# Fractionally-addressed delay lines

Davide Rocchesso
Dipartimento Scientifico e Tecnologico, Università di Verona
rocchesso@sci.univr.it

**Abstract**

While traditional implementations of digital delay lines are based on a circular buffer accessed by two pointers, we propose an implementation where a single fractional pointer is used both for reading and writing operations. On modern general-purpose architectures, the proposed method is nearly as efficient as the popular interpolated circular buffer, but it offers better performance in terms of frequency-dependent attenuation and response to delay-length modulations.

## 1 Prior Art

The classic implementation of the digital delay line uses a circular buffer, which is accessed by a writing pointer followed by a reading pointer [5]. When the delay length has to be made variable, the relative distance between the reading pointer and the writing pointer is varied sample by sample. In order to allow for fractional lengths and click-free length modulation, some form of interpolation has to be applied at the reading point [10, 3, 9]. The following properties should be ensured by the interpolation device: (i) flat frequency response, (ii) linear phase response, (iii) transient-free response to variations of the delay length.

All of the prior realizations, as far as a fixed delay length is considered, are linear and time-invariant systems, thus being completely described by their frequency response. Vice versa, we are proposing a realization which is time-varying even in the case of constant delay. While this realization was first proposed and implemented as part of a thesis work [6], signal and performance analyses were done very poorly at that time. In the following sections we describe and analyze the novel realization, with special focus on implementations on general purpose architectures.

## 2 A Fractionally-Addressed Delay Line

The key idea behind the proposed realization is to use a single pointer for both the read and write accesses. If the delay line has fixed integer length $B$, it is possible to use a buffer exactly $B$-cells long and a single pointer whose entry is first read and then written. In the same buffer we can also implement any delay which is an integer fraction $B/I$ just by incrementing the pointer at steps of $I$ samples. We are going to show how this scheme can be generalized to non-integer fractions of the total buffer length. The resulting technique can be seen as an extension of the table-lookup oscillator [4, 2], with the fundamental difference that every read is followed by one or more writes, in such a way that the waveform is continuously re-stored while being read.

Given a buffer size of $B$ samples, and a sample rate $F_s$, a (fractional) increment of $I$ samples gives a delay in seconds equal to

$$D = \frac{B}{I \cdot F_s} \tag{1}$$

.

Since this realization is related to waveform generation by fractional addressing [2], we call it the Fractionally-Addressed Delay (FAD) line.

### 2.1 Realization

As far as the value being read out of the delay line is concerned, the FAD line behaves similarly to the table-lookup oscillator, being possible to apply truncation, linear interpolation, or multirate interpolation techniques [5, 10]. More complicated is the injection of a new value, to be done right after the read, in such a way that no "holes" are left in the current pass through the buffer. For instance, for $I = 2$, two writes have to be performed for every read. A fractional increment would correspond to a variable number of writes at each step. Several interpolation techniques can also be applied at the write stage. Our quantitative analysis will be conducted on a realization where linear interpolation has been used in reading and quadratic interpolation in writing.

The pseudo-C code in general form looks as follows:

```
loop
    fph = floor(phase);
    output = interpolated_read(table[fph],
            table[fph+1], ...);
    ph = (phase_old + 1) % length_table;
    while (ph <= fph) {
```

```
        table[ph] = interpolated_write(
          ..., table[phase_old], input);
        ph = (ph + 1) % length_table;
      }
      phase_old = fph;
      phase = (phase + Increment);
      if (phase > length_table)
           phase = phase - length_table;
endloop
```

Notice that the `interpolated_read` uses samples following the `phase` pointer, while the `interpolated_write` uses samples preceding the pointer.

# 3 Input-Output Analysis

The FAD line is a time-varying system, and therefore it is difficult to characterize in terms of frequency response. When a sinusoidal input feeds the FAD line, spurious components are added to the main spectral line [7]. The magnitude of these components might be dependent on the frequency of the input sine wave and the initial (fractional) phase of the FAD-line pointer [1]. The signal-to-noise error ratio (SNR) as a function of these two parameters shows a very mild dependence on initial phase. Therefore, it makes sense to plot the average SNR as a function of the input frequency only (fig. 1). We can see that low frequencies are affected by high SNR, thus indicating that the FAD line has an acceptable behavior for practical sounds. Fig. 1 also shows a comparison with the linearly-interpolated (FIR) delay line. The noise error has been computed as the sum of the squared differences between the input and output waveform samples[2] [4]. The noise error is larger in the FIR case even though that implementation has no spurious components in the output spectrum. This is due to the fact that the FIR attenuation is larger on average.

