Time Varying Delay Effects
Julius Smith and Nelson Lee
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305
June 5, 2008

It is often necessary for a delay line to vary in length. (Consider simulating a sound ray when either the source or listener is moving.) In this case, separate read and write pointers are normally used.

Variable Delay Lines

Let $A$ denote an array of length $N$. Then we can implement an $M$-sample variable delay line in the C programming language as shown in Fig. 3. We require, of course, $M \leq N$.

The $M$-sample variable delay line using separate read- and write-pointers:

```c
static double A[N];
static double *rptr = A; // read ptr
static double *wptr = A; // write ptr

double setdelay(int M) {
    rptr = wptr - M;
    while (rptr < A) { rptr += N }
}

double delayline(double x)
{
    double y;
    A[wptr++] = x;
    y = A[rptr++];
    if ((wptr-A) >= N) { wptr -= N }
    if ((rptr-A) >= N) { rptr -= N }
    return y;
}
```

The Synthesis Tool Kit, Version 4 [?1] contains the C++ class "Delay" which implements this type of variable (but non-interpolating) delay line. There are additional subclasses which provide interpolating reads by various methods. In particular, the class DelayL implements continuously variable delay lengths using linear interpolation. The code listing in Fig. 3 can be modified to use linear interpolation by replacing the line

$$y = A[rptr++]$$

with

$$\text{long rpi = (long)floor(rptr);}$$
$$\text{double a = rptr - (double)rpi;}$$
$$\text{y = a * A[rpi] + (1-a) * A[rpi+1];}$$
$$\text{rptr += 1;}$$

To implement a continuously varying delay, we add a "delay growth parameter" $g$ to the delayline function in Fig. 3, and change the line

$$rptr += 1; // pointer update$$

above to

$$rptr += 1 - g; // pointer update$$

When $g$ is 0, we have a fixed delay line. When $g > 0$, the delay grows $g$ samples per second, which we may also interpret as seconds per second, i.e., $D_t = g$. In [?1], this will be applied to simulation of the Doppler effect.

Delay-Line Interpolation

When delay lines need to vary smoothly over time, some form of interpolation between samples is usually required to avoid "zipper noise" in the output signal as the delay length changes. There is a hefty literature on "fractional delay" in discrete-time systems, and the survey in [?2] is highly recommended.

This section will describe the most commonly used cases. Linear interpolation is perhaps most commonly used because it is very straightforward and inexpensive, and because it sounds very good when the signal bandwidth is small compared with half the sampling rate. For a delay line in a nearly lossless feedback loop, such as in a vibrating string simulation, allpass interpolation is usually a better choice since it costs the same as linear interpolation in the first-order case and has no gain distortion. (Feedback loops can be very sensitive to gain distortions.) Finally, in Appendix ??, some higher order interpolation schemes are outlined.

Linear Interpolation

Linear interpolation works by effectively drawing a straight line between two neighboring samples and returning the appropriate point along that line.

More specifically, let $\eta$ be a number between 0 and 1 which represents how far we want to interpolate a signal $y$ between time $n$ and time $n + 1$. Then we can define the linearly interpolated value
\[ \hat{y}(n + \eta) \] as follows:
\[ \hat{y}(n + \eta) = (1 - \eta) \cdot y(n) + \eta \cdot y(n + 1) \]

For \( \eta = 0 \), we get exactly \( \hat{y}(n) = y(n) \), and for \( \eta = 1 \), we get exactly \( \hat{y}(n + 1) = y(n + 1) \). In between, the interpolation error \( |\hat{y}(n + \eta) - y(n + \eta)| \) is nonzero, except when \( y(t) \) is a linear function between \( y(n) \) and \( y(n + 1) \).

Note that by factoring out \( \eta \), we can obtain a one-multiply form,
\[ \hat{y}(n + \eta) = y(n) + \eta \cdot [y(n + 1) - y(n)]. \]

Thus, the computational complexity of linear interpolation is one multiply and two additions per sample of output.

A linearly interpolated delay line is depicted in Fig. ??.

\[ y(n) \rightarrow \begin{array}{c} \text{M samples delay} \\ 1 - \eta \\ \eta \end{array} \rightarrow \frac{1}{z^{1-\eta}} \rightarrow \hat{y}(n - M - \eta) \]

Linearly interpolated delay line.

The C++ class implementing a linearly interpolated delay line in the Synthesis Tool Kit (STK) is called DelayL.

The frequency response of linear interpolation for fixed fractional delay (\( \eta \) fixed in Fig. ??) is shown in Fig. ???. From inspection of Fig. ??, we see that linear interpolation is a one-zero FIR filter.

When used to provide a fixed fractional delay, the filter is linear and time-invariant (LTI). When the delay provided changes over time, it is a linear time-varying filter.

The difference equation is
\[ \hat{x}(n - \Delta) \triangleq y(n) = \eta \cdot x(n) + x(n - 1) - \eta \cdot y(n - 1) \]
\[ = \eta \cdot x(n) - \eta \cdot y(n - 1) + x(n - 1). \]

Thus, like linear interpolation, first-order allpass interpolation requires only one multiply and two additions per sample of output.

The frequency response is
\[ H(z) = \frac{\eta + z^{-1}}{1 + \eta z^{-1}}. \]

At low frequencies \( (z \rightarrow 1) \), the delay becomes
\[ \Delta \approx \frac{1 - \eta}{1 + \eta}. \]

Figure ?? shows the phase delay of the first-order digital allpass filter for a variety of desired delays at dc. Since the amplitude response of any allpass is 1 at all frequencies, there is no need to plot it.

Intuitively, ramping the coefficients of the allpass gradually "grows" or "hides" one sample of delay. This tells us how to handle resets when crossing sample boundaries.

The difference equation is
\[ x(n - \Delta) \triangleq y(n) = \eta \cdot x(n) + x(n - 1) - \eta \cdot y(n - 1) \]
\[ = \eta \cdot x(n) - \eta \cdot y(n - 1) + x(n - 1). \]

Thus, like linear interpolation, first-order allpass interpolation requires only one multiply and two additions per sample of output.

