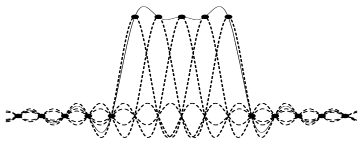
Digital Audio Resampling Home Page



Abstract

This document describes digital audio sampling-rate conversion and related concepts. Open-source software is provided, and pointers are given to related projects and papers.

Contents

1 What is Bandlimited Interpolation?

Bandlimited interpolation of discrete-time signals is a basic tool having extensive application in digital signal processing. In general, the problem is to correctly compute signal values at arbitrary continuous times from a set of discrete-time samples of the signal amplitude. In other words, we must be able to interpolate the signal between samples. Since the original signal is always assumed to be *bandlimited* to half the sampling rate, (otherwise aliasing distortion would occur upon sampling), *Shannon's sampling theorem* tells us the signal can be exactly and uniquely reconstructed for all time from its samples by bandlimited interpolation.

There are many methods for interpolating discrete points. For example, *Lagrange interpolation* is the classical technique of finding an order N polynomial which passes through $N+1$ given points.

The technique known as *cubic splines* fits a third-order polynomial through two points so as to achieve a certain slope at one of the points. (This allows for a smooth chain of third-order polynomial passing through a set of points.)

You may also have heard of *Bezier splines* which interpolate a set of points using smooth curves which don't necessarily pass through the points. (Bezier curves are commonly used in graphics and drawing programs, such as Adobe Illustrator.)

The above methods are suitable for graphics and other uses, but they are not ideal for digital audio. In digital audio, what matters is the *audibility* of interpolation error between samples. Since Shannon's sampling theorem says it is possible to restore an audio signal *exactly* from its samples, it makes sense that the best digital audio interpolators would be based on that theory. Such "ideal" interpolation is called *bandlimited interpolation.*

A bandlimited interpolation algorithm designed along these lines is described in the theory of operation tutorial.[1] There is also free open-source software[2] available in the C programming language. An excellent online tool[3] can be used to compare various sampling-rate conversion implementations, both commercial and FOSS.

2 Free Resampling Software

- **Note:** `libresample` and `sndfile-resample` (from `libsamplerate`) are already included in the Planet CCRMA Distribution.[4] If you are running Red Hat Linux, check out the Planet.

- There is also SoX[5] (which uses `libsoxr`, the SoX resampler library) to change sampling rates by this method. Say `port search sox` (Mac), `yum search sox` (Linux), etc., to find it. See also `ssrc` (from Shibatch) which is a fixed-point sampling-rate conversion library installable packages[6] for Debian and Ubuntu. There is a project combining `ssrc` and `sox` on SourceForge:
  `https://sourceforge.net/projects/resamplerv/`[7]

[1]`http://ccrma.stanford.edu/~jos/resample/Theory_Operation.html`
[2]`http://ccrma.stanford.edu/~jos/resample/Available_Software.html`
[3]`http://src.infinitewave.ca/`
[4]`http://ccrma.stanford.edu/planetccrma/software/`
[5]`http://sox.sourceforge.net/`
[6]`http://lacocina.nl/2013/12/11/shibatch-ssrc-packages/`
[7]`https://sourceforge.net/projects/resamplerv/`

