An adaptive approach to the synthesis of musical signals is presented. The system provides an alternative method to determine the FIR damped filter coefficients in the extended Karplus-Strong string physical modeling. The system can also be viewed as an adaptive system to design a series of FIR filters for any unknown plants. Two types of input signals, namely, a sinusoid and the first period of the musical signal, were used as input to simulate three musical signals: a plucked string guitar, a piano and a violin sound. The LMS algorithm is used as the adaptive algorithm applied in its adaptive filtering structure simulation. Analysis and synthesis musical results using the converged adaptive weights are presented. With the sinusoid input, the result shows that the output signal optimally matches with the original signal in the least square sense. With the input is the first period of the desired signal, the simulated signal has more distortion and poorer sound quality. This project also presents an interesting result of how a piano sound can smoothly change to a plucked guitar string.

1. INTRODUCTION

In *The Technology of Computer Music*[1], Max Mathews wrote "The two fundamental problems in sound synthesis are (1) the vast amount of data needed to specify a pressure function--hence the necessity of a very fast program--and (2) the need for a simple, powerful language in which to describe a complex sequence of sounds."

The first problem has been solved to a large extent by the match of technology of the now days fast digital processor performance. At present, multiple voices of many synthesis techniques can be sustained in real time on a personal computer or many real time DSP built-in audio workstations. [2]

Problem 2 remains unsolved, and cannot, in principle, ever be completely solved. Since it takes millions of samples to make a sound, nobody has the time to type in every sample of sound for a musical piece. Therefore, sound samples must be synthesized algorithmically, or derived from recordings of natural phenomena. In any case, a large number of samples must be specified or manipulated according a much smaller set of numbers. This implies a great sacrifice of generality.

The fundamental difficulty of digital synthesis is finding the smallest collection of synthesis techniques that span the gamut of musically desirable sounds with minimum redundancy. It is helpful when a technique is intuitively predictable. Table 1 shows today's best-known synthesis techniques.

The idea of the system presented in this paper can be viewed as a predictor, parallels to the Linear Predictive Coding (LPC) in frequency domain. LPC has been used for music synthesis and it is listed as a spectral modeling technique because there is evidence that the reason for the success of LPC in sound synthesis has more to do with the fact that the upper spectral envelope is estimated by the LPC algorithm than the fact that it has an interpretation as an estimator for the parameters of an all-pole model for the vocal tract. The adaptive system in this project serves as a predictor for the next period of the signal in time domain, if the input signal chose to be a
A sinusoid with the same fundamental frequency and with the length of the desire signal, the system serves as a time domain envelop for the desire signal. LPC has been proved valuable for estimating loop-filter coefficients in waveguide models of strings and instrument bodies, so it could also be entered as a tool for sampling loop-filters in the "Physical Model" column. And the adaptive system is exactly designing the loop-filter coefficients if the input signal is chosen to be a sinusoid with the length of a period, equals to 1 / (fundamental frequency). As a synthesis technique, LPC has the same transient-smearing problem that spectral modeling based on the short-time Fourier transform has. But the adaptive system does not have the problem for simulating the transient frequency.

As mentioned, a better way to look at the system is that it's a Karplus-Strong (KS) algorithm [3] for string physical model, while the damping filter in KS become the converged weights in LMS. The KS model can also be interpreted as a simplified digital waveguide model or a feed-backward comb filter. The period length of the system is the delay length of the tip delay line of KS. Fig. 1 shows the extended KS model.

