Computational Acoustic Modeling with Digital Delay

Julius Smith and Nelson Lee

RealSimple Project*
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305

June 5, 2008

Outline

- Delay lines
- Echo simulation
- Comb filters
- Vector Comb Filters (Feedback Delay Networks)
- Tapped Delay Lines and FIR Filters
- Allpass filters

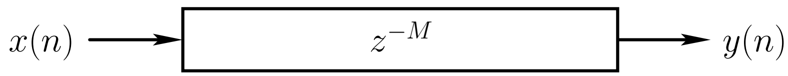
*Work supported by the Wallenberg Global Learning Network

Delay lines

Delay lines are important building blocks for many audio effects and synthesis algorithms, including

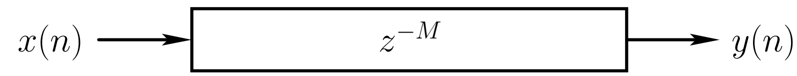
- Digital audio effects
  - Phasing
  - Flanging
  - Chorus
  - Leslie
  - Reverb
- Physical modeling synthesis
  - Acoustic propagation delay (echo, multipath)
  - Vibrating strings (guitars, violins, ...)
  - Woodwind bores
  - Horns
  - Percussion (rods, membranes)

The M-Sample Delay Line



- $y(n) = x(n - M)$, $n = 0, 1, 2, \ldots$
- Must define $x(-1), x(-2), \ldots, x(-M)$ (usually zero)

Delay Line as a Digital Filter



Difference Equation

$$y(n) = x(n - M)$$

Transfer Function

$$H(z) = z^{-M}$$

- M poles at $z = 0$
- M zeros at $z = \infty$

Frequency Response

$$H(e^{j\omega T}) = e^{-jM\omega T}, \quad \omega T \in [-\pi, \pi)$$

- “Allpass” since $\left|H(e^{j\omega T})\right| = 1$
- “Linear Phase” since $\angle H(e^{j\omega T}) = -M\omega T = \alpha\omega$

Delay Line in C

C Code:

```
static double D[M]; /* initialized to zero */
static long ptr=0;  /* read-write offset */

double delayline(double x)
{
  double y = D[ptr]; /* read operation */
  D[ptr++] = x;      /* write operation */
  if (ptr >= M) { ptr -= M; } /* wrap ptr */
  return y;
}
```

- *Circular buffer* in software
- Shared read/write pointer
- Length not easily modified in real time
- Internal state ("instance variables") = length M array + read pointer

Ideal Traveling-Wave Simulation

$$x(n) \longrightarrow \boxed{\quad z^{-M} \quad} \longrightarrow y(n)$$

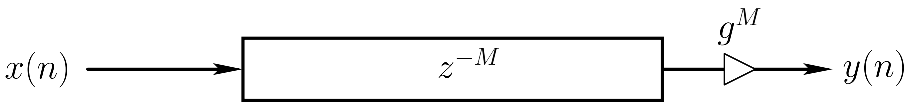
Acoustic Plane Waves in Air

- $x(n) =$ *excess pressure* at time nT, at some fixed point $p_x \in \mathbf{R}^3$ through which a *plane wave* passes
- $y(n) =$ excess pressure at time nT, for a point p_y which is McT meters "downstream" from p_x along the direction of travel for the plane wave, where
  - T denotes the *time sampling interval* in seconds
  - c denotes the *speed of sound* in meters per second
  - In one temporal sampling interval (T seconds), sound travels one spatial sample ($X = cT$ meters)

Transverse Waves on a String

- $x(n) =$ *displacement* at time nT, for some point on the string
- $y(n) =$ transverse displacement at a point McT meters away on the string

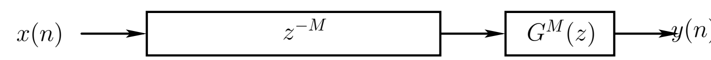
Lossy Traveling-Wave Simulator



- Propagation delay $= M$ samples
- Attenuation $= g^M < 1$ is *lumped* at one point along the ray
- *Exponential decay* in direction of wave travel
- *Distributed attenuation* is *lumped* at one point
- Input/output simulation is *exact* at the sampling instants
- Only deviation from ideal is that simulation is *bandlimited*

Traveling-Wave Simulation with Frequency-Dependent Losses

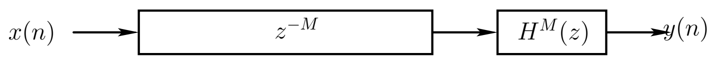
In all acoustic systems of interest, propagation losses *vary with frequency*.



- Propagation delay $= M$ samples $+$ filter delay
- Attenuation $= \left|G(e^{j\omega T})\right|^M$
- Filter is linear and time-invariant (LTI)
- Propagation delay and attenuation can now vary with frequency
- For physical passivity, we require
$$\left|G(e^{j\omega T})\right| \leq 1$$
for all ω.

Dispersive Traveling-Wave Simulation

In many acoustic systems, such as piano strings, wave propagation is also *dispersive*.



This is simulated using a filter having *nonlinear phase*.

For lossless, dispersive wave propagation, the filter is "allpass," i.e.,

$$\left|H(e^{j\omega T})\right| \equiv 1, \ \forall\omega$$

Note that a delay line is a special case of an allpass filter:

$$\left|e^{j\omega MT}\right| \equiv 1, \ \forall\omega$$

Allpass Filters

In general, (finite-order) allpass filters can be written as

$$H(z) = e^{j\phi} z^{-K} \frac{\tilde{A}(z)}{A(z)}$$

where

$$\begin{aligned} A(z) &= 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N} \\ \tilde{A}(z) &\stackrel{\Delta}{=} z^{-N}\overline{A}(z^{-1}) \\ &\stackrel{\Delta}{=} \overline{a}_N + \overline{a}_{N-1} z^{-1} + \cdots + \overline{a}_1 z^{-(N-1)} + \cdots + z^{-N} \end{aligned}$$

The polynomial $\tilde{A}(z)$ can be obtained by reversing the order of the coefficients in $A(z)$ and conjugating them.

