Simple Interpolators suitable for Real Time Fractional Delay Filtering

**Linearly Interpolated Delay Line (1st-Order FIR)**

\[ \hat{y}(n - \eta) = (1 - \eta) \cdot y(n) + \eta \cdot y(n - 1) \]

where \( \eta = \text{desired fractional delay} \).

**Allpass Interpolated Delay Line (1st-Order)**

\[ \Delta \approx \frac{1 - \eta}{1 + \eta} \]

Outline

- **Low-Order (Fast) Interpolators**
  - Linear
  - Allpass
- **High-Order Interpolation**
  - Ideal Bandlimited Interpolation
  - Windowed-Sinc Interpolation
- **High-Order Fractional Delay Filtering**
  - Lagrange
  - Farrow Structure
  - Thiran Allpass
- **Optimal FIR Filter Design for Interpolation**
  - Least Squares
  - Comparison to Lagrange

Linear Interpolation

Simplest of all, and the most commonly used:

\[ \hat{y}(n - \eta) = (1 - \eta) \cdot y(n) + \eta \cdot y(n - 1) \]

where \( \eta = \text{desired fractional delay} \).

One-multiply form:

\[ \hat{y}(n - \eta) = y(n) + \eta \cdot [y(n - 1) - y(n)] \]

- Works best with lowpass signals
  (Natural spectra tend to roll off rapidly)
- Works well with over-sampling
First-Order Allpass Interpolation

\[
\hat{x}(n - \Delta) \triangleq y(n) = \eta \cdot x(n) + x(n - 1) - \eta \cdot y(n - 1)
\]

\[
= \eta \cdot [x(n) - y(n - 1)] + x(n - 1)
\]

\[
H(z) = \frac{\eta + z^{-1}}{1 + \eta z^{-1}}
\]

- Low frequency delay given by
  \[
  \Delta \approx \frac{1 - \eta}{1 + \eta} \quad \text{(exact at DC)}
  \]

- Same complexity as linear interpolation
- Good for delay-line interpolation, not random access
- Best used with fixed fractional delay \(\Delta\)
- To avoid pole near \(z = -1\), use offset delay range, e.g.,
  \[
  \Delta \in [0.1, 1.1] \iff \eta \in [-0.05, 0.82]
  \]

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.
Ideal Bandlimited Interpolation

Ideal interpolation for digital audio is bandlimited interpolation, i.e., samples are uniquely interpolated based on the assumption of zero spectral energy for $|f| \geq f_s/2$.

Ideal bandlimited interpolation is sinc interpolation:

$$y(t) = (y \ast h_s)(t) = \sum_{n=0}^{N-1} y(nT)h_s(t - nT)$$

where

$$h_s(t) \triangleq \text{sinc}(f_s t)$$

$sinc(x) \triangleq \frac{\sin(\pi x)}{\pi x}$

(Proof: sampling theorem)

Applications of Bandlimited Interpolation

Bandlimited Interpolation is used in (e.g.)

- Sampling-rate conversion
- Wavetable/sampling synthesis
- Virtual analog synthesis
- Oversampling D/A converters
- Fractional delay filtering

Fractional delay filtering is a special case of bandlimited interpolation:

- Fractional delay filters only need sequential access $\Rightarrow$ IIR filters can be used
- General bandlimited interpolation requires random access $\Rightarrow$ FIR filters normally used

Fractional Delay Filters are used for (among other things)

- Time-varying delay lines (flanging, chorus, leslie)
- Resonator tuning in digital waveguide models
- Exact tonehole placement in woodwind models
- Beam steering of microphone / speaker arrays

Example Application of Fractional Delay Filtering and Bandlimited Interpolation

Digital Waveguide String Model

- "Pick-up" needs Bandlimited Interpolation
- "Tuning" needs Fractional Delay Filtering

The Sinc Function ("Cardinal Sine")

$$sinc(t) \triangleq \frac{\sin(\pi t)}{\pi t}$$

The sinc function is the impulse response of the ideal lowpass filter which cuts off at half the sampling rate
Ideal D/A Conversion

Each sample in the time domain scales and locates one sinc function in the unique, continuous, bandlimited interpolation of the sampled signal.