- Check out `http://src.infinitewave.ca/` for a large list of implementations and their relative performance. Seriously check it out—the coverage is amazing.
- *ReSampler*: `https://github.com/jniemann66/ReSampler`
  High quality command-line audio sample rate converter - LGPL-2.1 License
- *resampy*: `https://github.com/bmcfee/resampy`
  Python (Cython) implementation released under a BSD-style (ISC) license, since 2016
- *Brick* is an arbitrary-quality audio resampler, pitch-shifter, and format converter written and maintained by William Andrew Burnson:
  `http://camil.music.illinois.edu/software/brick/`
  `https://github.com/burnson/Brick`
- *Smarc* describes itself as a fast and high quality audio rate converter, allowing conversion between any two sampling rates, and optimized for standard audio rates, available as a command-line program or C library:
  `http://audio-smarc.sourceforge.net/`
- `resample-1.8.1.tar.gz`[8] (502 Kbytes) (v1.8.1 released November 11, 2006)
  The `resample` software package contains free sampling-rate conversion and filter design utilities written in C, including a stand-alone command-line sampling-rate conversion utility called `resample`. The package compiles readily under Linux and most other UNIX operating systems. It is released under the GNU Lesser General Public License (LGPL).
  - Older Version for NeXT Computers (49 Kbytes)[9]
  - Original 1983+ source for the PDP KL-10[10] (Released under the Revised BSD License)
- Erik de Castro Lopo's "SecretRabbitCode" `libsamplerate`[11]
  - Same basic algorithm
  - Floating-point (`resample` is fixed-point—sometimes needed on DSP chips)
  - Configures for non-Linux platforms such as Mac OS X and Windows
  - Includes the command-line program `sndfile-resample` which works like `resample`
  - Uses Erik's `libsndfile` (`resample` uses Bill Schottstaedt's `sndlib`)
  - License is either GPL or commercial (`resample` is LGPL)
- `libresample`[12] (LGPL) (version 1.3 released Jan. 7, 2004) by Dominic Mazzoni (used in Audacity) is a real-time library for sampling rate conversion providing several useful features relative to `resample-1.7` on which it is based:
  - It should build "out of the box" on more platforms, including Linux, Solaris, and Mac OS X (using the included `configure` script). There is also a Visual C++ project file for building under Windows.

[8] `http://ccrma.stanford.edu/~jos/gz/resample-1.8.1.tar.gz`
[9] `http://ccrma.stanford.edu/~jos/gz/resample-1.2.tar.gz`
[10] `./original-source.html`
[11] `http://www.mega-nerd.com/SRC/`
[12] `http://ftp.debian.org/pool/main/libr/libresample/`

  - Input and output signals are in memory (as opposed to sound files).
  - Computations are in floating-point (instead of fixed-point).
  - Filter table increased by a factor of 32, yielding more accurate results, even without linear interpolation (which also makes it faster).
  - Data can be processed in small chunks, enabling time-varying resampling ratios (ideal for time-warping applications and supporting an "external clock input" in software).
  - Easily applied to any number of simultaneous data channels

- `libresample4j`[13] is a Java port of `libresample` for real-time sampling-rate conversion by `dnault-laszlo`.[14]
- The free Open Source Audio Library Project (OSALP)[15] (LGPL) contains a C++ class based on `resample`.
- The *Speex*[16] speech coder/decoder (based on CELP) contains a variation of the `resample` algorithm in the file `resample.c`, is free and open-source, and is released under a BSD-style license (*i.e.*, free for both commercial and noncommercial uses). The code is characterized as "working, but it's in a very early stage, so it may have artifacts ... the API isn't stable ...". However, this appears to be the best BSD-licensed starting point.
- *SRDoubler*[17] (MIT License) is specialized for a factor of 2 stereo upsampling, allowing it to achieve significant computational savings at a perceptually non-degraded quality level.
- Website comparing various sampling-rate converters: `http://src.infinitewave.ca/`
- Paper on optimal FIR interpolation filter design: (.pdf)[18]

To report bugs, or contribute enhancements, please send email to `jos` (`jos@ccrma.stanford.edu`).

[13]`https://github.com/dnault/libresample4j/`
[14]`https://github.com/dnault`
[15]`http://sourceforge.net/projects/osalp/`
[16]`http://www.speex.org`
[17]`https://github.com/levmin/SRDoubler/`
[18]`http://ccrma.stanford.edu/~jos/pdf/optfir.pdf`

3 Theory of Operation

3.1 Abstract

This tutorial describes a technique for *bandlimited interpolation* of discrete-time signals which supports signal evaluation at an "arbitrary" time, and which performs well for smoothly changing sampling rates, such as needed for digital audio "scrubbing." The method is based on interpolated look-up in a table of filter coefficients, so as to make the filter impulse response available effectively in continuous-time form. A single pre-computed filter table handles all interpolation times and sampling-rate conversion ratios. Formulas are given for determining the look-up table size needed for a given precision requirement. This tutorial is an expansion of the conference paper in [**?**].

