Music Transcription:

Identification of Musical Note Played on a Piano

Ayano Hiranaka
Mentored By Amirhossein Taghvaei
University of Illinois at Urbana-Champaign
June 2019

Contents

1. Introduction

The main objective of this project is to identify, in real-time, the note being played on a piano. An audio file of piano notes being played are inputted into an adaptive filtering algorithm which compares the input signal to each note's ideal signal models and predicts the note being played through a probabilistic approach. The oscillator model is used to model the audio signals, and the filter used is the coupled oscillator feedback filter.

2. Model Generation and Visualization

An oscillator model used to model the audio signals is developed. The ideal oscillator position (angle in radians) θ at any instance in time k (up to K) and the observation signal Y_k, or the real audio signal with noise, are modeled as follows: is described as follows:

Ideal Oscillator: $$\theta_{k+1} = (\theta_k + \omega\Delta t)\ \%2\pi \tag{1}$$

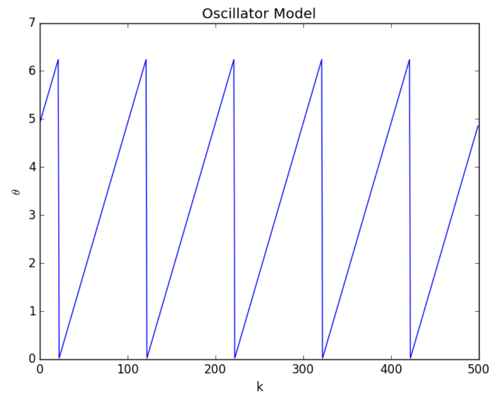
Observation Signal: $$Y_k = h(\theta_k) + \sigma_w W_k \tag{2}$$

Where ω is the signal frequency in rad/s and Δt is the time step size, $h(\theta_k)$ is the observation function, W_k is the normally distributed noise, σ_w is the noise strength ($\sigma_w \geq 0$), and θ is assumed to have an initial value of θ_o at $k = 0$ selected randomly from an uniform distribution $[0, 2\pi]$.

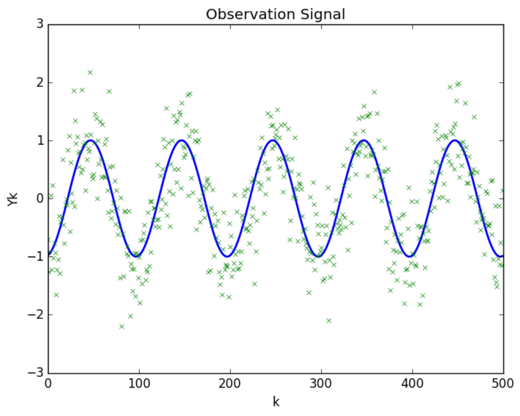
First, the observation function is assumed to take the following form where c is a constant:

Observation Function: $$h(\theta_k) = csin\theta$$

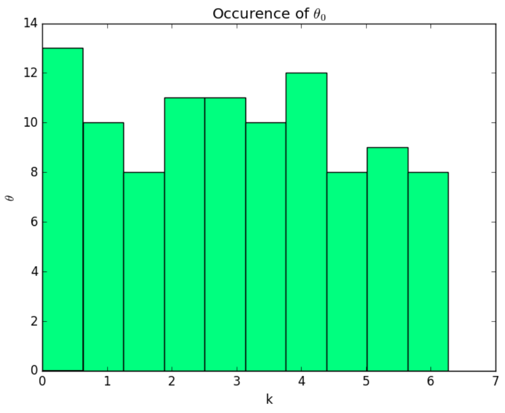
Figures 1 and 2 showing plots of the ideal oscillator and the observation signal are obtained using $\omega = 2\pi,\ \Delta t = 0.01, \sigma_w = 0.1, c = 1.0, K = 500$, and shows the change in the ideal and observed oscillator's behavior with respect to time. The observation signal's behavior is similar to that of the ideal signal, and varies according to the normally distributed noise. Figures 3 and 4 show the occurrence of θ_o and θ_{50} after the system is ran $M = 100$ times. The occurrences of θ values are generally uniform.