The frequency response is
\[ H(z) = \frac{\eta + z^{-1}}{1 + \eta z^{-1}}. \]

At low frequencies \( (z \rightarrow 1) \), the delay becomes
\[ \Delta \approx \frac{1 - \eta}{1 + \eta}. \]

Figure ?? shows the phase delay of the first-order digital allpass filter for a variety of desired delays at dc. Since the amplitude response of any allpass is 1 at all frequencies, there is no need to plot it.

The first-order allpass interpolator is generally controlled by setting
its dc delay to the desired delay. Thus, for a given desired delay $\Delta$, the allpass coefficient is (from Eq. (2))

$$\eta \approx \frac{1 - \Delta}{1 + \Delta}$$

From Eq. (1), we see that the allpass filter’s pole is at $z = -\eta$, and its zero is at $z = -1/\eta$. A pole-zero diagram for $\Delta = 0.1$ is given in Fig. ???. Thus, zero delay is provided by means of a pole-zero cancellation! Due to inevitable round-off errors, pole-zero cancellations are to be avoided in practice. For this reason and others (discussed below), allpass interpolation is best used to provide a delay range lying wholly above zero, e.g.,

$\Delta \in [0.1, 1.1] \iff \eta \in [-0.05, 0.82]$
Since the time constant of decay, in samples, of the impulse response of a pole of radius $R$ is approximately

$$\tau_T \approx \frac{1}{1 - R},$$

and since a 60-dB decay occurs in about 7 time constants ("$t_{60}$") \cite{p. 38}, we can limit the pole of the allpass filter to achieve any prescribed specification on maximum impulse-response duration.

For example, suppose 100 ms is chosen as the maximum $t_{60}$ allowed at a sampling rate of $f_s = 10,000$. Then applying the above constraints yields $\eta \leq 0.993$, corresponding to the allowed delay range $[0.00351, 1.00351]$.

1 Large Delay Changes

When implementing large delay length changes (by many samples), a useful implementation is to cross-fade from the initial delay line configuration to the new configuration:

- Computational requirements are doubled during the cross-fade.
- The cross-fade should occur over a time interval long enough to yield a smooth result.
- The new delay interpolation filter, if any, may be initialized in advance of the cross-fade, for maximum smoothness. Thus, if the transient response of the interpolation filter is $N$ samples, the new delay-line + interpolation filter can be "warmed up" (executed) for $N$ time steps before beginning the cross-fade. If the cross-fade time is long compared with the interpolation filter duration, "pre-warming" is not necessary.
- This is not a true "morph" from one delay length to another since we do not pass through the intermediate delay lengths. However, it avoids a potentially undesirable Doppler effect.
- A single delay line can be shared such that the cross-fade occurs from one read-pointer (plus associated filtering) to another.

Specific Time-Varying Delay Effects

Time varying delay lines are fundamental building blocks for delay effects, synthesis algorithms, and computational acoustic models of musical instruments.

In the category of delay effects, variable delay lines are used for

- Phasing
- Flanging
- Chorus
- Leslie
- Reverb

(While reverberators need not be time varying, nowadays they typically are \cite{,}.)

In digital waveguide synthesis, variable delay lines are used for

- Vibrating strings (guitars, violins, . . .)
- Woodwind bores
- Horns
- Tonal percussion (rods, membranes)

The following sections will elaborate on the use of variable delay lines for effects. Their use in digital waveguide models will be deferred to a later section.

Flanging

Flanging is a delay effect that has been available in recording studios since at least the 1960s. Surprisingly little literature exists, although there is some \cite{, , , , , ,}.

According to lore \cite{,}, the term "flanging" arose from the way the effect was originally achieved by two tape machines set up to play the same tape in unison, with their outputs are added together (mixed equally), as shown in Fig. 1. To achieve the flanging effect, the flange of one of the supply reels is touched lightly to make it play a littler slower. This causes a delay to develop between two tape machines. The flange is released, and the flange of the other supply reel is touched lightly to slow it down. This causes the delay to gradually disappear and then begin to grow again in the opposite direction. The delay is kept below the threshold of echo perception (e.g., only a few milliseconds in each direction). The process is repeated as desired, pressing the flange of each supply reel in alternation. The flanging effect has been described as a kind of "whoosh" passing by through the sound \cite{}. The effect is also compared to the sound of a jet passing overhead, in which the direct signal and ground reflection arrive at a varying relative delay \cite{}. If flanging is done rapidly enough, an audible Doppler shift is introduced which approximates the "Leslie" effect commonly used for organs (see §??).

Flanging is modeled quite accurately as a feedforward comb filter, as discussed in §??, in which the delay $M$ is varied over time. Figure \cite{} depicts such a model. The input-output relation for a basic flanger

\[\text{http://www.harmony-central.com/Effects/Articles/Flanging/}.\]
over time, it is clearly necessary to use an interpolated delay line

\[ M \]

to provide non-integer values of "LFO" waveform. We may say that the delay length is modulated is typically modulated by a low-frequency oscillator (LFO). Since \( M(n) \) must vary smoothly over time, it is clearly necessary to use an interpolated delay line to provide non-integer values of \( M \) in a smooth fashion.

\[ g(n) = x(n) + gx[n - M(n)] \quad (3) \]

where \( x(n) \) is the input signal amplitude at time \( n = 0, 1, 2, \ldots \), \( y(n) \) is the output at time \( n \), \( g \) is the "depth" of the flanging effect, and \( M(n) \) is the length of the delay-line at time \( n \). The delay length \( M(n) \) is typically varied according to a triangular or sinusoidal waveform. We may say that the delay length is modulated by an "LFO" (Low-Frequency Oscillator). Since \( M(n) \) must vary smoothly over time, it is clearly necessary to use an interpolated delay line to provide non-integer values of \( M \) in a smooth fashion.

\[ M(n) \text{ samples of delay} \]

The basic flanger effect.

As shown in Fig. ?? on page ??, the frequency response of Eq. (3) has a "comb" shaped structure. For \( g > 0 \), there are \( M \) peaks in the frequency response, centered about frequencies

\[ \omega_k^{(p)} = k \frac{2\pi}{M} \quad k = 0, 1, 2, \ldots, M - 1. \]

For \( g = 1 \), the peaks are maximally pronounced, with \( M \) notches occurring between them at frequencies \( \omega_k^{(p)} = \omega_k^{(1)} + \pi/M \). As the delay length \( M \) is varied over time, these "comb teeth" squeeze in and out like the pleats of an accordion. As a result, the spectrum of any sound passing through the flanger is "massaged" by a variable comb filter.

