MUS424:
Signal Processing Techniques for Digital Audio Effects `Handout #8`
Jonathan Abel, David Berners `April 18, 2019`

Homework #3: Time-Varying Resonant Filters [25 points]
Due Date: April 25, 2019

Time-Varying Resonant Filters

Submission Instructions

Submit via Canvas. Create **a single compressed file** (`.zip`, `.tar` or `.tar.gz`) containing all your submitted files. Upload the compressed file to the homework submission dropbox. Name the file using the following convention:

`<suid>_hw<number>.zip`

where `<suid>` is your Stanford username and `<number>` is the homework number. For example, for Homework #1 my own submission would be named `cavdir_hw1.zip`.

For coding problems (either C++ or Matlab code), submit all the files necessary to compile/run your code, including instructions on how to do it. In case of theory problems, submit the solutions in **PDF format only**. LaTeX or other equation editors are preferred, but scans are also accepted. In case of scanned handwriting, make sure the scan is legible. Illegible homework will not be graded.

Lab3: Time-Varying Resonant Filters

In this lab, we will explore resonant and time-varying filters. The provided file contains two plug-ins incorporating a resonant low-pass filter. The first plug-in, shown in Figure 1, explores a simple resonant low-pass filter with varying cutoff frequency and resonance. The second plug-in, shown in Figure 2, uses an LFO to vary the filter cutoff frequency for a Wah-Wah effect. For all questions, please turn in your code, any audio files rendered for the problem, and a list of parameters chosen to produce the desired output.

Problem 1. [10 Points]

The resonant low-pass filter is specified by its cutoff frequency ω_c and resonance Q. Its transfer function is given by

$$H(s) = \frac{1}{(s/\omega_c)^2 + \frac{1}{Q}(s/\omega_c) + 1}. \quad (1)$$

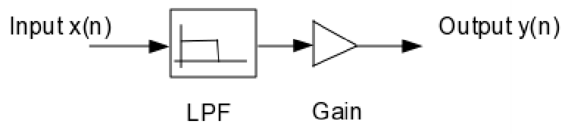

Figure 1: *Resonant Low Pass.*

1(a). [**4 Points**] Plot the transfer function magnitude and phase, and sketch the trajectory of the transfer function poles for

- $Q = 2^{[-4:2]}$ $\omega_c = 2\pi \cdot 1000$ radians/second
- $Q = 2$ $\omega_c = 2\pi \cdot 1000 \cdot 2^{[-2:2]}$ radians/second

1(b). [**6 Points**] A quasi-complete implementation of the filter described in Figure 1 is provided in the *ResonantLowPass* project. Complete the functions used to convert the cutoff frequency and resonance into the needed second-order digital filter coefficients. Apply the filter to the white noise sequence supplied, using various cutoff frequencies and resonances.

Problem 2. [**15 Points**]

In this exercise, the resonant low-pass filter of Problem 1 will be modified to include a low-frequency oscillator (LFO) which drives the filter cutoff frequency.

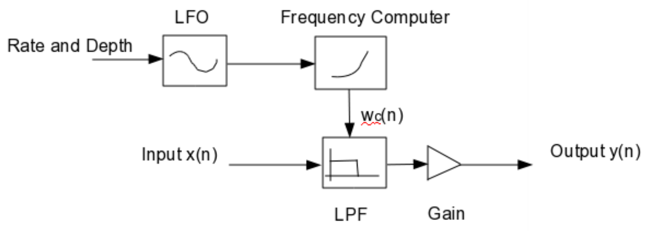

Figure 2: *LFO-Driven Wah-Wah*

The *WahWah* project is an augmented version of *ResonantLowpass.* Feel free to add the code you wrote for the previous problem into this code.

2(a). [**2 Points**] So as to make the filter smoothly varying over time, the user controls will be tracked using leaky integrators. For example, the resonance control will generate a sequence of targets Q_T, which drive the tracking filter,

$$Q(n) = (1-\alpha)Q_T + \alpha Q(n-1), \tag{2}$$

to produce a smoothly varying resonance sequence $Q(n)$, according to the forgetting factor α. Implement leaky integrators for the controls ω_c, Q, and the filter gain g. Design α to produce a 20 ms time constant. You will only need to implement a few lines in the *SlewedParameter* class. Note: You may also need to add an additional leaky integrator on the time-varying filter center frequency to avoid unwanted clicks or pops.

2(b). [**4 Points**] Implement a low-frequency oscillator (LFO) with output signal $\mu(n)$ either by

- Forming the sine of a phase counter,

  $$\mu(n) = \sin(\phi(n)), \qquad \phi(n) = \mathrm{rem}(\phi(n-1) + \delta(\omega_m), 2\pi), \tag{3}$$

  where the phase increment $\delta(\omega_m)$ is given by

  $$\delta(\omega_m) = \omega_m / f_s, \tag{4}$$

  with f_s being the sampling rate. See the matlab `rem` for help.

- The *magic circle* algorithm. See the canvas site for a paper reference.

2(c). [**4 Points**] Form a *frequency computer* to convert the LFO state $\mu(n)$ and tracked cutoff frequency control $\omega_c(n)$ into a filter cutoff frequency which sweeps exponentially between a factor $1/\sqrt{\rho}$ and $\sqrt{\rho}$ of the specified cutoff frequency, ρ being the frequency ratio control or depth. Use the guitar track provided to verify that the filter cutoff frequency smoothly changes with time.

2(d). [**5 Points**] Finally, you will Implement a *stereo side-chain.* Note that while is possible to implement "proper" side-chaining in VST 2, this feature is not fully supported (VST 3 offers "official" support for side-chaining). In this problem, we will simply use the right (second) channel to control the cutoff frequency of the resonant lowpass filter.

Borrowing code from the previous labs, implement a detector to estimate the level of the right channel $\lambda(n)$. Use $\lambda(n)$ to control the cutoff frequency of the resonant lowpass filter. Use the following expression to compute the actual cutoff frequency as a function of the user specified cutoff ω_c and the detected level λ:

$$\omega_c e^{\lambda(n)\log_{10}\rho} \tag{5}$$

Apply the lowpass filter to both channels. Test the modified plugin by using a stereo recording with a guitar track on the left channel and a drum track on the right channel.