3.2 Introduction

Bandlimited interpolation of discrete-time signals is a basic tool having extensive application in digital signal processing. In general, the problem is to correctly compute signal values at arbitrary continuous times from a set of discrete-time samples of the signal amplitude. In other words, we must be able to interpolate the signal between samples. Since the original signal is always assumed to be *bandlimited* to half the sampling rate, (otherwise aliasing distortion would occur upon sampling), Shannon's sampling theorem tells us the signal can be exactly and uniquely reconstructed for all time from its samples by bandlimited interpolation.

Considerable research has been devoted to the problem of interpolating discrete points. A comprehensive survey of "fractional delay filter design" is provided in [**?**]. A comparison between classical (e.g., Lagrange) and bandlimited interpolation is given in [**?**]. The book **Multirate Digital Signal Processing** [**?**] provides a comprehensive summary and review of classical signal processing techniques for sampling-rate conversion. In these techniques, the signal is first interpolated by an integer factor L and then decimated by an integer factor M. This provides sampling-rate conversion by any rational factor L/M. The conversion requires a digital lowpass filter whose cutoff frequency depends on $\max\{L, M\}$. While sufficiently general, this formulation is less convenient when it is desired to resample the signal at arbitrary times or change the sampling-rate conversion factor smoothly over time.

In this tutorial, a public-domain resampling algorithm is described which will evaluate a signal at any time specifiable by a fixed-point number [**?**]. In addition, one lowpass filter is used regardless of the sampling-rate conversion factor. The algorithm effectively implements the "analog interpretation" of rate conversion, as discussed in [**?**], in which a certain lowpass-filter impulse response must be available as a continuous function. Continuity of the impulse response is simulated by linearly interpolating between samples of the impulse response stored in a table. Due to the relatively low cost of memory, the method is quite practical for hardware implementation.

In section 2, the basic theory is presented, section 3 addresses practical issues, and implementation details are discussed in section 4. Finally, section 5 discusses numerical requirements on the length, width, and interpolation accuracy of the filter coefficient table.

3.3 Theory of Ideal Bandlimited Interpolation

We review briefly the "analog interpretation" of sampling rate conversion [**?**] on which the present method is based. Suppose we have samples $x(nT_s)$ of a continuous absolutely integrable signal $x(t)$, where t is time in seconds (real), n ranges over the integers, and T_s is the sampling period. We assume $x(t)$ is bandlimited to $\pm F_s/2$, where $F_s = 1/T_s$ is the sampling rate. If $X(\omega)$ denotes the Fourier transform of $x(t)$, i.e., $X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$, then we assume $X(\omega) = 0$ for $|\omega| \geq \pi F_s$. Consequently, Shannon's sampling theorem gives us that $x(t)$ can be uniquely reconstructed from the samples $x(nT_s)$ via

$$\hat{x}(t) \triangleq \sum_{n=-\infty}^{\infty} x(nT_s)h_s(t - nT_s) \equiv x(t), \tag{1}$$

where

$$h_s(t) \triangleq \text{sinc}(F_s t) \triangleq \frac{\sin(\pi F_s t)}{\pi F_s t}. \tag{2}$$

To resample $x(t)$ at a new sampling rate $F_s' = 1/T_s'$, we need only evaluate Eq. (1) at integer multiples of T_s'.

When the new sampling rate F_s' is less than the original rate F_s, the lowpass cutoff must be placed below half the new lower sampling rate. Thus, in the case of an ideal lowpass, $h_s(t) = \min\{1, F_s'/F_s\}\text{sinc}(\min\{F_s, F_s'\}t)$, where the scale factor maintains unity gain in the passband.

A plot of the sinc function $\text{sinc}(t) \triangleq \sin(\pi t)/(\pi t)$ to the left and right of the origin $t = 0$ is shown in Fig. 1. Note that peak is at amplitude 1, and zero-crossings occur at all nonzero integers. The sinc function can be seen as a hyperbolically weighted sine function with its zero at the origin canceled out. The name *sinc function* derives from its classical name as the *sine cardinal* (or *cardinal sine*) function.