As is evident from Fig. ?? on page ??, at any given time there are \( M(n) \) notches in the flanger’s amplitude response (counting positive- and negative-frequency notches separately). The notches are thus spaced at intervals of \( f_s/M \) Hz, where \( f_s \) denotes the sampling rate. In particular, the notch spacing is inversely proportional to delay-line length.

The time variation of the delay-line length \( M(n) \) results in a "sweeping" of uniformly-spaced notches in the spectrum. The flanging effect is thus created by moving notches in the spectrum. Notch motion is essential for the flanging effect. Static notches provide some coloration to the sound, but an isolated notch may be inaudible [?]. Since the steady-state sound field inside an undamped acoustic tube has a similar set of uniformly spaced notches (except at the ends), a static row of notches tends to sound like being inside an acoustic tube.

Flanger Speed and Excursion

As mentioned above, the delay-line length \( M(n) \) in a digital flanger is typically modulated by a low-frequency oscillator (LFO). The oscillator waveform is usually triangular, sinusoidal, or exponential (triangular on a log-frequency scale). In the sinusoidal case, we have

\[ M(n) = M_0 \cdot \left[ 1 + A \sin(2\pi f n T) \right] \]

where \( f \) is the "speed" (or "rate") of the flanger in cycles per second, \( A \) is the "excursion" or "sweep" (maximum delay swing) which is often not brought out as a user-controllable parameter, and \( M_0 \) is the average delay length controlling the average notch density (also not normally brought out as a user-controllable parameter).

Flanger Depth Control

To obtain a maximum effect, the depth control, \( g \) in Fig. ?? should be set to 1. A depth of \( g = 0 \) gives no effect.

Flanger Inverted Mode

A different type of maximum depth is obtained for \( g = -1 \). In this case, the peaks and notches of the \( g = 1 \) case trade places. In practice, the depth control \( g \) is usually constrained to the interval \([0, 1]\), and a sign inversion for \( g \) is controlled separately using a "phase inversion" switch.

Inverted mode, unless the delay \( M \) is very large, the bass response will be weak, since the first notch is at dc. This case usually sounds high-pass filtered relative to the "in-phase" case (\( g > 0 \)).

As the notch spacing grows very large (\( M \) shrinks), the amplitude response approaches that of a first-order difference

\[ g(n) = x(n) - x(n - 1), \]

which approximates a differentiator

\[ g(t) = \frac{d}{dt} x(t). \]

An ideal differentiator eliminates dc and provides a progressive high-frequency boost rising 6 dB per octave (specifically, the amplitude response is \(|H(\omega)| = |\omega|\).

Flanger Feedback Control

Many modern commercial flangers have a control knob labeled "feedback" or "regen." This control sets the level of feedback from the output to the input of the delay line, thereby creating a feedback comb filter in addition to the feedforward comb filter, in the same manner as in the creation of a Schroeder allpass filter.

More generally, outputs of any subset of the allpass sections can be fed back to the input (in small amounts) to produce different sounding effects [?].

Summary of Flanging

In view of the above, we may define a flanger in general as any filter which modulates the frequencies of a set of uniformly spaced notches and/or peaks in the frequency response. The main parameters are

- Depth \( g \in [0, 1] \) — controlling notch depth
• Speed $f$ — speed of notch movement
• Phase — switch to subtract instead of adding the direct signal with the delayed signal

Possible additional parameters include
• Average Delay $M_0$
• Excursion or Sweep $A$ — amount by which the delay-line grows or shrinks
• Feedback or Regeneration $a,M \in (-1, 1)$ — feedback coefficient from output to input

Note that flanging provides only uniformly spaced notches. This can be considered non-ideal for several reasons. First, the ear processes sound over a frequency scale that is more nearly logarithmic than linear [2]. Therefore, exponentially spaced notches (uniformly spaced on a log frequency scale) should sound more uniform perceptually.

Secondly, the uniform peaks and notches of the flanger can impose a discernible "resonant pitch" on the program material, giving the impression of being inside a resonant tube. Third, it is possible for a periodic tone to be completely annihilated by harmonically spaced notches if the harmonics of the tone are unlucky enough to land exactly on a subset of the harmonic notches. In practice, exact alignment is unlikely; however, the signal loudness can be modulated to a possibly undesirable degree as the notches move through alignment with the signal spectrum. For this reason, flangers are best used with noise-like or inharmonic sounds. For harmonic signals, it makes sense to consider methods for creating non-uniform moving notches.

In analog hardware, the first-order allpass transfer function [2, Appendix C, Section 6] is

$$H_s(j\omega) = \frac{s - \omega_0}{s + \omega_0}.$$  \hspace{1cm} (4)

In discrete time, the general first-order allpass has the transfer function

$$H_d(z) = \frac{z^{-1} + g_1}{1 + g_2 z^{-1}}.$$  

We now consider the analog and digital cases, respectively.

Classic Analog Phase Shifters

Setting $s = j\omega$ in Eq. (4) gives the frequency response of the analog-phase shifter transfer function to be

$$H_s(j\omega) = \frac{\omega - \omega_0}{j\omega + \omega_0}.$$  

The phase response is readily found to be

$$
\Theta(s) = \pi - 2 \tan^{-1}\left(\frac{\omega}{\omega_0}\right).
$$

Phasing

The phaser, or phase shifter, is closely related to the flanger in that it also works by sweeping notches through the spectrum of the input signal. While the term phasing is sometimes used synonymously with flanging [2], commercial phase shifters have been observed to implement nonuniformly spaced notches [3]. We will therefore define a phaser as any linear filter which modulates the frequencies of a set of non-uniformly spaced notches, while a flanger will remain any device which modulates uniformly spaced notches.

Phasing with First-Order Allpass Filters

The block diagram of a typical inexpensive phase shifter for guitar players is shown in Fig. 3. It consists of a series chain of first-order allpass filters, each having a single time-varying parameter $g_1(n)$ controlling the pole and zero location over time, plus a feedforward path through gain $g$ which is a fixed depth control.

$$x[n] \rightarrow \underbrace{\text{AP} - \text{AP}- \text{AP}- \text{AP}}_{\text{4 Notches}} \rightarrow y[n].$$

Structure of a phaser based on four first-order allpass filters.