Phase Delay and Group Delay

Phase Response:

$$\boxed{\Theta(\omega) \stackrel{\Delta}{=} \angle H(e^{j\omega T})}$$

Phase Delay:

$$\boxed{P(\omega) \stackrel{\Delta}{=} -\frac{\Theta(\omega)}{\omega}} \quad \text{(Phase Delay)}$$

Group Delay:

$$\boxed{D(\omega) \stackrel{\Delta}{=} -\frac{d}{d\omega}\Theta(\omega)} \quad \text{(Group Delay)}$$

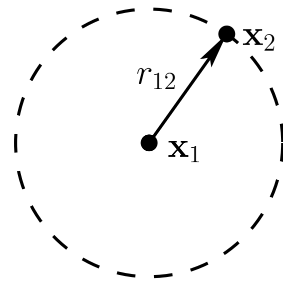
- For a slowly modulated sinusoidal input signal $x(n) = A(nT)\cos(\omega nT + \phi)$, the output signal is

  $$\boxed{y(n) \approx G(\omega)A[nT - D(\omega)] \cdot \cos\{\omega[nT - P(\omega)] + \phi\}}$$

  where $G(\omega) \stackrel{\Delta}{=} |H(e^{j\omega T})|$ is the *amplitude response*.

- *Unwrap* phase response $\Theta(\omega)$ to uniquely define it:
  - $\Theta(0) \stackrel{\Delta}{=} 0$ or $\pm\pi$ for real filters
  - Discontinuities in $\Theta(\omega)$ cannot exceed $\pm\pi$ radians
  - Phase jumps $\pm\pi$ radians are *equivalent*
  - See Matlab function `unwrap`

Acoustic Point Source



- Let $\mathbf{x} = (x, y, z)$ denote the *Cartesian coordinates* of a point in 3D space
- Point source at $\mathbf{x} = \mathbf{x}_1 = (x_1, y_1, z_1)$
- Listening point at $\mathbf{x} = \mathbf{x}_2 = (x_2, y_2, z_2)$
- Propagation distance:

  $$r_{12} = \|\,\mathbf{x}_2 - \mathbf{x}_1\,\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Acoustic pressure peak amplitude (or rms level) at $\mathbf{x} = \mathbf{x}_2$ is given by

$$p(\mathbf{x}_2) = \frac{p_1}{r_{12}}$$

where p_1 is the peak amplitude (or rms level) at $r_{12} = \|\,\mathbf{x}_2 - \mathbf{x}_1\,\| = 1$.

Inverse Square Law for Acoustics

The *intensity* of a sound is proportional to the *square* of its sound pressure p.

Therefore, the *average intensity* at distance r_{12} away from a point source of average-intensity $I_1 \propto < |p_1|^2 >$ is given by

$$I(\mathbf{x}_2) = \frac{I_1}{r_{12}^2}$$

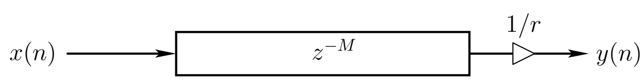
This is a so-called *inverse square law*.

Remember that far away from a finite sound source,

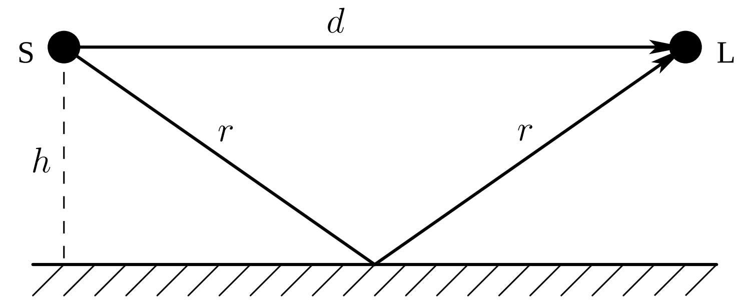
- pressure falls off as $1/r$
- intensity falls off as $1/r^2$

where r is the distance from the source.

Point-to-Point Spherical Pressure-Wave Simulation:



Acoustic Echo



- Source S, Listener L
- Height of S and L above floor is h
- Distance from S to L is d
- Direct sound travels distance d
- Floor-reflected sound travels distance $2r$, where

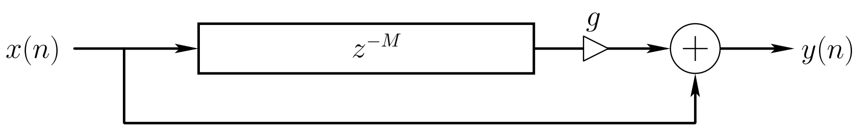
$$r^2 = h^2 + \left(\frac{d}{2}\right)^2$$

- Direct sound and reflection *sum* at listener L

$$p_L(t) \;\propto\; \frac{p_S\left(t - \frac{d}{c}\right)}{d} + \frac{p_S\left(t - \frac{2r}{c}\right)}{2r}$$

- Also called *multipath*

Acoustic Echo Simulator



- Delay line length set to *path-length difference*:

$$M = \frac{2r - d}{cT}$$

where

c = sound speed
T = sampling period

- Gain coefficient g set to *relative attenuation*:

$$g = \frac{1/2r}{1/d} = \frac{d}{2r} = \frac{1}{\sqrt{1 + (2h/d)^2}}$$

- M typically *rounded* to nearest integer
- For non-integer M, delay line must be *interpolated*