Especially for applications such as waveguide modeling of musical instruments [8], it is important to consider the attenuation that different frequencies are subject to when fed into the delay line. The attenuation of the main peak of the output spectrum turns out to be highly dependent on the initial phase. Therefore, for the sake of comparison with the FIR line, we plot in fig. 2 the minimum, maximum, and mean attenuation as a function of the frequency of the input sine wave. Notice that at 2/3 of the Nyquist frequency the FIR line shows an attenuation of 1.2dB while the FAD line shows a mean attenuation of 0.5dB.

[1] As an example of dependence on initial phase, consider the increment $I = 1$. If the initial phase is 0 the pointer always falls on samples. If the initial phase is 0.5 the pointer always falls between samples.

[2] A normalizing factor $\sqrt{2/N}$ has been applied, being $N$ the number of samples per period.

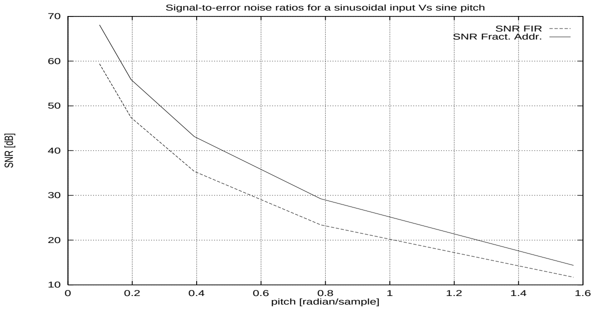


Figure 1: Signal-to-error noise ratio Vs. sine frequency for the FIR line and for the FAD line

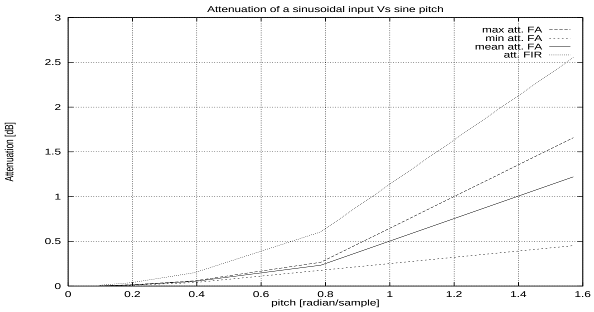


Figure 2: Attenuation of a sinusoidal input Vs. sine frequency for the FIR line and the FAD line

## 3.1 Behavior for time-varying delay

The FAD line shows an unconventional behavior when the delay length is dynamically varied [7]. Suppose to vary the delay length as a linear function of time $t$, starting from the nominal delay $\tau_0$ and decreasing it at the rate of $k$ seconds per second:

$$D(t) = \tau_0 - kt \tag{2}$$

The FIR line responds with an instantaneous pitch shift in the output signal. In other words, we get a Doppler effect and the pitch shift is

$$\Delta f = 1 + k \tag{3}$$

On the other hand, the FAD line provides a steady pitch shift

$$\Delta f = e^k \tag{4}$$

after a transient time

$$\tau_i = \frac{\tau_0}{k}(1 - e^{-k}) \tag{5}$$

The transient time can be calculated by feeding the delay line with an impulse at time 0. It will come out of the line at the time instant $\tau_i$ such that

$$\int_0^{\tau_i} I(t)dt = \frac{B}{F_s} \tag{6}$$

where $I(t)$ is the time-dependent increment which produces the desired ramp in delay length. Equation (6) can be rewritten, using (1), as

$$\int_0^{\tau_i} \frac{1}{\tau_0 - kt} dt = 1 \tag{7}$$

which is solved by (5).

The steady-state transposition can be calculated by observing that a second impulse entering the line at time $T_i$ "sees" an instantaneous delay of $\tau_0 - kT_i$ seconds. It gets out of the line at time $\frac{\tau_0 - kT_i}{k}(1 - e^{-k}) + T_i$, exactly $T_i e^{-k}$ seconds after the impulse which entered at time 0.

A different behavior is also reported in response to sinusoidal modulations of the delay length [7]. These modulations are essential for effects such as flanging or phasing.