![Fig. 1 Extended Karplus-Strong (KS) model](image)

Table 1: Synthesis techniques Summary

<table>
<thead>
<tr>
<th>Processed Recording</th>
<th>Spectral Model</th>
<th>Physical Model</th>
<th>Abstract Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concrete</td>
<td>Wavetable F</td>
<td>Ruiz String</td>
<td>VCO, VCA, VCF</td>
</tr>
<tr>
<td>Wavelet T</td>
<td>Additive</td>
<td>Karplus-Strong</td>
<td>Feedback FM</td>
</tr>
<tr>
<td>Sampling Vector</td>
<td>Phase Vocoder</td>
<td>Waveguide</td>
<td>Waveshaping</td>
</tr>
<tr>
<td>Granular</td>
<td>Sines + Noise</td>
<td>Modal</td>
<td>Phase Distortion</td>
</tr>
<tr>
<td>Prin. Comp. T</td>
<td>Chowning FM</td>
<td>Cordis-Anima</td>
<td></td>
</tr>
<tr>
<td>Wavelet T</td>
<td>LPC</td>
<td>Mosaic</td>
<td></td>
</tr>
</tbody>
</table>

The purpose of this project is to study and explore the feasibility of an audio signal synthesis system using adaptive LMS algorithm [4] as loop filter identification in the KS physical model, or as a signal predictor in time domain. Four different ways of realization of the same idea were tried and compared. The system is tested to simulate not only the plucked guitar string, but also the piano and violin sound.

### 2. SYSTEM SETUP

Four different system setups are implemented to achieve the converged adaptive weights in the adaptive system. Table 2 shows the input and desire signal for each setup.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Input Signal</th>
<th>Desire Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>I (Fig. 2)</td>
<td>Full length sinusoid</td>
<td>Full length audio</td>
</tr>
<tr>
<td>II (Fig. 3)</td>
<td>One period sinusoid</td>
<td>One period audio</td>
</tr>
<tr>
<td>III (Fig. 4)</td>
<td>One period audio</td>
<td>Next period audio</td>
</tr>
<tr>
<td>VI (Fig. 2)</td>
<td>Full length piano</td>
<td>Full length guitar</td>
</tr>
</tbody>
</table>

Table 2: Four different system setups

![Fig. 2 System diagram for Algorithm I and IV](image)

![Fig. 3 System diagram for Algorithm II](image)
2.1. Desire Signal

The desire signals in this project are: a plucked guitar string, a piano and a violin. They are all generated by different physical models and saved as wav format with 44.1 KHz, 16bits. Although they might have less spectrum complexity than PCM recording, they should serve as the equal resource in our project. It’s usually more difficult to simulate the transient frequency for any musical instruments by using mathematical physical modeling methods, but with the adaptive predictor, besides the number of iteration vs. number of FIR filters trade off, there is no such problem since the system output is the best matched signal in the least square sense. There is no need for any data preprocessing with this adaptive system while usually other audio application will require fade in (zero at DC) and even ignore the attack part of the signal to avoid transient frequencies.

2.2. Input Signal

Two types of signal are used as input signal: 1) sinusoid at the same fundamental frequency. 2) The first period (period = 1 / fundamental frequency) of the desire signal. The sinusoid adapts perfectly and has the optimal result in the least square sense. While with the second input signal, the system has heavy distortion and noisy behavior, causing a poor audio quality result. The output sounds like an audio signal with low sampling rate. And if we study its spectrum, the magnitude of the fundamental harmonic is not the largest, while the system introduces some harmonics with higher magnitude than the first harmonic. Psychoacoustic explain parts of the result, although the first harmonic is not the one with the largest magnitude, out brain somehow can fix this problem and automatically find the fundamental frequency, so that one still “hear” the pitch but it will sound much thinner than the original signal.

3. PROCEDURE / ALGORITHM

3.1 Part I – Algorithm I

In the first part of the experiment, a complete sine wave with the same length as the desired signal is used, with frequency chosen to be the same as the desired signal. In this case, a D4 tone of a guitar string, piano string, or violin string is set as the desired signal, which has peak frequency component around 294 Hz. The LMS algorithm is applied to implement the adaptive filter, with two weights (N = 2) and different μ values (0.01, 0.1, 0.4, 0.5, and 1/(2λ max)). The data length is the length of all the valid signals read from the wav file. For piano the data length is about 407,969 samples, for guitar is 124,210 samples, and for violin it is 23,534 samples.