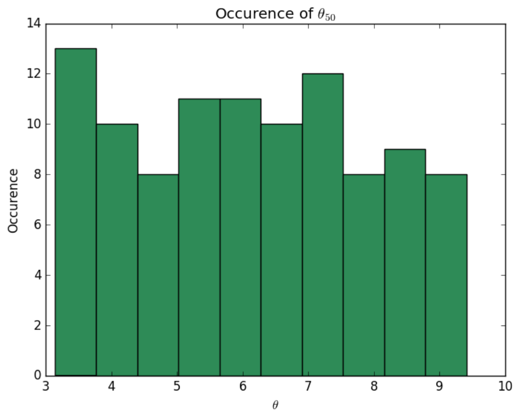
(a) Time Trace of Ideal Oscillator



(b) Ideal and Observation Signals



(c) Histogram of Occurrence of θ_o



(d) Histogram of Occurrence of θ_{50}

Figure 1: Behavior of the Oscillator Model and the Observation Signal

3. Feedback Particle Filter Algorithm

In this step, assume that only the observation signal is available (the ideal signal the observation signal represents is unknown). The goal of this section is to predict θ_k given Y_k with N oscillators. Each of the N oscillators are initialized with a frequency ω_0^i and a position θ_0^i randomly selected from uniform distributions with ranges $[\omega_0 - \delta, \omega_0 + \delta]$ and $[0, 2\pi]$, respectively. The FPF algorithm is applied to each of the N oscillators to predict the ideal oscillator the observation signal represents. The oscillators evolve in the following way:

FPF: $$\theta_{k+1}^{i} = \theta_{k}^{i} + \omega^{i}\Delta t + K^{i}\left(Y_k - \frac{h\left(\theta_k^i\right) + \hat{h}}{2}\right)\frac{1}{\sigma_w^2} \ \%2\pi \tag{3}$$

Where $\delta = 0.1$, $\hat{h} = \frac{1}{N}\sum_{i=1}^{N} h(\theta_k^i)$ and K^i is the gain. The gain is calculated in the following algorithm:

Algorithm 1: Calculate the Gain in FPF Algorithm

```
Input: ${\theta^i}_{i=1}^N$, ${h(\theta^i)}_{i=1}^N$
Output: ${K^i}_{i=1}^N$
```

$$\hat{h} = \frac{1}{N}\sum_{i=1}^{N} h(\theta^i)$$

$$A_{11} = \frac{1}{N}\sum_{i=1}^{N} sin^2(\theta^i)$$

$$A_{22} = \frac{1}{N}\sum_{i=1}^{N} cos^2(\theta^i)$$

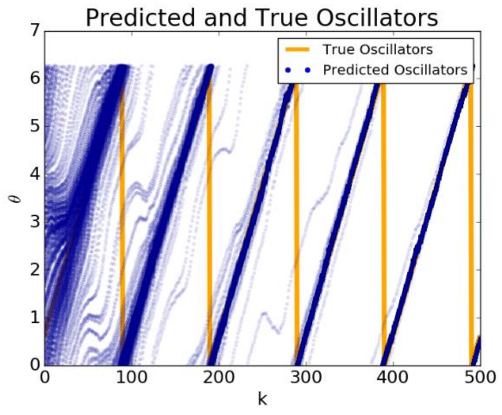
$$A_{12} = A_{21} = -\frac{1}{N}\sum_{i=1}^{N} sin(\theta^i)cos(\theta^i)$$

$$b_1 = \frac{1}{N}\sum_{i=1}^{N} cos(\theta^i)(h(\theta^i) - \hat{h})$$

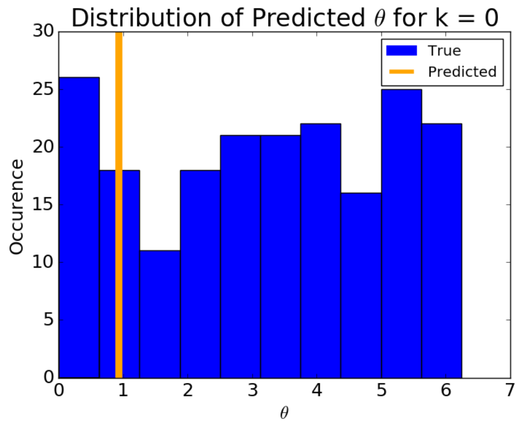
$$b_2 = \frac{1}{N}\sum_{i=1}^{N} sin(\theta^i)(h(\theta^i) - \hat{h})$$

$$\mathrm{A} = [[A_{11}, A_{12}], [A_{21}, A_{22}]]$$
$$\mathrm{b} = [b_1, b_2]$$
$$\mathrm{c} = A^{-1}b$$
$$K^i = -c_1\sin\left(\theta^i\right) + c_2\cos\left(\theta^i\right) \ for\ i = 1, \dots, N$$