### 3.2 Physical Interpretation

If the dynamic behavior of the delay lines is closely analyzed, we see that the FAD and the FIR realizations actually simulate two different physical phenomena. In both cases, the lines can be thought of as a one-dimensional medium where waves propagate. However, when the delay length is dynamically reduced we have two physical analogies in the two cases. The shortening of the FIR line corresponds to the receiver getting closer to the transmitter, and therefore we have a tight simulation of the Doppler effect. On the other hand, the shortening of the FAD line corresponds to incresing the velocity of propagation in the medium while maintaining the same physical distance between the two ends.

Figures 3–4 illustrate what happens when using different implementations of the delay lines in the simulation of a one-dimensional waveguide resonator, such as a string. In this application there is a couple of delay lines in a feedback connection, each representing propagation of waves in one direction. For the sake of simplicity, the terminations are supposed to be perfectly reflecting. If one of the delays is fed by three periods of a fast sine wave, this packet propagates, gets reflected, feeds the other line, and comes back to the excitation point for another reflection. Under ideal conditions, the packet keeps going back and forth without attenuation or losses. Suppose that, right after a reflection at the excitation end, suddenly the string gets lengthened by some amount, as we would do for lowering the pitch of a string. This can be simulated, in the classic FIR line implementation, by moving the reading pointer backwards. However, this operation exposes again the wave packet which has just been reflected, thus modifying the "duty cycle" of the waveform, as reported on fig. 3. A correct waveform can be obtained by adding a write operation right after the read to the FIR line implementation. This write takes care of erasing the waveform as it passes the reading pointer. Another way of lowering the pitch is that of instantaneously changing the string tension. In a waveguide simulation this corresponds to changing the spatial sampling via a change in the waveguide speed of propagation [8]. This can be achieved by the FAD line implementation just by changing the phase increment, and the result is illustrated in fig. 4. A mixture of tension and length increase might be obtained by using the FAD line with a variable buffer size.

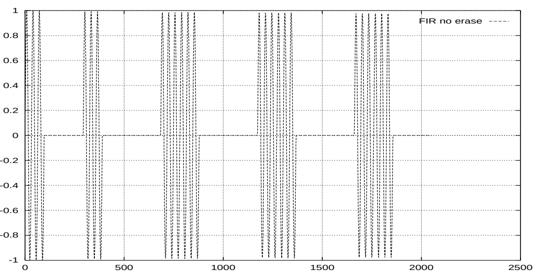


Figure 3: Note transition: FIR line

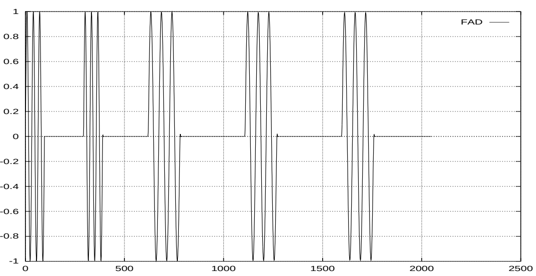


Figure 4: Note transition: FAD line

The two ways of producing a pitch shift are not equivalent as far as timbre is concerned. In fact, the FAD-line pitch change produces a contraction of the whole spectrum, thus modifying the position of formants, while the FIR-line pitch change comes without moving the formants.

Summarizing, different implementations of the delay line do have practical consequences on the timbre produced by dynamically-varying waveguide models.

# 4 Performance

The FAD line has only one pointer for accessing data in the buffer. It exhibits spatial locality because any short sequence of accesses spans over a small neighborhood of the pointed buffer cell. On the other hand, a FIR line has two pointers, thus exhibiting two distinct spatial localities. As a consequence, we expect that the FAD line makes better use of the cache in general purpose computer architectures. However, the FAD performs more writes than reads. In order to attain a 50% of delay variability, we have to accept up to two writes for every read.