Figure ??a shows the phase responses of four first-order analog allpass filters with $\omega_0$ set to $2\pi(100, 200, 400, 800)$. Figure ??b shows the resulting normalized amplitude response for the phaser, for $g = 1$ (unity feedforward gain). The amplitude response has also been normalized by dividing by 2 so that the maximum gain is 1. Since there is an even number (four) of allpass sections, the gain at dc is $1 + (-1)(-1)(-1)(-1) = 1$. Put another way, the initial phase of each allpass section at dc is $\pi$, so that the total allpass chain phase at dc is $4\pi$. As frequency increases, the phase of the allpass chain decreases. When it comes down to $3\pi$, the net effect is a sign inversion by the allpass chain, and the phase has a notch. There will be another notch when the phase falls down to $\pi$. Thus, four allpass sections give two notches. For each notch in the desired response we must add two new allpass sections.

The maximum response $|H(s)|_{\max}$ may be approximated by a slope-constant line at height $G_{\max} = 20\log_{10}(s)$ from dc to $\omega_0$, followed by an intersecting line with negative slope of $20\log_{10}(s)$ for all higher frequencies. At the break frequency, the true gain is down $3\mathrm{dB}$ from $G_{\max}$. The piecewise-linear approximation is extremely accurate. Such an approximate amplitude response is called a Bode plot. Such plots are covered in any introductory course on control systems design (also called the design of control mechanisms).
where \( \omega \) transformation Eq. (5) gives

From Fig. ??b, we observe that the first notch is near \( f = 100 \) Hz. This happens to be the frequency at which the first allpass pole “breaks,” i.e., \( \omega = \omega_1 \). Since the phase of a first-order allpass section at its break frequency is \( \pi/2 \), the sum of the other three sections must be approximately \( 2\pi + \pi/2 \). Equivalently, since the first section has “given up” \( \pi/2 \) radians of phase at \( \omega = \omega_1 = 2\pi100 \), the other three allpass sections combined have given up \( \pi/2 \) radians as well (with the second section having given up more than the other two).

In practical operation, the break frequencies must change dynamically, usually periodically at some rate.

### Classic Virtual Analog Phase Shifters

To create a virtual analog phaser, following closely the design of typical analog phasers, we must translate each first-order allpass to the digital domain. Working with the transfer function, we must map from \( s \) plane to the \( z \) plane. There are several ways to accomplish this goal [?]. However, in this case, an excellent choice is the bilinear transform (see §??? on page ??), defined by

\[
s \longleftarrow \frac{z - 1}{z + 1}
\]

where \( c \) is chosen to map one particular frequency to exactly where it belongs. In this case, \( c \) can be chosen for each section to map the break frequency of the section to exactly where it belongs on the digital frequency axis. The relation between the analog and digital frequency axes follows immediately from Eq. (5) as

\[
j\omega_c = \frac{e^{j\omega \pi T/2} - 1}{e^{j\omega \pi T/2} + 1} = \frac{e^{j\omega \pi T/2} (e^{-j\omega \pi T/2} - e^{-j\omega \pi T/2})}{e^{j\omega \pi T/2} (e^{-j\omega \pi T/2} + e^{-j\omega \pi T/2})} = \frac{\sin(\omega_c \pi T)}{\cos(\omega_c \pi T/2)} = jct\tan(\omega_c \pi T/2).
\]

This is a good example of when the bilinear transform performs very well.

Thus, given a particular desired break frequency \( \omega_b = \omega_d = \omega_1 \), we can set

\[
c = \omega_1 \cot(\omega_1 \pi T/2)
\]

Recall from Eq. (3) on page ?? that the transfer function of the first-order analog allpass filter is given by

\[
H_a(s) = \frac{s - \omega_b}{s + \omega_b}
\]

where \( \omega_b \) is the break frequency. Applying the general bilinear transformation Eq. (3) gives

\[
H_d(z) = H_a \left( \frac{1 - z^{-1}}{1 + z^{-1}} \right) = \frac{\left( \frac{1 - z^{-1}}{1 + z^{-1}} \right) - \omega_b}{\left( \frac{1 - z^{-1}}{1 + z^{-1}} \right) + \omega_b} = \frac{p_d - z^{-1}}{-p_d z^{-1}}
\]

where we have denoted the pole of the digital allpass by

\[
p_d = \frac{c - \omega_b}{c + \omega_b} = \frac{\cot(\omega_b T/2) - 1}{\cot(\omega_b T/2) + 1} = \frac{1 - \tan(\omega_b T/2)}{1 + \tan(\omega_b T/2)}
\]

Figure ?? shows the digital phaser response curves corresponding to the analog response curves in Fig. ??, While the break frequencies are preserved by construction, the notches have moved slightly, although this is not visible from the plots. An overlay of the total phase of the analog and digital allpass chains is shown in Fig. ??, We see that the phase responses of the analog and digital allpass chains diverge visibly only above 9 kHz. The analog phase response approaches zero in the limit as \( \omega_b \to \infty \), while the digital phase response reaches zero at half the sampling rate, 10 kHz in this case.
The allpass structure proposed in [?], [?], circuit leads to a delay-free loop in the digitized system for an example in which the presence of feedback in the analog sampling rate by 15% or so. See the case of digitizing the Moog V CF.

Critical bands of the notches, but what is the ideal non-uniform spacing? On one hand,

Allpass Phaser Notch Distribution

As mentioned above, it is desirable to avoid exact harmonic spacing of the notches, but what is the ideal non-uniform spacing? One possibility is to space the notches according to the critical bands of hearing, since essentially this gives a uniform notch density with respect to ”place” along the basilar membrane in the ear. There is no need to follow closely the critical-band structure, and many simple functional relationships can be utilized to tune the notches [?]. Due to the immediacy of the relation between notch characteristics and the filter coefficients, the notches can easily be placed under musically meaningful control.

Vibrato Simulation

The term vibrato refers to small, quasi-periodic variations in the pitch of a tone. On a violin, for example, vibrato is produced by wiggling the finger stopping the string on the fingerboard; a violin vibrato frequency can be very slow, or a bit faster than 6 Hz. A typical vibrato depth is on the order of 1 percent (a semitone is $2^{1/12}$ ≈ 0 percent). In the singing voice, vibrato is produced by modulating the tension of the vocal folds. Vibrato is typically accompanied by tremolo, which is amplitude modulation at the same frequency as the vibrato which causes it. For example, in the violin, the frequency-modulations of the string vibrations are translated into amplitude modulations by the complex variations in the frequency response of the violin body.

