Physically-Informed Tape Echo Model using the Synthesis Tool Kit (December 2018)

Mark E.R. Bennett, McGill University

***Abstract*—A Tape Echo Model developed using the Synthesis Tool Kit ("STK") is presented. Building upon a variety of methods proposed in Virtual Analog ("VA") literature, the newly developed model simulates the response of a "speed-style" tape delay, and is capable of processing audio in real-time. Most notably, the model emulates the pitch modulation behaviour associated with a speed-style tape delay during periods of increasing and decreasing tape speed. The model also incorporates the saturation and bandwidth characteristics of a magnetic tape reel, a pseudo-randomized modulation algorithm to simulate the effects of an aging tape reel, and incorporates a preamp. The model was tested using an electric guitar and amplifier, and the resultant sound quality and response is comparable to that of commercially available tape echo models. Finally, some recommendations for future development are given.**

***Index Terms*— Physical Modelling, Tape Echo, Delay line, STK**

I. Introduction

Delay and echo effects have been prominently featured in all genres of music and music production since their invention. Original introduced in the 1950s, the earliest commercially available examples used a magnetic tape reel as their delay line. While these vintage units are cumbersome to use, expensive, and difficult to maintain, their nuanced sound is revered by musicians, and many prefer them to modern digital delays. In line with the ongoing digitization of all equipment used in music production [1], many researchers have developed physically-informed, digital models of famous tape echo machines [2][3]. That is, models have been developed based on the physical hardware found in a vintage tape machine, as opposed to simply modelling its performance. Perhaps the most recent contribution to tape echo modelling was presented by Zavalishin and Parker at DAFx 2018 [4]. They describe a novel method for efficiently emulating the characteristic pitch modulation behaviour exhibited by a "speed-style" tape echo using a standard variable-length ring-buffer.

This paper presents an implementation of their method and several other physically-informed characteristics of a tape echo using the software tools provided by the Synthesis Tool Kit ("STK"). It simulates the pitch modulation behaviour of a speed-style tape echo, and emulates a pre-amp, tape bandwidth and saturation, and aging tape modulation. The model is capable of processing audio in real-time, and has several user-controllable parameters, provided by a simple graphical user interface ("GUI"). It is of note that the tape echo model described in this paper is not based upon any particular tape machine, but rather incorporates aspects of various classic tape machines. Namely, it incorporates features from both the Maestro Echoplex, and the Roland Space Echo.

The block diagram of the model is shown in figure 1. In essence, it comprises of a pre-amplifier, a delay line, a feedback path, and a mixing amplifier. There are several additional blocks in the delay line path that are used to model the physical characteristics of tape, as described.

The remainder of this paper will read as follows: The software development platforms that were used are discussed in section II. The algorithm proposed by Zavalishin and Parker and its implementation is described in detail in section III. In sections IV, V, and VI, the pre-amp, tape bandwidth and saturation, and aging tape modulation implementation are discussed, respectively. In section VII some additional control parameters are discussed. The results and some recommendations for future work are given in sections VIII and IX, and finally in section X we conclude.

II. Software Development

To develop this model, only regular C++ 11, and the classes provided by STK were used. STK is a set of open source audio signal processing and algorithmic synthesis classes written in C++ [5]. Originally developed by researchers at the Center for Computer Research in Music and Acoustics (CCRMA), It provides developers and researchers the means to quickly create audio processing software, with support for real-time audio and MIDI. Since it is simply a collection of classes, it is incredibly portable, and requires no special software or programs to run and use. Given its generality, it served as an ideal platform on which to build the tape echo model. In fact, the model was developed using only default classes that come with STK – with the exception of one very minor adjustment. A full list of the STK-provided classes, tutorials, and documentation can be found in [5].

Aside from the C++ and STK software used to develop the functionality of model, a simple graphical user interface ("GUI") was developed using Tcl [6]. Combined with the real-time MIDI support provided by STK, this interface allows the user to control various parameters of the model in real time.

