Music 206: Delay and Digital Filters II

Tamara Smyth, trsmyth@ucsd.edu
Department of Music,
University of California, San Diego (UCSD)

April 18, 2019

Flanger

- The well known *flanger* is a feedforward comb filter with a time-varying delay $M(n)$ (see flanging.mov).
- Flanging, used in recording studios since the 1960s, creates a rapidly varying high-frequency sound by adding a signal to an image of itself that is delayed by a short, variable amount of time.
- Flanging was accomplished in analog studios by summing the outputs of two tape machines playing the same tape.

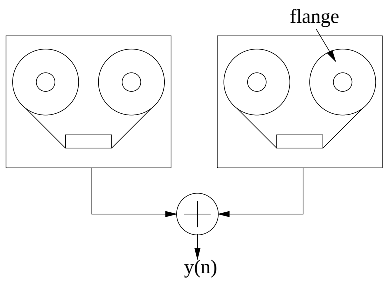


Figure 1: Two tape machines are used to produce the flanging effect.

- By touching (and releasing) the flange on one supply reel, it would s to slow it down (and speed it up).

Flange Comb Filter Parameters

- The flange simulation is a *feedforward* comb filter, where the delay $M(n)$ is a function of time,

$$y(n) = x(n) + gx(n - M(n)).$$

  - coefficient g (DEPTH parameter), determines the prominence of the flanging effect.
  - flange is typically swept from a few milliseconds to 0 to produce characteristic “flange” sound.

- The time-varying delay can be handled by modulating $M(n)$ with a low-frequency oscillator (LFO) sinusoid:

$$M(n) = M_0[1 + A\sin(2\pi f n T)],$$

  where

$$\begin{aligned} f &\triangleq \text{rate or speed of the flanger, in Hz} \\ A &\triangleq \text{“excursion” (maximum delay swing)} \\ M_0 &\triangleq \text{average delay length controlling the} \\ &\quad\ \text{average notch density.} \end{aligned}$$

Fractional Delay using Linear Interpolation

- For a successful flanging effect, $M(n)$ must change *smoothly* over time:
  - $M(n)$ should not have jumps in values associated with rounding to the nearest integer.
- One of the simplest ways to handle fractional delay is by using **linear Interpolation**:
  - the *linear interpolator* effectively "draws a line" between neighbouring samples, and returns the appropriate value on that line.

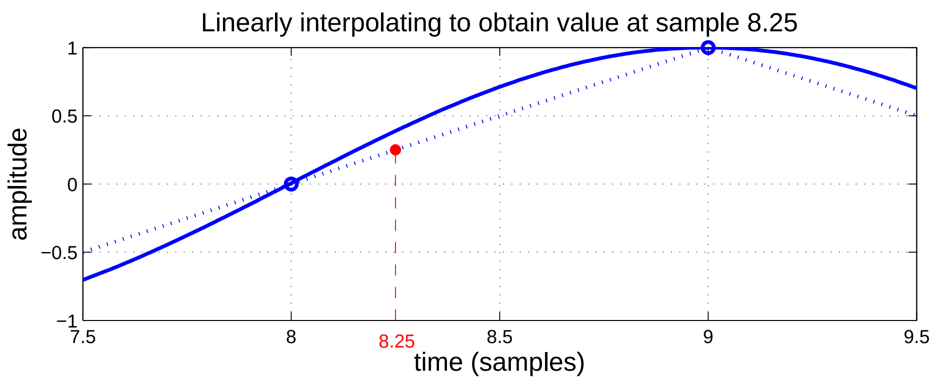


Figure 2: Linear Interpolation.

Linear Interpolation (Implementation)

- The fractional part of the delay, δ, effectively determines how far to go along the line between samples.
- A fractional delay $\hat{x}(n-(M+\delta))$, reads from the delay line at neighbouring delays M and $M+1$, and takes the weighted sum of the outputs:

  $$\hat{x}(n-(M+\delta)) = (1-\delta)x(n-M)+\delta x(n-(M+1)),$$

  where M is the integer and δ is the fractional part.
- Notice that if $\delta = 0$, the fractional delay reduces to the regular integer delay.
- Linear interpolation in a circular delay line (Matlab):

```
if (outPtr==1)
  z = (1-delta)*dline(outPtr) + delta*dline(Mmax);
else
  z = (1-delta)*dline(outPtr) + delta*dline(outPtr-1);
end
```

Tapped Delay Line

- A *tap* refers to the extraction of the signal at a certain position within the delay-line.
- The tap may be interpolating or non-interpolating, and also may be scaled.
- A tap implements a shorter delay line within a larger delay line.

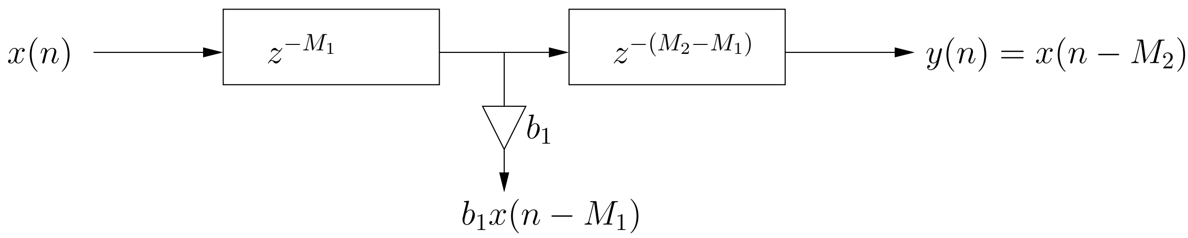


Figure 3: A delay line *tapped* after a delay of M_1 samples.

Multi-Tap Delay Line Example

- Multi-Tapped delay lines efficiently simulate multiple echoes from the same source signal.