STK Program for Digital Echo Simulation

The Synthesis Tool Kit (STK)[1] is an object-oriented C++ tool kit useful for rapid prototyping of real-time computational acoustic models.

```
/* Acoustic echo simulator, main C++ program.
   Compatible with STK version 4.2.1.
   Usage: main inputsoundfile
   Writes main.wav as output soundfile
 */

#include "FileWvIn.h"  /* STK soundfile input support  */
#include "FileWvOut.h" /* STK soundfile output support */
#include "Stk.h"       /* STK global variables, etc.   */

static const int M = 20000;   /* echo delay in samples */
static const StkFloat g = 0.8; /* relative gain factor */

#include "delayline.c" /* defined previously */

int main(int argc, char *argv[])
{
  unsigned long i;
  FileWvIn input(argv[1]); /* read input soundfile */
  FileWvOut output("main"); /* creates main.wav */
  unsigned long nframes = input.getSize();
  for (i=0;i<nframes+M;i++)   {
    StkFloat insamp = input.tick();
    output.tick(insamp + g * delayline(insamp));
  }
}
```

[1] http://ccrma.stanford.edu/CCRMA/Software/STK/

General Loss Simulation

The substitution

$$z^{-1} \leftarrow gz^{-1}$$

in any transfer function *contracts all poles by the factor* g.

Example (delay line):

$$H(z) = z^{-M} \;\rightarrow\; g^M z^{-M}$$

Thus, the contraction factor g can be interpreted as the *per-sample propagation loss factor*.

Frequency-Dependent Losses:

$$z^{-1} \leftarrow G(z)z^{-1}, \quad \left|G(e^{j\omega T})\right| \leq 1$$

$G(z)$ can be considered the *filtering per sample* in the propagation medium. A lossy delay line is thus described by

$$Y(z) = G^M(z) z^{-M} X(z)$$

in the frequency domain, and

$$y(n) = \underbrace{g * g * \ldots * g *}_{M \text{ times}} x(n-M)$$

in the time domain.

Air Absorption

From Moorer 1979:

$$I(r) = I_0 e^{-r/\tau_r}$$

where

$$\begin{aligned} I_0 &= \text{intensity at the source} \\ I(r) &= \text{intensity } r \text{ meters from the plane-source} \\ \tau_r &= \text{intensity decay time constant (meters)} \\ &\quad \text{(depends on frequency, temperature, humidity} \\ &\quad \text{and pressure)} \end{aligned}$$

Relative Humidity	Frequency in Hz 1000	2000	3000	4000
40	5.6	16	30	105
50	5.6	12	26	90
60	5.6	12	24	73
70	5.6	12	22	63

Attenuation in dB per kilometer at 20°C and standard atmospheric pressure.

Acoustic Intensity

Acoustic Intensity may be defined by

$$\boxed{\underline{I} \stackrel{\Delta}{=} p\underline{v}} \qquad \left(\frac{\text{Energy Flux}}{\text{Area} \cdot \text{Time}} = \frac{\text{Power Flux}}{\text{Area}}\right)$$

where

$$\begin{aligned} p &= \text{acoustic pressure} \quad \left(\frac{\text{Force}}{\text{Area}}\right) \\ \underline{v} &= \text{acoustic particle velocity} \quad \left(\frac{\text{Length}}{\text{Time}}\right) \end{aligned}$$

For a *plane traveling wave*, we have

$$\boxed{p = Rv}$$

where

$$R \stackrel{\Delta}{=} \rho c$$

is called the *wave impedance* of air, and

$$\begin{aligned} c &= \text{sound speed} \\ \rho &= \text{mass density of air} \quad \left(\frac{\text{Mass}}{\text{Volume}}\right) \\ v &\stackrel{\Delta}{=} |\underline{v}| \end{aligned}$$

Therefore, in a plane wave,

$$\boxed{I = pv = Rv^2 = \frac{p^2}{R}}$$

Acoustic Energy Density

The two forms of energy in a wave are *kinetic* and *potential*:

$$\begin{aligned} w_v &= \frac{1}{2}\rho v^2 = \frac{1}{2c}Rv^2 \quad \left(\frac{\text{Energy}}{\text{Volume}}\right) \\ w_p &= \frac{1}{2}\frac{p^2}{\rho c^2} = \frac{1}{2c}\frac{p^2}{R} \quad \left(\frac{\text{Energy}}{\text{Volume}}\right) \end{aligned}$$

These are called the *acoustic kinetic and potential energy densities*, respectively.

In a plane wave, where $p = Rv$ and $I = pv$, we have

$$\begin{aligned} w_v &= \frac{1}{2c}Rv^2 = \frac{1}{2} \cdot \frac{I}{c} \\ w_p &= \frac{1}{2c}\frac{p^2}{R} = \frac{1}{2} \cdot \frac{I}{c} \end{aligned}$$

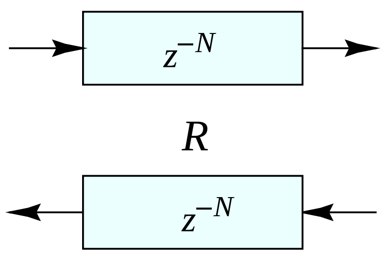
Thus, half of the acoustic intensity I in a plane wave is kinetic, and the other half is potential:[2]

$$\frac{I}{c} = w = w_v + w_p = 2w_v = 2w_p$$

Note that acoustic intensity I has units of *energy per unit area per unit time* while the acoustic energy density $w = I/c$ has units of *energy per unit volume*.

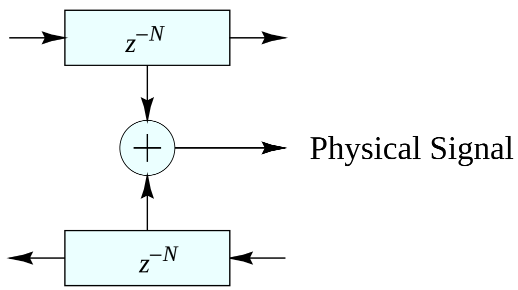
[2] This was first pointed out by Rayleigh.

