MUMT 618: Week #1

- Digital Signals and Filtering:
  - Students are expected to already be familiar with basic aspects of discrete-time audio signal processing;
  - That said, the material presented here is intended to provide a minimum level of understanding of how digital filters are characterized and implemented in Matlab;
  - As well, the general idea of characterizing a system in terms of its time-domain impulse response and corresponding frequency response will be essential throughout the course.
- Delay Lines:
  - Digital delay lines will be used extensively in this course to simulate wave propagation over time and/or space.
  - As well, we will see that several different effects algorithms can be implemented with delay lines.

1 Discrete-Time Signals & Digital Filtering

This course is fundamentally concerned with the manipulation and/or synthesis of audio or other physical signals. Because we work with computers, these signals are uniformly sampled, or *discretized*, in time and/or space.

1.1 Discrete-Time Signals

- Signals typically represent physical variables such as displacement, velocity, pressure, energy, ...
- In most cases, we are concerned with variables that are *time-dependent*.
- The discrete-time or digital time index is generally specified by $t_n = nT_s$, where n is an integer and T_s is the sampling interval or period.
- It is common to drop the explicit reference to T_s (or assume $T_s = 1$) and index discrete-time signals by the letter n.
- A discrete-time signal that is only dependent on time can be represented by $x[n]$ for $n = 0, 1, 2, ...$
- We typically assume our systems are *causal* or do not depend on future inputs (they should have zero "activity" before a non-zero input is applied). This can also be stated as $x[n] \overset{\Delta}{=} 0$ for $n < 0$.

1.2 Frequency-Domain Representation and Ideal Signals

- When working with signals and systems, it is generally useful to consider both time- and frequency-domain representations.
- The frequency-domain representation of a signal indicates how that signal can be constructed in terms of sinusoidal components, each with a specific frequency, gain and phase offset.

- We will often make use of “ideal” signals, such as sinusoids, noise and impulses, as inputs to our systems.
- **Sinusoids:** can be represented mathematically in the time-domain with sin or cos functions; in the frequency domain, they consist of a single non-zero frequency component.
- **Noise:** in the time domain, noise signals consist of a continuous stream of randam values; in the frequency domain, they contain sinusoidal energy over a specified wide range of frequencies.
- **Impulses:** in the time domain, a signal of very short time duration and an area of one; in the frequency-domain, impulses contains sinusoidal energy over a full range of frequencies.
- A discrete-time unit impulse signal is defined in the time domain by:

$$\delta[n] \triangleq \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

- The frequency-domain representation of a discrete-time signal is computed using the discrete fourier transform (DFT). The resulting frequency-domain signal is given in terms of complex numbers of the form $a + jb$, which can be plotted in terms of *magnitude* and *phase* values. Magnitude values are calculated as $\sqrt{a^2 + b^2}$, while phase values are calculated as $\tan^{-1}(b/a)$.
- Figure 1 illustrates example sinusoidal, noise and impulse signals, plotted in both the time and frequency domains (magnitude only).

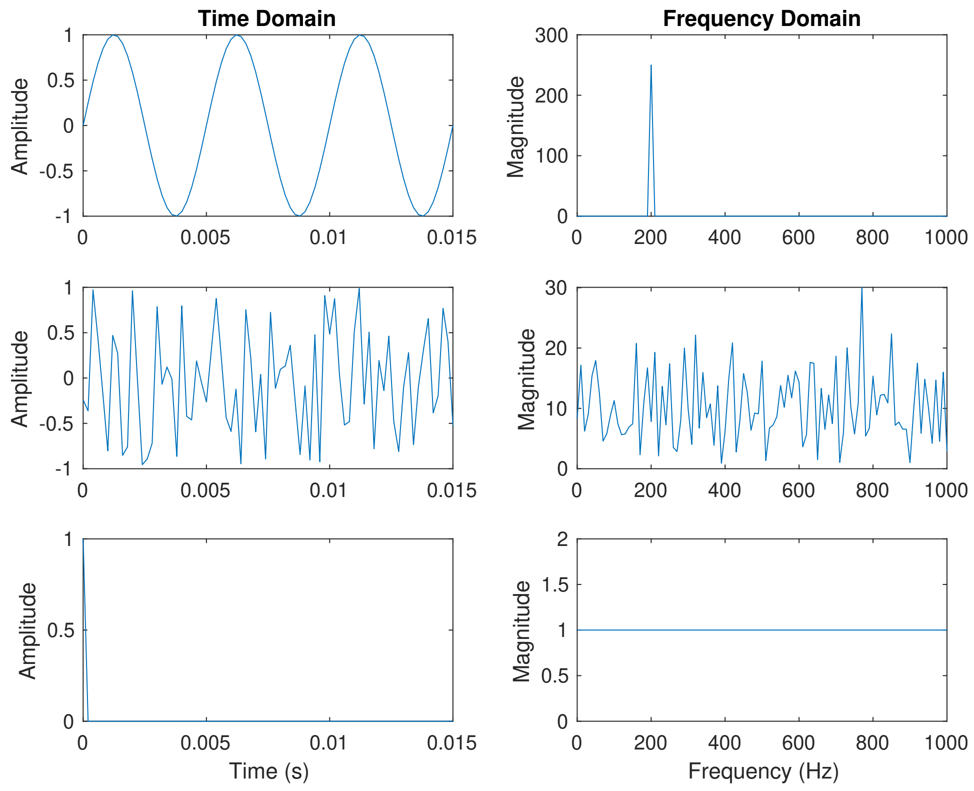

Figure 1: Sinusoidal (top), noise (middle) and impulse (lower) signals: time- and frequency-domain representations.