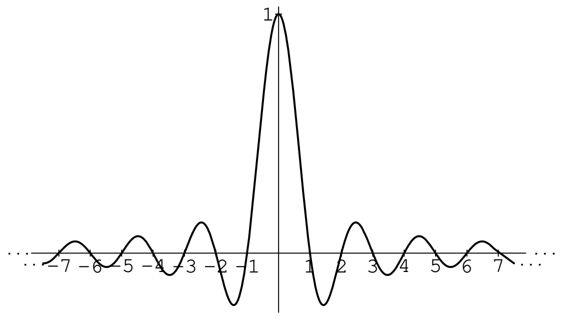


Figure 1: The sinc function plotted for seven zero-crossings to the left and right.

If "$*$" denotes the convolution operation for digital signals, then the summation in Eq. (1) can be written as $(x * h_s)(t)$.

Equation Eq. (1) can be interpreted as a superpositon of shifted and scaled sinc functions h_s. A sinc function instance is translated to each signal sample and scaled by that sample, and the instances are all added together. Note that zero-crossings of $\text{sinc}(z)$ occur at all integers except $z = 0$. That means at time $t = nT_s$, (i.e., on a sample instant), the only contribution to the sum is the single sample $x(nT_s)$. All other samples contribute sinc functions which have a zero-crossing at time $t = nT_s$. Thus, the interpolation goes precisely through the existing samples, as it should.

A plot indicating how sinc functions sum together to reconstruct bandlimited signals is shown in Fig. 2. The figure shows a superposition of five sinc functions, each at unit amplitude, and displaced by one-sample intervals. These sinc functions would be used to reconstruct the bandlimited interpolation of the discrete-time signal $x = [\ldots, 0, 1, 1, 1, 1, 1, 0, \ldots]$. Note that at each sampling instant $t = nT_s$, the solid line passes exactly through the tip of the sinc function for that sample;

this is just a restatement of the fact that the interpolation passes through the existing samples. Since the nonzero samples of the digital signal are all 1, we might expect the interpolated signal to be very close to 1 over the nonzero interval; however, this is far from being the case. The deviation from unity between samples can be thought of as "overshoot" or "ringing" of the lowpass filter which cuts off at half the sampling rate, or it can be considered a "Gibbs phenomenon" associated with bandlimiting.

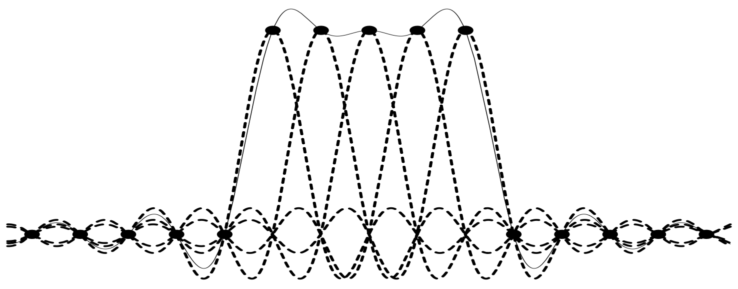


Figure 2: Bandlimited reconstruction of the signal The dots show the signal samples, the dashed lines show the component sinc functions, and the solid line shows the unique bandlimited reconstruction from the samples obtained by summing the component sinc functions.

A second interpretation of Eq. (1) is as follows: to obtain the interpolation at time t, shift the signal samples under *one* sinc function so that time t in the signal is translated under the peak of the sinc function, then create the output as a linear combination of signal samples where the coefficient of each signal sample is given by the value of the sinc function at the location of each sample. That this interpretation is equivalent to the first can be seen as a result of the fact that convolution is commutative; in the first interpretation, all signal samples are used to form a linear combination of shifted sinc functions, while in the second interpretation, samples from one sinc function are used to form a linear combination of samples of the shifted input signal. The practical bandlimited interpolation algorithm presented below is based on the second interpretation.

