

University of Sheffield

COM3502-4502-6502

Speech Processing



Programming Assignment

Instructions

Prof. Roger K. Moore & Dr. Stefan Goetze

Department of Computer Science

24th October 2020

Contents

1	Overview (please read carefully)	2
1.1	Logistics	2
1.2	Provided Materials	2
1.3	Hand-In Procedure	3
2	Part-I: Voice Manipulation	4
2.1	Continuous Speech Input	4
2.2	Filtering in the Time-Domain	5
2.3	Filtering in the Frequency-Domain	6
2.4	Low Frequency Oscillator	7
2.5	Amplitude Modulation: <i>Tremolo</i>	9
2.6	Ring Modulation	9
2.7	Frequency Shifting	10
2.8	Frequency Modulation: <i>Vibrato</i>	12
2.9	Time Delay Effects: <i>Echo, Comb Filtering and Flanger</i>	12
2.10	Feedback: <i>Reverberation</i>	13
3	Part-II: Real-Time Voice Changer	15
3.1	General Requirements	15
3.2	COM4502-6502 Only	15
3.3	Final Steps	15

1 Overview (please read carefully)

Before proceeding with this assignment you should have completed the ‘Pure Data’ (Pd) Introductory Programming Exercise, and completed the quiz on Blackboard. If not, then you are strongly advised to do so before moving on to the programming assignment described below. In any event, it may be advisable to review the introductory exercise, especially Sections 7 and 8 which cover working with real-time audio and speech.

Note: *This programming assignment is worth 55% of the overall course mark.*

This is an individual assignment.

You are permitted to re-use any of the Pd examples provided during the course.

1.1 Logistics

You are free to complete this assignment in your own time. However, feedback, advice and guidance is available from the post-graduate teaching assistants via the appropriate discussion board on Blackboard. Note that they will try to help you as much as possible, but it is not their job to debug your code or provide solutions to the assignment itself.

Note: *It will take some time to complete this assignment, so plan your work accordingly over the coming weeks. Read these instructions carefully, and pay particular attention to the marks associated with each component - especially the last.*

Note: *Please be aware that students registered on COM4502 and COM6502 have additional tasks to perform. These are marked ‘COM4502-6502 Only’.*

1.2 Provided Materials

In addition to these instructions, you have been provided with a .zip file containing a number of items¹ that you will need for this assignment. In particular, you have been supplied with a L^AT_EX template - `YourName.tex` - which will you use to compile your response sheet.

L^AT_EX (pronounced [lɑːtek] or [lertek]) is a *free* document preparation system for high-quality typesetting², and having a working knowledge of L^AT_EX is a key skill for any computer scientist³. L^AT_EX is very easy to learn⁴, and there are many resources available on the web, e.g. https://www.sharelatex.com/learn/Learn-LaTeX_in_30_minutes.

L^AT_EX distributions are available for Linux, Mac OSX and Windows (see: <https://www.latex-project.org/get/>). However, for this assignment, you might like to use an on-line collaborative environment such as Overleaf (<https://www.overleaf.com>).

¹The provided `screenshot.jpg` is simply a placeholder which you will replace with your own images.

²L^AT_EX includes features designed for the production of technical and scientific documentation, so it is the de-facto standard for the publication of scientific/technical papers and reports. Unlike WYSIWYG (“*What You See Is What You Get*”) word processors (such as *Microsoft Word* or *OpenOffice*), L^AT_EX is based on plain-text source files (containing html-style markup) which are compiled into a typeset document. This separation of ‘content’ from ‘style’ facilitates the production of very professional scientific documents in terms of their consistency, readability and design. You can find out more about L^AT_EX here: <https://www.latex-project.org>.

³The Department strongly recommends that you consider using L^AT_EX for your Dissertation, and Prof. Moore has provided a template here: http://staffwww.dcs.shef.ac.uk/people/R.K.Moore/campus_only/USFD_Academic_Report-LaTeX-Template.zip.

⁴Students who took **COM2009-3009 Robotics** should be familiar with L^AT_EX already.