To apply vibrato to a sound, it is necessary to apply a quasi-periodic frequency shift. This can be accomplish using a modulated delay line. This works because a time-varying delay line induces a simulated Doppler shift on the signal within it.
The Doppler effect causes the pitch of a sound source to appear to rise or fall due to motion of the source and/or listener relative to each other. You have probably heard the pitch of a horn drop lower as it passes by (e.g., from a moving train). As a pitched sound-source moves toward you, the pitch you hear is raised; as it moves away from you, the pitch is lowered. The Doppler effect has been used to enhance the realism of simulated moving sound sources for compositional purposes [?], and it is an important component of the “Leslie effect” (described in §2). As derived in elementary physics texts, the Doppler shift is given by

\[
\omega_i = \omega_0 \frac{1 + \frac{v_i}{c}}{1 - \frac{v_i}{c}}
\]

where \(\omega_0\) is the radian frequency emitted by the source at rest, \(\omega_i\) is the frequency received by the listener, \(v_i\) denotes the speed of the listener relative to the propagation medium in the direction of the source, \(v_s\) denotes the speed of the source relative to the propagation medium in the direction of the source, and \(c\) denotes sound speed. Note that all quantities in this formula are scalars.

Vector Formulation

Denote the sound-source velocity by \(\mathbf{v}_s(t)\) where \(t\) is time. Similarly, let \(\mathbf{v}_l(t)\) denote the velocity of the listener, if any. The position of source and listener are denoted \(\mathbf{x}_s(t)\) and \(\mathbf{x}_l(t)\), respectively, where

\[
\mathbf{x}(t) = (x_1, x_2, x_3)^T
\]

is 3D position. We have velocity related to position by

\[
\mathbf{v}_s = \frac{d}{dt}\mathbf{x}_s(t) \quad \mathbf{v}_l = \frac{d}{dt}\mathbf{x}_l(t).
\]

Consider Doppler shift from a physical point of view. The air can be considered as analogous to a magnetic tape which moves from source to listener at speed \(c\). The source is analogous to the write-head of a tape recorder, and the listener corresponds to the read-head. When the source and listener are fixed, the listener receives what the source records. When either moves, a Doppler shift is observed by the listener, according to Eq. (6).

Doppler Simulation via Delay Lines

This magnetic tape is now the delay line, the tape read-head is the read-pointer of the delay line, and the write-head is the delay-line write-pointer. In this analogy, it is readily verified that modulating delay by changing the read-pointer increment from 1 to \(1 + v_s/c\) (thereby requiring interpolated reads) corresponds to motion away from the source at speed \(v_s\). It also follows that changing the write-pointer increment from 1 to \(1 + v_l/c\) corresponds source motion toward the listener at speed \(v_l\). When this is done, we must use interpolating writes into the delay memory. Interpolating writes are often called de-interpolation [?], and they are formally the graph-theoretic transpose of interpolating reads (ordinary “interpolation” [?]. A review of time-varying, interpolating, delay-line reads and writes, together with a method using a single shared pointer, are given in [?].

Consider a Fourier component of the source at frequency \(\omega_s\). We wish to know how this frequency is shifted to \(\omega_i\) at the listener due to the Doppler effect.

Velocity Projection

The Doppler effect depends only on velocity components along the line connecting the source and listener [?, p. 453]. We may therefore orthogonally project the source and listener velocities onto the vector \(\mathbf{x}_d = \mathbf{x}_l - \mathbf{x}_s\) pointing from the source to the listener. (See Fig. 4.1 on page 44 for a specific example.)

The orthogonal projection of a vector \(\mathbf{x}\) onto a vector \(\mathbf{y}\) is given by

\[
\mathbf{P}_{\mathbf{y}}(\mathbf{x}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{y}\|^2}\mathbf{y} = \frac{\mathbf{y}^T\mathbf{y}}{\|\mathbf{y}\|^2}\mathbf{y}.
\]

Therefore, we can write the projected source velocity as

\[
\mathbf{v}_d = \mathbf{P}_{\mathbf{x}_d}(\mathbf{v}_s) = \frac{\langle \mathbf{v}_s, \mathbf{x}_d \rangle}{\|\mathbf{x}_d\|^2}\mathbf{x}_d = \frac{\langle \mathbf{v}_s, \mathbf{x}_d \rangle}{\|\mathbf{x}_d\|^2} = \frac{\langle \mathbf{v}_s - \mathbf{v}_l, \mathbf{x}_d \rangle}{\|\mathbf{x}_d\|^2} = \frac{(\mathbf{v}_s - \mathbf{v}_l) \cdot \mathbf{x}_d}{\|\mathbf{x}_d\|^2}.
\]

In the far field (listener far away), Eq. (8) reduces to

\[
\mathbf{v}_d \approx \frac{\langle \mathbf{v}_s, \mathbf{x}_d \rangle}{\|\mathbf{x}_d\|^2} = \mathbf{P}_{\mathbf{x}_d}(\mathbf{v}_s) \quad (\|\mathbf{x}_d\| \gg \|\mathbf{x}_d\|).
\]

Time-Varying Delay-Line Reads

If \(x(t)\) denotes the input to a time-varying delay, the output can be written as

\[
y(t) = x(t - D_t),
\]

where \(D_t\) denotes the time-varying delay in seconds. In discrete-time implementations, when \(D_t\) is not an integer multiple of the sampling interval, \(x(t - D_t)\) may be approximated to arbitrary accuracy (in a finite band) using bandlimited interpolation (see §2.3 or other techniques for implementation of fractional delay [?, ?]).