Convolving a sampled signal \( y(n) \) with sinc \((n - \eta)\) “evaluates” the signal at an arbitrary continuous time \( \eta \in \mathbb{R} \):

\[
y(\eta) = \sum_{n=0}^{N-1} y(n) \text{sinc}(\eta - n)
= \text{SAMPLE}\{ y \ast \text{SHIFT}_\eta(\delta) \}
\]

Ideal D/A Example

Reconstruction of a bandlimited rectangular pulse \( x(t) \) from its samples \( x = [\ldots, 0, 1, 1, 1, 1, 1, \ldots] \):

Bandlimited Rectangular Pulse Reconstruction

Catch

- Sinc function is infinitely long and noncausal
- Must be available in continuous form

Optimal Least Squares Bandlimited Interpolation Formulated as a Fractional Delay Filter

Note that interpolation is a special case of linear filtering. (Proof: Convolution representation above.)

Consider a filter which delays its input by \( \Delta \) samples:

- Ideal impulse response = bandlimited delayed impulse = delayed sinc
  \[
h_\Delta(t) = \text{sinc}(t - \Delta) \triangleq \frac{\sin[\pi(t - \Delta)]}{\pi(t - \Delta)}
\]
- Ideal frequency response = “brick wall” lowpass response, cutting off at \( f_s/2 \) and having linear phase \( e^{-j\omega\Delta T} \)
  \[
  H_\Delta(e^{j\omega T}) \triangleq \text{DTFT}(h_\Delta) = \begin{cases} e^{-j\omega\Delta}, & |\omega| < \pi f_s \\ 0, & |\omega| \geq \pi f_s \end{cases}
  \]
  \[
  \Rightarrow H_\Delta(e^{j\omega T}) = e^{-j\omega\Delta T}, \quad -\pi \leq \omega T < \pi
  \]
  \[
  \leftrightarrow \text{sinc}(n - \Delta), \quad n = 0, \pm 1, \pm 2, \ldots
  \]

The sinc function is an infinite-impulse-response (IIR) digital filter with no recursive form \( \Rightarrow \) non-realizable

To obtain a finite impulse response (FIR) interpolating filter, let’s formulate a least-squares filter-design problem:

Desired Interpolator Frequency Response

\[
H_\Delta(e^{j\omega T}) = e^{-j\omega\Delta T}, \quad \Delta = \text{Desired delay in samples}
\]

FIR Filter Frequency Response

\[
\hat{H}_\Delta(e^{j\omega T}) = \sum_{n=0}^{L-1} \hat{h}_\Delta(n) e^{-j\omega n T}
\]

Error to Minimize

\[
E(e^{j\omega T}) = H_\Delta(e^{j\omega T}) - \hat{H}_\Delta(e^{j\omega T})
\]

L\(^2\) Error Norm

\[
J(h) \triangleq \| E \|_2^2 = \frac{T}{2\pi} \int_{-\pi/T}^{\pi/T} \left| E(e^{j\omega T}) \right|^2 d\omega
\]

\[
= \frac{T}{2\pi} \int_{-\pi/T}^{\pi/T} \left| H_\Delta(e^{j\omega T}) - \hat{H}_\Delta(e^{j\omega T}) \right|^2 d\omega
\]

By Parseval’s Theorem

\[
J(h) = \sum_{n=0}^{\infty} \left| h_\Delta(n) - \hat{h}_\Delta(n) \right|^2
\]

Optimal Least-Squares FIR Interpolator

\[
\hat{h}_\Delta(n) = \begin{cases} \text{sinc}(n - \Delta), & 0 \leq n \leq L - 1 \\ 0, & \text{otherwise} \end{cases}
\]
Truncated-Sinc Interpolation

Truncate sinc(t) at 5th zero-crossing to left and right of time 0 to get

Frequency Response: Rectangular Window

- Vertical axis in dB, horizontal axis in spectral samples
- Optimal in least-squares sense
- Poor stop-band rejection (∼80 dB)
- "Gibbs Phenomenon" gives too much "ripple"
- Ripple can be reduced by tapering the sinc function to zero instead of simply truncating it.