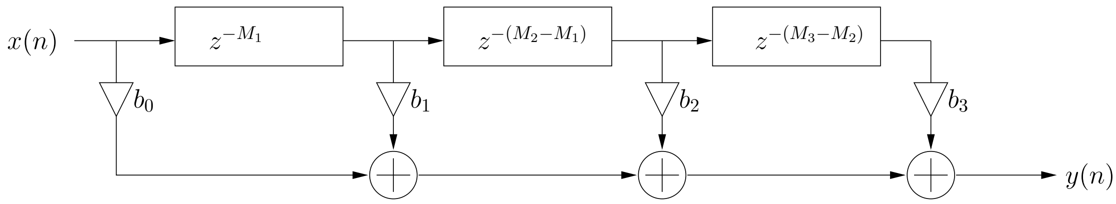


Figure 4: A multi-tapped delay with length M_3.

- In the above figure, the total delay line length is M_3 samples, and the internal taps are located at delays of M_1 and M_2 samples, respectively.
- The output signal is a linear combination of the input signal $x(n)$, the delay-line output $x(n - M_3)$, and the two tap signals $x(n - M_1)$ and $x(n - M_2)$.
- The difference equation is given by

$$y(n) = b_0x(n)+b_1x(n-M_1)+b_2x(n-M_2)+b_3x(n-M_3)$$

- *Convolution* is equivalent to tapping a delay line every sample and multiplying the output of each tap by the value of the impulse response for that time.

Chorus

- A Chorus is produced when several musicians play simultaneously, but inevitably with small changes in the amplitudes and timings between each individual's sound.
- The chorus *effect* is a signal processing unit that changes the sound of a single source to a chorus by implementing the variability occurring when several sources attempt to play in unison.

Chorus Implementation

- A chorus effect may be efficiently implemented using a *multi-tap fractional* delay line:
  - taps are not fixed and usually range from 10 to 50 ms.
  - their instantaneous delay may be determined using a random noise generator or, as in the flanger, a Low Frequency Oscillator (LFO).

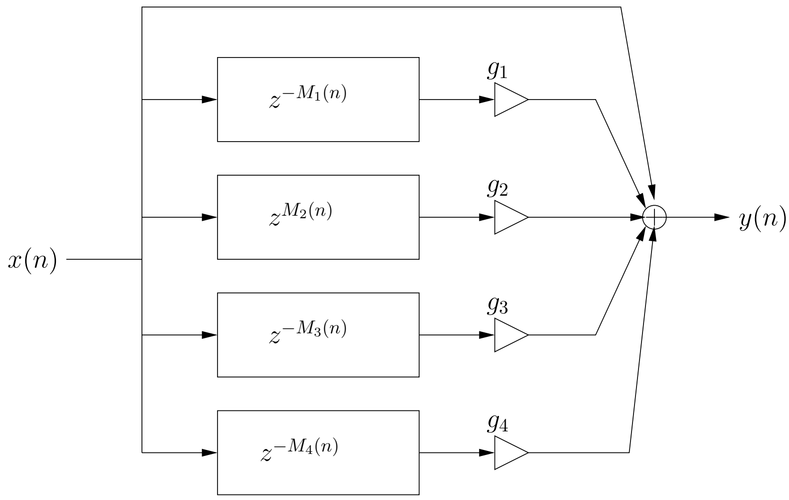


Figure 5: A bank for variable delay lines realize the chorus effect.

- The chorus is similar to the flanger, only there are multiple delayed copies of the input, and the delay times are typically longer (where a flanger is about 1-10 ms, a chorus is about 10-50 ms).

A Simple Recursive (IIR) Filter

- Using FIR filters to reproduce a desired frequency response often requires a very high-order filter, i.e., a greater number of coefficients and more computation.
- It is often possible to reduce the number of **feedforward** coefficients by introducing **feedback** coefficients.
- A simple first-order recursive low-pass filter is given by

$$y(n) = x(n) + .9y(n-1)$$

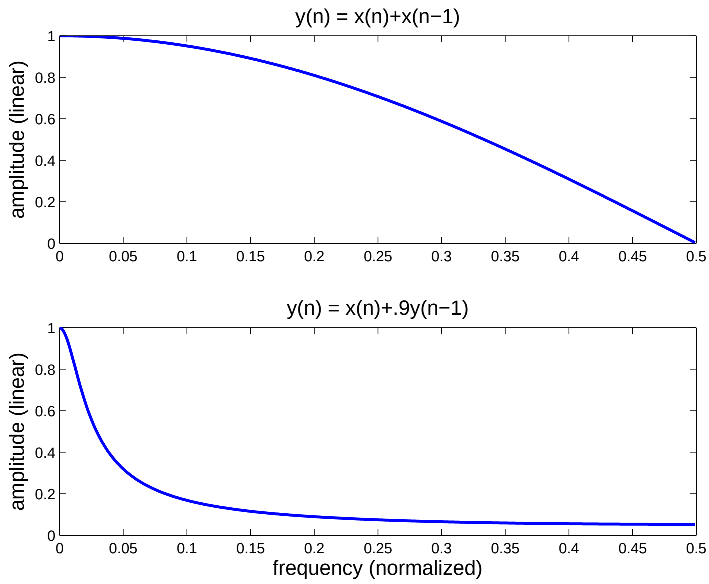


Figure 6: The spectral magnitude of the first-order FIR and IIR (recursive) lowpass filters.