3.4 From Theory to Practice

The summation in Eq. (1) cannot be implemented in practice because the "ideal lowpass filter" impulse response $h_s(t)$ actually extends from minus infinity to infinity. It is necessary in practice to *window* the ideal impulse response so as to make it finite. This is the basis of the *window method* for digital filter design [**?**, **?**]. While many other filter design techniques exist, the window method is simple and robust, especially for very long impulse responses. In the case of the algorithm presented below, the filter impulse response is very long because it is heavily oversampled. Another approach is to design optimal decimated "sub-phases" of the filter impulse response, which are then interpolated to provide the "continuous" impulse response needed for the algorithm [**?**].

Figure 3 shows the frequency response of the ideal lowpass filter. This is just the Fourier transform of $h_s(t)$.

If we truncate $h_s(t)$ at the fifth zero-crossing to the left and the right of the origin, we obtain the frequency response shown in Fig. 4. Note that the stopband exhibits only slightly more than 20 dB rejection.

If we instead use the Kaiser window [**?**, **?**] to taper $h_s(t)$ to zero by the fifth zero-crossing to the

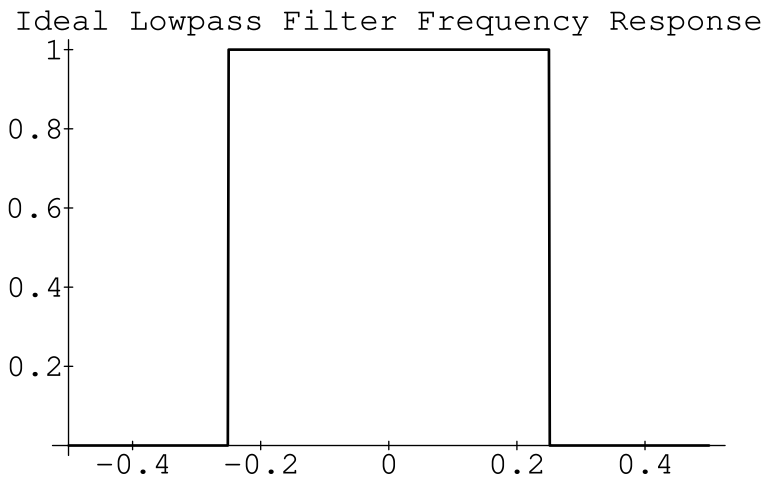


Figure 3: Frequency response of the ideal lowpass filter.

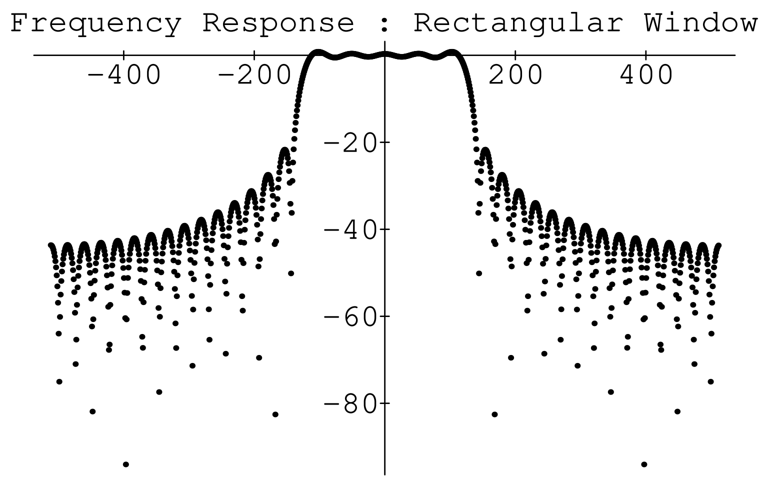


Figure 4: Frequency response of the ideal lowpass filter after rectangularly windowing the ideal (sinc) impulse response at the fifth zero crossing to the left and right of the time origin. The vertical axis is in units of decibels (dB), and the horizontal axis is labeled in units of spectral samples between plus and minus half the sampling rate.

left and the right of the origin, we obtain the frequency response shown in Fig. 5. Note that now the stopband starts out close to −80 dB. The Kaiser window has a single parameter which can be used to modify the stop-band attenuation, trading it against the transition width from pass-band to stop-band.