The FAD line, despite of its higher complexity, does not run much slower than the FIR line on a general purpose computer. A rough benchmark has been performed on an $AMD - K6$ architecture by repeatedly delaying a soundfile stored in an array. The experiment was done using a Linux operating system with the machine in stand-alone single-user configuration. The first run was neglected because it seemingly involves instruction and data loading in the cache. An average of the following 13 repetitions gave us the results which are summarized in fig. 5, where benchmarks are reported for varying buffer size. A first comment is about the small difference in performance of the three algorithms, especially if we consider that the quadratically-interpolated FAD line has about 5 times as many floating point operations as the FIR line. A second comment is about the fact that the curves are monotonically increasing. This indicates that more and more cache misses are encountered when using larger buffers. The fact that the slope of the FIR curve is higher than the other curves confirms that having only one locality helps[3]. In the FIR curve it is also possible to see two peaks of steepness right before reaching the sizes of 4096 and 65536 samples which, when translated in bytes, give respectively the size of the level-1 data cache (32 KBytes) and the size of the level-2 cache[4]. Using the FIR line and eliminating the phase increment it is possible to measure the overall cost of caching, wich is also visible from fig. 5 as the difference between the two lower curves. Since it turns out to be less than 6% even for very large buffers, we can argue that delay lines are not affected much by the memory hierarchy.

In our implementations, we have not done aggressive code optimization, so that the relative performance of the three realizations might vary in practice from what we have shown. In particular, we found that float-to-integer conversions are expensive, and it would be wise to perform them by direct bit manipulation, as suggested in [1]. In any case, our results show that alternative implementations might be considered for sound-processing building blocks, even when the cost in terms of pure floating point operations seems daunting.

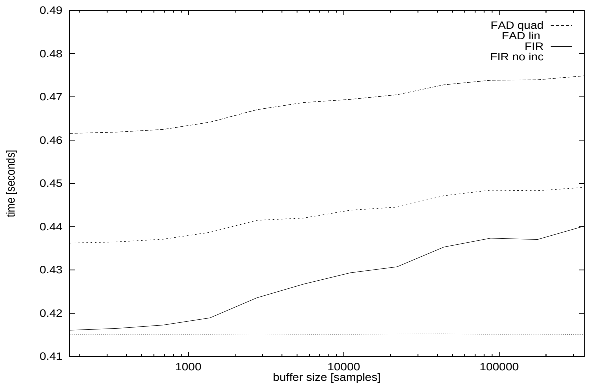


Figure 5: Performance on an AMD-K6 of the quadratically-interpolated FAD line, linearly-interpolated FAD line, linearly-interpolated FIR line, and FIR line with no phase increment, as a function of buffer size.

# 5 Conclusion

We have proposed a realization of the digital delay line which is based on an extension of the table-lookup oscillator. The proposed realization exploits the features of modern computer architectures and shows improved performance in terms of frequency-dependent attenuation and dynamic behavior. We expect this delay line will be considered as a building block for physically-based sound synthesis and for sound effects such as flangers and choruses.

# References


[1] R. B. Dannenberg and N. Thompson. Real-time software synthesis on superscalar architectures. *Computer Music J.*, 21(3):83–94, 1997.

[2] W. M. Hartmann. Digital waveform generation by fractional addressing. *J. Acoustical Soc. of America*, 82(6):1883–1891, 1987.

[3] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine. Splitting the Unit Delay—Tools for Fractional Delay Filter Design. *IEEE Signal Processing Magazine*, 13(1), Jan 1996.

[4] F. R. Moore. Table lookup noise for sinusoidal digital oscillators. *Computer Music J.*, 1(1):26–29, 1977.

[5] S. J. Orfanidis. *Introduction to Signal Processing.* Prentice Hall, Englewood Cliffs, N.J., 1996.

[6] D. Rocchesso. Realizzazione di Risuonatori Dispersivi in Tempo Reale. Tesi di laurea, Università di Padova, Dipartimento di Elettronica e Informatica, Feb. 1992.

[7] D. Rocchesso. A digital delay line based on fractional addressing. In *Proc. XII Colloquium Mus. Inform.*, Gorizia, Italy, Sept. 1998. AIMI.

[8] J. O. Smith III. *Principles of Digital Waveguide Models of Musical Instruments*, volume Applications of Digital Signal Processing to Audio and Acoustics, pages 417–466. Kluwer Academic Publishers, 1998. M. Kahrs and K. Brandenburg, eds.

[9] S. Tassart and P. Depalle. Analytical approximations of fractional delays: Lagrange interpolators and allpass filters. In *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Munich*, pages 455–458, Apr. 1997.

[10] U. Zölzer. *Digital Audio Signal Processing.* John Wiley and Sons, Inc., Chichester, England, 1997.


[3] However, this phenomenon doesn't show up when the same benchmark runs on an Intel Pentium II.

[4] And also of the memory covered by the Translation Lookaside Buffer, which is responsible for fast translation of virtual addresses to real addresses.