Windowed Sinc Interpolation

・Sinc function can be windowed more generally to yield

\[ h_{a}(n) = \begin{cases} 
\frac{w(n-\Delta)}{\alpha(n-\Delta)}, & 0 \leq n \leq L-1 \\
0, & \text{otherwise} \end{cases} \]

・Example of window method for FIR lowpass filter design applied to sinc functions (ideal lowpass filters) sampled at various phases (corresponding to desired delay between samples)
・For best results, \( \Delta \approx \frac{L}{2} \)
・w(n) is any real symmetric window (e.g., Hamming, Blackman, Kaiser).
・Non-rectangular windows taper truncation which reduces Gibbs phenomenon, as in FFT analysis

Spectrum of Kaiser-windowed Sinc

Frequency Response: Kaiser Window

- Stopband now starts out close to −80 dB
- Kaiser window has a single parameter which trades off stop-band attenuation versus transition-bandwidth from pass-band to stop-band

Lowpass Filter Design

・In the transition band, frequency response "rolls off" from 1 at \( \omega_c = \omega_s/(2\alpha) \) to zero (or ≈ 0.5) at \( \omega_s/2 \).
・Interpolation can remain "perfect" in pass-band

Online references (FIR interpolator design)

・Music 421 Lecture 2 on Windows
- Music 421 Lecture 3 on FIR Digital Filter Design
- Optimal FIR Interpolator Design

http://ccrma.stanford.edu/~jos/FIR/Windows.html
http://ccrma.stanford.edu/~jos/FIR/WINILT.html
Oversampling Reduces Filter Length

• Example 1:
  - $f_s = 44.1$ kHz (CD quality)
  - Audio upper limit = 20 kHz
  - Transition band = 2.05 kHz
  - FIR filter length $\Delta \approx L_1$

• Example 2:
  - $f_s = 48$ kHz (e.g., DAT)
  - Audio upper limit = 20 kHz
  - Transition band = 4 kHz
  - FIR filter length $\approx \frac{L_1}{2}$

• Required FIR filter length varies inversely with transition bandwidth
  ⇒ Required filter length in example 1 is almost double ($\approx 4/2.1$) the required filter length for example 2

• Increasing the sampling rate by less than ten percent reduces the filter expense by almost fifty percent

The Digital Audio Resampling Home Page

• C++ software for windowed-sinc interpolation
• C++ software for FIR filter design by window method
• Fixed-point data and filter coefficients
• Can be adapted to time-varying resampling
• Open source, free
• First written in 1983 in SAIL
• URL: http://ccrma.stanford.edu/~jos/resample/
• Most needed upgrade:
  - Design and install a set of optimal FIR interpolating filters

Lagrange Interpolation

• Lagrange interpolation is just polynomial interpolation
• $N$th-order polynomial interpolates $N + 1$ points
• First-order case = linear interpolation

Problem Formulation

Given a set of $N + 1$ known samples $f(x_k)$, $k = 0, 1, 2, \ldots, N$, find the unique order $N$ polynomial $g(x)$ which interpolates the samples

Solution (Waring, Lagrange):

$$g(x) = \sum_{k=0}^{N} l_k(x)f(x_k)$$

where

$$l_k(x) \Delta = \frac{(x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_N)}{(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_N)}$$

• Numerator gives a zero at all samples but the $k$th
• Denominator simply normalizes $l_k(x)$ to 1 at $x = x_k$
• As a result,
  $$l_k(x_j) = \delta_{kj} = \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases}$$