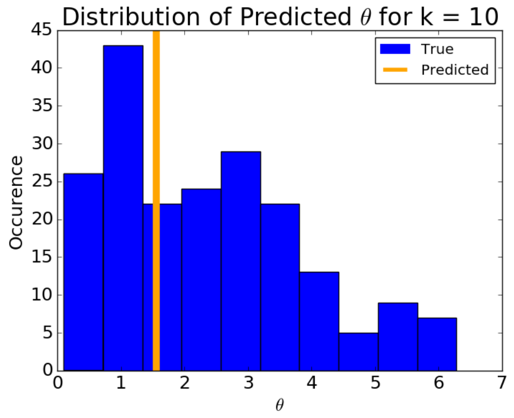
Figures 5 to 8 shows the results of the FPF algorithm. As more time elapses, the *N* oscillators converge to the true oscillator's position. At *k=0*, the *N* oscillator's prediction of the true oscillator's position is random, as shown in Figure 6. For larger values of *k*, such as 100, the prediction becomes more accurate, as shown by Figure 8.



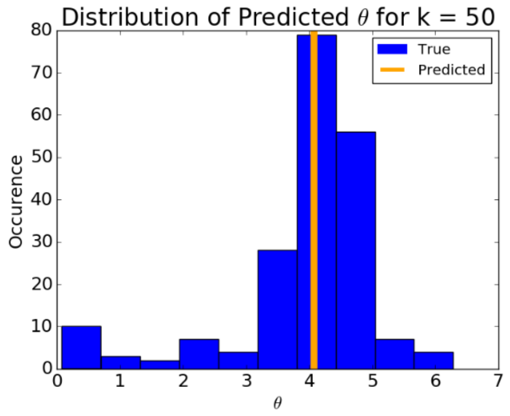
(a) Time Trace of True and Predicted Oscillators



(b) Histogram of Distribution of Oscillators at Time k=0



(c) Histogram of Distribution of Oscillators at Time k=10



(d) Histogram of Distribution of Oscillators at Time k=50

Figure 2: Results of the FPF Algorithm

4. Empirical Studies on Performance of FPF Algorithm

In this section, the performance of the FPF algorithm is investigated when: (1) the number of oscillators *N* is varied, (2) there is uncertainty in the ideal oscillator model, and (3) there is uncertainty in the observation model. In order to quantify the performance, the error of the algorithm is defined in the following way:

Error: $$e_n = \frac{1}{n}\sum_{k=0}^{n-1}\left[\left(\frac{1}{N}\sum_{i=1}^{N}\sin(\theta_k^i) - \sin(\theta_k)\right)^2 + \left(\frac{1}{N}\sum_{i=1}^{N}\cos(\theta_k^i) - \cos(\theta_k)\right)^2\right] \quad (4)$$

4.1 Effect of the Number of Oscillators *N*

The effect of the number of oscillators is investigated by plotting the error with respect to the number of oscillators *n* for *N* = {10, 20, 50, 100, 200, 500, 1000}. The result is shown in Figure 3a. The error was expected to decrease as the number of oscillators used are increased, but Figure 3a shows an interesting result suggesting that there is little difference in error resulting from a change in the number of oscillators used.

4.2 Effect of Model Uncertainty

Next, the effect of uncertainty in the ideal signal model is investigated. Practically, parameters of the ideal signal model will most likely be unknown, for example, the true frequency of the signal, *ω,* may be unknown. In the algorithm, the *N* oscillators' frequencies ω^i are selected from the range $[\omega_0 - \delta, \omega_0 + \delta]$. It is predicted that the FPF filter will function properly when the true frequency of the input signal is within this range. This section investigates the FPF algorithm's performance when the true frequency is outside of this range. For this purpose, the ideal oscillator frequency is defined as $\omega = 2\pi(1 + \alpha)$ and $\delta = 0.2$, and the error values are compared for $\alpha = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$. Results are shown in Figure 3b. As expected, the error is higher for higher values of α, or when the actual frequency deviates more from the expected range.

4.3 Effect of Observation Model Uncertainty

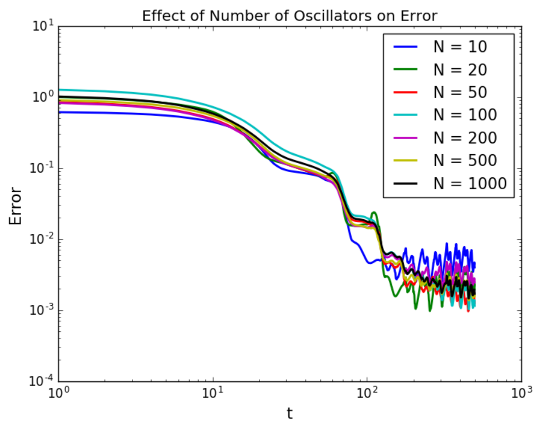
For this analysis, the observation function is assumed to take the form:

$$h(\theta) = c_1 sin\theta + c_2 cos\theta.$$

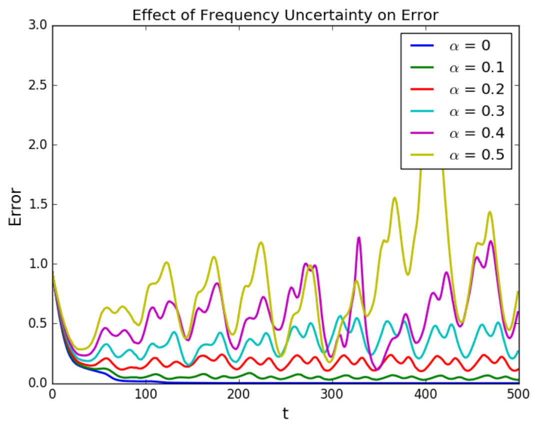
To investigate the effect of observation model uncertainty on error, the FPF algorithm will be applied to the oscillators assuming that the observation function takes the form as initially defined:

$$h(\theta_k) = c_0 sin\theta.$$

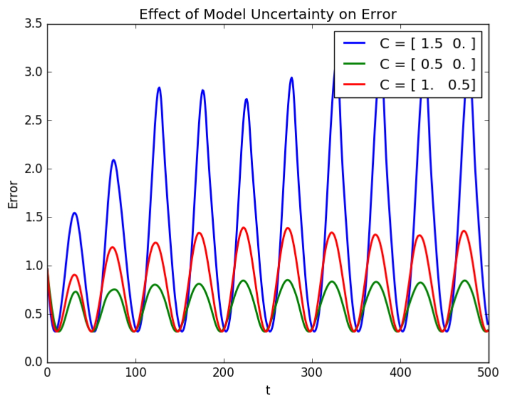
Let$c_0 = 1.0$ and $\mathrm{C} = [(c_1, c_2)] = [(1.5, 0.0), (0.5, 0.0), (1.0, 0.5)]$. The resulting error is shown in Figure 3c. $(c_1, c_2) = (1.5, 0.0)$ results in the highest error while $(c_1, c_2) = (0.5, 0.0)$ results in the lowest error, suggesting that the error becomes high when the observation model coefficient exceeds the coefficient of the true model. The performance when $(c_1, c_2) = (1.0, 0.5)$ suggests that the error is not affected significantly when the actual signal contains a harmonics not assumed by the algorithm ($c_2 cos\theta$) if the weight of this component is sufficiently low.



(a) Time Progression of Error for Different Number of Oscillators N



(b) Effect of Model Frequency Uncertainty in Error Progression



(c) Effect of Observation Model Uncertainty on Error Progression

Figure 3: Results of Empirical Studies on FPF Performance

5. Adaptive Filtering

Results of section 4.3 reveals that the FPF algorithm's performance is sensitive to how well the assumed observation model matches the actual observed signal.

5.1 Coefficient Adapting

This section considers a way to adapt the coefficients of the assumed observation function to fit the actual observation function. To better accommodate various observation functions, assume the function takes the following form:

Observation Function: $$h(\theta) = c_1 sin\theta + c_2 \sin(2\theta) + c_3 \cos(2\theta) \tag{5}$$

In every iteration of the FPF algorithm, the coefficients c_1, c_2, and c_3 evolves in the following way for $n = 1, 2, 3$:

$$c_n \hookleftarrow c_n - \frac{\varepsilon_{amp}}{\sigma_w^2} \frac{\partial}{\partial c_n} \underbrace{\left(Y_k - \frac{1}{N} \sum_{i=1}^{N} h\left(\theta_k^i\right) \right)^2}_{error} \tag{6}$$

Where $\varepsilon_{amp} = 0.1$ is the *amplitude correction factor*. In this approach, the coefficients are adjusted along the direction of the error gradient, reducing the error each time the coefficients are updated.