Digital Waveguide



- A (lossless) *digital waveguide* is a *bidirectional delay line* at some wave impedance R
- Each delay line contains a *sampled acoustic traveling wave*
  - right-going wave on top
  - left-going wave on bottom
- *Losses* and *dispersion* handled with add-on digital filters

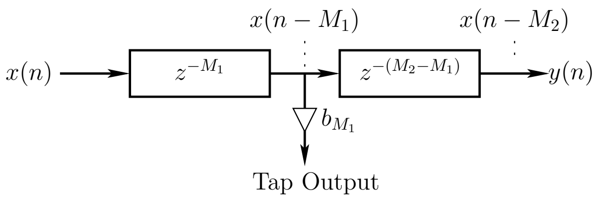
Physical Outputs of a Digital Waveguide



- Any 1D acoustic field is given by the *sum* of these delay lines
  - vibrating strings
  - woodwind bores
  - pipes
  - horns
- Physical output signals (force, pressure, velocity, ...) are obtained by *summing* the left- and right-going *traveling-wave components*
- Delay lines need *taps* for forming physical outputs

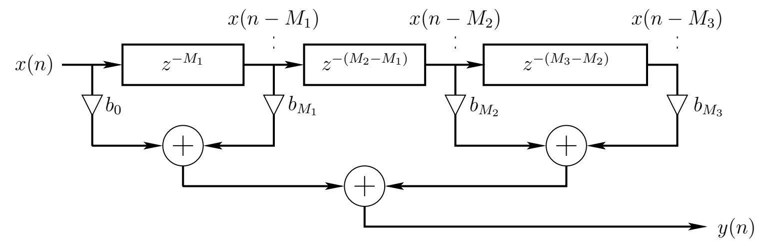
Tapped Delay Lines (TDL)

- A *tapped delay line* (TDL) is a delay line with at least one "tap".
- A *tap* brings out and scales a signal inside the delay line.
- A tap may be *interpolating* or *non-interpolating*.



- TDLs efficiently simulate *multiple echoes* from the *same source*.
- Extensively used in *artificial reverberation*.

Example Tapped Delay Line



- Two internal taps
- Total delay is M_3 samples
- Taps at M_1 and M_2 samples

Difference equation:

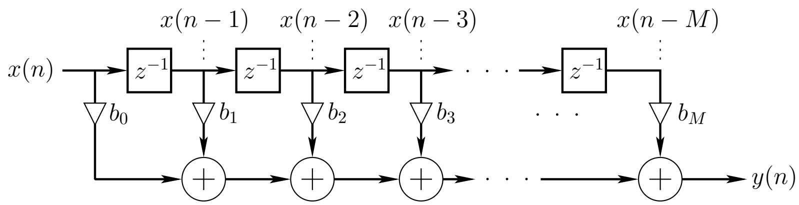
$$y(n) = b_0 x(n) + b_{M_1} x(n - M_1) + b_{M_2} x(n - M_2) + b_{M_3} x(n - M_3)$$

Transfer function:

$$H(z) = b_0 + b_{M_1} z^{-M_1} + b_{M_2} z^{-M_2} + b_{M_3} z^{-M_3}$$

General Causal FIR Filters

The most general case of a TDL having a tap after *every* delay element is the general causal *finite impulse response (FIR)* filter,



- Causal
  ($y(n)$ may not depend on *"future"* input samples $x(n+1)$, $x(n+2)$, etc.)
- Also called a *transversal filter*.

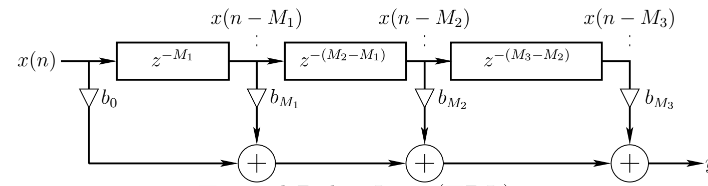
Difference Equation:

$$y(n) = b_0x(n) + b_1x(n-1) + b_2x(n-2) + b_3x(n-3) + \cdots + b_Mx(n-M)$$

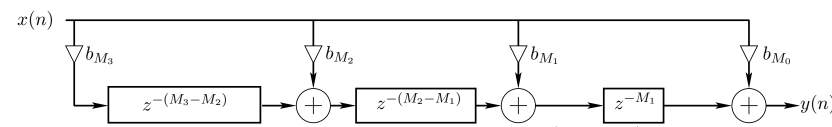
Transfer Function:

$$H(z) = b_0 + b_1z^{-1} + b_2z^{-2} + b_3z^{-3} + \cdots + b_Mz^{-M} = \sum_{m=0}^{M} b_mz^{-m} \triangleq B(z)$$

Transposed Tapped Delay Line (TTDL)



Tapped Delay Line (TDL).



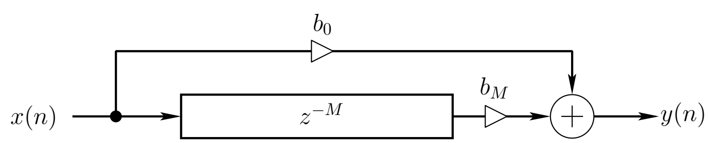
Transposed Tapped Delay Line (TTDL).

A flow-graph is *transposed* (or "reversed") by reversing all signal paths.