• Generalized bandlimited impulse = generalized sinc function:
  Each $l_k(x)$ goes through 1 at $x = x_k$ and zero at all other sample points
  I.e., $l_k(x)$ is analogous to sinc($x - x_k$)

Example Lagrange Basis Functions

Lagrange Basis Polynomials, Order = 8. Random \( x \) (marked by dotted lines)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
</tr>
<tr>
<td>4</td>
<td>0.8</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
</tr>
<tr>
<td>6</td>
<td>1.2</td>
</tr>
<tr>
<td>7</td>
<td>1.4</td>
</tr>
<tr>
<td>8</td>
<td>1.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( n )</th>
<th>( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
</tr>
<tr>
<td>4</td>
<td>0.8</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
</tr>
<tr>
<td>6</td>
<td>1.2</td>
</tr>
<tr>
<td>7</td>
<td>1.4</td>
</tr>
<tr>
<td>8</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Lagrange Interpolation Optimality

In the uniformly sampled case, Lagrange interpolation can be viewed as ordinary FIR filtering:

- Lagrange interpolation filters maximally flat in the frequency domain about dc:
  \[
  \frac{d^m E(e^{j\omega})}{d\omega^m} \bigg|_{\omega=0} = 0, \quad m = 0, 1, 2, \ldots, N,
  \]
  where
  \[
  E(e^{j\omega}) = e^{-j\Delta} - \sum_{n=0}^{N} h(n)e^{-jn\omega}
  \]
  and \( \Delta \) is the desired delay in samples.

- Same optimality criterion as Butterworth filters in classical analog filter design.
- Can also be viewed as “Pade approximation” to a constant frequency response in the frequency domain.

Proof of Maximum Flatness at DC

The maximum flat fractional-delay FIR filter is obtained by equating to zero all \( N+1 \) leading terms in the Taylor (Maclaurin) expansion of the frequency-response error at dc:

\[
0 = \frac{d^m E(e^{j\omega})}{d\omega^m} \bigg|_{\omega=0} = \frac{d^m}{d\omega^m} \left[ e^{-j\Delta} - \sum_{n=0}^{N} h(n)e^{-jn\omega} \right] \bigg|_{\omega=0} = (-j\Delta)^m - \sum_{n=0}^{N} (-jn)^m h(n)
\]

\[
\Rightarrow \sum_{n=0}^{N} n^k h(n) = \Delta^k, \quad k = 0, 1, \ldots, N
\]

This is a linear system of equations of the form \( Vh = \Delta \), where \( V \) is a Vandermonde matrix. The solution can be written as a ratio of Vandermonde determinants using Cramer’s rule. As shown by Cauchy (1812), the determinant of a Vandermonde matrix \( [p_i^{j-i}] \), \( i, j = 1, \ldots, N \) can be expressed in closed form as

\[
[p_i^{j-i}] = \prod_{i<j}(p_j - p_i) = (p_2 - p_1)(p_3 - p_1) \cdots (p_N - p_1) \cdots (p_1 - p_2)(p_4 - p_2) \cdots (p_N - p_2) \cdots (p_{N-1} - p_N-2)(p_N - p_{N-2}) \cdots (p_N - p_{N-1})
\]

Making this substitution in the solution obtained by Cremer’s rule yields that the impulse response of the order \( N \) maximally flat fractional-delay FIR filter may be written in closed form as

\[
h(n) = \prod_{k \neq n}^N \frac{D - k}{n - k}
\]

which coincides with the formula for Lagrange interpolation when the abscissae are equally spaced on the integers from 0 to \( N - 1 \). (Online Reference: Vesa Välimäki’s thesis, Chapter 3, Part 2, pp. 82–84)
Lagrange Interpolator Frequency Responses: Orders 1, 2, and 3

Explicit Formula for Lagrange Interpolation Coefficients

\[ h_\Delta(n) = \prod_{k=0}^{N} \frac{\Delta - k}{n - k}, \quad n = 0, 1, 2, \ldots, N \]

