CREATING VIBRATO USING MATLAB

Adrian Clarke 306164981(SID)

Digital Audio Systems, DESC9115, Semester 1 2012
Graduate Program in Audio and Acoustics
Faculty of Architecture, Design and Planning, The University of Sydney

The purpose of this report is to review and explain a function used to create a digital audio effect within the program Matlab. The effect I will be investigating is vibrato and I have sourced a Matlab function from *DAFX – Digital Audio Effects* by Udo Zölzer. Vibrato can be defined as a periodic variation in time of pitch, above and below the fundamental frequency of a sound. The rate at which the pitch is varied and the extent by which it is varied can be considered the two fundamental parameters affecting the character sound of a vibrato effect.

The audio file used in my project is a one second long 800Hz sine tone created in Matlab (800Hz.wav). This audio file can be seen visualised in Figure 1.1 and Figure 1.2. The output of the function under examination is the 800Hz sine tone with a vibrato applied (800Hz_with_vibrato.wav). The parameters of the vibrato are a 5ms average delay modulated by a 10Hz sine tone, the effect of this vibrato can be seen in Figure 2.1 as the frequency range is spread either side of the 800Hz fundamental and Figure 2.2 illustrates the modulation over time of the frequency. The signal flow of a vibrato effect can be seen in Figure 3. As is shown vibrato is basically an FIR Comb Filter without the delayed signal being mixed back with the input. The output is purely the input *x(n)* being delayed by *M* samples to create the output *y(n)* as shown in Equation 1. The value *M* of the delay is being constantly varied as determined by the frequency of the modulating LFO (low frequency oscillator).

The vibrato function I am examining (`vibrato.m`) takes an input wave and creates an output wave with a vibrato effect applied. The character of the vibrato effect is determined by four input arguments: `inwave`, `fs`, `Modfreq` and `Width`. The `inwave` argument is our input wave that has been imported by Matlab's built-in `wavread` function. The argument `fs` is the sampling frequency of the input wave, which in the case of the audio files provided with my report is 44100. `Modfreq` is the frequency in Hz of the LFO used to modulate the delay time, this can be considered the rate at which the frequency is being varied and is typically a value of approximately 5-14Hz. The input argument `Width` is the average delay length in seconds that is to be modulated by the LFO, typical values for this are 5-10ms (0.005-0.01 seconds). If the values of the `Modfreq` and `Width` arguments exceed the typical ranges mentioned the vibrato effect is lost.

When the vibrato function is called in Matlab, these four input arguments must be specified. The rest of the code of the function performs operations on these input

arguments to ultimately create an output wave with the desired vibrato effect, according to the input arguments specified. The earlier part of the function is fairly self-explanatory, and comments are provided in green with a basic description of the operations being performed. For example the average delay in number of samples (`DELAY`) is created by multiplying the two input arguments `Width` and `fs`. The modulation width in number of samples (`WIDTH`) is created by the same method. Built-in Matlab functions are used on our input arguments to create new variables relevant to the purpose of our function. For example the `length` function is used on our `inwave` to determine the number of samples it contains (`LEN`). Variable `L` is created from the `DELAY` and `WIDTH` variables to give the length of the delay, in-built function `zeros` is then applied to `L` to create a vector of length `L` containing only zeros (`Delayline`). This vector is the memory allocation for the delay and is used later in the function to input the sample values of the delayline. Memory allocation for the output wave (`outwave`) is created by applying the `zeros` function to the `inwave`.

The next part of the function uses what is known as a 'for loop'. This is a section of code to be repeated a specific number of times. In this instance the code is repeated to calculate the delay length at every sampling point (`n`) as the delay length is being constantly modulated by our LFO. The variable `n` is a vector of length `LEN` going from 1 to `LEN`-1 in increments of one. The next two lines of the function use our `Modfreq` input argument to calculate the value of our LFO (`MOD`) at each point (`n`) by using the built-in `sin` function. `MOD` is used in the next line to calculate the length of the delay (`TAP`) at each point (`n`) as it is modulated by the LFO sine wave (`MOD`). As the delay length is being continuously modulated by the LFO we get values of delay that are non-integer values of the sampling interval. Using the built-in `floor` function we can get the closest integer value of the sampling interval (`i`) below the factional delay value (`TAP`). By subtracting `i` from `TAP` we get the fractional value (`frac`). The last line of the function uses interpolation to calculate the integer sample value of best fit from `frac` and `TAP`. There are multiple different ways of interpolating the output signal however for the purpose of this assignment I have only included one method. Once the function has finished running it will create `outwave`, shown in Figure 2.1 and 2.2 which is the input wave with vibrato applied.

Bibliography

Vibrato.m - Udo Zölzer, ed. *DAFX: Digital Audio Effects*. John Wiley & Sons, 2002, Chapter 3, pp. 68-69. ISBN: 0-471-49078-4.