III. Tape-like Pitch Modulation

In general, the delay line of a tape machine is comprised of four main components. The tape reel itself, the "read" head, the "write" head, and the "erase" head. The incident signal is written onto the tape at the write heads' position, travels along the spinning tape, and is read off at the read head, creating the delay. The erase head sits directly behind the write head, so that the write head always writes to a clean tape. The delay time between write/read is determined by two variables:

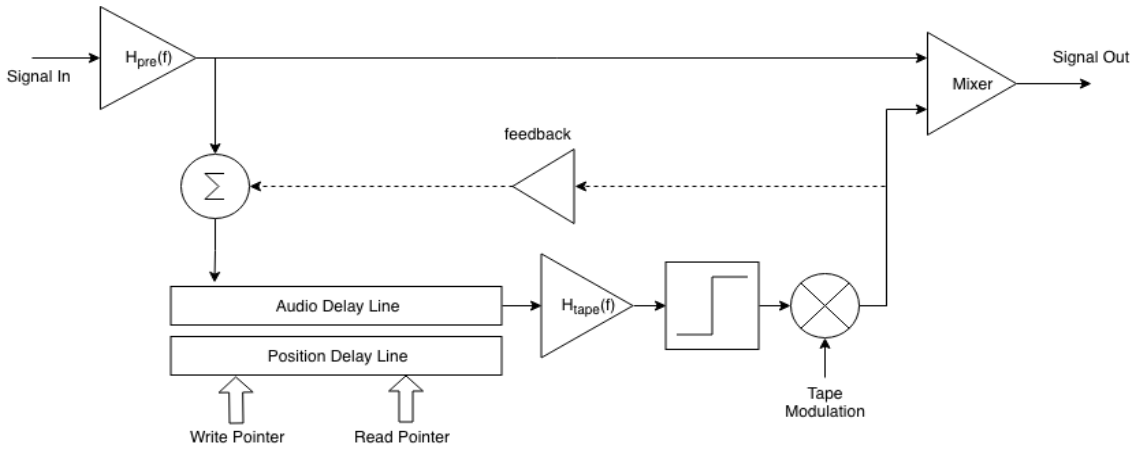

Figure 1: Block diagram of the proposed tape model. Two delay lines are used to accommodate speed style operation. The delay line also features a tape bandwidth filter, a nonlinear, memoryless saturation section, and a tape modulation block.

The speed at which the tape spins, and the distance between the two heads.

The distinction can then be made between two different styles of tape machines: "length-style" delays, such as the Maestro Echoplex, and "speed-style" delays, such as the Roland Space Echo. In a length-style delay, the tape moves at a constant speed, and the read head position is adjusted to change the delay time. In a speed-style delay, the read/write head positions are fixed, and the tapes' motor speed is adjusted to change the delay time. The pitch modulation behaviour of a speed-style delay is typically preferred as the signal pitch can be modulated up and down very smoothly, allowing it to be easily controlled by the user during feedback. In a length style delay, the pitch modulation is much more sporadic as delay time changes, making it difficult to predict and control. The goal, then, as outlined in [4], is to use a standard variable-length digital ring buffer that simulates the response of a speed-style delay.

A. The Tape Equation

To accomplish this, Zavalishin and Parker developed the *Tape Equation*, which defines a relationship between changes in delay time (and therefore buffer length), and changes in tape speed. The derivation of the tape equation is well explained in [4], so a shorthand version is presented here. To examine the tape equation, a few definitions must be established first. Let x_w and x_r represent the write and read head positions at time *t,* and let n_w and n_r be the write and read pointer, respectively. If *L* is taken as the fixed tape length, it then follows that the write and read positions are related by:

$$x_w(t) - x_r(t) \equiv L > 0 \tag{1}$$

And since the speed at time *t* is constant at all points along the tape:

$$\dot{x}_w(t) = \dot{x}_r(t) = v(t) \tag{2}$$

Then, the *effective delay time* between when a signal is written and read from the tape is denoted as *T(t).* It follows that:

$$x_r(t) = x_w(t - T(t)) \tag{3}$$

That is, from *t-T(t)* to *t,* the signal must travel a distance of *L.* So, if the *total distance travelled by the tape* is defined as *V(t):*