The General Difference Equation for LTI filters

- The general difference equation for LTI filters includes feedback terms, and is given by

$$\begin{aligned} y(n) = b_0 x(n) &+ b_1 x(n-1) + \cdots + b_M x(n-M) \\ &- a_1 y(n-1) - \cdots - a_N y(n-N) \end{aligned}$$

- This can be implemented in Matlab using the `filter` function:

```
B = ...; % feedforward coefficients
A = ...; % feedback coefficients
y = filter(B, A, x);
```

- Matlab specifies coefficients according to the filter transfer function and NOT the difference equation:
  - all **feedback** coefficients (except the first) have a sign *opposite* to that in the difference equation;
  - this is explained by moving the y terms in the difference equation to the left of the equal sign (a step in arriving at the filter *transfer function*):

$$y(n) + a_1 y(n-1) + \cdots = b_0 x(n) + b_1 x(n-1) + \cdots$$

The Simple Feedback Comb Filter

- What happens when we multiply the output of a delay line by a gain factor g then feed it back to the input?

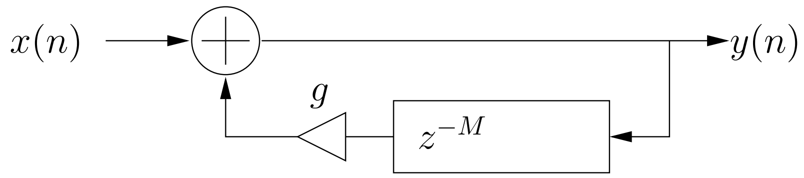


Figure 7: The signal flow diagram of a comb filter.

- The difference equation for this filter is

$$y(n) = x(n) + gy(n - M),$$

- If the input to the filter is an impulse

$$x(n) = \{1, 0, 0, \ldots\}$$

the output (impulse response) will be ...

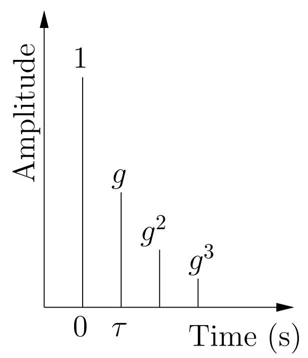


Figure 8: Impulse response for filter $y(n) = x(n) + gy(n - M)$, where $\tau = M/f_s$.

Effect of Feedback Delay

- Since the pulses are equally spaced in time at an interval equal to the loop time $\tau = M/f_s$ seconds, it is periodic and will sound at the frequency $f_0 = 1/\tau$.

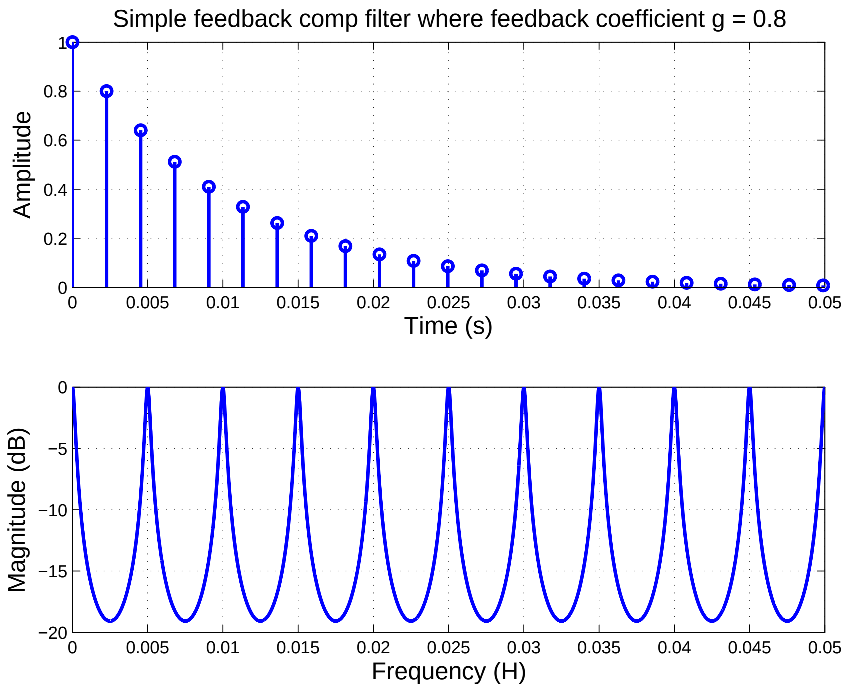


Figure 9: Impulse and magnitude response of a comb filter with feedback g = 0.8.

- The spacing between the maxima of the "teeth" is equal to the natural frequency f_0.

Effect of the Feedback coefficient g

- Coefficient g is the *depth* parameter, where values closer to 1 yield more extreme maxima and minima.

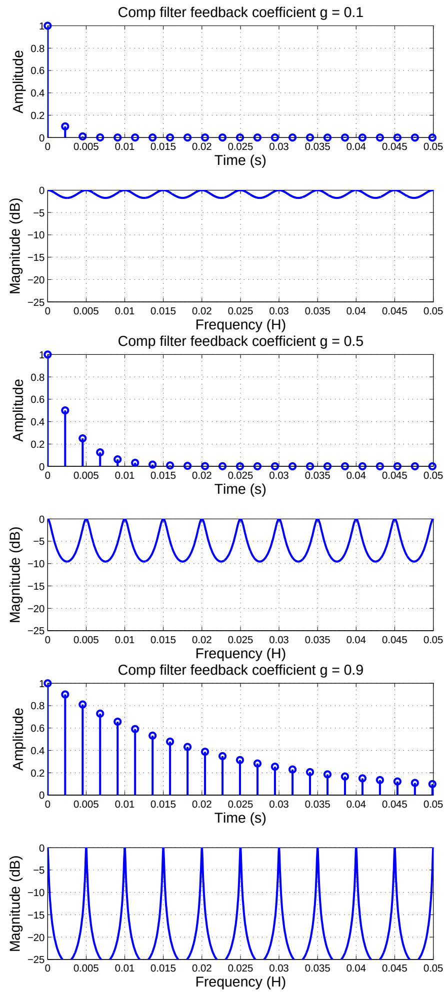


Figure 10: Impulse and Magnitude Response with increasing feedback coefficient.