Let’s analyze the frequency shift caused by a time-varying delay by setting \(x(t)\) to a complex sinusoid at frequency \(\omega_s\):

\[
x(t) = e^{j\omega_s t}\]

The output is now

\[
y(t) = x(t - D_t) = e^{j\omega_s (t - D_t)}.
\]

The instantaneous phase of this signal is

\[
\theta(t) = \angle y(t) = \omega_s \cdot (t - D_t)
\]

which can be differentiated to give the instantaneous frequency

\[
\dot{\omega}_t = \omega_s (1 - \frac{D_t}{D_s})
\]

where \(\omega_s\) denotes the output frequency, and \(D_s \triangleq \frac{1}{\omega_s}\) denotes the time derivative of the delay \(D_t\). Thus, the delay growth-rate, \(D_t\), equals the relative frequency downshift:

\[
D_t = \frac{\omega_s - \dot{\omega}_t}{\omega_s}.
\]
Comparing Eq. (10) to Eq. (6), we find that the time-varying delay most naturally simulates Doppler shift caused by a moving listener, with

$$D_t = \frac{v_{ls}}{c}$$

That is, the delay growth-rate, $D_t$, should be set to the speed of the listener away from the source, normalized by sound speed $c$.

Simulating source motion is also possible, but the relation between delay change and desired frequency shift is more complex, viz., from Eq. (6) and Eq. (10),

$$D_t = -\frac{\nu_r + \nu_d}{1 - \frac{\nu_d}{c}} \approx -\left(\frac{\nu_r}{c} + \frac{\nu_d}{c}\right)$$

where the approximation is valid for $v_d \ll c$. In Section 1.1, a simplified approach is proposed based on moving the delay input instead of its output.

The time-varying delay line was described in §1.1) or any number of moving sources. As discussed there, to implement a continuously varying delay, we add a “delay growth parameter” $g$ to the delayline function in Fig. 1.1. To simplify the layout, the input and output signals are simply sum into the shared delay line.

$$\text{rpdr} += 1; \quad \text{// pointer update}$$

$$\text{rpdr} += 1 - g; \quad \text{// pointer update}$$

When $g$ is 0, we have a fixed delay line, corresponding to $D_t = 0$ in Eq. (10). When $g > 0$, the delay grows $g$ samples per sample, which we may also interpret as seconds per second, i.e., $D_t = g$. By Eq. (11), we see that we need

$$g = -\frac{v_{ls}}{c}$$

to simulate a listener traveling toward the source at speed $v_{ls}$.

Note that when the read- and write-pointers are driven directly from a model of physical propagation-path geometry, they are always separated by predictable minimum and maximum delay intervals. This implies it is unnecessary to worry about the read-pointer passing the write-pointers or vice versa. In generic frequency shifters [3], or in a Doppler simulator not driven by a changing geometry, a pointer cross-fade scheme may be necessary when the read- and write-pointers get too close to each other.

**Multiple Read Pointers**

Using multiple read pointers, multiple listeners can be simulated. Furthermore, each read-pointer signal can be filtered to simulate propagation losses and radiation characteristics of the source in the direction of the listener. The read-pointers can move independently to simulate the different Doppler shifts associated with different listener motions and relative source directions.

**Multiple Write Pointers**

It is interesting to consider also what effects can be achieved using multiple de-interpolating write pointers. From the considerations in general, we need as many delay lines as there are sources or listeners, whichever is smaller. More precisely, if there are $N$ moving sources and $M$ moving listeners, simulation requires $\min(N, M)$ delay lines.

**Stereo Processing**

As a special case, stereo processing of any number of sources can be accomplished using two delay lines, corresponding to left and right stereo channels. The stereo mix may contain a panned mixture of any number sources, each with its own stereo placement, path filtering, and Doppler shift. The two stereo outputs may correspond to “left and right ears,” or, more generally, to left- and right-channel microphones in a studio recording set-up.

**System Block Diagram**

A schematic diagram of a stereo multiple-source simulation is shown in Fig. 1.1. To simplify the layout, the input and output signals are all on the right in the diagram. For further simplicity, only one input source is shown. Additional input sources are handled identically, summing into the same delay lines in the same way.

The input source signal first passes through filter $H_R(z)$, which provides time-invariant filtering common to all propagation paths. The left- and right-channel filters $H_L(z)$ and $H_R(z)$ are typically low-order, linear, time-varying filters implementing the time-varying

---

The “first” write-pointer is defined as the one writing farthest ahead in time; it must overwrite memory, instead of summing into it, when a circular buffer is being used, as is typical.
characteristics of the shortest (time-varying) propagation path from
the source to each listener. (The \((n)\) superscript here indicates
a time-varying filter.) These filter outputs sum into the delay lines at
arbitrary (time-varying) locations using interpolating writes
(de-interpolation). The zero signals entering each delay line on
the left can be omitted if the left-most filter overwrites delay memory
instead of summing into it.

The outputs of \(H_{zL}(z)\) and \(H_{zR}(z)\) in Fig. 1.1 correspond to the
“direct signal” from the moving source, when a direct signal exists.
These filters may incorporate modulation of losses due to the
changing propagation distance from the moving source to each
listener, and they may include dynamic equalization corresponding
to the changing radiation strength in different directions from the
moving (and possibly turning) source toward each listener.

The next trio of filters in Fig. 1.1 \(H_{1}(z)\), \(H_{zL}(z)\), and \(H_{zR}(z)\),
correspond to the next-to-shortest acoustic propagation path,
typically the “first reflection,” such as from a wall close to the source.
Since a reflection path is longer than the direct path, and since a
reflection itself can attenuate (or scatter) an incident sound ray,
there is generally more filtering required relative to the direct signal.
This additional filtering can be decomposed into its fixed component
\(H_{1}(z)\) and time-varying components \(H_{zL}(z)\) and \(H_{zR}(z)\).

Note that acceptable results may be obtained without implementing
all of the filters indicated in Fig. 1.1. Furthermore, it can be
convenient to incorporate \(H_{1}(z)\) into \(H_{zL}(z)\) and \(H_{zR}(z)\) when
doing so does not increase their orders significantly.

Note also that the source-filters \(H_{zL}(z)\) and \(H_{zR}(z)\) may include
HRTF filtering \([?, ?]\) in order to impart illusory angles of arrival in
3D space.

Chorus Effect

The *chorus effect* (or “choralizer”) is any signal processor which
makes one sound source (such as a voice) sound like many such
sources singing (or playing) in *unison*. Since performance in unison
is never exact, chorus effects simulate this by making independently
modified copies of the input signal. Modifications may include
(1) delay,
(2) frequency shift, and
(3) amplitude modulation.