1.3 The Discrete Fourier Transform (DFT)

- Given a discrete-time signal $x[n]$, we can determine its frequency response using the *Discrete Fourier Transform (DFT)*:

$$X[k] \triangleq \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N}, \qquad k = 0, 1, 2, \dots, N-1.$$

- The complementary *inverse DFT* is given by:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j2\pi kn/N}, \qquad n = 0, 1, 2, \dots, N-1.$$

- The DFT can be interpreted as the sum of projections of $x[n]$ onto a set of N sampled complex sinusoids or sinusoidal basis functions at (normalized) radian frequencies given by $\omega_k = 2\pi k/N$ with $k = 0, 1, 2, \dots, N-1$.
- In this way, the DFT and its inverse provide a "recipe" for reconstructing a given discrete-time signal in terms of sampled complex sinusoids.
- Various computational tools exist that allow for the transformation of a discrete-time signal into its frequency-domain representation.

1.4 Digital Filtering

Digital filters are fundamental to digital audio processing. While we only have time here for a cursory overview of the essential features of filters, students are encouraged to pursue more advanced courses and references in filter analysis and design.

- In general, we need to manipulate our signals. Even if we only seek to measure and analyze "real world" signals, we still typically need to "process" these signals in order to compensate for measurement system "biases".
- The processing of signals is called *filtering*. When applied to discrete-time signals, this processing is called *digital filtering*.
- Digital filters are defined by their *impulse response*, $h[n]$, or the filter output given a unit sample impulse input signal.
- The filtering operation in the time domain is referred to as convolution, defined as

$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} h[m]x[n-m],$$

  where h is the filter impulse response and x is the input signal to the filter.
- Digital filters are often more intuitively understood in terms of their *frequency response*. That is, how is a sinusoidal signal of a given frequency affected by the filter.
- One way to find the frequency response of a digital filter is by taking the DFT of the filter impulse response, assuming the impulse response has sufficiently decayed before being truncated.
- The frequency response of a filter consists of its *magnitude* and *phase* responses. The magnitude response indicates the ratio of a filtered sine wave's output amplitude to its input amplitude. The phase response describes the phase "offset" or time delay in radians experienced by a sine wave passing through a filter.
- If the input to a filter is a sinusoid, the output from the filter will also be a sinusoid of the same frequency, though potentially scaled in amplitude or shifted in time. This is a fundamental property of *linear systems*.

1.5 Filter Types and Descriptions

- Digital filter types are generally described in terms of their frequency magnitude response characteristic.

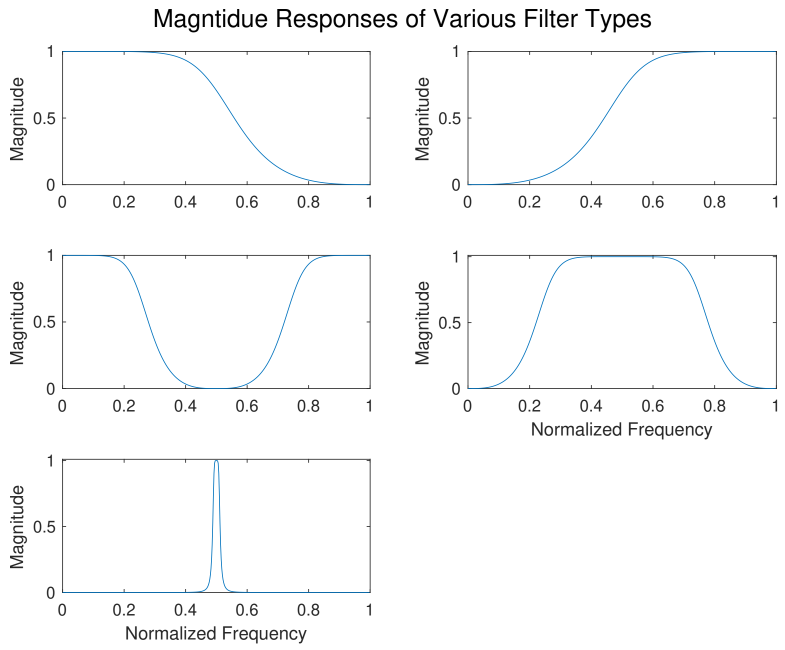

Figure 2: Magnitude responses for several different filter types.

- Lowpass filters signficantly attenuate sinusoids above a certain "cutoff frequency" (f_c), ideally only passing frequency components below the cutoff.
- Highpass filters significantly attenuate sinusoids below a certain "cutoff frequency" and pass components higher than the cutoff.
- The cutoff frequency of a filter is defined as the frequency at which the power transmitted is 1/2 the maximum power transmitted in the passband. This corresponds to a -3dB drop on a decibel scale. As the power of a signal is related to its amplitude squared, the cutoff frequency corresponds to an amplitude reduction of $1/\sqrt{2} = 0.707$.
- Bandpass filters only pass sinusoids above and below a certain frequency region, while bandstop filters are designed to pass all sinusoids except those within a certain frequency region.
- Resonance filters accentuate sinusoids within a certain frequency region. They may or may not pass sinusoidal frequencies outside that region.
- Resonance filters are typically described in terms of their *center frequency* and *quality factor (Q)*, which is given by the center frequency divided by the -3dB bandwidth. A higher Q indicates a "sharper" resonance.
- Allpass filters do not affect the magnitude characteristics of a signal (their magnitude response is equal to 1 for all frequencies) but have frequency-dependent phase characteristics.