Feedback Comb Filter Decay Rate

- The response decays exponentially as determined by the loop time and gain factor g (values near 1 yield longer decay times).

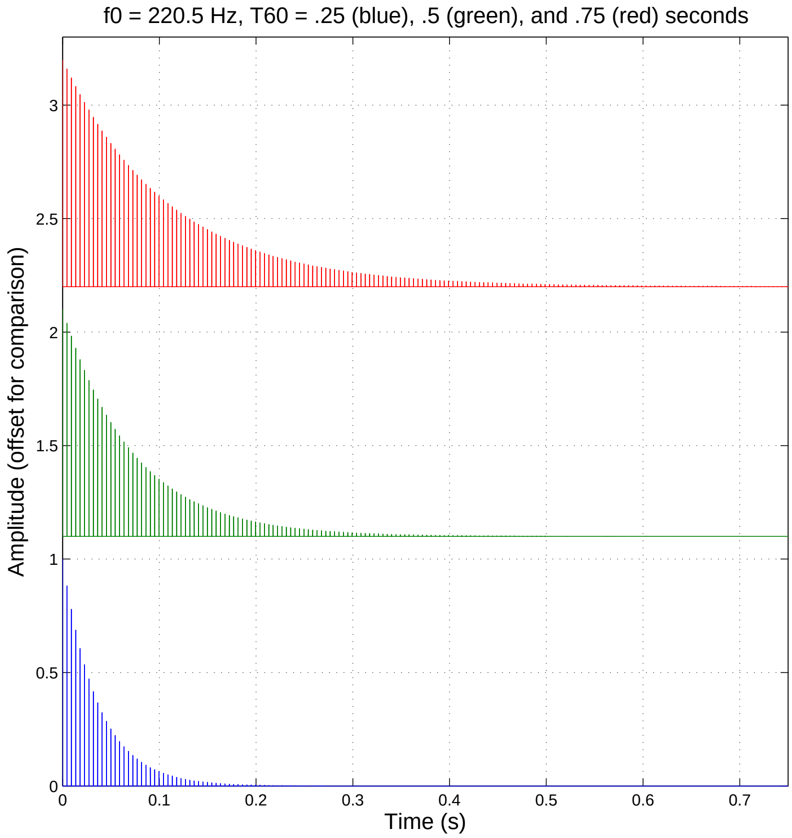


Figure 11: Comb filter impulse responses with a changing the decay rate.

Obtaining a desired T_{60}

- The T_{60} is the time to decay to an inaudible level of -60 dB or by 0.001 on a linear scale.
- Given a loop time of M samples (frequency f_0) and a desired T_{60}, what should be the value of g?
- If the loop has a delay of M samples, **the number of trips through the loop** after n samples, or after t seconds is
$$\frac{n}{M} = \frac{tf_s}{M} = tf_0,$$
where f_0 is the fundamental frequency of the loop.
- Attenuation at time t is given by
$$\alpha(t) = g^{tf_0}.$$
- At time $t = T_{60}$, the attenuation is 0.001,
$$\alpha(T_{60}) = g^{T_{60}f_0} = g^{T_{60}f_s/M} = 0.001,$$
and solving for g yields
$$g = 0.001^{M/(f_s T_{60})}.$$

General Comb Filter

- Combining both the feedforward and feedback comb filter yields the general comb filter, given by the difference equation

$$y(n) = x(n) + g_1 x(n - M_1) - g_2 y(n - M_2)$$

where g_1 and g_2 are the feedforward and feedback coefficients, respectively.

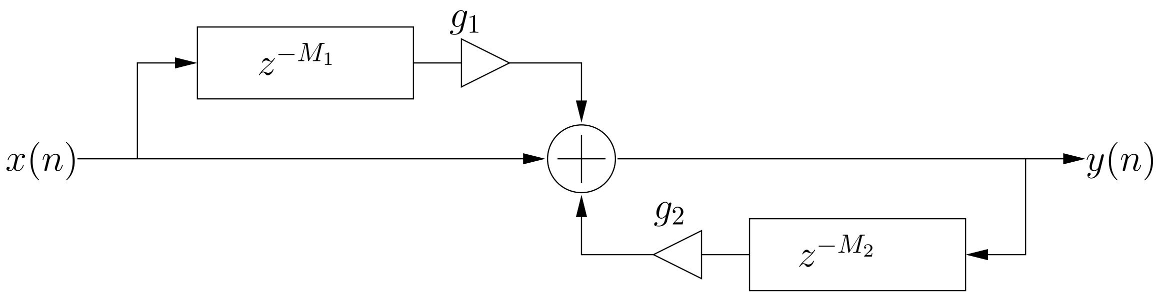


Figure 12: Signal flow diagram for digital comb filters.

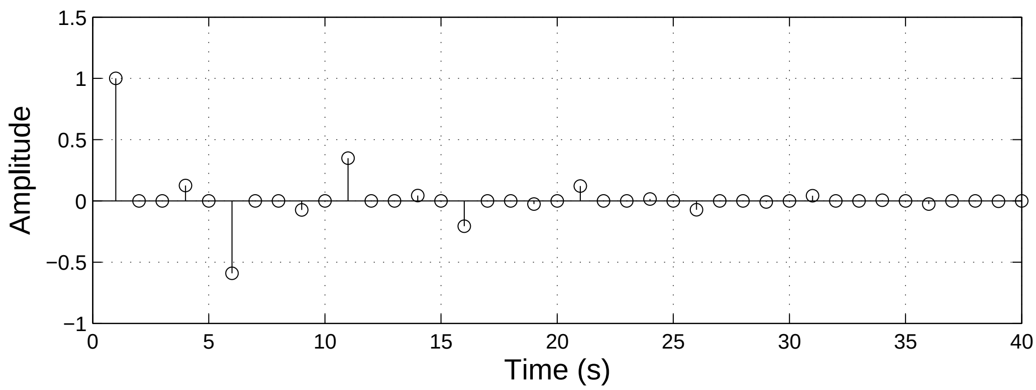


Figure 13: Comb Filter Impulse Response.