★ **Step 0:** Before you start working with L^AT_EX, edit the filename of the provided `.tex` file by replacing `YourName` with your name.

Your edited `.tex` file should compile to produce a `.pdf` document with your name on the front cover, followed by your responses to a number of questions. You will be submitting the `.pdf` file when you have completed the assignment, *not* the `.tex` file.

1.3 Hand-In Procedure

Once you have completed the assignment, one member of each pair should submit a `.zip` file (via Blackboard) containing your response sheet (in `.pdf` format) and the requested Pd source files. **Do not** submit your L^AT_EX source files. The `.zip` filename should be of the form `YourName.zip`.

Please make sure that your name is shown correctly on the front page of your response sheet (as explained in the previous section).

Standard departmental penalties apply for late hand-in⁵ and plagiarism⁶.

Feedback (including provisional marks) will be provided via Blackboard within three working weeks of the hand-in date.

The deadline for handing-in this assignment (via MOLE) is ...

COM3502-4502: 17:00 Friday 18th December 2020

COM6502: 17:00 Friday 22nd January 2021

⁵<https://sites.google.com/sheffield.ac.uk/comughandbook/general-information/assessment/late-submission>

⁶<https://sites.google.com/sheffield.ac.uk/comughandbook/general-information/assessment/unfair-means>

2 Part-I: Voice Manipulation

As you saw in Lecture 1, speech processing has many applications across a wide variety of market sectors. One area of particular interest is the creation of appropriate voices for fictional characters in television, cinema, games and related areas of edutainment⁷. Such robot, alien or cartoon voice effects are often achieved by manipulating the speech of a voice actor (either in real-time or in post-production⁸) - a process that is referred to as ‘Voice FX’ or ‘Vocal FX’.

The aim of this assignment is to implement a range of speech processing algorithms for *real-time* voice manipulation. The final outcome will be a ‘Voice Changer’ that can be configured to perform a variety of different manipulations on your own voice. Part-I involves creating the core components in Pd, and Part-II brings them together into a single [VoiceChanger] application. Please work through the following sections in the order given, providing responses the relevant questions as you come to them (by editing your .tex file).

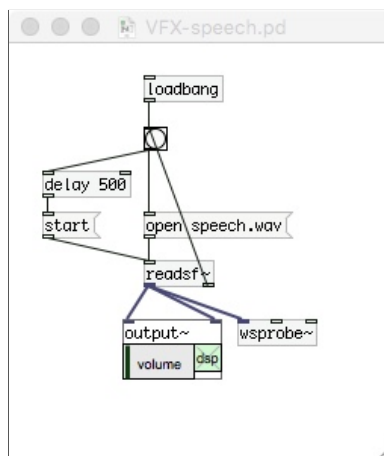
Note: *In Part-II you will reuse many of the components you create in Part-I. Hence, you are strongly advised to adopt professional working practices, e.g. saving versioned copies of each piece of software that you develop.*

Note: *Some of the steps below require you to listen critically to sounds generated by your software. It is thus advisable for you to use headphones and/or work in a quiet environment. Take care not to annoy other people with your sound output.*

2.1 Continuous Speech Input

Your first task is to program a Pd patch that can provide a continuous stream of real-time speech that can be fed into the various manipulation techniques which follow.

★ **Step 1:** Create the following patch . . .



Note: *Recall from the introductory programming exercise that the audio output control will appear when you create [output~] object.*

⁷Wilson, S., & Moore, R. K. (2017). Robot, alien and cartoon voices: implications for speech-enabled systems. In *1st Int. Workshop on Vocal Interactivity in-and-between Humans, Animals and Robots (VIHAR-2017)*. Skovde, Sweden.

⁸Rose, J. (2012). *Audio Postproduction for Film and Video*. Taylor & Francis.

★ **Step 2:** Click on the **bang** at the top of the patch, increase the volume slider on **[output~]** and verify that you can hear the speech⁹ on a continuous loop.

★ **Step 3:** Now click on **[wsprobe~]**. This should open a separate patch that displays the waveform and spectrum in real-time (as you have seen several times in the lectures). Experiment with the various options provided in the **[wsprobe~]** GUI, and take particular notice of the differences between voiced and voiceless sounds.