5.2 Noise Strength Adapting

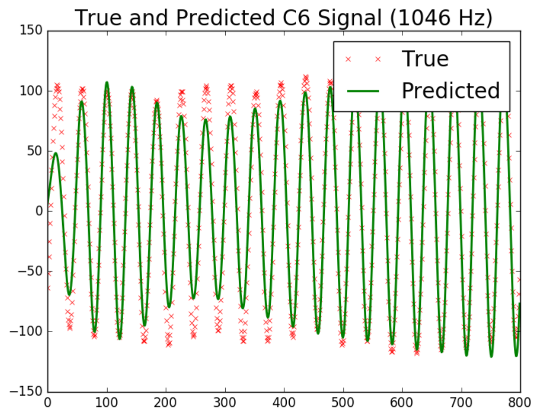
The performance of the FPF algorithm is sensitive to the error strength σ_w. The noise level may be unknown or varying, and will be adapted in the following way:

$$\sigma_w^2 \hookleftarrow (1 - \varepsilon_{noise})\sigma_w^2 + \varepsilon(Y_k - \hat{h}_k)^2 \tag{7}$$

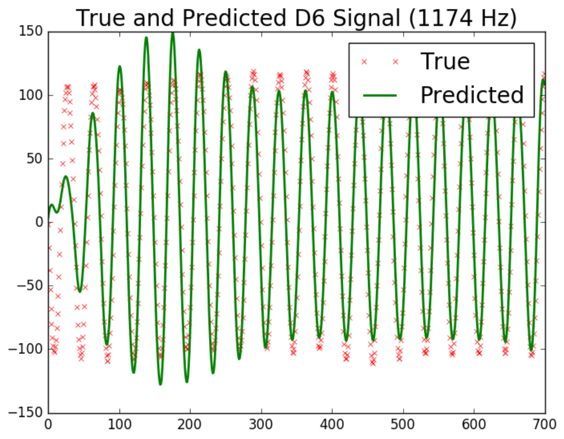
Where $\varepsilon_{noise} \ll 1$ is the *noise correction factor*.

6. Application on Real Signals

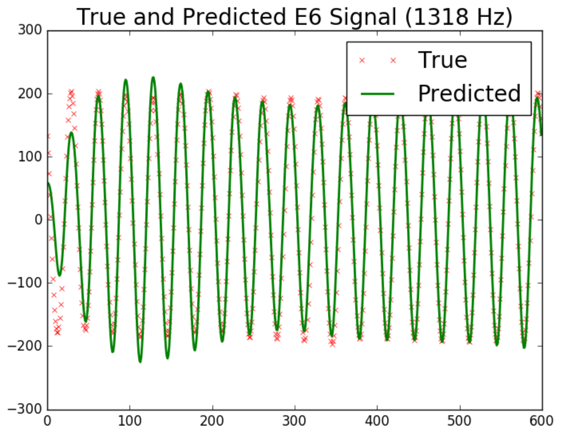
In this section, the FPF algorithm is applied to a real piano note signal. Audio files of single piano keys played once are inputted to the algorithm and its responses are observed. The notes selected as inputs are $\{C6, D6, E6, F6, G6, A6, B6, C7\}$. Prior to entering signals into the system, each signal was investigated and reasonable starting values for the observation model coefficients (c_1, c_2, c_3) and the noise strength σ_w^2 are determined for each notes signal. The results are shown in Figure 4. It is observed that the algorithm is able to trace each note accurately.