$$V(t) = \int v(\tau)\, d\tau \tag{4}$$

where *V(t)* is an arbitrary antiderivative of the tape speed, evaluating this integral from *t-T(t)* to *t* must then equal the tape length *L:*

$$V(t) = \int_{t-T(t)}^{t} v(\tau)\, d\tau = L \tag{5}$$

This is the integral form of the tape equation, which establishes a relationship between the tape length, tape speed, and the effective delay time. The tape equation can be expressed in several forms, including the difference form and the differential form, given in (6) and (7), respectively:

$$V(t) - V\big(t - T(t)\big) = L \tag{6}$$

$$\frac{d}{dt}T(t) = 1 - \frac{v(t)}{v(t - T(t)} \tag{7}$$

A clear relationship between changes in tape speed and changes in the effective delay time is established in (7). There are three scenarios relating speed changes to changes in delay time (and therefore the pitch modulation behaviour) that are observed:

1) If there is no change in speed, $dT(t)/dt = 0$, and no pitch modulation occurs

2) If the speed decreases, $dT(t)/dt > 0$, and the delay time increases. This causes a downward modulation of the signals' pitch.

3) If the speed increases, $dT(t)/dt < 0$, and the delay time decreases. This causes an upward modulation of the signals' pitch.

The effective delay time can then be used to re-size the audio buffer as appropriate – which effectively creates speed-style delay behaviour with a standard variable-length ring buffer.

B. Implementation and Initialization

To implement this algorithm, the write/read head position is stored in a separate channel of the audio ring buffer. By tracking the write/read position, the write/read pointers can be incremented accordingly to account for changes in speed. For each processing loop, the write pointer will be constantly incremented by one sample. Then, based on changes in the tape speed, the read pointer will be adjusted to either increase, decrease, or maintain the buffers' length. To help see how the algorithm operates, it is convenient to visualize the position buffer in a circular fashion as shown in figure 2. The buffer is stationary, and the circumference is twice the tape length, *2L*. The read/write pointers then spin anti-clockwise around the buffer at the tape speed, *v(t)*. The write head travels from [0, L], before getting wrapped around to zero again. The read head then travels from [-L, 0], relative to the write head.

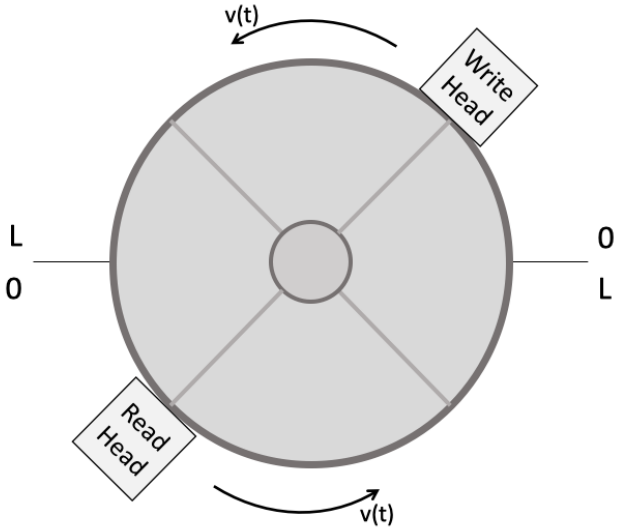

Figure 2: Circular representation of the position ring buffer

The algorithm is then implemented as follows. The continuous time variable *t* is now replaced with the discrete time variable, *n*, and the sample rate is *fs*:

0. At the start of each loop, it is assumed that:
   - n_w is pointing to the buffer element about to be written to.
   - $n_r + \delta_r$ points to the buffer element read out in the previous loop, where δ_r is the fractional component of the read pointer.

1. The desired delay time is read from GUI, and it is used to calculate the current tape speed in meters per sample:

$$v[n] = L/(dtime * f_s) \tag{8}$$

2. Advance the write head position based on the current tape speed, and save it to the position buffer at n_w:

$$x_w[n] = V[n_w - 1] + v[n] = x_w[n-1] + \dot{x}_w[n] \tag{9}$$

$$V[n_w] = x_w[n] \tag{10}$$

This step strongly resembles the first-order *forward Euler method* ("FE") commonly used for transient analysis in circuit simulation theory, where the step size is a single sample, and the change in write head position is assumed to be constant for a single sample.