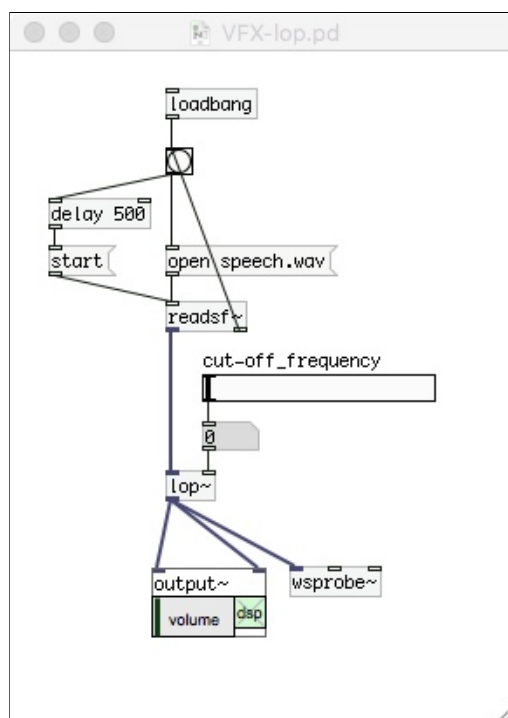
QUESTION 1 (worth up to 5 marks)

Provide a screenshot of **[wsprobe~]** for a typical voiced sound, and explain the features in the waveform and spectrum that distinguish it from an unvoiced sound. *Hint: use the 'snapshot' feature in [wsprobe~] to obtain a static display.*

2.2 Filtering in the Time-Domain

One of the most basic manipulations in Voice FX is to 'filter' a signal in order to increase or decrease energy at particular frequencies. The simplest types of filter are 'low-pass', 'high-pass' and 'band-pass', and these may be implemented in Pd using the **[lop~]**, **[hip~]** and **[bp~]** objects respectively.

★ **Step 4:** Modify your patch to include a low-pass filter with a 'cut-off frequency' controlled by a horizontal slider covering the range 0 to 10,000 Hz as shown here¹⁰ ...



★ **Step 5:** Experiment with different values for the cut-off frequency, and observe the consequences for the output sound as well as the waveform and spectrum displays.

⁹The provided **speech.wav** file is the same as the one used in the Lectures (and as provided in **com3502-4502-6502_pd-examples.zip** available on MOLE).

¹⁰Recall that the slider's range can be set by right-clicking and adjusting its 'properties'.

QUESTION 2 (worth up to 5 marks)

Which sounds are most affected when the low-pass cut-off frequency is set to around 500 Hz - vowels or consonants - and why?

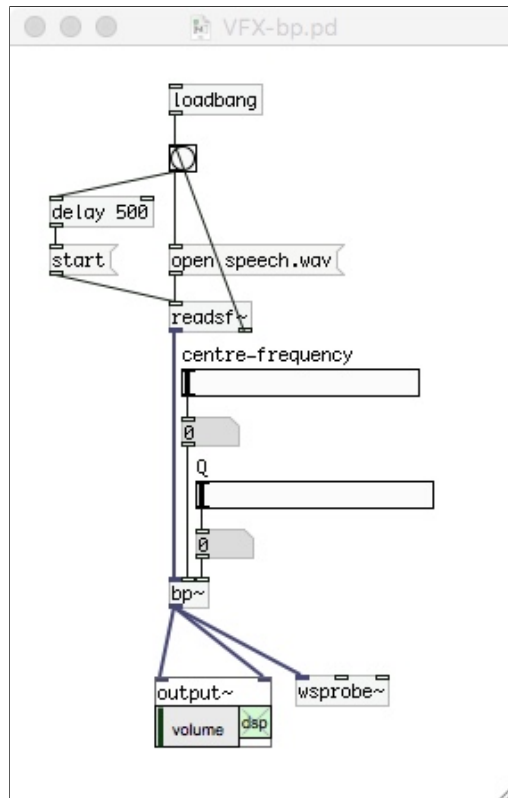
★ **Step 6:** Edit the [lop~] object to become a [hip~] object, and again experiment with different values for the cut-off frequency.