Lagrange Interpolation Coefficients

<table>
<thead>
<tr>
<th>( N )</th>
<th>( h_\Delta(0) )</th>
<th>( h_\Delta(1) )</th>
<th>( h_\Delta(2) )</th>
<th>( h_\Delta(3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 - ( \Delta )</td>
<td>( \Delta )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( \frac{\Delta^2 - \Delta}{2} )</td>
<td>( -\Delta(\Delta - 2) )</td>
<td>( \frac{3\Delta^2}{2} )</td>
<td>( \frac{3\Delta^2}{2} )</td>
</tr>
<tr>
<td>3</td>
<td>( \frac{\Delta^3 - 2\Delta^2 + \Delta}{6} )</td>
<td>( -\Delta(\Delta - 2)(\Delta - 3) )</td>
<td>( \frac{5\Delta^3}{6} )</td>
<td>( \frac{5\Delta^3}{6} )</td>
</tr>
</tbody>
</table>

– For \( N = 1 \), Lagrange interpolation reduces to linear interpolation \( h = [1 - \Delta, \Delta] \), as before
– For order \( N \), desired delay should be in a one-sample range centered about \( \Delta = N/2 \)

Matlab Code For Lagrange Fractional Delay

```matlab
function h = lagrange(N, delay)
%LAGRANGE h=lagrange(N,delay) returns order N FIR filter h which implements given delay.
% N (in samples). For best results, delay should be near N/2 +/- 1.

n = 0:N;
h = ones(1, N+1);
for k = 0:N
    index = find(n ~= k);
h(index) = h(index) * (delay-k) ./ (n(index)-k);
end
```

Relation of Lagrange Interpolation to Windowed Sinc Interpolation

- For an infinite number of equally spaced samples, with spacing \( x_{k+1} - x_k = \Delta \), the Lagrange-interpolation basis polynomials converge to shifts of the sinc function, i.e.,

\[ l_k(x) = \text{sinc}\left(\frac{x - k\Delta}{\Delta}\right), \quad k = \ldots, -2, -1, 0, 1, 2, \ldots \]

Proof: As \( \text{order} \to \infty \), the binomial window \( \to \) Gaussian window \( \to \) constant (unity).

Alternate Proof: Every analytic function is determined by its zeros and its value at one nonzero point. Since \( \text{Sinc}(\pi x) \) is zero on all the integers except 0, and since sinc(0) = 1, it therefore coincides with the Lagrangian basis polynomial for \( N = \infty \) and \( k = 0 \).
Variable FIR Interpolating Filter

Basic idea: Each FIR filter coefficient \( h_n \) becomes a polynomial in the delay parameter \( \Delta \):

\[
h_\Delta(n) \triangleq \sum_{m=0}^{P} c_m(m) \Delta^m, \quad n = 0, 1, 2, \ldots, N
\]

\(\Longleftrightarrow H_\Delta(z) \triangleq \sum_{n=0}^{N} h_\Delta(n) z^{-n} \)

\[
= \sum_{n=0}^{N} \left( \sum_{m=0}^{P} c_m(m) \Delta^m \right) z^{-n} \\
= \sum_{m=0}^{P} \left( \sum_{n=0}^{N} c_m(m) z^{-n} \right) \Delta^m \\
= \sum_{m=0}^{P} C_m(z) \Delta^m
\]

- More generally: \( H_\Delta(x) = \sum_{m=0}^{P} \alpha(\Delta) C_m(z) \) where \( \alpha(\Delta) \) is provided by a table lookup
- Basic idea applies to any one-parameter filter variation
- Also applies to time-varying filters (\( \Delta \leftarrow t \))

Farrow Structure for Variable Delay FIR Filters

When the polynomial in \( \Delta \) is evaluated using Horner’s rule,

\[
\hat{X}_{\Delta} \Delta(z) = X + \Delta \left[ C_1 \Delta X + \Delta \left[ C_2 \Delta X + \ldots \right] \right],
\]

the filter structure becomes

\[
x(n) \rightarrow C_1(z) \rightarrow \Delta \rightarrow \ldots \rightarrow \Delta \rightarrow C_N(z) \rightarrow (n - \Delta)
\]