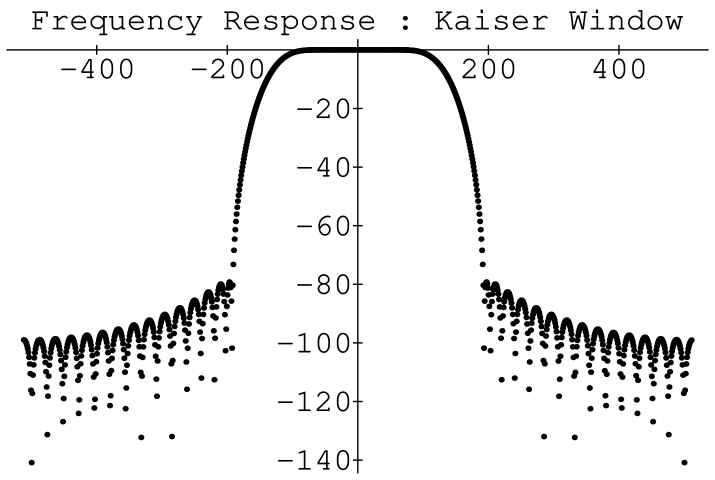


Figure 5: Frequency response of the ideal lowpass filter Kaiser windowed at the fifth zero crossing to the left and right.

3.5 Implementation

The implementation below provides signal evaluation at an arbitrary time, where time is specified as an unsigned binary fixed-point number in units of the input sampling period (assumed constant). Figure 6 shows the time register t, and Figure 7 shows an example configuration of the input signal and lowpass filter at a given time. The time register is divided into three fields: The leftmost field gives the number n of samples into the input signal buffer, the middle field is an initial index l into the filter coefficient table $h(l)$, and the rightmost field is interpreted as a number η between 0 and 1 for doing linear interpolation between samples l and $l+1$ (initially) of the filter table. The concatenation of l and η are called $P \in [0,1)$ which is interpreted as the position of the current time between samples n and $n+1$ of the input signal.

Let the three fields have n_n, n_l, and n_η bits, respectively. Then the input signal buffer contains $N = 2^{n_n}$ samples, and the filter table contains $L = 2^{n_l}$ "samples per zero-crossing." (The term "zero-crossing" is precise only for the case of the ideal lowpass; to cover practical cases we generalize "zero-crossing" to mean a multiple of time $t_c = 0.5/f_c$, where f_c is the lowpass cutoff frequency in cycles per sample.) For example, to use the ideal lowpass filter, the table would contain $h(l) = \text{sinc}(l/L)$.

Our implementation stores only the "right wing" of a symmetric finite-impulse-response (FIR) filter (designed by the window method based on a Kaiser window [**?**]). Specifically, if $h(l)$, $l = -LN_z, -LN_z+1, \ldots, -1, 0, 1, 2, \ldots, LN_z$, denotes a symmetric finite impulse response, then the *right wing* of h is defined as the set of samples $h(l)$ for $l = 0, 1, 2, \ldots, LN_z$. By symmetry, the *left wing* can be reconstructed as $h(-l) = h(l)$, $l = 1, 2, \ldots, LN_z$.

Our implementation also stores a table of *differences* $\hbar(l) = h(l+1) - h(l)$ between successive FIR sample values in order to speed up the linear interpolation. The length of each table is $N_h = LN_z + 1$, including the endpoint definition $\hbar(N_h) = 0$.