3. Calculate the read head position based on the updated write head position:

$$x_r[n] = x_w[n] - L \tag{11}$$

4. Index the read pointer to find the element in the position buffer that corresponds to the new read position (more on this search in the next subsection):

$$V[n_r + \delta_r] = x_r \tag{12}$$

5. Read the delay output from the audio buffer at $n_r + \delta_r$

6. Send the output through the feedback loop to the delays' input

7. Write the new delay input to the audio buffer at n_w

8. Increment the write pointer, wrap the pointer and the write head position if necessary.

Essentially, changes in tape speed (how fast the write/read head spin around the circular buffer) are translated into how much the read pointer increments on each pass through the loop. The

three scenarios for changes in tape speed can then be re-examined to observe how they affect indexing of the read pointer:

1) If there is no change in speed, the read pointer increments by one sample per loop. The buffer length and delay time remain the same. No pitch modulation occurs.

2) If the speed decreases, the read pointer increments by less than one sample per loop. The buffer length increases, and the delay time increases. A downward pitch modulation is heard.

3) If the speed increases, the read pointer increments by more than one sample per loop. The buffer length decreases, and the delay time decreases. An upward pitch modulation is heard.

C. Updating the read position

The read pointer must be updated each loop using an incremental search. Since the tape speed is always positive, it is only necessary to search forward from the previous read pointer position. More precisely, the integer part of the read pointer is found with an incremental search, then the fractional part can be calculated with linear interpolation. This was accomplished in software with a simple while loop, after the new read position was calculated:

```
while( V[nr + δr] < xr ) { nr++; }
nrp1 = 1;
nr -=1;
```

Upon exiting the while loop, the read pointer will be pointing to the first value that is larger than the updated read position. So, this value is saved as $n_r + 1$, the read pointer is decremented, and then the fractional portion of the read pointer is calculated:

$$\delta_r = \frac{x_r[n] - V[n_r]}{V[n_r + 1] - V[n_r]} \tag{13}$$

To improve the CPU cost of this search, it is noted in [4] that a logarithmic search can be used instead of the linear search presented here during quick increases in tape speed. However, this logarithmic search was not implemented.

It is also possible for aliasing to occur during quick increases in tape speed. A method for anti-aliasing is also described in [4], but was not implemented. The pre-amp and tape bandwidth filtering that will be described in subsequent sections band limit the signal enough that aliasing is effectively prevented for the 44.1kHz sampling rate used.

It is also worth noting that in [4] fixed-point representation was used for all tape-related variables, such as write/read position, tape speed, and total distance travelled by the tape. Since the position buffer is essentially a running sum, it is possible that rounding errors introduced by floating-point representation will accumulate over time, and cannot be efficiently corrected. Using fixed-point representation alleviates these errors, as any values added to the buffer will be exactly subtracted back out when the write position is wrapped. However, Fixed point representation was not used in this model, and all variables use a floating-point representation.

D. Initialization

At the beginning of the algorithm, the variables are initialized as follows:

$$V[0] = -L$$
$$V[1] = 0$$
$$n_r = 0$$
$$n_w = 2$$
$$x_w = 0$$

With this initialization, the read pointer will remain between 0 and 1, and the read head will remain stationary until the write head reaches distance L (the opposite side of the circular buffer!). Then, both pointers will begin incrementing together until the speed value is disrupted.

IV. Preamp Modelling

The preamp found in the Maestro Echoplex EP-3 is notorious and revered by guitar players in and of itself. Thought to add warm to the overall sound of the guitar, there are several commercially available boost pedals based solely on the preamp found in the original Echoplex units. The Schematic of the original EP-3 preamp is shown in figure 3. It is a fixed gain, JFET-based amplifier with some passive filtering. Besides filtering and amplification, it also acts as a buffer which helps to maintain signal integrity through long cable runs.