1.6 Finite Impulse Response (FIR) Filters

- Finite Impulse Response (FIR) filters are defined by scaled and time-delayed versions of the filter input signal only, as given by the following *difference equation*:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] \dots, \qquad n = 0, 1, 2, \dots,$$

where the input $x[n] \triangleq 0$ and output $y[n] \triangleq 0$ for $n < 0$.

- An FIR filter can be represented by a *block diagram* as shown in Fig. 3 below.

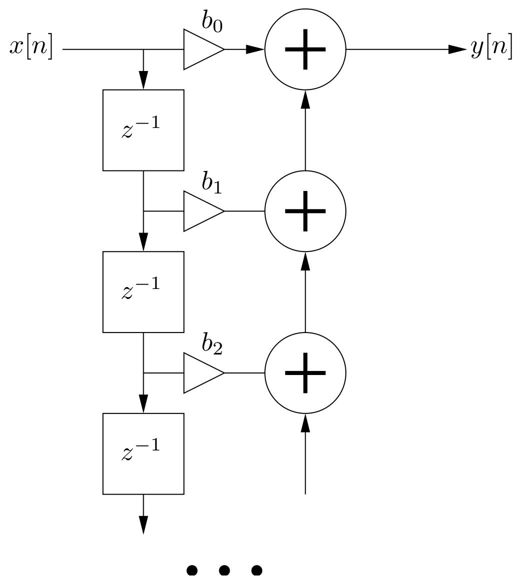

Figure 3: An FIR filter block diagram.

- The z^{-1} terms represent unit delays. Note that while this representation for a delay element is common and widely accepted in the signal processing community, the specification of delay in terms of powers of z is a z-domain characterization (to be described below) while the block diagram itself is a time-domain representation.
- With respect to the filter block diagram, FIR filters make use of *feed-forward* terms only.
- The impulse response of an FIR filter is only as long as the maximum delayed input term in its difference equation.
- The summation of feedforward input terms can result in destructive signal interference, or cancellations, at certain frequency values.
- The FIR filter is said to have an *order* equivalent to the number of unit delays in its difference equation.

1.7 Infinite Impulse Response (IIR) Filters

- Infinite Impulse Response (IIR) filters include delayed and scaled versions of the output signal that are fed back into the current output.
- IIR filters are described by the following difference equation:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] \ldots - a_1y[n-1] - a_2y[n-2] \ldots, \qquad n = 0, 1, 2, \ldots,$$

where the input $x[n] \triangleq 0$ and output $y[n] \triangleq 0$ for $n < 0$.

- An IIR filter can be represented by a block diagram as shown in Fig. 4 below.
- With respect to the filter block diagram, IIR filters make use of both *feed-forward* and *feedback* terms.
- Given these feedback terms, IIR filters can become unstable based on the values of the feedback coefficients (a_k terms).

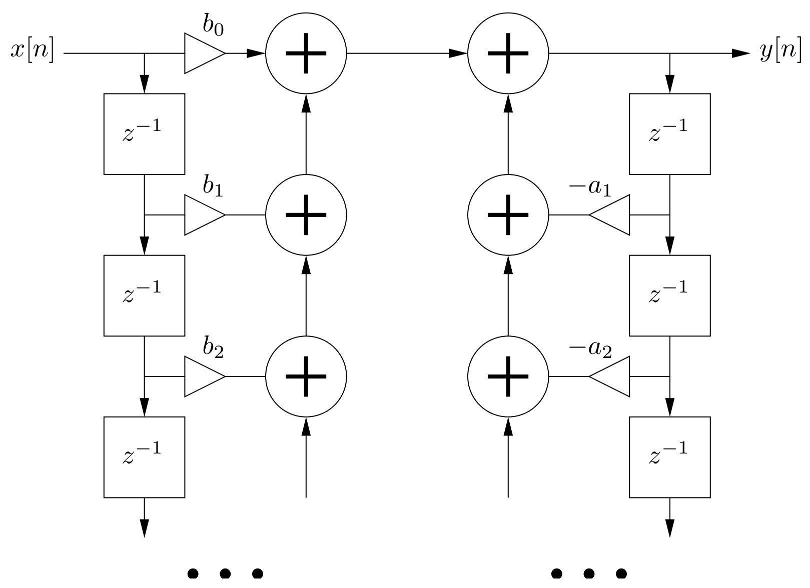

Figure 4: A general IIR digital filter block diagram.

- Because of this constructive feedback, an IIR filter can produce strong peaks, or resonances, in its frequency magnitude response.
- The feedback terms also result in a long filter impulse response. Though a stable filter's impulse response will decay *toward* zero over time, it technically never exactly reaches zero (ignoring issues of finite precision arithmetic).
- The order of an IIR filter is equivalent to the greater of the number of its delayed input or output terms.

