebyshev. Theoretically, our results indicate that the Butterworth filter should also return a message closer to the original as the bode plot had reached unity gain (0 dB) as compared to the Butterworth which reached -5 dB.

Despite utilizing three voice messages with distinct frequency ranges, there was no perceptible difference in how the different messages were handled by the filters. The output in each case still sounded like incomprehensible "monkey garble."

Changing the passband or cut-off frequencies had no effect on the perceived output, regardless of voice. Low and high-frequency noise is still present throughout even when these values are changed (which in turn change ω_c). The noise above the passband frequency will be cut out, but decreasing the passband frequency to cut the majority of the noise out cuts the actual voice message out.

5 Appendices

5.1 *Appendix A*: Signal Processing Script

Please find attached to the submission `signal_processing.mlx`, the MATLAB Live Script developed for processing and creating the figures.

5.2 *Appendix B*: Filter Implementations

```
syms s;

wc = 55017605.63 / 1000;
butterworth_filter  = tf(wc, [1 wc wc^2]) * tf(wc, [1 wc]);

wp = wc / 1.1675;
chebyshev_filter    = tf(0.71570 * wp^2, [1 0.62646 * wp 1.41245 * wp^2]) *
   tf(wp, [1 wp]);
```

Figure 6: MATLAB implementation of the Chebyshev and Butterworth filters using calculated ω_c and the Symbolic Toolbox