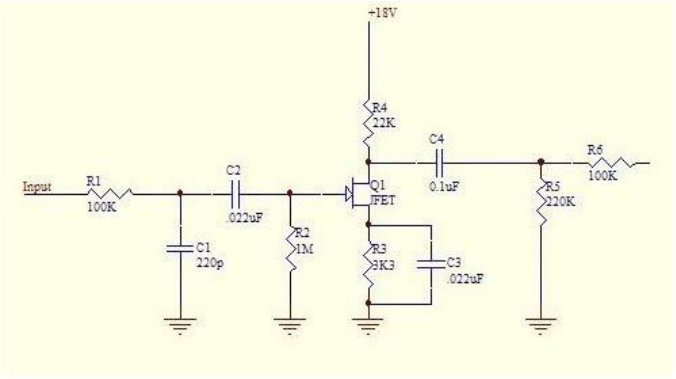

Figure 3: Schematic of the Echoplex EP-3 preamp

To model this preamp, a frequency domain analysis was performed using LTSPICE. The simulation results are shown in figure 4. The curve was then analyzed, and it was matched using a combination of two filters. To model the high-pass filtering, a second order Butterworth filter with a cut-off frequency of 10Hz was used. This HPF was then cascaded with a second order Chebyshev low-pass filter with a cut-off frequency of 10kHz and a filter Q of 1.5924. Finally, the output of both filters was amplified by a constant, scalar gain factor of 12dB. These filters were simulated using MATLAB, allowing their Z-domain coefficients to be calculated. These filters were then implemented as "Biquad" objects in STK

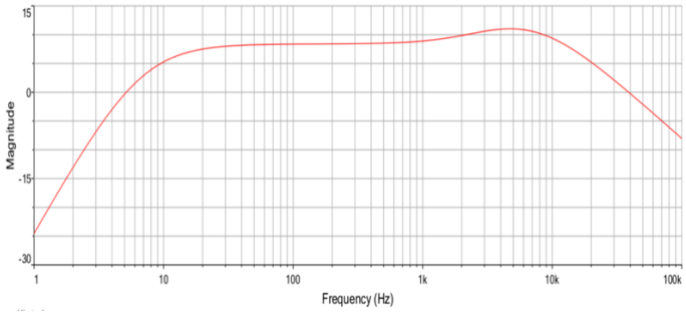

Figure 4: Frequency response of the Echoplex EP-3 preamp

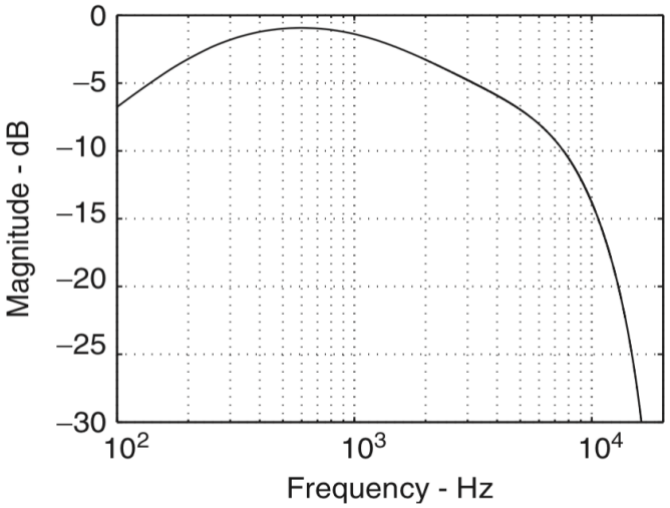

Figure 5: Frequency response of a tape reel. Source: [2]

V. Tape Saturation and Bandwidth

To more accurately model the response of a magnetic tape reel, a tape bandwidth filter and memoryless saturation block were added to the delay line path of the model.