QUESTION 3 (worth up to 5 marks)

How is it that the speech is still quite intelligible when the high-pass cut-off frequency is set to 10 kHz?

As you have seen, these simple low-pass and high-pass filters have only one control parameter - their cut-off frequency. A simple band-pass filter has two parameters - its 'centre frequency' and its ' Q factor'. The Q of a band-pass filter is defined as $Q = 1/B$, where B is the bandwidth (in Hz). Hence a high- Q filter has a narrow bandwidth and *vice versa*.

★ **Step 7:** Edit the [hip~] object to become a [bp~] object, and add another horizontal slider covering the range 0 to 100 as shown here ...



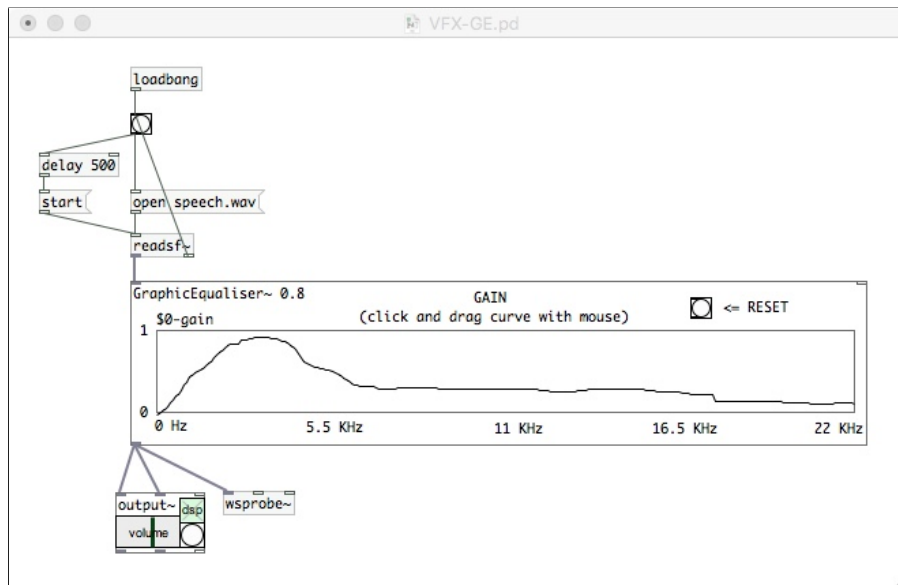
★ **Step 8:** Experiment with different values for the centre frequency and Q .

2.3 Filtering in the Frequency-Domain

The [lop~], [hip~] and [bp~] objects operate directly on the speech waveform. Hence, they are examples of 'time-domain' signal processing. An alternative to time-domain processing is to process signals in the 'frequency-domain', i.e. by modifying the spectrum rather than the waveform.

A classic example of filtering in the frequency-domain is a ‘graphic equaliser’. This is a common component of a high-fidelity (HiFi) audio system, and it allows the spectrum to be shaped by setting gains across a range of frequencies. You have been provided with a Pd ‘abstraction’¹¹ [GraphicEqualiser~] that can perform this function.

★ **Step 9:** Remove your [bp~] object and its slider controls, and insert the provided [GraphicEqualiser~]. Connect it between [readsf~] and [output~], and experiment with different filter profiles by clicking on the graph (in ‘run mode’) and dragging the curve into different shapes. Note how you can make low-pass, high-pass and band-pass filters (and many more), simply by drawing the appropriate shapes, as shown here . . .



★ **Step 10:** Open the [GraphicEqualiser~] abstraction to see how it works (by right-clicking on the object, and selecting **open**). Take particular note of how the GUI objects (the graph and reset button) are made to appear in the parent patch using the ‘**graph-on-parent**’ option in the sub-patch properties¹².

QUESTION 4 (worth up to 5 marks)

COM3502-4502-6502: The [GraphicEqualiser~] object uses an FFT internally; what does FFT stand for and what does an FFT do?