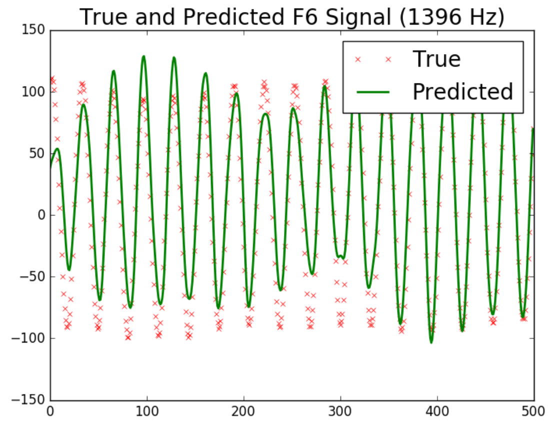
(a) C6



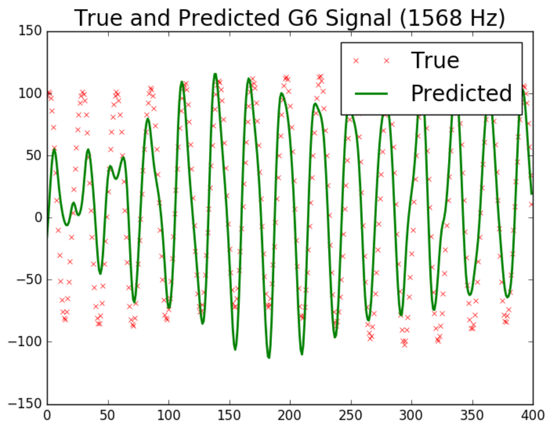
(b) D6



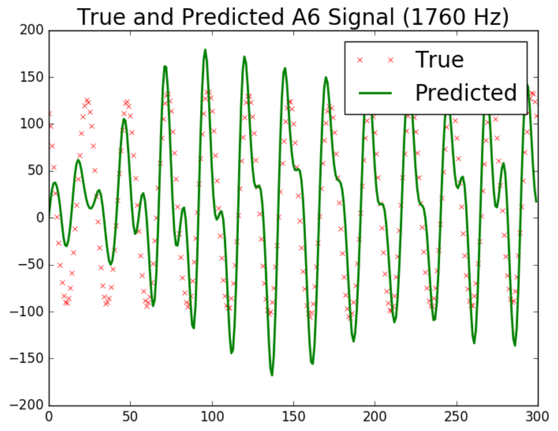
(c) E6



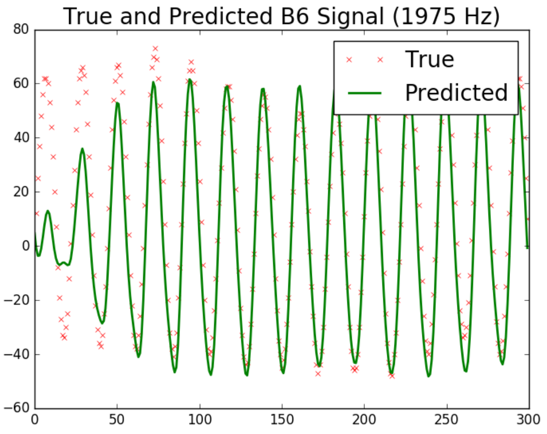
(d) F6



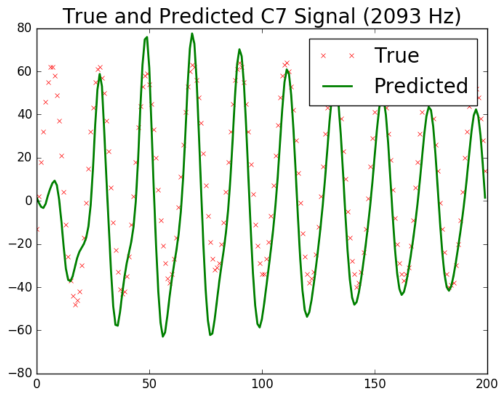
(e) G6



(f) A6



(g) B6



(h) C7

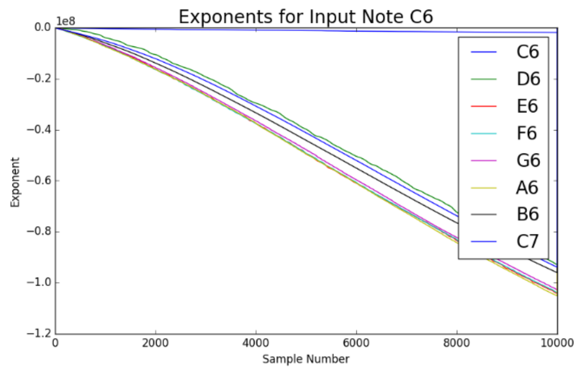
Figure 4: Extracted Portions of Time Traces of True Input Signals and FPF Predicted Signals $\{C6, D6, E6, F6, G6, A6, B6, C7\}$

7. Note Classification

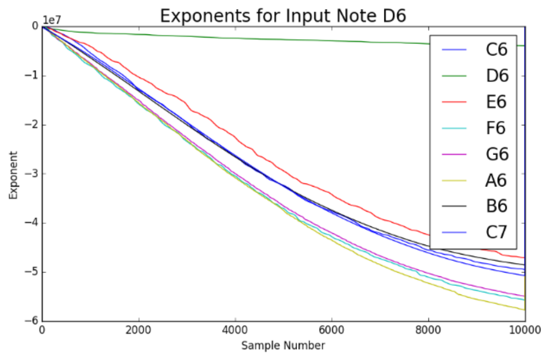
The objective of this section is to identify which note is being played given one of the notes from $\{C6, D6, E6, F6, G6, A6, B6, C7\}$ as input. The FPF algorithm is applied to the input signal eight times, each time assuming that the input signal is one of the eight notes: First, the algorithm is applied assuming that the input note is *C6*, the next iteration of the algorithm is applied assuming that the input note is *D6*. This process is repeated until all eight notes are considered. Each time, the prediction oscillator attempts to trace the input signal, and the prediction signal that is able to trace the input signal most accurately is identified as the note played. To quantify how well the input is predicted, the filter exponents $\{\mu_k^{(1)}, \mu_k^{(2)}, \dots, \mu_k^{(8)}\}$ are initialized at zero and update according to:

