Assignment 2: Vibrato

Documentation of the Class Design:

class name: Vibrato
private members:

*CRingBuffer<float> **ppfRingBuff*- This is a double pointer object of the Ring Buffer class, that consists of float values.

LFO ppcLFO- This is an object of the LFO class, that is used to create the LFO within this vibrato.

The above two objects are integral members of this class and are hence kept private in access.

Enumeration:

```
enum VibratoParam {
    kModFreq,
    kWidth,
    kDelay,
    kNumParams
};
```

This enumeration is used in Setting the vibrato parameters like Modulation Frequency, Modulation width and Delay length.

Other Private Variables:

```
float fVibParam[kNumParams];//ModFreq,Width,
float fVibParamRange[kNumParams][3];
float SamplingRate;
int iNumChannels;   //Width in seconds,ModFreq in Hz
int iFramesProcessed;
```

The first is a float array of the all the parameters,in the order Mod.Freq, Width and Delay.

The second is a float two dimensional array that consists of the ranges of each of the parameters , which are stored within the class itself.

The Sampling frequency,Number of channels and the number of frames processed are commonly used variables which have been made a part of the class itself for easier function

public members:

*void process(float **ppfInBuff,float **ppfOutbuff, int iNumOfFrames):*

This is the process function which accepts two float double pointer buffers (one with the input and the other for the output), and does a frame by frame processing of the vibrato effect giving the output in the process class.

void setVibratoParam(VibratoParam eVibParam, float fParamVal);

float getVibratoParam(VibratoParam eVibParam) const;

The setter and getter function for the parameters of the Vibrato class. The setVibratoParam can be used to change individual parameters based on the enumerations:

kModFreq- Modulation Frequency

kWidth- Modulation Width

kDelay- Modulation Delay

Upon entering one of these three,we can enter the float value to which we wish the particular parameter to be changed.

The getVibratoParam shall get the value of a particular parameter during the process.The enumeration used is the same.

Vibrato(float fVParam[3], int UserNumChannels,float UserSamplingRate, int iMaxDelayInSec); and virtual ~Vibrato():

The constructor and destructor are used for the instantiation and destruction of the Vibrato class. The float array fVParam is an array consisting of the parameter values. Hence, we can instantiate the class with values from within the constructor itself, which reduces the number of external operations.

Design related Questions and thier Answers:

What parameters can be adjusted during processing?
The Modulation Frequency, Modulation Width and the Delay can be adjusted during processing using the set function.

When (and by whom) is the maximum allowed delay set?
The prescribed Parameter ranges are now maintained within the Vibrato class and are used to check if the parameters are within the range and not. If not, default values for each parameter are set.

What parameters can be known during instantiation?
Modulation Frequency, Modulation Width , Delay, Sampling Frequency, and Maximum Delay Length(Equal to block length) are the parameters that shall be known during instantiation.

How/Where do you set them?
They are set in the constructor function of the Vibrato class itself, and can also be changed in process, using the set function.

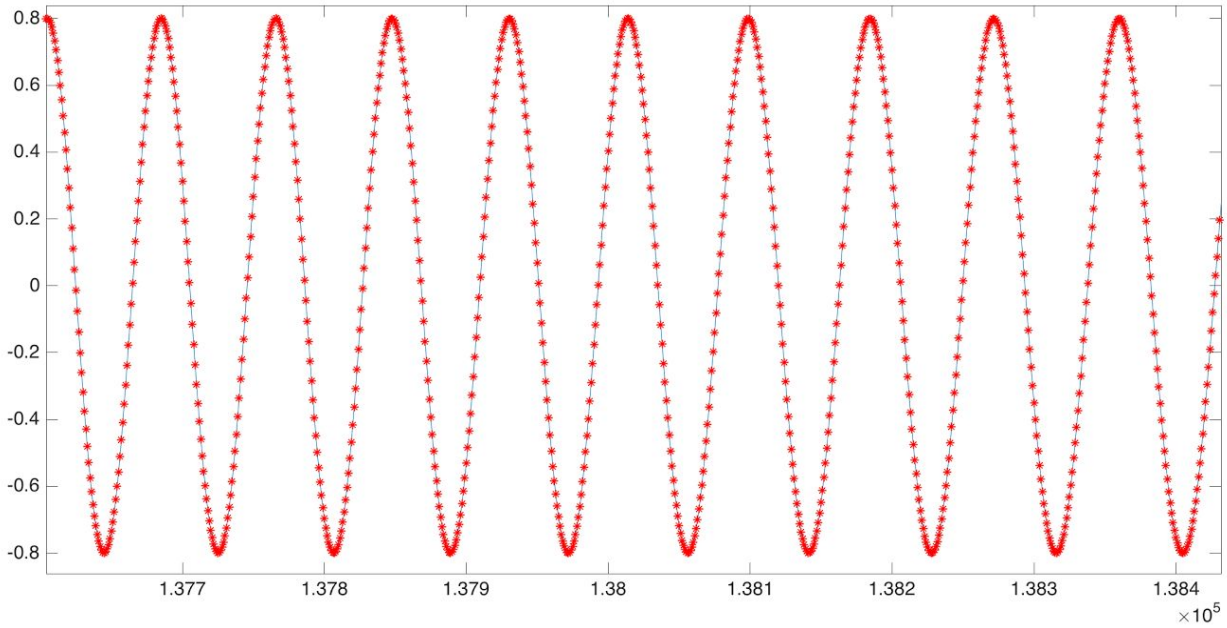
Will your interface work for all samplerates and all channel configurations?
Yes.