COM4502-6502 ONLY: What is a DFT and how is it different from an FFT?

★ **Step 11:** Use [GraphicEqualiser~] to simulate the effect of a land-line telephone by eliminating all energy below 300 Hz and above 3,400 Hz.

2.4 Low Frequency Oscillator

Many ‘Voice FX’ are achieved by modifying some characteristic of the speech using a low frequency oscillator or ‘LFO’. LFOs typically have two controls: **speed** (which is specified by the frequency in Hertz) and **depth** (which specifies the magnitude of the effect). Your [VoiceChanger] will require several LFOs, so it makes sense to implement one as a Pd ‘abstraction’.

¹¹Refer back to the introductory programming exercise if you have forgotten what an abstraction is.

¹²Note that you can also look inside [output~] or [wsprobe~] to see the same ‘graph-on-parent’ functionality.

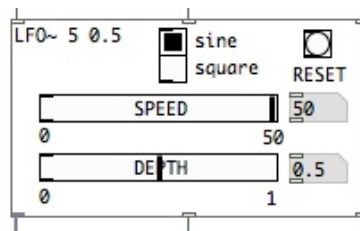
★ **Step 12:** Using the skills you acquired in the introductory programming exercise, create a Pd abstraction [LFO~] that is capable of generating an audio sine/square wave between 0 and 50 Hertz (e.g. using the [osc~] object). The **speed** and **depth** of the LFO should be controllable by suitable sliders, and they should appear in the object using the ‘graph-on-parent’ approach.

★ **Step 13:** Add an option to select either ‘sine’ or ‘square’ wave output using a Vradio button. *Hint: to obtain a square wave, multiply the output of the [osc~] object by a very large number and pass the result through [clip~ -1 1] before being multiplied by depth.*

Additional features that will be useful later in the assignment are:

- [number] GUI objects for **speed** and **depth**
- a **reset** function
- external inputs for **speed**, **depth** and **reset**
- external outputs for **speed** and **depth**
- an ability to handle creation arguments for **speed** and **depth** (also linked to **reset**)
- an ability to reset the phase of the oscillator when **speed** = 0 in order to ensure consistent output when initialising/resetting

Your resulting [LFO~] abstraction should look something like this (note the use of **speed** = 5 and **depth** = 0.5 as creation arguments) ...



Although your [LFO~] object outputs audio, you are unlikely to be able to hear it as the frequency is so low. However, you can check that it is functioning correctly by connecting it to [wsprobe~] and/or by using the appropriate **number** GUI object to wind the **speed** up to an audible frequency.

★ **Step 14:** Test your [LFO~] object by creating an [LFO~-help] object.

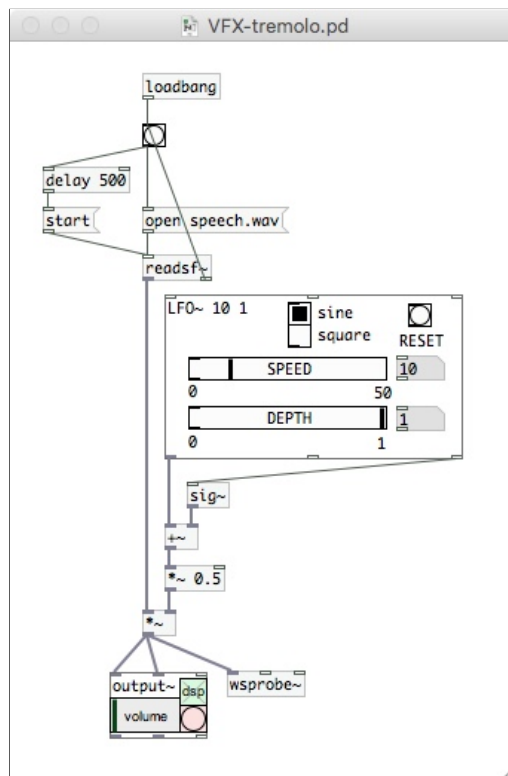
QUESTION 5 (worth up to 10 marks)

With **speed** = 50 and **depth** = 0.5, what are the minimum and maximum amplitudes of your LFO output, and how do they vary with changes in these two settings? Also, please provide two screenshots: (a) your [LFO~-help] object and (b) the internal structure of your [LFO~] object.