A very simple string model

- A very simple string model can be implemented using a single delay line and our simple first-order low pass filter $H(z)$ to model frequency-dependent loss.

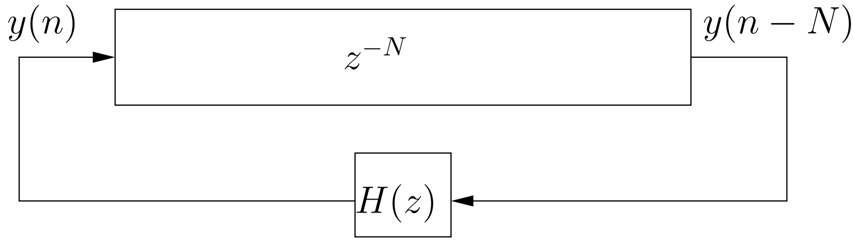


Figure 14: A very simple model of a rigidly terminated string.

- Though losses are distributed along the length of the string, in an LTI system they may be *lumped* to a single observation point and approximated with $H(z)$.
- Different quality string sounds can be created by changing this filter.
- This model may be interpreted as a **feedback comb filter** with lowpassed feedback or a simplified **digital waveguide** model.
- How is this model excited? How is the string *plucked*?

Karplus-Strong Pluck String

- When the delay-line initial conditions consist of *white noise*, the algorithm is known as the **Karplus-Strong** algorithm.
- White noise is a sequence of *uncorrelated* random values. It can be generated in Matlab as follows:

```
N = ...;          % length of vector

y = randn(1, N); % N samples of Gaussian white noise
                 % with zero mean and unit variance

x = rand(1, N);  % N samples of white noise,
                 % uniform between 0 and 1

xn = 2*(x-0.5);  % uniform between -1 and 1
```

- Filling the delay line with white noise is akin to plucking the string with a random initial displacement—a very energetic excitation.
- What are the control parameters of this model?

Controlling Karplus-Strong

- Controlling Dynamics:
  - Limit the range of random numbers—change the Matlab line

    ```
    xn = 2*(x-0.5); % uniform between -1 and 1
    ```

  - Filter the white noise serving as the initial conditions. The cut-off frequency of the filter will control the effective dynamic level (since acoustic instruments are usually brighter at louder dynamic levels).
- Sounding frequency (pitch)
  - Change the delay line length, where

    $$f_0 = f_s/(N + 1/2).$$

  - The $1/2$ term in the denominator is due to the low-pass filter's phase delay of 1/2 sample.
  - Notice that the delay-line length is of an integer size. This limits the resolution of possible sounding frequencies.

Limits of integer-length delay lines

- At low frequencies (large N), this is less of a problem, but becomes increasingly problematic at higher frequencies when delay-line lengths are small and a single sample delay can make a bigger difference.

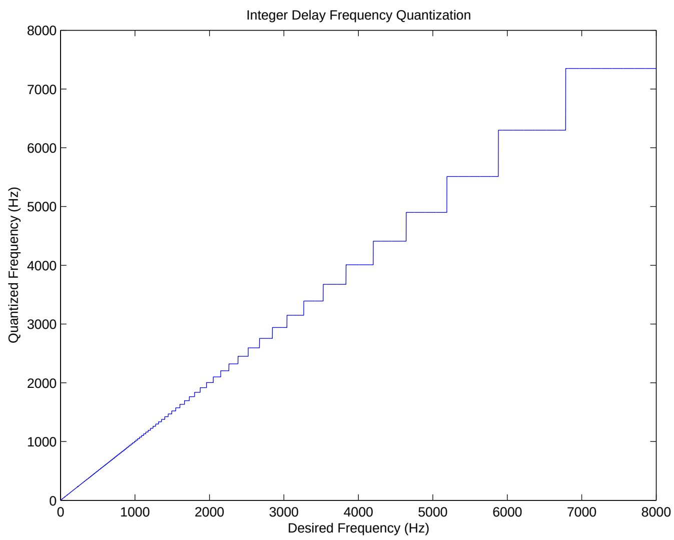


Figure 15: As the desired frequency gets higher, it is quantized to fewer possible values; there is a single frequency value for all desired frequencies between 7000 and 8000 Hz.

- Example: at $f_s = 44100$,
  - to obtain a frequency of 882 Hz, a delay of $f_s/882 = 50$ samples is required;
  - the next highest possible frequency with an integer number of samples is $f_s/49 = 900$ Hz.

Frequency-dependent decay rate

- Another (control) problem with KS is that, because of the low-pass filter in the feedback loop, the decay rate is dependent on frequency.