Is your process function in place or not?
Yes, we use a process function to process the Vibrato effect.

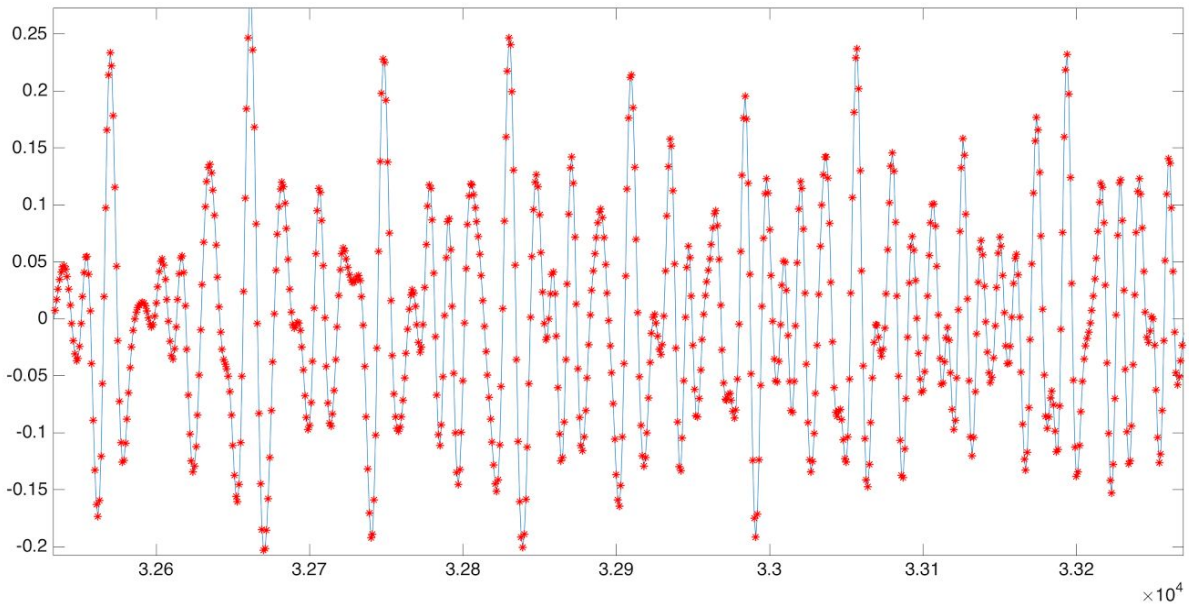
Document your thoughts on the design, solutions, and answers to these questions.
We have tried to keep the design of the class simple and less cumbersome, based on the feedbacks from the last assignment. We use the constructor to instantiate the class, and accept the parameters as an array from the user using an enumeration. A distinct advantage of this shall be one-by-one changability of the parameters using the setParamVal() function.

MATLAB vs C++ implementation

The red stars show the output from the C++ implementation, whereas the Blue plots show the MATLAB plots for the implementation.



Plot-1: Plot for sinwav441Hz.wav



Plot-2: Plot for Piano-Shepard.wav