3.2 Part II – Algorithm II

In the second phase, we only take one period of a sine wave (with frequency 293.66 Hz) to simulate each period of the instrument signal, the results are accumulated for each period and output as a single wave file. The weight N = 2, and μ is set to be 0.4.

3.3 Part III – Algorithm III

This algorithm works as a prediction model, where it takes current period signal to simulate next period’s signal. This scheme works like a predictor, where the input signal is the delayed version of the desired signal. The frequency and weight setting are the same as previous one.

3.4 Part VI – Algorithm VI

This algorithm inputs a piano signal to simulate the guitar signal, which could give an interesting sound effect between piano and guitar. The piano signal is also a D4 signal with the same frequency 294.56 Hz. The weights and frequency setting are unchanged.

3.5 Part V

This part tries to input sinusoids with very high or very low frequencies, and see if the expected waveform can still be obtained. This will show if an appropriate frequency is really necessary to start with in order to achieve the desired output.

3.6 Part VI

Finally, a wave file with variable Doppler Effect (therefore has variable frequency) is taken as the desired signal, and algorithm I is implemented to simulate the desired signal. This is to verify that the input signal must take appropriate frequency value to perform well.

Fig. 5 shows the waveform of a plucked guitar string, a piano and a violin signal result from different physical models. [5]
4. RESULT / ANALYSIS

4.1 Part I – Algorithm I

The results obtained from the first part is very close to the original signal, except that there are small distortion in the signal, the sound is almost identical to the human ear. The drawback of this algorithm is that it takes about 10 minutes to run the simulation for guitar, and 100 minutes for piano signal. Since violin file has short data length, it only takes 22 sec to run, but the performance is not excellent. Fig.6 shows the synthesis result with the algorithm I for guitar and piano signal:

4.2 Part II – Algorithm II

In the second part, the algorithm takes much less time to compute, around 16 sec for guitar, 70 sec for piano, and 3 sec for violin. The quality of the sound for guitar is about the same as the previous one, with very little distortion, but the piano signal has worse performance on this algorithm. The output sounds more like a string than a piano. For violin, one can also find some distortions when comparing with the original signal. Fig. 7 shows the synthesis result with algorithm II for guitar and piano sound.
Fig. 7 (a) Synthesis output and (b) learning curve with Algorithm II for Guitar. (c) Output and (d) learning Curve with Algorithm II for Piano. Note that the learning curve accumulates the MSE measurement of each training period and outputs as a single plot.

4.3 Part III – Algorithm III

In the prediction model, it is obvious that the sound quality has significantly deteriorated. The noise is very noticeable and the waveform is very different from the desired one. The calculation time is about 16 sec for guitar and 70 sec for piano, but with much worse result. However, violin signal performs quite well with this algorithm, since it is a pretty uniform signal for each period. Fig. 8 is the plot with algorithm III for guitar and violin signal.

Fig. 8 (a) Output and (b) learning curve with Algorithm III for Guitar. (c) output and (d) Learning Curve with Algorithm III for Violin.

4.4 Part IV – Algorithm IV

In this part, the piano sound is taken to simulate the guitar sound, which kind of mix the two sound components together and gives an exotic result. Since the reverberation is very long in piano signal, it takes a while to complete the calculation. The result is not supposed to be very accurate due to the long reverberation time of the piano. Fig 9 shows the synthesis result by algorithm IV.

The piano does smoothly changed to the guitar sound. And the MSE is pretty close to zero, although the algorithm does not reduce much MSE in this case. Table 3 describes the calculation time for each algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Piano</th>
<th>Guitar</th>
<th>Violin</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>6636 s</td>
<td>637.6 s</td>
<td>21.62 s</td>
</tr>
<tr>
<td>II</td>
<td>70.50 s</td>
<td>15.66 s</td>
<td>2.937 s</td>
</tr>
<tr>
<td>III</td>
<td>70.78 s</td>
<td>16.13 s</td>
<td>2.843 s</td>
</tr>
<tr>
<td>IV</td>
<td>640.4 s</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3 Calculation Time with Four Different Algorithms
The piano signal takes about four times longer to calculate in algorithm II and III since it has a longer reverberation time and the redundant zeros cannot be deleted like it was done in the guitar signal. It even takes 10 times longer when implementing algorithm I. For violin, due to its short data size, the calculation time is significantly less. Therefore, algorithm II is decided to be the primary algorithm in future calculation.