Filter Exponents:
$$\mu_{k+1}^{(m)} = \mu_k^{(m)} - \underbrace{\frac{1}{2\sigma_W^{2\,(m)}}\left(y_k - \hat{h}_k{}^{(m)}\right)^2}_{prediction\ error} \tag{8}$$

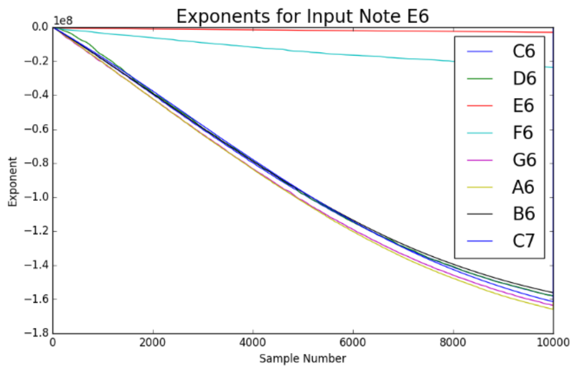
Where $\{\hat{h}_k^{(1)}, \hat{h}_k^{(2)}, \dots, \hat{h}_k^{(8)}\}$ are prediction outputs for each of the eight notes, obtained by averaging the values of the N oscillators at each time step k. Results when each of the eight notes are inputted is shown in Figure 5. Notes $\{C6, D6, E6, F6, G6, A6, B6\}$ are identified correctly. The difference in the exponents of *E6* and *F6* are small when the input note is *F6,* and *C7* is incorrectly identified as *B6*, suggesting that the algorithm may not be able to reliably distinguish between notes that are a half-note apart.