The tape bandwidth filter was implemented based on measurements by Arnardottier et al. in [2]. The frequency response of a magnetic tape reel based on their measurements is shown in figure 5. Following the advice of Downing and Terjesen in [3], this frequency response was modelled with a second order Chebyshev bandpass filter. The filter was simulated in MATLAB, and its Z-domain coefficients calculated. It was then implemented in STK as a "Biquad" object. This filter gives the model the characteristic low-end roll off associated with tape echoes. Each repeat becomes increasingly "brighter" as the tape filter continually attenuates the low frequencies in the signal.

Tape saturation creates a non-linear, distorted input versus output amplitude characteristic due to a combination of attributes such as tape magnetization, tape head construction, tape material, width, speed, equalization, biasing, and the corresponding amplitudes and dynamics of the source signal [3]. To emulate this saturation, the delayed signal is processed by the Gaussian Error Function ("erf()") as recommended in [3]. For C++ 11 and later, erf() can be found in the standard <cmath> header included in C++. A graph depicting the gaussian error function is shown in figure 6. The "tape saturation" control slider found in the GUI determines how much the signal is amplified before passing through the error function.

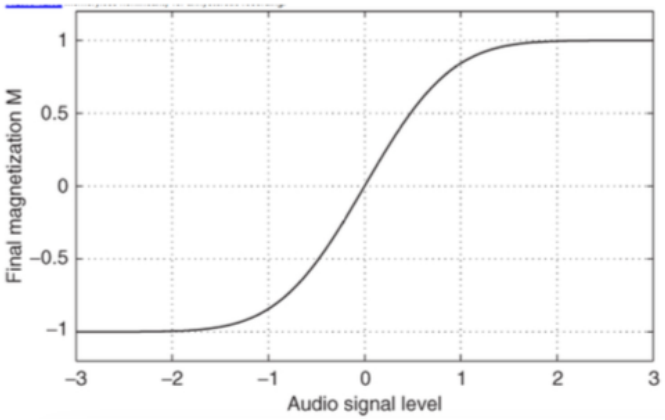

Figure 6: Final Magnetization M vs Audio Signal level for a tape reel, this very strongly agrees with the Gaussian Error Function, used in statistical analysis. Source: [3]

VI. Tape Age Modulation

To simulate some of the imperfections of a tape reel, a slightly randomized modulation was incorporated into the delay line of the model.

A tape reels' surface degrades over time as it is subject to many write, read, and erase cycles, causing random warping and modulation of the signal to occur. In many commercially-available tape echo models, this aging process is simulated by adding a semi-random sinusoidal modulation to the delay line.

This modulation was incorporated into the model using a short delay line, and a slightly modified "Modulate" STK object. Originally designed to simulate the vibrato of a human voice, the Modulate class produces a sinusoidal wave with randomized noise superimposed onto it. The user can control the vibrato frequency and depth, as well as the randomization gain. The class was modified for the tape echo model by allowing the user to also control the frequency of randomized noise. This allowed the modulation to be tuned to match that of an aging tape, as opposed to the vibrato of a human voice. The "tape age" slider found in the GUI controls the modulation depth. Since this feature adds to the overall length of the delay line path, the length of the modulation delay line is compensated for when calculating the tape speed, to ensure that it is accurate.

VII. Additional Controls and Features

A. Feedback Control

As noted in [1], the feedback coefficient of both the Maestro Echoplex and the Roland Space Echo ranges from 0 to 2. And, for values greater than or equal to 1, the tape echo will self-oscillate. To simulate this feature, the feedback coefficient in the model is also allowed to vary linearly from 0 to 2.

B. Volume and Mix Controls

For usability, an output volume and wet/dry mix control were

added to the model. The volume control simply scales the output signal, and the mix control changes the volume balance between the input and delayed signals. A mix value of zero mutes the delayed signal, while a mix value of one mutes the input signal. Values in between give a blend of the two signals.

C. Delay Time Control

As noted in [4], one of the biggest advantages of a digital delay line over its vintage, analog counterpart is the range of possible delay times. Most tape echoes have a delay time range of 60 to 600ms [1]. Theoretically, however, the only limit on the delay time in a digital delay is the available memory of the system. And, with some slight alterations to the algorithm, negative tape speeds can even be incorporated, allowing the delay line to be read backwards [4].