Table 4 shows that change of $\mu$ does not affect the calculation time too much when using the same algorithm with the same desired signal (guitar in this case).

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>Time 1/(2$L_{max}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>653.1 s</td>
</tr>
<tr>
<td>0.1</td>
<td>634.3 s</td>
</tr>
<tr>
<td>0.4</td>
<td>637.6 s</td>
</tr>
<tr>
<td>0.5</td>
<td>633.8 s</td>
</tr>
<tr>
<td>1/(2$L_{max}$)</td>
<td>648.5 s</td>
</tr>
</tbody>
</table>

Table 2 Calculation Time with Different $\mu$ Values using Algorithm I on Guitar Signal

The value of $\mu$ also changes the resulting amplitude of the output signal. The larger the $\mu$ is, the larger the signal output is, and vice versa. Therefore it is desirable to have the maximum amplitude around but less than 1 to avoid clipping and utilize the full range of the audio signal. It turns out that when $\mu = 0.4$ it gives the best result.

4.5 Part V

This part returns very poor signal simulation, obviously an appropriate frequency is required to synthesize the desired signal at certain frequency: an input frequency that is integer multiple or fraction of the desired signal frequency will usually work well, but for arbitrary signal, the LMS will not work at all. Fig 10 shows the synthesis waveform of doubling and halving the normal input frequency. And the results are good simulation to the desired signal.

4.6 Part VI

For the last part of the project, the simulated wave has good convergence in the beginning but gets worse and worse as time goes on. This is because the first half of the signal has a frequency that is closer to the peak frequency while the latter half does not. It proves that the input signal requires some engineering to get a nicer output signal. Fig 11 shows the desired variable frequency wave and the simulation.

4.6 Part VI

For the last part of the project, the simulated wave has good convergence in the beginning but gets worse and worse as time goes on. This is because the first half of the signal has a frequency that is closer to the peak frequency while the latter half does not. It proves that the input signal requires some engineering to get a nicer output signal. Fig 11 shows the desired variable frequency wave and the simulation.

5. Conclusion / Future Work

It is shown that the second algorithm has the best performance with the fastest calculation time while maintaining the accuracy of the desired signal for guitar. For violin signal, algorithm III performs the best. It implies that different signal pattern might require different algorithm for performance optimization. For periodic signal with uniform amplitude, algorithm III can predict the next period’s signal fairly accurately, and for impulse signal like piano and guitar, a single period sine wave (algorithm II) works well.

Since the calculation time grows as the data length gets longer, a reasonable data length is required to run this
algorithm. In order to get the calculation time under 10 minutes, a sample length around 50,000 ~ 100,000 is recommended, with sampling frequency being 44,100 Hz.

For further research, musical signals like drum, wind instrument, or signals with time-variable frequencies, could be used as the desire signal to study the convergence properties of the LMS algorithm. The complexity of a variable frequency signal cannot be simulated by simply applying a constant frequency input signal. It is shown that when input frequency signal is too far off from the desired signal, the LMS algorithm cannot converge to what is expected. Therefore, to simulate a variable frequency wave, it is required to have a special algorithm to determine what frequency to be applied to the input signal at different time frame to achieve the convergence, and this can also be achieved by another adaptive scheme.

It is also necessary to improve the simulation quality if the algorithm is to be used on professional audio systems, where harmonic distortion and noise have to be very low (a Tone Harmonic Distortion + Noise level is considered to be accept if it is below -70 dB). Therefore some new algorithms are required to improve the current setting.

6. REFERENCES