The typical chorus effect today is based on several *time-varying delay lines* which accomplishes (1) and (2) in a qualitative fashion. Reverb
generally provides (3) incidentally. Before digital delay lines, analog
microphone stand in Fig. ?? \([?, ?]\). Two horns are apparent, but one is a
dummy, serving mainly to cancel the centrifugal force of the other
during rotation. The Model 44W horn is identical to that of the
Model 600, and evidently standard across all Leslie models \([?]\). For a
circularly rotating horn, the source position can be approximated as
\[
\mathbf{L}(t) = \begin{bmatrix}
r_c \cos(\omega_s t) \\
rs \sin(\omega_s t)
\end{bmatrix}
\]

where \(r_c\) is the circular radius and \(\omega_s\) is angular velocity. This
expression ignores any *directionality* of the horn radiation, and
approximates the horn as an omnidirectional radiator located at
the same radius for all frequencies. In the Leslie, a *diffuser* is inserted
into the end of the horn in order to make the radiation pattern closer
to uniform \([?]\), so the omnidirectional assumption is reasonably
accurate.

By Eq. (12), the source velocity for the circularly rotating horn is
\[
\mathbf{L}_s(t) = \frac{d}{dt} \mathbf{L}(t) = \begin{bmatrix}
-r_c \omega_s \sin(\omega_s t) \\
rs \omega_s \cos(\omega_s t)
\end{bmatrix}
\]

Note that the source velocity vector is always orthogonal to the
source position vector, as indicated in Fig. 1.1.

**The Leslie**

The Leslie, named after its inventor, Don Leslie \([?]\) is a popular audio
processor used with electronic organs and other instruments \([?, ?]\). It
employs a rotating horn and rotating speaker port to “choralize” the
sound. Since the horn rotates within a cabinet, the listener hears
multiple reflections at different Doppler shifts, giving a kind of *chorus
effect*. Additionally, the Leslie amplifier distorts at high volumes,
producing a pleasing “growl” highly prized by keyboard players.

The Leslie consists primarily of a rotating horn and a rotating
speaker port inside a wooden cabinet enclosure \([?]\). We first consider
the rotating horn.

**Rotating Horn Simulation**

The heart of the Leslie effect is a rotating horn loudspeaker. The
rotating horn from a Model 600 Leslie can be seen mounted on a

**Reference**

http://en.wikipedia.org/wiki/Leslie_effect
Since $v_s$ and $x_s$ are orthogonal, the projected source velocity Eq. (8) simplifies to
\[ v_{sl} = \mathbf{P}_{x_s} (v_s) = \frac{(v_s, x_s)}{\|v_s - x_s\|} (v_s - x_s). \tag{14} \]

Arbitrarily choosing $x_l = (r_l, 0)$ (see Fig. 1.1), and substituting Eq. (12) and Eq. (13) into Eq. (14) yields
\[ v_{sl} = \frac{-r_l \omega_m \sin(\omega_m t)}{r_l^2 + 2r_l r_s \cos(\omega_m t) + r_s^2} \left[ \begin{array}{c} r_l - r_s \cos(\omega_m t) \\ -r_s \sin(\omega_m t) \end{array} \right]. \tag{15} \]

In the far field, this reduces simply to
\[ v_{sl} \approx -r_s \omega_m \sin(\omega_m t) \begin{array}{c} 1 \\ 0 \end{array} \] \tag{16} \]

Substituting into the Doppler expression Eq. (6) with the listener velocity $v_l$ set to zero yields
\[ \omega_l = \frac{\omega_s}{1 + r_s \omega_m \sin(\omega_m t) / c} \approx \omega_s \left[ 1 - \frac{r_s \omega_m}{c} \sin(\omega_m t) \right]. \tag{17} \]

where the approximation is valid for small Doppler shifts. Thus, in the far field, a rotating horn causes an approximately sinusoidal multiplicative frequency shift, with the amplitude given by horn length $r_s$ times horn angular velocity $\omega_m$ divided by sound speed $c$.

Note that $r_s \omega_m$ is the tangential speed of the assumed point of horn radiation.

**Leslie Free-Field Horn Measurements**

The free-field radiation pattern of a Model 600 Leslie rotating horn was measured using the experimental set-up shown in Fig. ?? [?]. A matched pair of Panasonic microphone elements (Crystal River Snapshot system) were used to measure the horn response both in the plane of rotation and along the axis of rotation (where no Doppler shift or radiation pattern variation is expected). The microphones were mounted on separate boom microphone stands, as shown in the figure. A close-up of the plane-of-rotation mic is shown in Fig. ??.

**Rotating horn recording set up (from [?]).**

The horn was set manually to fixed angles from -180 to 180 degrees in increments of 15 degrees, and at each angle the impulse response was measured using 2048-long Golay-code pairs [?].

Figure ?? shows the measured impulse responses and Fig. ?? shows the corresponding amplitude responses at the various angles. Note that the beginning of each impulse response contains a fixed portion which does not depend significantly on the angle. This is thought to be due to “leakage” from the base of the horn. It arrives first since the straight-line path from the enclosed speaker to the microphone is shorter than that traveling through the horn assembly.

**Measured impulse-responses of the Leslie 600 rotating-horn at multiples of 15 degrees. The middle trace is recorded with the microphone along the axis of the horn (from [?]).**
Separating Horn Output from Base Leakage

Note that Fig. ?? indicates the existence of fixed and angle-dependent components in the measured impulse responses. An iterative algorithm was developed to model the two components separately [?].

Let $M = 256$ denote the number of impulse-response samples in each measured impulse response, and let $N = 25$ denote the number of angles ($-180:15:180$) at which impulse-response measurements were taken. We denote the $M \times N$ impulse-response matrix by $h$. Each column of $h$ is an impulse response at some horn angle. (Figure ?? can be interpreted as a plot of the transpose of $h$.)

We model $h$ as

$$h = \alpha + \gamma \cdot \text{diag}(z^{-\tau}) + e$$

where $\tau_i$ is the arrival-time delay, in samples, for the horn output in the $i$th row (the delays clearly visible in Fig. ?? as a function of angle). These arrival times are estimated as the location of the peak in the cross-correlation between the $i$th impulse response and the same impulse response after converting it to minimum phase [?]. The diagonal matrix $\text{diag}(z^{-\tau})$ denotes a shift operator which delays the $i$th column of $\gamma$ by $\tau_i$ samples. Thus, $\gamma$ contains the horn-output impulse response (without the base leakage) shifted to time zero (i.e., the angle-dependent delay is removed). Finally, the error matrix $e$ is to be minimized in the least-squares sense.