1.8 Steady State and Transient Response

- Before a signal is applied to the input of a digital filter, the filter's internal "state" is assumed equal to zero.
- Digital filters are linear systems. One property of linear systems is that a sinusoidal input will produce a sinusoidal output of the same frequency.
- However, when a sinusoidal signal is first applied to the input of a digital filter, the output initially exhibits a region of transition (referred to as its *transient response*).
- For FIR filters, this transition region (or "warm-up" period) has a duration in samples equal to the filter order.
- For IIR filters, the length of the transition region is dependent on the filter order and the feedback coefficient values.
- Assuming a continued application of the sinusoidal input, the filter will subsequently settle into its *steady-state* region.
- If the input changes frequency or displays a discontinuity of any sort, another transient region will occur in the filter output.
- The frequency response of a digital filter is understood to represent its steady-state behavior.

1.9 The z-Transform

- The z-transform is a mathematical tool that is extensively used to evaluate the properties of discrete-time systems such as digital filters. In particular, it is convenient for determining the stability of a system. It is the discrete-time equivalent of the Laplace transform, which is used for continuous-time systems.
- The unilateral z-Transform of a discrete-time signal $x[n]$ is given by:

$$X(z) \triangleq \sum_{n=0}^{+\infty} x[n]z^{-n},$$

  where z is a complex variable.
- The z-transform maps a discrete-time signal to a function of the complex variable z.
- A convenient property of the z-transform is given by the *Shift Theorem*,

$$x[n-\Delta] \leftrightarrow z^{-\Delta}X(z),$$

  which says that a delay of Δ samples in the time domain corresponds to a multiplication by $z^{-\Delta}$ in the z domain.
- Using the shift theorem, we can easily calculate the z-transform of a digital filter's difference equation. Given the following second-order difference equation,

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] - a_1y[n-1] - a_2y[n-2],$$

  the z-transform can immediately be written (assuming the system is linear)

$$Y(z) = b_0X(z) + b_1z^{-1}X(z) + b_2z^{-2}X(z) - a_1z^{-1}Y(z) - a_2z^{-2}Y(z).$$

- From this expression, we can determine the transfer function, $H(z) = Y(z)/X(z)$, of the filter:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}}.$$

- The transfer function of a system is evaluated in the complex z-plane, as illustrated below:
- The z-transform is a more general version of the *Discrete-Time Fourier Transform*, which itself can be viewed as the limiting form of the DFT when its length N is allowed to approach infinity.
- We can determine the frequency response of a system from its z-transform by setting $z = e^{j\omega T_s}$, where ω is in radians per second and T_s is the sample period. In the complex z-plane, this is equivalent to evaluating the z-transform on the "unit circle" defined by $z = e^{j\omega T_s}$.

1.10 Filters in Matlab

- Filter coefficients in Matlab are specified in terms of the following general filter difference equation:

$$a_0y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2]\ldots - a_1y[n-1] - a_2y[n-2]\ldots, \qquad n = 0, 1, 2, \ldots$$

- In this way, a filter with difference equation $y[n] = x[n] + x[n-1]$ would be specified with feedforward coefficients $b = [1\ 1]$ and a feedback coefficient $a = 1$.
- Likewise, a filter with difference equation $y[n] = x[n] + 0.9y[n-1]$ would be specified with a feedforward coefficient $b = 1$ and feedback coefficients $a = [1\ -0.9]$.
- In Matlab, we can filter an input signal x with a filter specified by coefficients b and a as `y = filter(b, a, x)`.
- In Matlab, we can plot a filter's frequency response using the `freqz` function: `freqz(b, a)`.

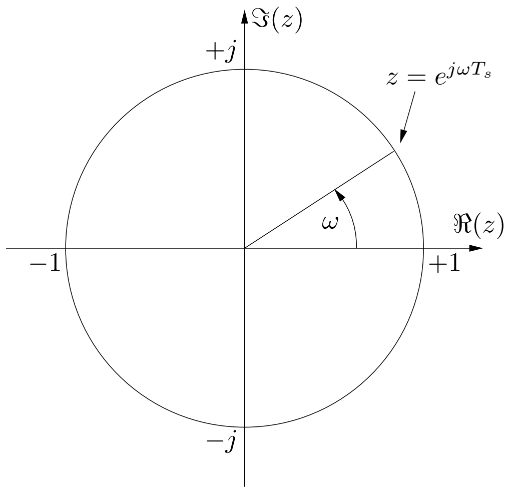

Figure 5: The complex z-plane.

1.11 Transfer Function & Pole-Zero Analysis:

- The numerator and denominator of a system's *transfer function* are polynomials in z. The *roots* of these polynomials can be determined by factorization.
- Roots of the numerator polynomial indicate values of z at which the transfer function evaluates to zero. These are called *zeros*.
- Roots of the denominator polynomial indicate values of z at which the transfer function evaluates to infinity. These are called *poles*.
- The zeros and poles of a transfer function can be plotted in the z-plane. Their locations with respect to the unit circle indicate radian frequencies at which the system's magnitude response has local minima (near zeros) or maxima (near poles).
- In Matlab, the functions `roots` and `zplane` can be used to determine and plot the poles and zeros of a system.
- When the coefficients of a transfer function are all real, complex roots are given by complex-conjugate pairs.
- For a system to have a stable frequency response, all of its poles must lie within the unit circle in the z-plane.