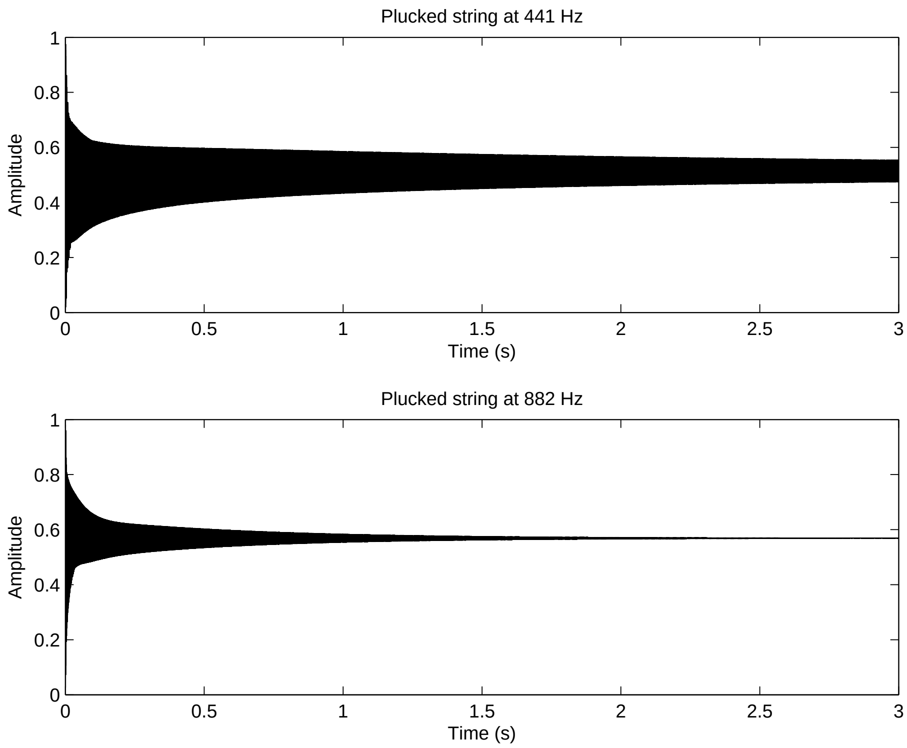


Figure 16: Decay rate is faster at higher frequencies.

- This behaviour is generally desired, as it is a characteristic of acoustic systems, but it is too extreme in the KS.

Selected Smith and Jaffe Extensions

- In a paper by Smith and Jaffe (Computer Music Journal, Summer 1983) extensions to the Karplus-Strong are developed in a musical context:
  - Tuning (fractional delay) using allpass filters as an alternative to linear interpolation
  - Decay rate shortening and stretching
  - Dynamics
  - Plucking position
  - Rests at the ends of notes (i.e. turning off the algorithm without hearing a click)
  - Glissandi and Slurs
  - Sympathetic String Simulation
- Find paper here

Tuning

- For large N (low pitches) the difference between N and $N+1$ is slight, but becomes increasingly noticeable for small N (high pitches).
- Recall, the fundamental frequency (which is inversely proportional to the period) is given by

$$f_1 \triangleq \frac{1}{(N+1/2)T_s} = \frac{f_s}{N+1/2}$$

which may be expressed more generally in terms of phase delay of our feedback filter:

$$f_1 = \frac{f_s}{N + P_a(f_1)}.$$

- We need to introduce a filter into the feedback loop that can contribute a small delay without alterning the loop gain.
- What kind of filters can introduce a frequency-dependent delay without having an effect on gain?

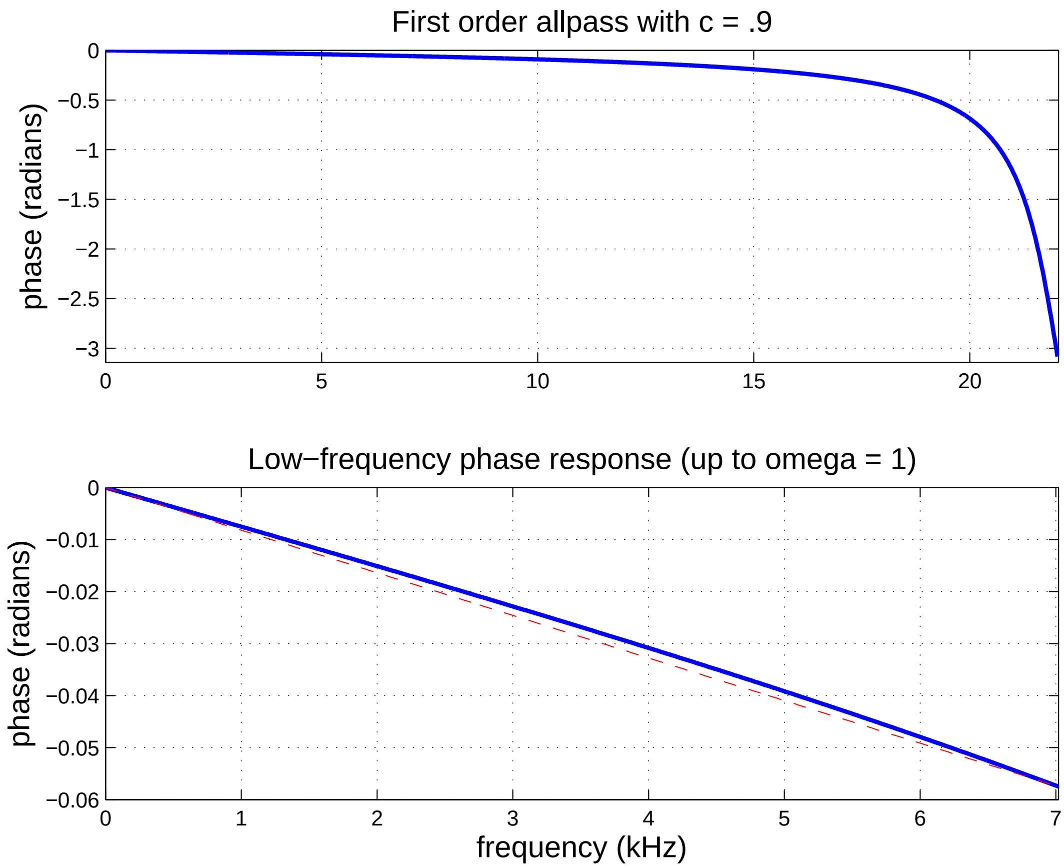
First-Order Allpass Filter

- The first-order allpass filter has difference equation

$$y(n) = Cx(n) + x(n-1) - Cy(n-1),$$

where $|C| < 1$ for stability.