Each column of the matrix $\alpha$ contains a copy of the estimated horn-base leakage impulse-response:

$$\alpha = g \cdot 1^T$$

where $1^T = [1, 1, \ldots, 1]$.

The estimated angle-dependent impulse-responses in $\gamma$ are modeled as linear combinations of $K = 5$ fixed impulse responses, viewed (loosely) as principal components:

$$\gamma = g \cdot w$$

where $g$ is the $M \times K$ orthonormal matrix of fixed filters (principal components), and $w$ is a $K \times N$ matrix of weights, found in the usual way by a truncated singular value decomposition (SVD) [?].

Algorithm

To start the separation algorithm, $\gamma_0$ is initialized to the zero-shifted impulse response data $h \cdot \text{diag}(z^0)$, ignoring the tails of the base-leakage they may contain. Then $\alpha_0$ is estimated as the mean of $h - \gamma_i \cdot \text{diag}(z^{-\tau_i})$. This mean is then subtracted from $h$ to produce $b_1 = (h - \alpha_0) \cdot \text{diag}(z^{-\tau})$ which is then converted to $\gamma_1 = g_1 \cdot w_1$ by a truncated SVD. A revised base-leakage estimate $\alpha_1$ is then formed as $h - \gamma_i \cdot \text{diag}(z^{-\tau_i})$, and so on, until convergence is achieved.

Results

Figure ?? plots the $K = 5$ weighted principal components identified for the angle-dependent component of the horn radiativity. Each component is weighted by its corresponding singular value, thus visually indicating its importance. Also plotted using the same line type are the zero-lines for each principal component. Note in particular that the first (largest) principal component is entirely positive.

First 5 principal components weighted by their corresponding singular values. Each angle-dependent impulse response is modeled as a linear combination of these angle-independent impulse-response components (from [?]).

Figure ?? shows the complete horn impulse-response model $(\alpha + \gamma \cdot \text{diag}(z^{-\tau}))$, overlaid with the original raw data $h$. We see that both the fixed base-leakage and the angle-dependent horn-output response are closely followed by the fitted model.
Figure ?? shows the estimated impulse response of the base-leakage component \( \alpha(n) \), and Fig. ?? shows the modeled angle-dependent horn-output components \( \gamma \) delayed out to their natural arrival times.

Modeled horn-output impulse-responses at multiples of 15 degrees (from [?]).

Figure ?? shows the average power response of the horn outputs. Also overlaid in that figure is the average response smoothed according to Bark frequency resolution [?]. This equalizer then becomes \( H_0(z) \) in Fig. ?? The filters \( H_{0L}(z) \) and \( H_{0R}(z) \) in Fig. ?? are obtained by dividing the Bark-smoothed frequency-response at each angle by \( H_0(z) \) and designing a low-order recursive filter to provide that equalization dynamically as a function of horn angle.

The impulse-response arrival times \( \tau_i \) determine where in the delay lines the filter-outputs are to be summed in Fig. ??

Average angle-dependent amplitude response overlaid with Bark-smoothed response to be used as a fixed equalization applied to the source (from [?]).

Figure ?? shows a spectrogram view of the angle-dependent amplitude responses of the horn with \( H_0(z) \) (Bark-smoothed curve in Fig. ??) divided out. This angle-dependent, differential equalization is used to design the filters \( H_{0L}(z) \) and \( H_{0R}(z) \) in Fig. ?? Note that...
below 12 Barks or so, the angle-dependence is primarily to decrease
amplitude as the horn points away from the listener, with high
frequencies decreasing somewhat faster with angle than low
frequencies.

Leslie normalized response power spectrum, theta = [−180:30:180]

Angle-dependent amplitude response divided by Bark-smoothed
average response to be used as the basis for design of time-varying,
angle-dependent equalization to be applied after \( H_0(z) \) (from \([?)\]).

1.2 Rotating Woofer-Port and Cabinet Simulation

It is straightforward to extend our computational model to include
the rotating woofer port and wooden cabinet enclosure as follows:

- In \([?)\], it is mentioned that an AM “throb” is the main effect of
  the rotating woofer port. A modulated lowpass-filter cut-off
  frequency has been used for this purpose by others. Our
  measured data will be used to construct angle-dependent filtering
  in a manner analogous to that of the rotating horn, and this
  “woofer filter” runs in parallel with the rotating horn model.
- The Leslie cabinet multiply-reflects the sound emanating from
  the rotating horn. The first few early reflections are simply
  handled as additional sources in Fig. \([?)\]. We can extend the
  impulse-response-component separation algorithm of \([?)\] to the
  case of superimposed early reflections in the impulse response.
  (Preliminary results are promising.)
- To qualitatively simulate later, more reverberant reflections in
  the Leslie cabinet, we feed a portion of the rotating-horn and
  speaker-port signals to separate states of an artificial reverberator
  (see Chapter \([?)\)). This reverberator may be configured as a
  “very small room” corresponding to the dimensions and
  scattering characteristics of the Leslie cabinet, and details of the
  response may be calibrated using measurements of the impulse
  response of the Leslie cabinet. Finally, in order to emulate the
  natural spatial diversity of a radiating Leslie cabinet in a room,
  “virtual cabinet vent outputs” can be extracted from the model
  and fed into separate states of a room reverberator.

1.3 Miscellaneous Effects

This section describes miscellaneous digital audio effects which the
author has seen applied in practice. For much more about signal
processing for digital audio effects, see, e.g., \([?)\).

Doubling Simulation

Doubling is a studio recording technique often used to “thicken”
vocals in which the same part is sung twice by the same person. In
other words, doubling is a “chorus of two”, where both parts are
sung “in unison” by the same person. As an example, the Beatles
used doubling very often, such as on the track “Hard Day’s Night”.
A single variable delay line can simulate doubling very effectively.

Slap Back

The term slap back refers to the use of a single echo on a recorded
track. The echo may be placed in a different spatial location in the
stereo mix. Normally the echo delay is just large enough to be heard
as a discrete echo on careful listening (e.g., on the order of tens of
milliseconds). Slap back is very popular in 1950s-style recordings
such as “rockabilly” tunes.