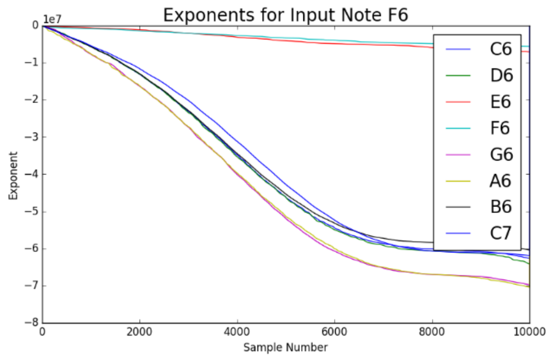
(a) C6 as Input



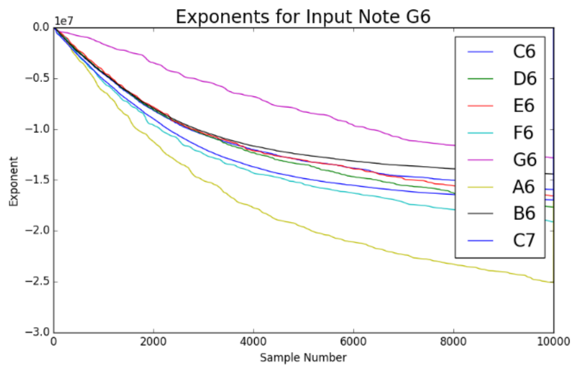
(b) D6 as Input



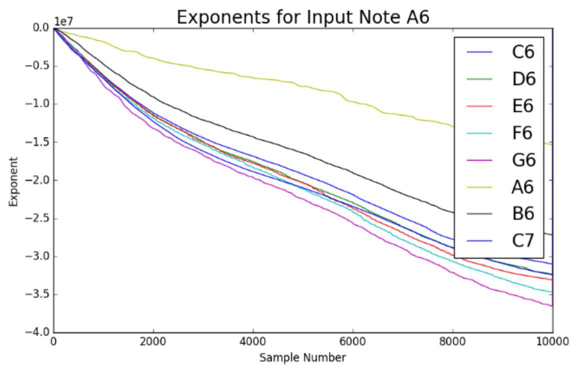
(c) E6 as Input



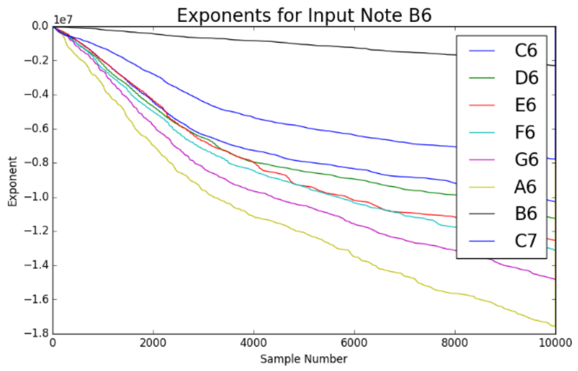
(d) F6 as Input



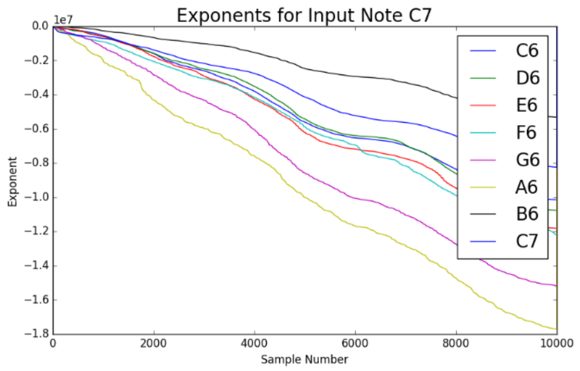
(e) G6 as Input



(f) A6 as Input



(g) B6 as Input



(h) C7 as Input

Figure 5: Exponents for Inputs $\{C6, D6, E6, F6, G6, A6, B6, C7\}$

8. Detecting Change in Note Played

In order to transcribe a musical piece played on a piano, the algorithm must be able to detect a change in the note being played. The objective of this section is to identify a change in the note being played in real-time using the exponents values.

8.1 Reducing Reaction Time

When, for instance, the note *C6* is played, the exponent for *C6* shows the highest value among the eight exponents. Then, when the note changes, for example, to *G6*, the exponent of *G6* begins to increase. However, the system will not identify the new note *G6* until the exponent for *G6* surpasses the exponent of *C6*. In order to reduce this delay, a threshold is implemented, preventing exponent values to decrease below a certain value. While a higher threshold increases reaction speed, it also increases the likelihood of error. An optimum level of the threshold is found using a portion of a piano recording of the "ABC Song" where the note changes from *C6* to *G6*. To allow the ideal samples to better fit the input signal, the input song signal's amplitude is reduced to 150. Results for $\varepsilon_{amp} = 0.01$, $\varepsilon_{noise} = 10^{-5}$, $threshold = -10^{6}$ is shown in Figure 6. The note played changes from *C6* to *G6* at the red vertical line, and the system identifies this change after a delay of approximately 0.263 seconds.

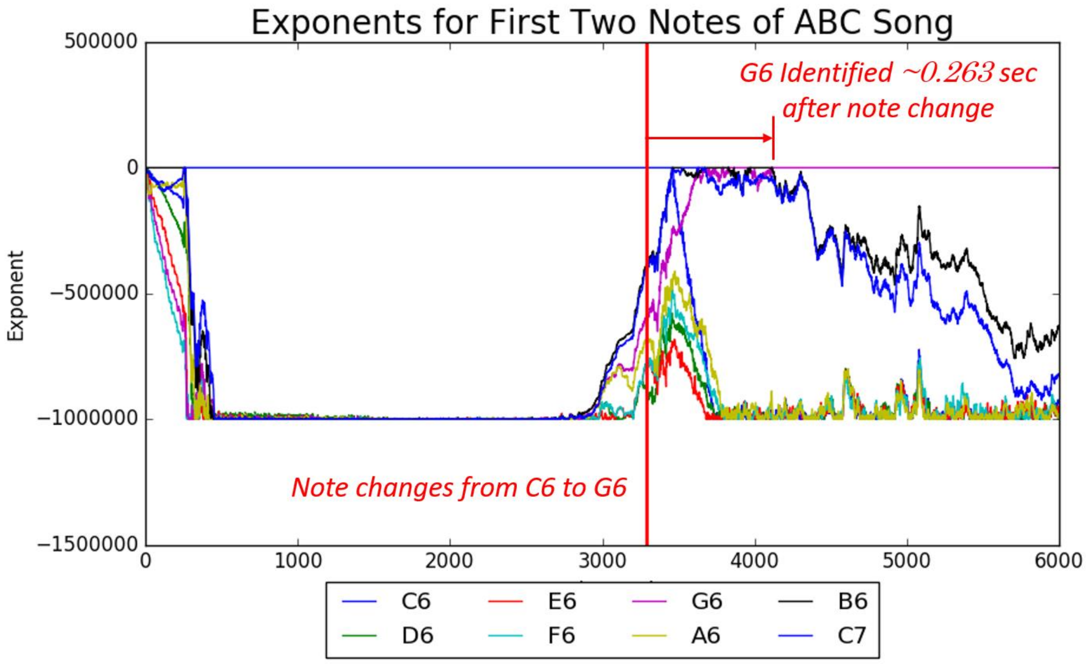


Figure 6: Change in Exponent as the Note Played Changes from C6 to G6