- The phase delay, unlike the 2-point averager, is dependent on frequency.



Phase delay of First-Order Allpass Filter

- The *low-frequency* phase delay may be approximated by
$$P_c(f) \approx \frac{1 - C}{1 + C}.$$
- The filter coefficient C may be solved as a function of the desired phase delay $P_c(f)$.
$$C \approx \frac{1 - P_c(f)}{1 + P_c(f)}.$$
- Notice if $P_c = 0$ then $C = 1$:
  - this produces a pole-zero cancellation on the unit circle which can cause an unstable filter due to round-off errors!
  - thus, the one-sample delay control is shifted to
$$\epsilon \leq P_c \leq (1 + \epsilon).$$

Setting the Allpass Phase Delay to a Desired Frequency

- A fundamental frequency f_1 has a corresponding period of

$$P_1 = f_s/f_1 \text{ samples.}$$

- The model should then have a phase delay of

$$N + P_a(f_1) + P_c(f_1) = P_1 \text{ samples.}$$

where $P_a(f_1) = 1/2$ for the two-point averager.

- The delay line length N becomes

$$N \triangleq \text{Floor}(P_1 - P_a(f_1) - \epsilon),$$

where ϵ is a number much less than 1, that was used to shift $P_c(f_1)$'s one-sampe delay range above 0 to 1.

- The fractional phase delay (in samples) for the allpass interpolator becomes

$$P_c(f_1) \triangleq P_1 - N - P_a(f_1).$$

Attenuation of "Harmonics"

- On each pass through the delay-line loop, a partial at frequency f is subject to an attenuation equal to the loop amplitude response $|H(\omega T)|$.
- The frequency response $H(\omega T)$ of the simple lowpass filter may be found by testing with a complex sinusoid $x(n) = e^{\omega nT}$:

$$\begin{aligned} y(n) &= x(n) + x(n-1) \\ &= e^{j\omega nT} + e^{j\omega(n-1)T} \\ &= e^{j\omega nT} + e^{j\omega nT}e^{-j\omega T} \\ &= (1 + e^{-j\omega T})e^{\omega nT} \\ &= (1 + e^{-j\omega T})x(n), \end{aligned}$$

  where $H(e^{j\omega T}) = (1 + e^{-j\omega T})$.
- The gain of the filter is given by

$$\begin{aligned} G(\omega) &= |H(e^{j\omega T})| \\ &= |(1 + e^{-j\omega T})| \\ &= |(e^{j\omega T/2} + e^{-j\omega T/2})e^{-j\omega T/2}| \\ &= |2\cos(\omega T/2)e^{-j\omega T/2}| \\ &= 2\cos(\omega T/2) \end{aligned}$$

Loop Attenuation at frequency f_1

- The gain of the low-pass filter at frequency f is
$$G_a(f) = \cos(\pi f T_s).$$
- After M passes through the delay-line loop, a partial at frequency f is subject to attenuation
$$\cos(\pi f T_s)^M.$$
- Since the round-trip time in the loop is $N + 1/2$ samples, the number of trips through the loop after n samples ($n = tf_s$) is given by
$$M = \frac{n}{N + 1/2} = \frac{tf_s}{N + 1/2} = tf_1.$$
- The attenuation factor at time $t = nT_s$ is given by
$$\alpha_f(t) \triangleq \cos(\pi f T_s)^{tf_1}.$$
- That is, a partial or harmonic of frequency f, having an initial amplitude of A at time 0, will have amplitude $A\alpha_f(t)$ at time t seconds.

Solving for corresponding time constant

- The *time constant* τ is the time to decay by $1/e$.
- To solve for τ_f, the time constant for frequency f,

$$
\begin{aligned}
\alpha_f(t) &= e^{-t/\tau_f} \\
\ln \alpha_f(t) &= -\frac{t}{\tau_f} \quad \text{(take log of both sides)} \\
\tau_f &= -\frac{t}{\ln \alpha_f(t)} \\
&= -\frac{t}{t f_1 \ln\left(\cos(\pi f T_s)\right)} \text{ seconds} \\
&= -\frac{1}{f_1 \ln(\cos(\pi f T_s))} \text{ seconds} \\
&= -\frac{(N + 1/2)\, T_s}{\ln(\cos(\pi f T_s))} \text{ seconds.}
\end{aligned}
$$

Attenuation and Decay with General Loss Filter

- The filter accounting for frequency-dependent loss may be other than a two-point averager.
- A general presentation of the attenuation factor for the k^{th} harmonics is given by

$$\alpha_k(t) = G_a(f_k)^{\frac{tf_s}{N+P_a(f_k)}},$$

and the decay for each harmonic becomes

$$\tau_k = -\frac{N + P_a(f_k)}{f_s \ln G_a(2\pi f_k T_s)},$$

where $G_a(f_k)$ and $P_a(f_k)$ are the gain and phase delays, respectively, of the filter used.

Relating to the T_{60}

- For audio/music, it is more useful to define the time constant as the time it takes to decay -60dB, or 0.001 times the initial value.
- The attenuation factor at time $t = T_{60}(f)$ is given by

$$\alpha_f(T_{60}(f)) = 0.001.$$

- Conversion from τ to T_{60} is done by

$$\begin{aligned} 0.001 &= e^{-T_{60}/\tau} \\ \ln(0.001) &= -\frac{T_{60}}{\tau} \\ T_{60} &= -\ln(0.001)\tau \\ &\approx 6.91\tau \end{aligned}$$

Decay of non-harmonics

- The previous analysis describes the attenuation due to "propagation" around the loop.
- Sinusoids that do not "fit" into the loop, are quicky destroyed by self interference.