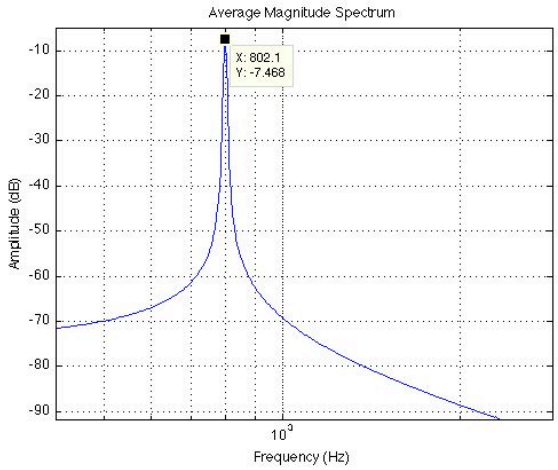

Figure 1.1 ***Average magnitude spectrum of '800Hz.wav' showing fundamental frequency of 800Hz. Window length of 8192 has been used to provide accurate frequency resolution.***

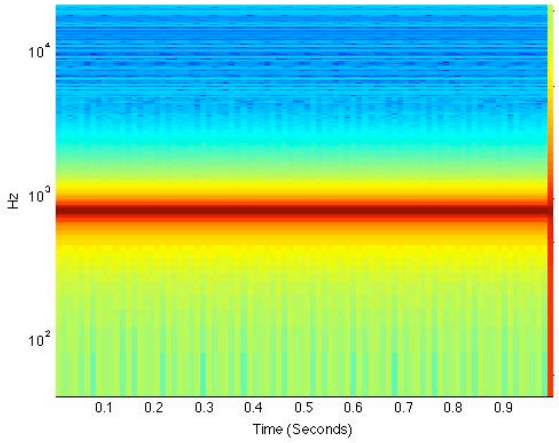

Figure 1.2 ***Spectrogram of '800Hz.wav' showing fundamental frequency of 800Hz. Window length of 1024 has been used to provide accurate time resolution.***

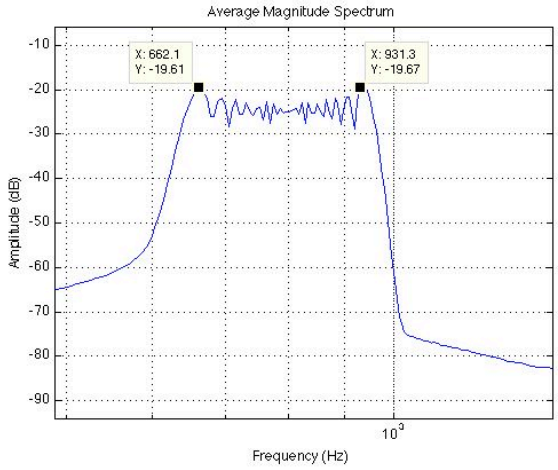

Figure 2.1 ***Average magnitude spectrum of '800Hz_with_vibrato.wav' illustrating vibrato effect. Window length of 8192 has been used to provide accurate frequency resolution.***

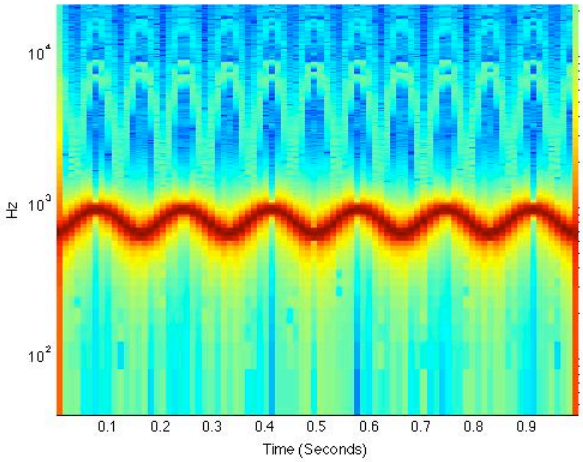

Figure 2.2 ***Spectrogram of '800Hz_with_vibrato.wav' showing fundamental frequency of 800Hz. Window length of 1024 has been used to provide accurate time resolution.***

$$y(n) = x(n - M)$$

Equation 1. ***Equation for vibrato effect***

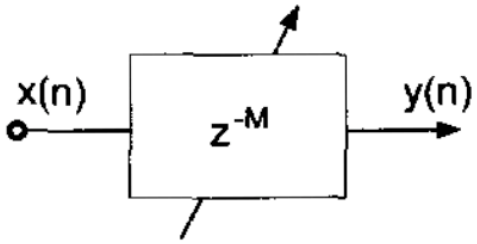

Figure 3. ***Signal flow of vibrato (Udo Zolzer, "DAFX – Digital Audio Effects")***