1.12 Filter Combinations

- Filters can be cascaded in series or parallel.
- The time-domain operation of a filter is referred to as *convolution* (and typically denoted with the symbol $*$). From the properties of linear systems, convolution in the time domain corresponds to multiplication in the frequency domain.
- Thus, the series combination of filter responses $H_1(z)$ and $H_2(z)$ in Fig. 6 above would produce a new response given in the frequency domain by $H_1 H_2$ or in the time domain by $h_1 * h_2$.
- The response of filters in parallel is given by simple addition (in both the time- and frequency-domains).
- In this way, the filter combination shown in Fig. 6 could be replaced by a single filter with frequency-domain response $(H_1 H_2) + H_3$ or time-domain response $(h_1 * h_2) + h_3$.

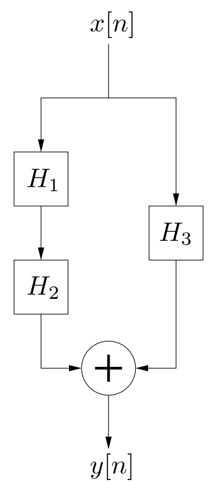

Figure 6: Cascaded digital filters.

2 Delay Lines

A variety of digital filters, audio effects and algorithms make use of delay lines. In this section, we study some properties of delay line systems and consider issues of implementation.

2.1 A Digital Delay Line

- A block diagram of a digital delay line is shown in Fig. 7 below. The length of the delay is given by the exponent of z (see the z-Transform).

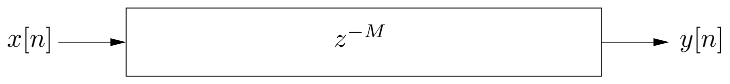

Figure 7: A digital delay line of length M samples.

- The input signal is given by $x[n], n = 0, 1, 2, \ldots$. For a delay line length of M samples, the output is given by the relation
$$y[n] = x[n - M], \qquad n = 0, 1, 2, \ldots,$$
where $x[n] \triangleq 0$ for $n < 0$.
- The transfer function for this system is given by $H(z) = z^{-M}$.
- The functioning of a delay line can be visualized as in Fig. 8, assuming $M = 4$ and a discrete-time unit impulse input signal defined as
$$x[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0. \end{cases}$$
- For every time step n, the signal value in the last (right-most) memory location is output from the delay line, the remaining stored values are propagated to adjacent memory locations (to the right), and a new input sample is written to the first (left-most) memory location.
- Thus, the sample value of 1 that is input at time $n = 0$ will appear at the delay line output at time step $n = 4$. For this particular example, all subsequent inputs and outputs to the delay line are zeroes.

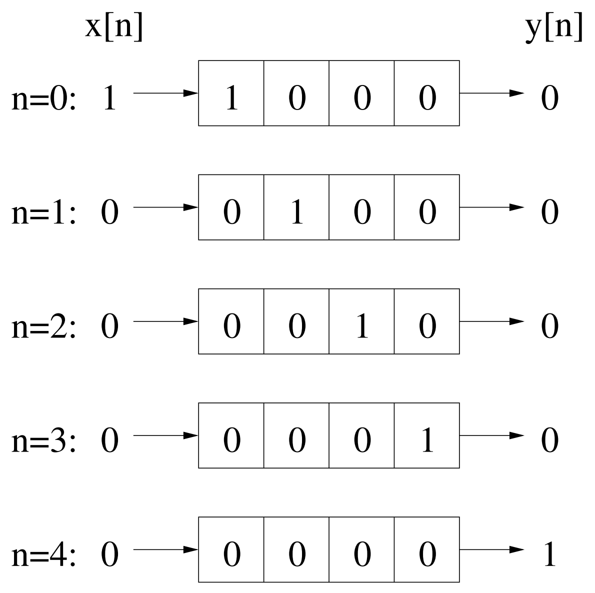

Figure 8: Signal flow in a digital delay line of length $M = 4$ samples.

In this way, the finite length impulse response of the length $M = 4$ digital delay line is given by $h = \{0, 0, 0, 0, 1\}$.

- The phase response for the example $M = 4$ delay line system is shown in Fig. 9, plotted both in radians (top) and as *phase delay* (negative phase divided by frequency) in samples (bottom), from which its linear phase response is clear.

2.2 Implementing Digital Delay Lines

- A digital delay line is implemented by allocating a buffer of values in memory.
- The input and output points are represented by one or more pointers, which are incremented at each time step.
- If a delay line length $M = 0$ is to be supported, it is necessary to use more than one pointer.
- Memory allocation can be a relatively time consuming operation in a realtime synthesis environment. Thus, in situations where the delay length may change over time, a large buffer of some maximum size is usually created during initialization.
- The *Synthesis ToolKit in C++ (STK)* includes the class, `Delay`, which provides non-interpolating delay functionality.

2.3 Feedforward Comb Filters

Comb filters get their name from the shape of their magnitude response and are important components of audio signal processing networks.