2.5 Amplitude Modulation: *Tremolo*

‘Tremolo’ is one of the most basic voice manipulations that makes use of an LFO. In this effect, the amplitude of a speech signal is modulated, i.e. the speech waveform is multiplied by a variable gain that ranges between 0 and 1. Your LFO outputs an audio signal between **-depth** and **+depth**. So, in order to modulate the amplitude of the speech correctly, the output of the LFO has to be scaled appropriately - in this case, by adding **depth** (e.g. using the audio math object `[+~]` connected to the **depth** output of your LFO) and dividing the result by 2 (e.g. using the audio math object `[*~ 0.5]`).

★ **Step 15:** Implement ‘tremolo’ using your `[LFO~]` object using the principles given above. The resulting program should look something like this (note the use of a `[sig~]` object to convert numerical data to audio data) ...



★ **Step 16:** Experiment with different settings for **speed** and **depth**. In particular, note that a square wave with **speed** between 3 and 4 Hz (and **depth** = 1) has a very destructive effect on the intelligibility of the output. This is because 3-4 Hz corresponds to the typical syllabic rate of speech.

2.6 Ring Modulation

Another basic effect is to *multiply* the speech signal by the output of an LFO. This is known as ‘ring modulation’.

★ **Step 17:** Modify your implementation of ‘tremolo’ by removing the two scaling objects (`[+~]` and `[*~ 0.5]`) and connecting the audio output of your `[LFO~]` directly to the `[*~]` object.

★ **Step 18:** Experiment with different settings for **speed** and **depth**, and note how the timbre of the resulting sound is subtly different from ‘tremolo’.

Note: *In the BBC TV series Dr. Who, the voices of the alien Daleks¹³ are generated by a ring modulator with an LFO set to around 30 Hz. The voice actors also spoke using a stilted monotonic intonation in order to enhance the effect. You can try this yourself by adding a ‘live speech input’ option to your code (i.e. using [adc~] and some means to toggle between the prerecorded and live input)¹⁴.*

QUESTION 6 (worth up to 5 marks)

In your own words¹⁵, why is this effect known as ‘ring modulation’?

2.7 Frequency Shifting

Many Vocal FX are the result of altering the frequencies present, e.g. changing the pitch of a voice. There are many algorithms for frequency shifting, but you have already implemented an approximate solution with your ring modulator.

★ **Step 19:** Use the **number** object in your [LFO~] GUI to wind up the **speed** of the LFO in your ring modulator to around 300 Hz. If all is working correctly, as you raise the modulation frequency, you will hear that the frequencies in the speech seem to be both increasing *and* decreasing. Indeed, if you listen carefully at a fixed **speed** of around 300 Hz, you may be able to detect two voices speaking simultaneously at different frequencies.

The reason for this result is that multiplying one signal by another - known as ‘heterodyning’ - produces an output which consists of the sums and differences of the frequencies of the two input signals. This means that the frequencies in the original speech are both raised and lowered by an amount corresponding to the **speed** of the LFO - hence the strange mixture of higher pitched and lower pitched voices in the output.

The sum and difference components are known as the upper and lower ‘sidebands’. So, to avoid the distortion caused by having both sidebands present in the output, we need to remove one of them, i.e. we only want one sideband. This can be achieved by creating two versions of the original signal that differ in phase by 90°, then modulating the two versions with two heterodynes that also differ in phase by 90°. These phase differences are such that, when the two out-of-phase heterodyned signals are subtracted, the lower sideband is cancelled out. This process is known as ‘single-sideband modulation’ or ‘SSB’¹⁶.

QUESTION 7 (worth up to 5 marks)

Why is SSB commonly used in long-distance radio voice communications?

Splitting a signal into two with a 90° phase difference between them can be achieved using the ‘Hilbert Transform’. Hence the following patch shows the Pd object [hilbert~] being used to create a frequency shifter based on SSB modulation ...