For demonstration purposes, the delay time range of the model was arbitrarily set from 60ms to 1s. These ranges can easily be extended to incorporate shorter and/or longer delay times, if desired.

VIII. Results

To test the model, and to aid in the development process, the model was evaluated in a series of playing tests. A Yamaha THR10C amplifier was connected as a USB interface to feed an electric guitar signal into the computer on which the model was running. The model takes audio input from the computers default audio input device, and sends it to the default audio output device. The THR10C was configured to act as both: unprocessed audio was sent from the THR10C to the computer, and the processed audio data was sent back to the THR10C for playback.

The result was a realistic sounding, easy-to-use tape echo model that accurately simulates speed-style pitch modulation behaviour in real-time. The model is capable of self-oscillation, and emulates many of the nuances of an actual tape machine very well. While the CPU cost of the model was not formally measured, any latency experienced by the player between audio input and output was negligible, and the sound quality of the processed audio closely matched that of the unprocessed audio signal.

IX. Recommendations for Future Work

While the presented model provides a realistic simulation of a vintage tape echo, there are several improvements that could be made in future iterations of the model.

Some of the features described in [4] were not implemented in this model. Namely, the use of fixed-point representation for all tape-related variables, and an anti-aliasing filter. It is recommended that all tape variables be converted to a fixed-point representation, and that the anti-aliasing filter described in [4] be incorporated into the model.

In [3], the “wow” and “flutter” characteristics of a Roland Space Echo, caused by nonlinearities in motor speed-ups and slow-downs, were incorporated by adapting the methods described in [7]. It is recommended that these characteristics be incorporated into the model in future iterations.

Finally, it is recommended that a more intricate preamp model be incorporated into the model. Due to the time constraints of this project, it was approximated as a pair of LTI filters. To increase the accuracy of the model, it is recommended that the preamps’ response be simulated more completely

X. Conclusion

A tape echo model that emulates the pitch modulation behaviour of a “speed-style” delay has been presented. This model is capable of processing audio in real-time, using only standard C++ code and the classes provided in the Synthesis Tool Kit. The model also incorporates a simulation of the Echoplex EP-3 preamp, and models the saturation and bandwidth properties of a magnetic tape reel as described in literature. The digital modelling of analog musical equipment is a consequence of the ongoing digitization of all equipment in music production, the tape echo has been no exception. While the originals are expensive, cumbersome to operate, and difficult to maintain, digital models simulating their behaviour are readily available, inexpensive, and reliable. These models accurately simulate the response of their vintage counterparts, and allow musicians and audio engineers to continue to use their warm, nuanced, sound for inspiration and musical production.

XI. References

[1] V. Valimaki, S. Bilbaio, J. O. Smith and J. S. Abel, "Virtual Analog Effects," in *DAFX: Digital Audio Effects, Second Edition*, John Wiley & Sons Ltd., 2011, pp. 473-523.

[2] S. Arnardottier, J. S. Abel and J. O. Smith, "A Digital Model of the Echoplex Tape Delay," in *Conference of the Audio Engineering Society*, San Francisco, CA, USA, 2008.

[3] J. Downing and C. Terjesen, "Real-Time Digital Modeling of the Roland Space Echo," University of Rochester, 2016.

[4] V. Zavalishin and J. D. Parker, "Efficient Emulation of Tape-Like Delay Modulation Behaviour," in *International Conference on Digital Audio Effects (DAFx)*, Aveiro, Portugal, 2018.

[5] P. R. Cooke and G. P. Scavone, "The Synthesis ToolKit in C++ (STK)," Center for Computer Research in Music and Acoustics, 1995-2017. [Online]. Available: https://ccrma.stanford.edu/software/stk/. [Accessed 30th November 2018].

[6] T. Davel, "Tcl8.6.9/Tk8.6.9 Documentation," University of California, 2012. [Online]. Available: https://www.tcl.tk/man/tcl/contents.htm. [Accessed 22nd November 2018].

[7] C. H. Chandler and R. C. o. America, "Flutter Measurement". United States of America Patent 2762013, 1 November 1954.