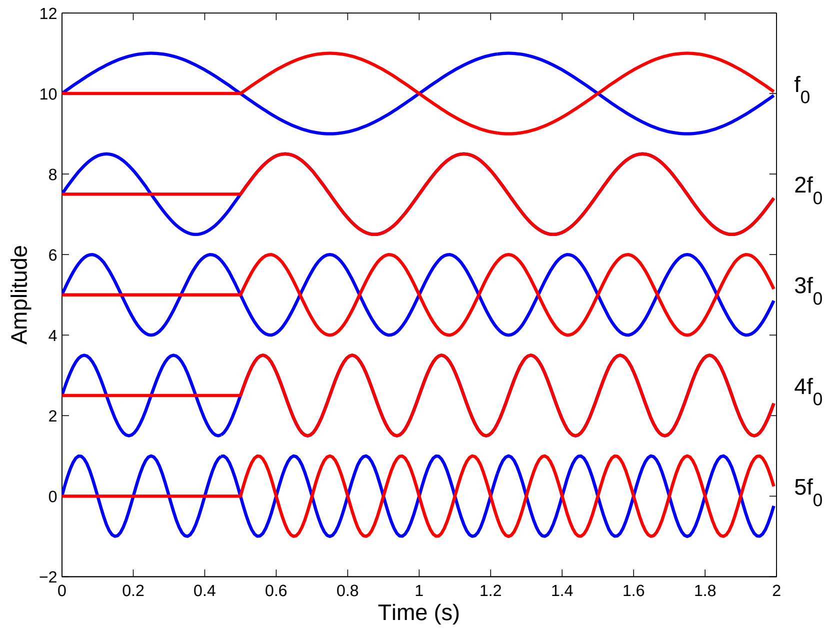


Figure 17: Destructive interference occurs at odd harmonics of the fundamental frequency.

- Though the loop is initialized with random numbers, after a very short time the primary frequencies remaining in the loop are those with an integer number of periods in $N + 1/2$ samples.

Decay-time Shortening

- To shorten the decay time, a loss factor of ρ can be introduced in the feedback loop, yielding

$$y(n) = x(n) + \rho \frac{y(n-N) + y(n-(N+1))}{2}$$

- The amplitude envelope of a sinusoid at frequency f, is now proportional to

$$\alpha_f(t, \rho) = |\rho \cos(\pi f T_s)|^{t f_1} = |\rho|^{t f_1} \alpha_f(t).$$

and the decay-time constant for the fundamental frequency becomes

$$\tau_1(\rho) = -\frac{1}{f_1 \ln |\rho \cos(\pi f_1 T_s)|}.$$

- Note that ρ cannot be used to lengthen the decay time, since the amplitude at 0 Hz would increase exponentially.
- $|\rho| \leq 1$ if the string is to be stable.
- ρ is used to shorten the low-pitch notes.

Setting ρ for a desired T_{60}

- For a desired T_{60}, determine the corresponding time constant τ
$$\tau \approx \frac{t_{60}}{6.91}.$$
- Use this value in solving for ρ,
$$\begin{aligned}
\tau &= -\frac{1}{f_1 \ln|\rho\cos(\pi f_1 T)|} \\
\ln|\rho\cos(\pi f_1 T)| &= -\frac{1}{f_1\tau} \\
|\rho\cos(\pi f_1 T)| &= e^{-\frac{1}{f_1\tau}} \\
|\rho| &= \frac{e^{-1/(f_1\tau)}}{|\cos(\pi f_1 T)|}
\end{aligned}$$

Decay Stretching

- To stretch the decay, and reduce the lowpass effect at high frequencies, the simple lowpass can be replaced with a two-point weigthed average

$$y(n) = (1 - S)x(n) + Sx(n - 1),$$

where S, the stretching factor, is between 0 and 1.

- For stability, S can't be greater than 1.
- When $S = 1/2$, the filter reduces the the previous two-point averager.
- When $S = 0$ or 1, the frequency-dependent term (delay) disappears, and the gain response is unity for all f.
- At intermediate values, $0 < S < 1$, the note duration is finite, with a minimum for $S = 1/2$.
- The resulting decay time is then a function of loss factor ρ and stretch factor S.

Effect of Decay Stretching on Tuning

- Changing S changes the effective loop length as a function of frequency since it changes the phase delay of the overall loop.
  - we must therefore compute $P_a(f_1)$ when using the allpass filter fractional delay to tune to the desired frequency.
- As shown in the paper by Smith and Jaffe, for low frequencies relative to the sampling rate, we may use the approximation

$$P_a(f, S) \approx S, \quad 0 \leq S \leq 1.$$

- See this in Matlab:

```
S = .6;
[H, omega] = freqz([1-S S], 1);

% start at index 2 to avoid division by 0
mean(angle(H(2:end))./omega(2:end));
```

- When $S = 1/2$, we have the basic string algorithm.

Time constant as a function of S

- Recall the gain of the simple 2-point averager is

$$G(\omega) = |1 + e^{-j\omega T}|$$

- The gain of the weighted 2-point averager is

$$\begin{aligned} G(S;\omega) &= |(1-S) + Se^{-j\omega T}| \\ &= |(1-S) + S[\cos(\omega T) + j\sin(\omega T))]| \\ &= \sqrt{[(1-S) + S(\cos(\omega T)]^2 + S^2\sin^2(\omega T))} \\ &= \sqrt{(1-S)^2 + 2S(1-S)\cos(\omega T) + S^2}. \end{aligned}$$

- The time constant is

$$\tau = -\frac{1}{f_0 \ln(G(S;\omega))} = -\frac{N+S}{f_s \ln(G(S;\omega))}$$