- A feedforward comb filter is illustrated in Fig. 10.
- The difference equation for this filter is given by:

$$y[n] = b_0 x[n] + b_M x[n - M].$$

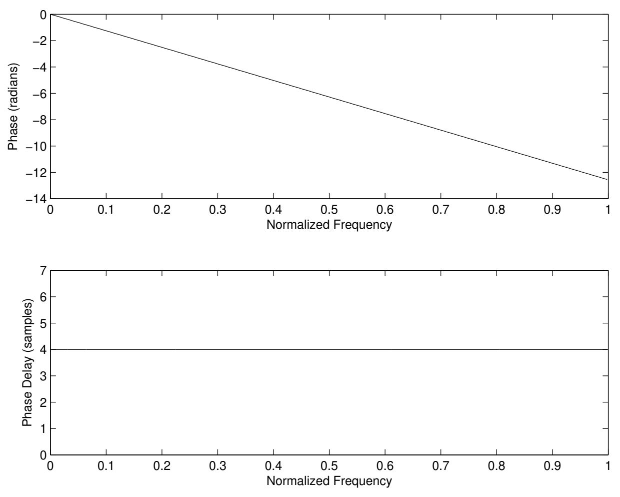

Figure 9: Phase response of a digital delay line of length $M = 4$ samples: (top) phase in radians; (bottom) phase delay in samples.

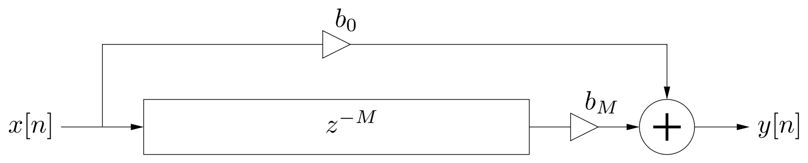

Figure 10: A feedforward comb filter block diagram.

- The transfer function of the feedforward comb filter is

$$H(z) = b_0 + b_M z^{-M},$$

  from which the amplitude response is found as

$$G(\omega) \triangleq \left|H(e^{j\omega})\right| = \left|b_0 + b_M e^{-j\omega M}\right|, \qquad -\pi \leq \omega \leq \pi.$$

- The magnitude response of a feedforward comb filter, calculated with the Matlab script combs.m, is shown in Fig. 11 for $M = 5$, $b_0 = 1$, and $b_M = 0.1$, 0.5, and 0.9.

2.4 Feedback Comb Filters

- A feedback comb filter is illustrated in Fig. 12.

  The difference equation for this filter is given by:

$$y[n] = b_0 x[n] - a_M y[n - M].$$

  For stability, the coefficient $|a_M| \leq 1$.

- The transfer function of the feedback comb filter is

$$H(z) = \frac{b_0}{1 + a_M z^{-M}},$$

  from which the amplitude response is found as

$$G(\omega) \triangleq \left|H(e^{j\omega})\right| = \left|\frac{b_0}{1 + a_M e^{-j\omega M}}\right|, \qquad -\pi \leq \omega \leq \pi.$$

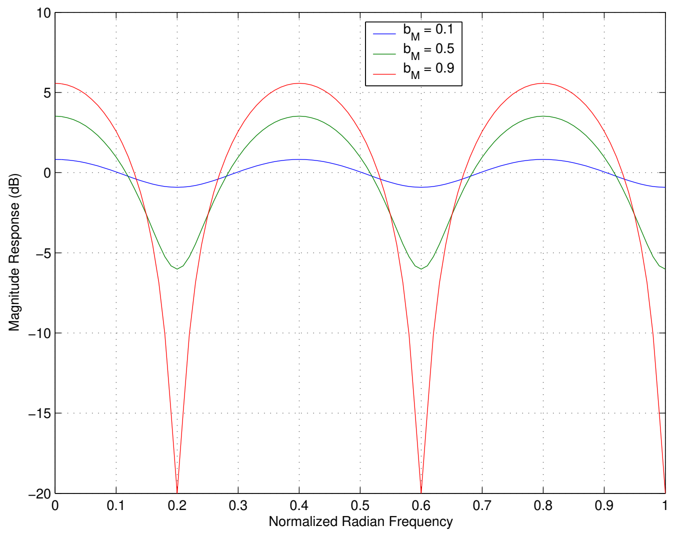

Figure 11: Magnitude response of a feedforward comb filter with $M = 5$, $b_0 = 1$, and $b_M = 0.1$, 0.5, and 0.9.

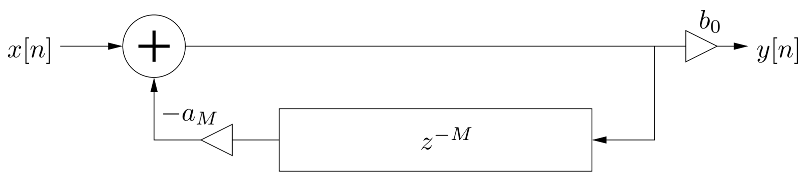

Figure 12: A feedback comb filter block diagram.

- The magnitude response of a feedback comb filter, calculated with the Matlab script combs.m, is shown in Fig. 13 for $M = 5$, $b_0 = 1$, and a_M = -0.1, -0.5, and -0.9.

2.5 Delay Line Interpolation: Linear Interpolation

- The previous discussion was based on the use of digital delay lines with integer sample delay lengths. In many contexts, it is necessary to compute the output of a delay line at fractional delay-line lengths.
- While signal values in a delay line are stored at integer multiples of the sample period T_s, it is possible to apply interpolation techniques to approximate values between those in memory.
- The most common interpolation techniques used in conjunction with delay lines are linear interpolation and allpass interpolation.
- Linear interpolation is an efficient technique for determining sample values at fractional delay line lengths. Intermediate values between two neighboring samples are found by "drawing" a line that connects those two samples and returning values along that line.
- Let Δ be a number between 0 and 1 that indicates how far to interpolate a signal y between time n and time $n-1$. The linearly interpolated value $\hat{y}[n-\Delta]$ is given by:

$$\begin{aligned}\hat{y}[n-\Delta] &= (1-\Delta)\cdot y[n] + \Delta\cdot y[n-1] \\ &= y[n] + \Delta\cdot(y[n-1]-y[n]).\end{aligned}$$

- A block diagram for linear interpolation is shown in Fig. 14.