Consider a sampling-rate conversion by the factor $\rho = F_s'/F_s$. For each output sample, the basic interpolation Eq. (1) is performed. The filter table is traversed twice—first to apply the left wing of the FIR filter, and second to apply the right wing. After each output sample is computed, the time register is incremented by $2^{n_l+n_\eta}/\rho$ (i.e., time is incremented by $1/\rho$ in fixed-point format). Suppose the time register t has just been updated, and an interpolated output $y(t)$ is desired. For $\rho \geq 1$, output is computed via

$$\begin{aligned} v &\leftarrow \sum_{i=0}^{h\text{ end}} x(n-i)\left[h(l+iL) + \eta\overline{h}(l+iL)\right] && (3)\\ P &\leftarrow 1 - P && (4)\\ y(t) &\leftarrow v + \sum_{i=0}^{h\text{ end}} x(n+1+i)\left[h(l+iL) + \eta\overline{h}(l+iL)\right], && (5) \end{aligned}$$

where $x(n)$ is the current input sample, and $\eta \in [0,1)$ is the interpolation factor. When $\rho < 1$, the initial P is replaced by $P' = \rho P$, $1-P$ becomes $\rho - P' = \rho(1-P)$, and the step-size through the filter table is reduced to ρL instead of L; this lowers the filter cutoff to avoid aliasing. Note that η is fixed throughout the computation of an output sample when $\rho \geq 1$ but changes when $\rho < 1$.

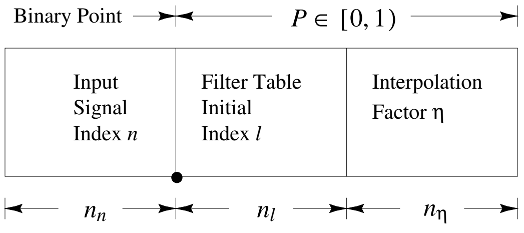


Figure 6: Time register format.

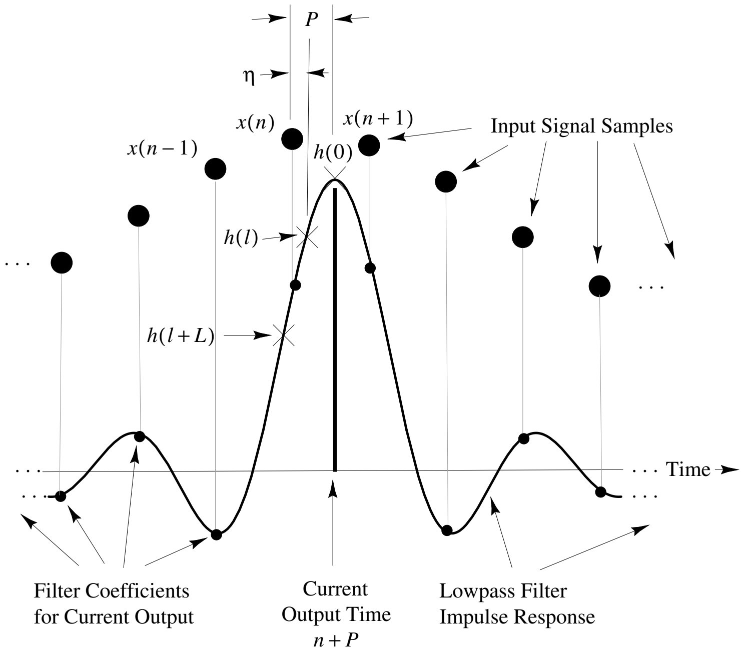

P
η
$x(n)$
$x(n+1)$
$x(n-1)$
$h(0)$
Input Signal Samples
$h(l)$
$h(l+L)$
Time
Filter Coefficients for Current Output
Current Output Time $n+P$
Lowpass Filter Impulse Response

Figure 7: Illustration of waveforms and parameters in the interpolator.

When $\rho < 1$, more input samples are required to reach the end of the filter table, thus preserving the filtering quality. The number of multiply-adds per second is approximately $(2N_z + 1)\max\{F_s, F_s'\}$. Thus the higher sampling rate determines the work rate. Note that for $\rho < 1$ there must be $\lceil N_z F_s / F_s' \rceil$ extra input samples available before the initial conversion time and after the final conversion time in the input buffer. As $\rho \to 0$, the required extra input data becomes infinite, and some limit must be chosen, thus setting a minimum supported ρ. For $\rho \geq 1$, only N_z extra input samples are required on the left and right of the data to be resampled, and the upper bound for ρ is determined only by the fixed-point number format, viz., $\rho_{\max} = 2^{n_l + n_\eta}$.