- Branchpoints become sums
- Sums become branchpoints
- Input/output exchanged
- Transfer function *identical* for SISO systems
- Derives from *Mason's gain formula*
- Transposition converts direct-form I, II digital filters to two more direct forms

Comb Filters

Feedforward Comb Filter



$$\begin{aligned} b_0 &= \text{Feedforward coefficient} \\ b_M &= \text{Delay output coefficient} \\ M &= \text{Delay-line length in samples} \end{aligned}$$

Difference Equation

$$y(n) = b_0 x(n) + b_M x(n-M)$$

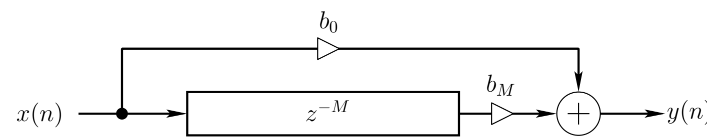
Transfer Function

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

Frequency Response

$$H(e^{j\omega T}) = b_0 + b_M e^{-jM\omega T}$$

Gain Range for Feedforward Comb Filter



For a sinewave input, with $b_0, b_M > 0$:

- Gain is maximum $(b_0 + b_M)$ when a *whole number of periods* fits in M samples:

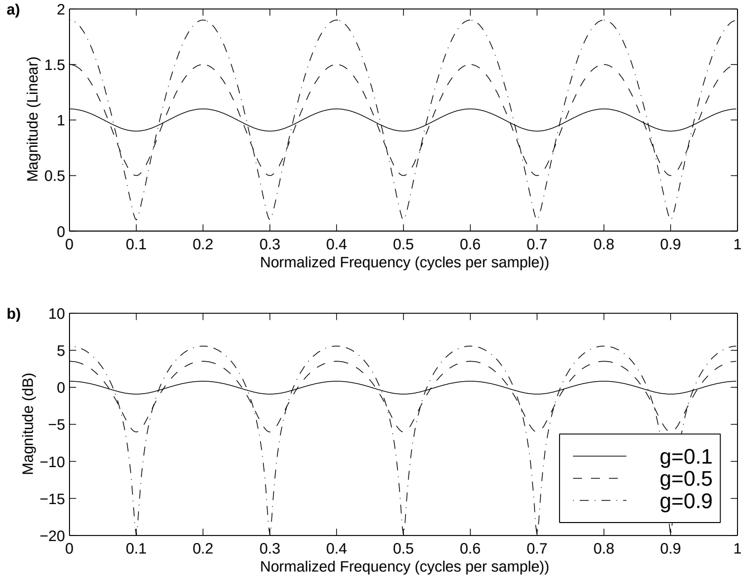
  $$\omega_k T = k\frac{2\pi}{M}, \quad k = 0, 1, 2, \ldots$$

  Note: These are the *DFT*$_M$ *basis frequencies*

- Gain is minimum $(|b_0 - b_M|)$ when an *odd number of half-periods* fits in M samples:

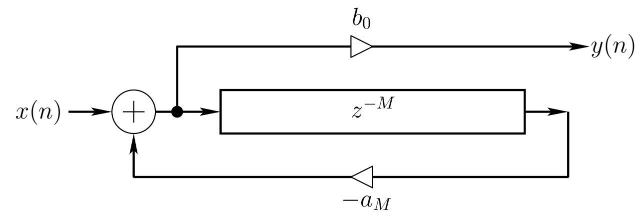
  $$\omega_k T = (2k+1)\frac{\pi}{M}, \quad k = 0, 1, 2, \ldots$$

Feed-Forward Comb-Filter Amplitude Response



- Linear (top) and decibel (bottom) amplitude scales
- $H(z) = 1 + gz^{-M}$
  - $M = 5$
  - $g = 0.1, 0.5, 0.9$
- $G(\omega) \stackrel{\Delta}{=} \left|H(e^{j\omega T})\right| = \left|1 + ge^{-jM\omega T}\right| \to 2\cos(M\omega T/2)$ when $g = 1$
- In *flangers*, these nulls slowly move with time

Feedback Comb Filter



$$\begin{aligned} -a_M &= \text{Feedback coefficient (need } |a_M| < 1 \text{ for stability)} \\ M &= \text{Delay-line length in samples} \end{aligned}$$

Direct-Form-II Difference Equation (see above figure)

$$\begin{aligned} v(n) &= x(n) - a_M y(n-M) \\ y(n) &= b_0 v(n) \end{aligned}$$

Direct-Form-I Difference Equation (commute gain b_0 to input)

$$y(n) = b_0 x(n) - a_M y(n-M)$$

Transfer Function

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

Frequency Response

$$H(e^{j\omega T}) = \frac{b_0}{1 + a_M e^{-jM\omega T}}$$

Simplified Feedback Comb Filter

Consider the special case $b_0 = 1$, $-a_M = g \;\Rightarrow$

$$\begin{aligned} y(n) &= x(n) + g\,y(n-M) \\ H(z) &= \frac{1}{1 - g\,z^{-M}} \end{aligned}$$

- Impulse response is a *series* of echoes, exponentially decaying and uniformly spaced in time
- Models a *plane wave between parallel walls*
- Models a *displacement wave on a guitar string*
- g = *round-trip attenuation*
  - two wall-to-wall traversals
  - two wall reflections

Simplified Feedback Comb Filter, Cont'd

For a sinewave input and $0 < g < 1$:

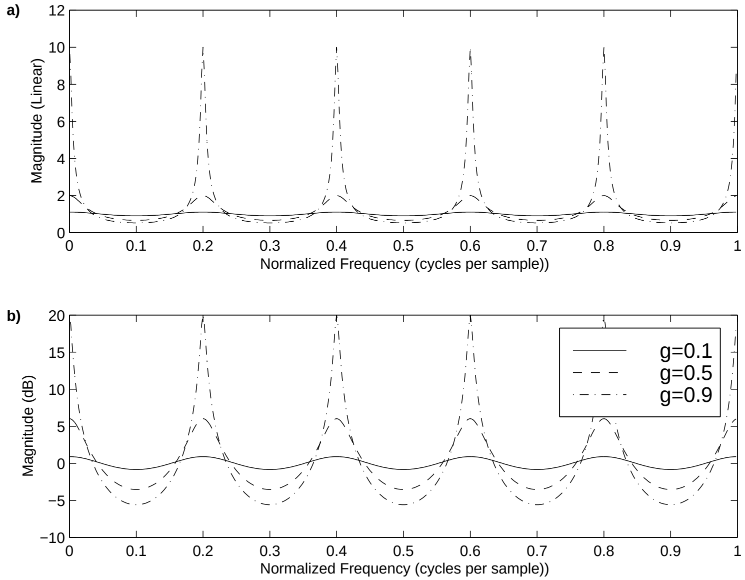
- Gain is maximum $[1/(1-g)]$ when a whole number of periods fits in M samples:

  $$\omega_k T = k\frac{2\pi}{M}, \quad k = 0, 1, 2, \ldots$$

  These are again the *DFT*$_M$ *basis frequencies*
- Gain is minimum $[1/(1+g)]$ when an odd number of half-periods fits in M samples:

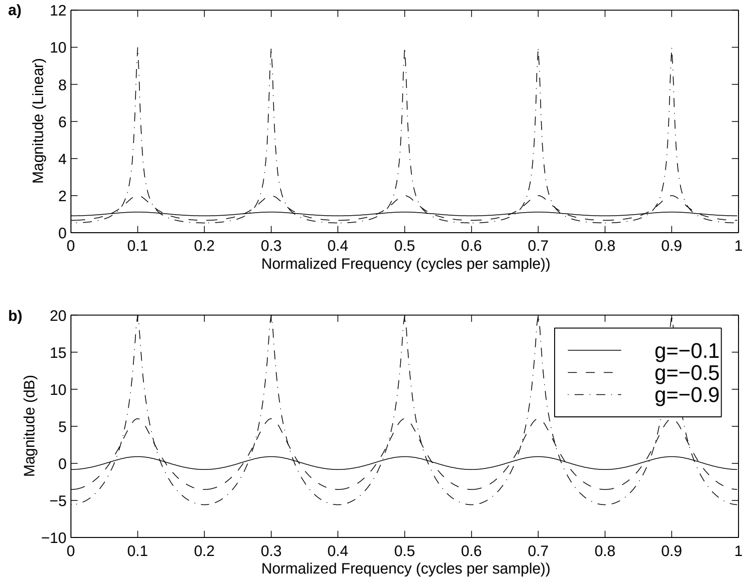
  $$\omega_k T = (2k+1)\frac{\pi}{M}, \quad k = 0, 1, 2, \ldots$$

Feed-Back Comb-Filter Amplitude Response



- Linear (top) and decibel (bottom) amplitude scales
- $H(z) = \frac{1}{1-gz^{-M}}$
- $M = 5, \quad g = 0.1, 0.5, 0.9$
- $G(\omega) \stackrel{\Delta}{=} \left|H(e^{j\omega T})\right| = \left|\frac{1}{1-ge^{-jM\omega T}}\right| \underset{g=1}{\rightarrow} \frac{1}{2\sin\left(\frac{M}{2}\omega T\right)}$

Inverted-Feed-Back Comb-Filter Amplitude Response



- Linear (top) and decibel (bottom) amplitude scales
- $H(z) = \frac{1}{1-gz^{-M}}$
- $M = 5, \quad g = -0.1, -0.5, -0.9$
- $G(\omega) \stackrel{\Delta}{=} \left|H(e^{j\omega T})\right| = \left|\frac{1}{1-ge^{-jM\omega T}}\right| \underset{g=-1}{\rightarrow} \frac{1}{2\cos\left(\frac{M}{2}\omega T\right)}$

Equivalence of Comb Filters to Tapped Delay Lines

We can easily show that a *parallel combination* of feedforward comb filters is equivalent to a tapped delay line:

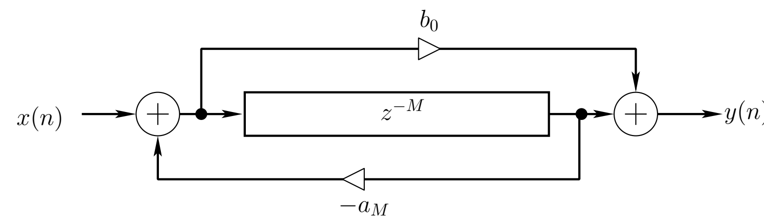
$$
\begin{aligned}
H(z) &= \left(1+g_1 z^{-M_1}\right)+\left(1+g_2 z^{-M_2}\right)+\left(1+g_3 z^{-M_3}\right)\\
&= 3+g_1 z^{-M_1}+g_2 z^{-M_2}+g_3 z^{-M_3}\\
&\Rightarrow \quad b_0=3,\ b_{M_1}=g_1,\ b_{M_2}=g_2,\ b_{M_3}=g_3
\end{aligned}
$$

We can also show that a *series combination* of feedforward comb filters produces a *sparse tapped delay line*:

$$
\begin{aligned}
H(z) &= \left(1+g_1 z^{-M_1}\right)\left(1+g_2 z^{-M_2}\right)\\
&= 1+g_1 z^{-M_1}+g_2 z^{-M_2}+g_1 g_2 z^{-(M_1+M_2)}
\end{aligned}
$$

$$
\begin{aligned}
\Rightarrow \quad b_0 &= 1, \quad b_{M_1}=g_1, \quad b_{M_2}=g_2, b_{M_3}=g_1 g_2\\
M_3 &= M_1+M_2
\end{aligned}
$$

Allpass Filters



- Used extensively in artificial reverberation
- Can be "vectorized" like comb filters (Gerzon '76)
- Transfer function:

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

- To obtain an allpass filter, set $b_0=\overline{a_M}$

  Proof:

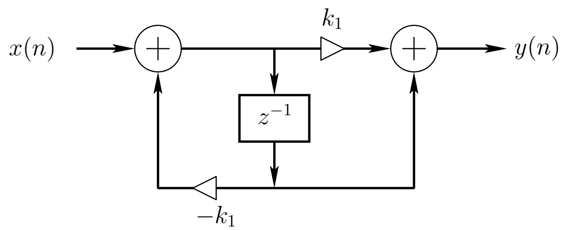
$$
\begin{aligned}
\left|H(e^{j\omega T})\right| &= \left|\frac{\overline{a}+e^{-jM\omega T}}{1+ae^{-jM\omega T}}\right| = \left|\frac{\overline{a}+e^{-jM\omega T}}{e^{jM\omega T}+a}\right|\\
&= \left|\frac{\overline{a+e^{jM\omega T}}}{a+e^{jM\omega T}}\right| = 1
\end{aligned}
$$

First-Order Allpass Filter

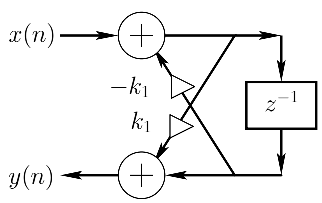
Transfer function:

$$H_1(z) = S_1(z) \triangleq \frac{k_1 + z^{-1}}{1 + k_1 z^{-1}}$$

(a)



(b)



(a) Direct form II filter structure

(b) Two-multiply lattice-filter structure

Nested allpass filter design:

- Any delay-element or delay-line inside a stable allpass-filter can be replaced with any stable allpass-filter

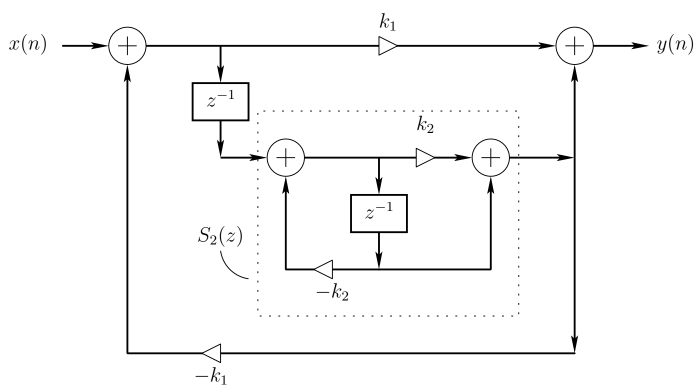
- More generally, any stable allpass can be replaced by any another stable allpass

Nested Allpass Filters

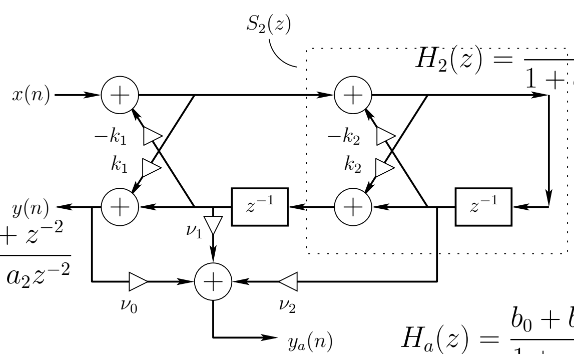
Transfer function:

$$H_2(z) = S_1\left((z^{-1}S_2(z))^{-1}\right) \triangleq \frac{k_1 + z^{-1}S_2(z)}{1 + k_1 z^{-1} S_2(z)}$$

(a)



(b)



$$H_2(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

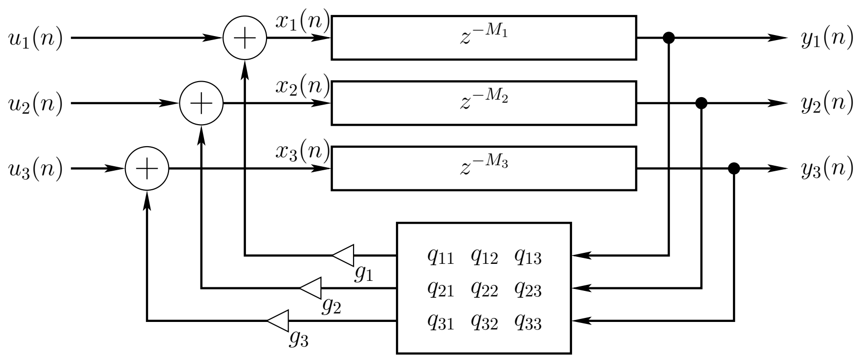
$$H_0(z) = \frac{a_2 + a_1 z^{-1} + z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

$$H_a(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

(a) *Nested* direct-form-II structures

(b) Two-multiply lattice-filter structure (equivalent)

Feedback Delay Network (FDN)



Order 3 MIMO FDN

- "Vectorized Feedback Comb Filter"
- Introduced by Gerzon ("orthogonal matrix feedback") for reverberation applications
- Refined by Stautner, Puckette, and Jot
- Closely related to *state-space representations of LTI systems* = "vectorized one-pole filter"

FDN Time Update

$$\begin{bmatrix} x_1(n) \\ x_2(n) \\ x_3(n) \end{bmatrix} = \begin{bmatrix} g_1 & 0 & 0 \\ 0 & g_2 & 0 \\ 0 & 0 & g_3 \end{bmatrix} \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} \begin{bmatrix} x_1(n-M_1) \\ x_2(n-M_2) \\ x_3(n-M_3) \end{bmatrix} + \begin{bmatrix} u_1(n) \\ u_2(n) \\ u_3(n) \end{bmatrix},$$

Outputs given by

$$\begin{bmatrix} y_1(n) \\ y_2(n) \\ y_3(n) \end{bmatrix} = \begin{bmatrix} x_1(n-M_1) \\ x_2(n-M_2) \\ x_3(n-M_3) \end{bmatrix}$$