As delay \( \Delta \) varies, “basis filters” \( C_i(z) \) remain fixed
\(\Rightarrow\) very convenient for changing \( \Delta \) over time

Farrow Structure Design Procedure

Solve the \( N_\Delta \) equations

\[
z^{-\Delta} = \sum_{i=0}^{N_\Delta} C_i(z) \Delta_i, \quad i = 1, 2, \ldots, N_\Delta
\]

for the \( N + 1 \) FIR transfer functions \( C_i(z) \), each order \( N_i \) in general

References: Laakso et al., Farrow

Thiran Allpass Interpolators

Given a desired delay \( \Delta = N + \delta \) samples, an order \( N \) allpass filter

\[
H(z) = \frac{z^{-N} A(z^{-1})}{A(z)} = \frac{a_N + a_{N-1} z^{-1} + \cdots + a_1 z^{-(N-1)} + z^{-N}}{1 + a_1 z^{-1} + \cdots + a_{N-1} z^{-(N-1)} + a_N z^{-N}}
\]

can be designed having maximally flat group delay equal to \( \Delta \) at DC using the formula

\[
a_k = (-1)^k \binom{N}{k} \prod_{n=0}^{N} \frac{\Delta - N + n}{\Delta - N + k + n}, \quad k = 0, 1, 2, \ldots, N
\]

where

\[
\binom{N}{k} = \frac{N!}{k!(N-k)!}
\]

denotes the \( k \)th binomial coefficient

- \( a_0 = 1 \) without further scaling
- For sufficiently large \( \Delta \), stability is guaranteed
- Rule of thumb: \( \Delta \approx \text{order} \)
- Mean group delay is always \( N \) samples
- For any stable \( N \)-th order allpass filter:

\[
\frac{1}{2\pi} \int_{-\pi}^{\pi} D(\omega) d\omega \triangleq -\frac{1}{2\pi} \int_{-\pi}^{\pi} \Theta(\omega) d\omega = -\frac{1}{2\pi} \left( \Theta(2\pi) - \Theta(0) \right) = N
\]

- Only known closed-form case for allpass interpolators of arbitrary order
- Effective for delay-line interpolation needed for tuning since pitch perception is most acute at low frequencies.

Frequency Responses of Thiran Allpass Interpolators for Fractional Delay

![Thiran Interpolating Filters, Del=Order=0.3, Order=[1,2,3,5,10,20](x)](image-url)
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.

- Computation doubled during cross-fade
- Cross-fade should be long enough to sound smooth
- Not a true “morph” from one delay length to another, since we do not pass through the intermediate delay lengths.
- A single delay line can be shared such that the cross-fade occurs from one read-pointer (plus associated filtering) to another.

L-Infinity (Chebyshev) Fractional Delay Filters

- Use Linear Programming (LP) for real-valued $L_\infty$-norm minimization
- Remez exchange algorithm (remez, cremez)
- In the complex case, we have a problem known as a Quadratically Constrained Quadratic Program
- Approximated by sets of linear constraints (e.g., a polygon can be used to approximate a circle)
- Can solve with code developed by Prof. Boyd’s group
- See Mohonk-97 paper for details.

Chebyshev FD-FIR Design Example

![Fractional delay min–max filters](image1)

![Modulus of the Error – (infinity norm)](image2)
Comparison of Lagrange and Optimal Chebyshev Fractional-Delay Filter Frequency Responses

Interpolation Summary

<table>
<thead>
<tr>
<th>Order</th>
<th>FIR</th>
<th>IIR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Linear</td>
<td>Alipass_1</td>
</tr>
<tr>
<td>Large N</td>
<td>Lagrange</td>
<td>Windowed Sinc</td>
</tr>
<tr>
<td>( \infty )</td>
<td>Windowed Sinc</td>
<td>Sinc</td>
</tr>
</tbody>
</table>