¹³<https://en.wikipedia.org/wiki/Dalek>

¹⁴You will need this feature later, in any case.

¹⁵I.e. do not plagiarise from Wikipedia.

¹⁶SSB is commonly used in long-distance radio voice communications, especially by amateur radio enthusiasts.

2.8 Frequency Modulation: *Vibrato*

Now that you have the ability to shift the frequencies in a speech signal, it is very easy to implement another common voice manipulation technique - ‘vibrato’. All that is required is for the frequency shifter to be controlled by the output of an LFO.

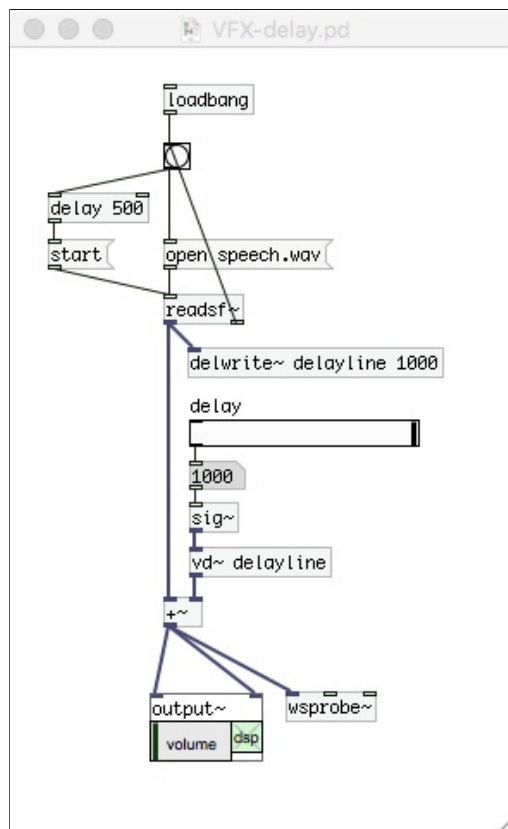
★ **Step 23:** Implement ‘vibrato’ by connecting an LFO to your frequency shifter, and experiment with different values for **speed** and **depth**. Note that the LFO output will need to be scaled to provide an appropriate frequency shift range and then added to the output of the frequency shift slider.

Note: With *speech.wav* as input, it is possible to simulate the voice of Gollum from the *Lord of the Rings* by setting *LFO = sine wave*, *speed = 50 Hz* and *depth = 350*.

2.9 Time Delay Effects: *Echo, Comb Filtering and Flanger*

Many interesting voice FX can be achieved by delaying the signal and recombining it with itself. [Pd] provides the possibility of writing to and reading from audio ‘delay lines’ using the objects [delwrite~] and [delread~]. However, for full flexibility in your [VoiceChanger] application, you need to be able to vary the time at which audio data is read out of the delay line. So, rather than using [delread~] it is better to use [vd~].

★ **Step 24:** Implement the following patch¹⁷ with the ‘delay’ slider’s properties set to operate *on a log scale* from 1 to 1000 msecs ...



¹⁷Of course, you may choose to replace the [+~] with the mixer you implemented earlier.

★ **Step 25:** Experiment with various values for the delay, and note the different effects you can achieve with delays (a) below 20 msecs, (b) between 20 and 100 msecs, and (c) above 100 msecs.

You should observe that, with delays below 20 msec, the signals combine to create a subtle ‘phasing’ effect. This is known as ‘comb filtering’ as the signal is effectively interfering with itself, and frequency components corresponding to multiples of the delay time are enhanced or cancelled out (due to ‘superposition’). Delays between 20 and 100 msecs give the effect of the voice being in a reverberant room. Delays above 100 msecs sound like distant echoes.

Finally, in this section, it is possible to use an LFO to vary the delay. The resulting effect is known as a ‘flanger’.