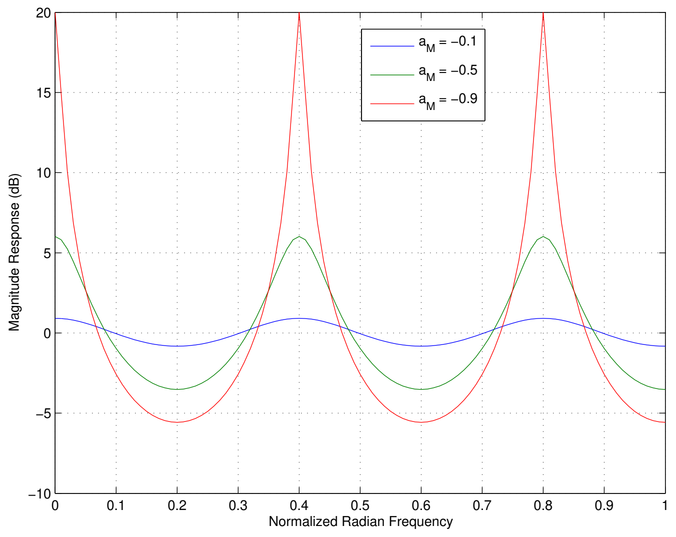

Figure 13: Magnitude response of a feedback comb filter with $M = 5$, $b_0 = 1$, and a_M = -0.1, -0.5, and -0.9.

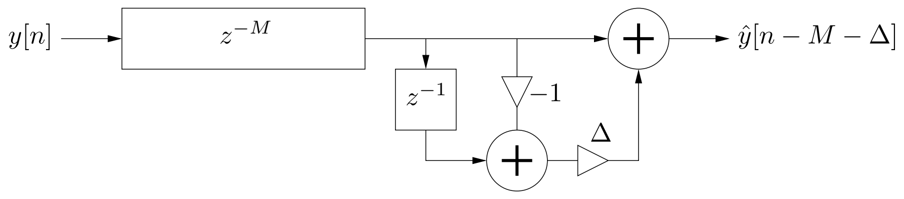

Figure 14: A linearly interpolated delay line.

- The magnitude and phase delay for a linear interpolation filter, calculated with the Matlab script interpolate.m, is shown in Fig. 15 for fractional delays between 0 and 1.
- From the frequency response plots, it is apparent that linear interpolation becomes less accurate for high frequencies. In particular, the "artificial" attenuation that is introduced at higher frequencies can be problematic in feedback loops where the loop gain should be close to 1.
- Linear interpolation is also known as first-order Lagrange interpolation. Lagrange interpolators are FIR filter structures. Higher order Lagrange interpolators can be more accurate but are less efficient to implement.

2.6 Delay Line Interpolation: Allpass Interpolation

- An allpass filter has unity magnitude response but variable phase delay properties.
- The difference equation for a first-order allpass interpolation filter is given by

$$\begin{aligned} \hat{x}[n-\Delta] \triangleq y[n] &= a \cdot x[n] + x[n-1] - a \cdot y[n-1] \\ &= a \cdot (x[n] - y[n-1]) + x[n-1]. \end{aligned}$$

- A block diagram for a first-order allpass interpolation filter is shown in Fig. 16.
- The frequency response of the first-order allpass interpolation filter is

$$H(z) = \frac{a + z^{-1}}{1 + az^{-1}}.$$

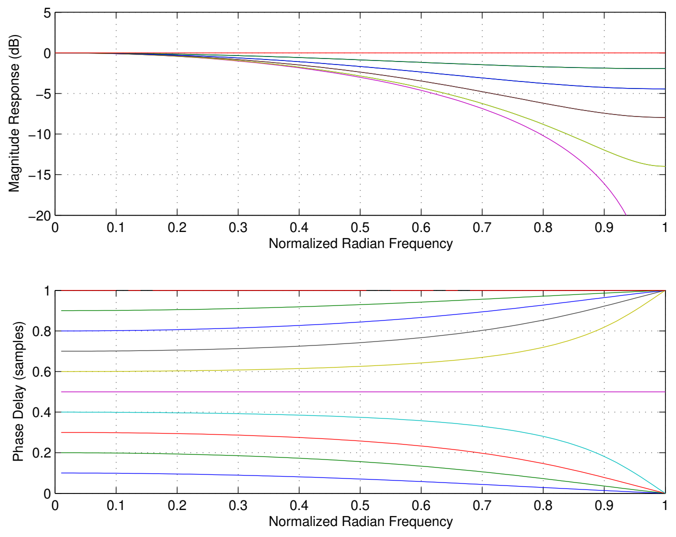

Figure 15: Magnitude and phase delay for linear interpolation with fractional delays between 0 and 1.