In frequency-domain vector notation,

$$\begin{aligned} \mathbf{X}(z) &= \mathbf{\Gamma Q D}(z)\mathbf{X}(z) + \mathbf{U}(z) \\ \mathbf{Y}(z) &= \mathbf{D}(z)\mathbf{X}(z) \end{aligned}$$

where

$$\mathbf{D}(z) \triangleq \begin{bmatrix} z^{-M_1} & 0 & 0 \\ 0 & z^{-M_2} & 0 \\ 0 & 0 & z^{-M_3} \end{bmatrix}$$

Relation of FDNs to State-Space Models

When the delay lines are only 1 sample long, a standard state-space model results:

$$\begin{aligned}\mathbf{x}(n+1) &= \mathbf{A}\mathbf{x}(n) + \mathbf{u}(n)\\ \mathbf{y}(n) &= \mathbf{x}(n)\end{aligned}$$

- The matrix $\mathbf{A} = \mathbf{\Gamma Q}$ is the *state transition matrix*
- The vector $\mathbf{x}(n) = [x_1(n), x_2(n), x_3(n)]^T$ contains the *state variables*
- The state vector $\mathbf{x}(n)$ completely determines the state of the system at time n.
- The *length* of the state vector $\mathbf{x}(n)$ is the *order* of the linear system.

When the delay-lines are not unit length, the state-space description of an FDN expands in a simple way.

Matlab includes many tools for state-space analysis and simulation, especially in the *Control Systems Tool Box*.

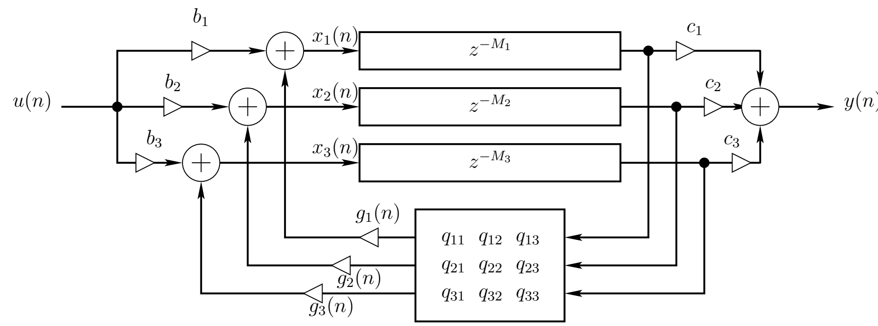
A Single-Input, Single-Output (SISO) FDN

Define

$$\mathbf{u}(n) = \mathbf{B}u(n)$$

where $\mathbf{B}$ is $N \times 1$. Similarly, define

$$y(n) = c_1 x_1(n - M_1) + c_2 x_2(n - M_2) + c_3 x_3(n - M_3)$$



Order 3 SISO Feedback Delay Network

By state-space analysis, the transfer function is

$$H(z) = \mathbf{C}^T \mathbf{D}(z) \left[\mathbf{I} - \mathbf{A}\mathbf{D}(z)\right]^{-1} \mathbf{B}$$

where

$$\mathbf{D}(z) \triangleq \begin{bmatrix} z^{-M_1} & 0 & 0 \\ 0 & z^{-M_2} & 0 \\ 0 & 0 & z^{-M_3} \end{bmatrix}$$

When $M_1 = M_2 = M_3 = 1$, this system can realize *any* transfer function of the form

$$H(z) = \frac{\beta_1 z^{-1} + \beta_2 z^{-2} + \beta_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}.$$

Applications of FDNs

- In FDN reverberation applications, $\mathbf{A} = \mathbf{\Gamma Q}$, where $\mathbf{Q}$ is an orthogonal matrix, and $\mathbf{\Gamma}$ is a diagonal matrix of lowpass filters, each having gain bounded by 1.
- In certain applications, the subset of orthogonal matrices known as *circulant matrices* have advantages.[3]

[3] http://ccrma.stanford.edu/~jos/cfdn/

Stability of FDNs

Stability is assured when some norm of the state vector $\mathbf{x}(n)$ does not increase over time for a zero input signal.

Sufficient condition for stability:

$$\| \mathbf{x}(n+1) \| < \| \mathbf{x}(n) \|,$$

for all $n \geq 0$, where

$$\mathbf{x}(n+1) = \mathbf{A} \begin{bmatrix} x_1(n - M_1) \\ x_2(n - M_2) \\ x_3(n - M_3) \end{bmatrix}.$$

Inequality holds under the L^2 norm whenever the feedback matrix $\mathbf{A}$ satisfies

$$\| \mathbf{A}\mathbf{x} \|_2 < \| \mathbf{x} \|_2$$

where

$$\| \mathbf{x} \|_2 \stackrel{\Delta}{=} \sqrt{x_1^2 + x_2^2 + \cdots + x_N^2}.$$

(the "L^2 norm") $\Leftrightarrow \| \mathbf{A} \|_2 < 1$

Stable Feedback Matrices

The matrix

$$\mathbf{A} = \mathbf{\Gamma Q}$$

always gives a stable FDN when $\mathbf{Q}$ is an *orthogonal matrix*, and $\mathbf{\Gamma}$ is a diagonal gain matrix having entries less than 1 in magnitude:

$$\mathbf{\Gamma} = \begin{bmatrix} g_1 & 0 & \dots & 0 \\ 0 & g_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & g_N \end{bmatrix}, \quad |g_i| < 1.$$

It is also possible to express FDNs as special cases of digital waveguide networks, in which case stability depends on the network being *passive*. Smith and Rocchesso 1994 This analysis reveals that the FDN is lossless if and only if the feedback matrix $\mathbf{A}$ has *unit-modulus eigenvalues and linearly independent eigenvectors* (see the Rocchesso and Smith 1996[4] for details).

[4]http://ccrma.stanford.edu/~jos/cfdn/Conditions_Losslessness.html