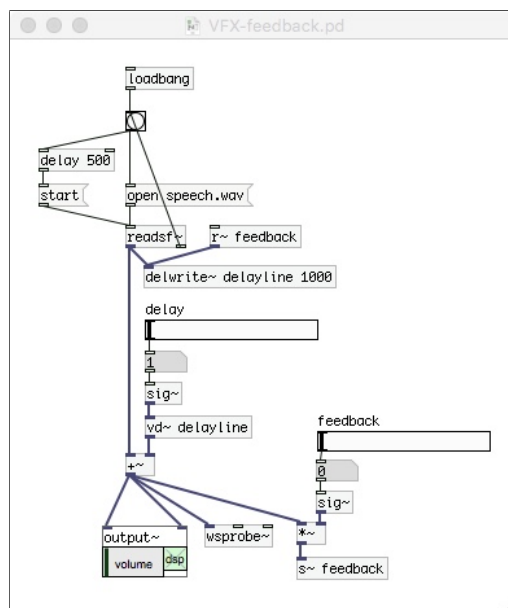
★ **Step 26:** Add an LFO to your ‘delay’ patch to create a ‘flanger’, and experiment with different settings. Note that you will need to scale the output of the LFO, and you will get different effects depending on whether the delayed signal is mixed with the original or not.

Note: Using ‘flanger’ it is possible to achieve an underwater effect by setting *LFO = sine wave*, *speed = 10 Hz*, *depth = 0.4*, *delay = 5 msecs* and *mix = 100% processed*.

2.10 Feedback: Reverberation

The previous section noted that an echo occurring between 20 and 100 msecs after the original sounds like reverberation. However, in a real environment there are many echoes caused by multiple reflections from the different surfaces present, and a signal might bounce around a very reverberant environment many times. A simple method for achieving such an effect is to feed a proportion of the output signal back to the input. This is known as ‘feedback’.

★ **Step 27:** Modify your delay patch as follows, and experiment with different values for delay and feedback ...



QUESTION 9 (worth up to 5 marks)

In a practical system, why is it important to keep the feedback gain less than 1?

Note: *A short delay with a lot of feedback can give a metallic sound to a voice, e.g. C3PO from Star Wars. Try $\text{delay} = 10 \text{ msecs}$, $\text{feedback} = 0.9$ and $\text{mix} = 90\%$ processed.*

Clearly feedback could be applied to any/all of the effects that you have implemented so far, thereby increasing the range of possible voice effects considerably - as you will discover in Part-II.

3 Part-II: Real-Time Voice Changer

3.1 General Requirements

The final part of this assignment is to compile all of the different components you developed in Part-I into a single [VoiceChanger] application. The application should allow ‘live’ speech input in addition to the ability to select a particular prerecorded file (i.e. no longer hard-wired for `speech.wav`). The GUI should be well thought out, easy to use and attractive, and it should not only allow the different effects to be controlled, but also allow them to be connected to each other in a logical sequence. You should also provide innovations, for example preset effects (using `Vradio` buttons to select particular combinations of settings).

3.2 COM4502-6502 Only

It is possible to convert a single voice into multiple voices by combining the outputs of parallel manipulations, especially different frequency shifts. This effect is known as ‘chorus’. Your objective is to add ‘chorus’ to your [VoiceChanger] application by careful use of abstractions and a method for selecting one or more output voices.

3.3 Final Steps

★ **Step 28:** Implement [VoiceChanger] according to the requirements stated above.

QUESTION 10 (worth up to 50 marks¹⁸)

Please provide a short¹⁹ description of the operation of your [VoiceChanger] application, together with a screenshot of your final GUI.

Congratulations, you have almost completed this programming assignment. All that is left now is to package up your code and submit it for marking.

★ **Step 29:** Create a .zip file of the form ‘YourName.zip’ containing ...

- your response sheet in .pdf format with your name on the front cover and containing your responses to all of the questions; and
- your final [VoiceChanger] code, together with any necessary abstractions (such as [LF0~], [LF0~-help] etc.).

Important: For marking, we expect your code to work ‘out of the box’.

Note: Do not include your .tex file (and associated image files) or your work-in-progress code examples.

★ **Step 30:** Hand-in your .zip file in accordance with the instructions at the beginning of this document.

That’s it - you’re done!

¹⁸25 for functionality, 15 for design/layout, 5 for Pd features, 5 for innovations

¹⁹no more than 500 words