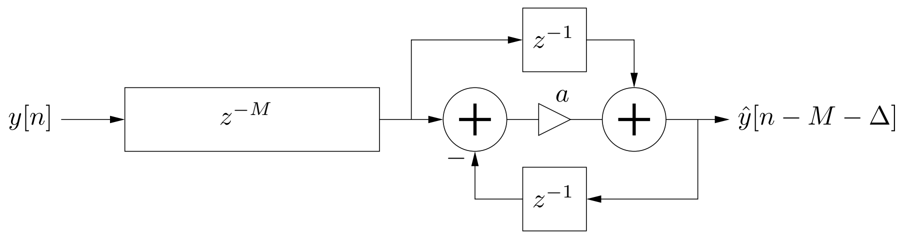

Figure 16: A first-order allpass interpolated delay line.

At low frequencies (as $z \rightarrow 1$), the delay becomes

$$\Delta \approx \frac{1-a}{1+a}.$$

- Figure 17 shows the phase delay, calculated with the Matlab script interpolate.m, of the first-order allpass filter for fractional delay values between 0.1 and 2.

- For a given desired fractional delay Δ, the allpass coefficient a is determined as

$$a = \frac{1-\Delta}{1+\Delta},$$

  where Δ is best maintained in the range $0.3 \leq \Delta \leq 1.3$ to achieve maximally flat phase delay response together with the fastest decaying impulse response, a desired characteristic when dealing with dynamic delay values and their associated transient responses.

- The transient responses, calculated with the Matlab script aptransient.m, of first-order allpass filters used to implement several different fractional delay values are shown in Fig. 18. Note how values of Δ closer to 0.0 have significantly longer transient "tails".

- First-order allpass interpolation is the most simple case of Thiran allpass interpolation.

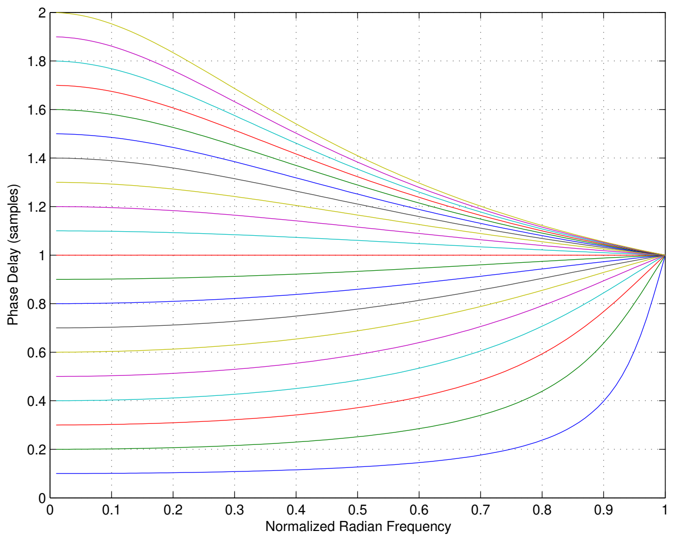

Figure 17: Phase delay for allpass interpolation with fractional delays between 0.1 and 2.

2.7 Tapped Delay Lines

- A tapped delay line is a delay line that provides access to its contents at arbitrary intermediate delay length values.
- The block diagram of Fig. 19 illustrates a tap delay line of total length M_2 samples and a tap located at a delay of M_1 samples. In general, a tap delay line is implemented as a single buffer in memory.
- Tap delay lines can be interpolating or non-interpolating.
- Tap delay lines can efficiently model multiple echoes from the same sound source and are useful in artificial reverberation.

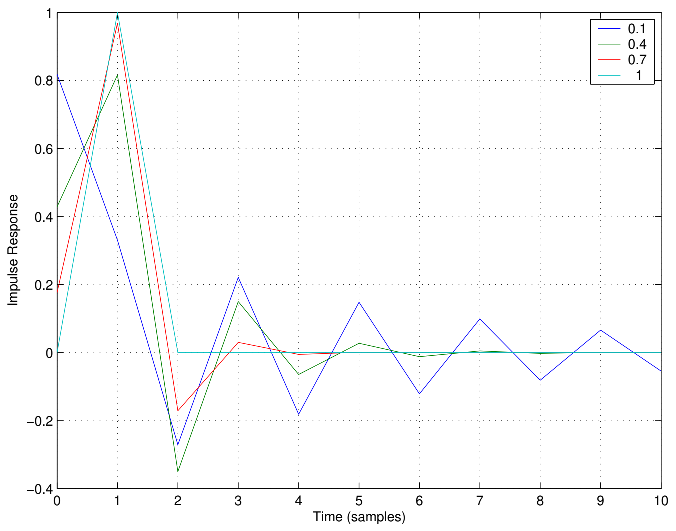

Figure 18: Impulse response of first-order allpass filters with fractional delay settings of $\Delta = 0.1$, 0.4, 0.7, and 1.0.

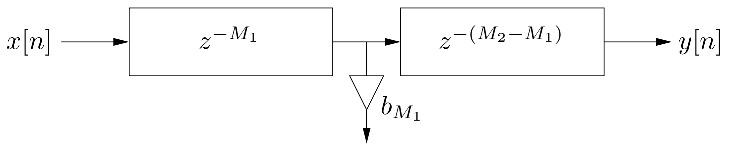

Figure 19: A tap delay line of total length M_2 and a tap at M_1.