As shown below, if n_c denotes the word-length of the stored impulse-response samples, then one may choose $n_l = 1 + n_c/2$, and $n_\eta = n_c/2$ to obtain $n_c - 1$ effective bits of precision in the interpolated impulse response.

Note that rational conversion factors of the form $\rho = L/M$, where $L = 2^{n_l}$ and M is an arbitrary positive integer, do not use the linear interpolation feature (because $\eta \equiv 0$). In this case our method reduces to the normal type of bandlimited interpolator [?]. With the availability of interpolated lookup, however, the range of conversion factors is boosted to the order of $2^{n_l+n_\eta}/M$. E.g., for $\rho \approx 1$, $n_l = 9, n_\eta = 8$, this is about 5.1 decimal digits of accuracy in the conversion factor ρ. Without interpolation, the number of significant figures in ρ is only about 2.7.

The number N_z of zero-crossings stored in the table is an independent design parameter. For a given quality specification in terms of aliasing rejection, a trade-off exists between N_z and sacrificed bandwidth. The lost bandwidth is due to the so-called "transition band" of the lowpass filter [?]. In general, for a given stop-band specification (such as "80 dB attenuation"), lowpass filters need approximately twice as many multiply-adds per sample for each halving of the transition band width.

As a practical design example, we use $N_z = 13$ in a system designed for high audio quality at 20% oversampling. Thus, the effective FIR filter is 27 zero crossings long. The sampling rate in this case would be 50 kHz.[19] In the most straightforward filter design, the lowpass filter pass-band would stop and the transition-band would begin at 20 kHz, and the stop-band would begin (and end) at 25 kHz. As a further refinement, which reduces the filter design requirements, the transition band is really designed to extend from 20 kHz to 30 kHz, so that the half of it between 25 and 30 kHz aliases on top of the half between 20 and 25 kHz, thereby approximately halving the filter length required. Since the entire transition band lies above the range of human hearing, aliasing within it is not audible.

Using 512 samples per zero-crossing in the filter table for the above example (which is what we use at CCRMA, and which is somewhat over designed) implies desiging a length $27 \times 512 = 13824$ FIR filter having a cut-off frequency near $\pi/512$. It turns out that optimal Chebyshev design procedures such as the Remez multiple exchange algorithm used in the Parks-McLellan software [?] can only handle filter lengths up to a couple hundred or so. It is therefore necessary to use an FIR filter design method which works well at such very high orders, and the window method employed here is one such method.

It is worth noting that a given percentage increase in the original sampling rate ("oversampling") gives a larger percentage savings in filter computation time, for a given quality specification, because the added bandwidth is a larger percentage of the filter transition bandwidth than it is of the original sampling rate. For example, given a cut-off frequency of 20 kHz, (ideal for audio work), the transition band available with a sampling rate of 44 kHz is about 2 kHz, while a 48 kHz sampling rate provides a 4 kHz transition band. Thus, a 10% increase in sampling rate *halves* the work per sample in the digital lowpass filter.

3.6 Quantization Issues

In this section, we investigate the requirements on the sampling density $L = 2^{n_l}$ of the lowpass-filter impulse response, and the number of bits n_η required in the interpolation factor η. These quantities are determined by computing the worst-case error and comparing it to the filter coefficient quantization error.

[19] We arbitrarily define the 20% guard band as a percentage of half the sampling rate actually used, not as 20% of the desired 20 kHz bandwidth which would call for a 48 kHz sampling rate.

3.6.1 Choice of Table Size

It is desirable that the stored filter impulse response be sampled sufficiently densely so that interpolating linearly between samples does not introduce error greater than the quantization error. We will show that this condition is satisfied when the filter table contains at least $L = 2^{n_c/2}$ entries per zero-crossing, where n_c is the number of bits allocated to each table entry.

3.6.2 Linear Interpolation Error Bound

Let $h(t)$ denote the lowpass filter impulse response, and assume it is twice continuously differentiable for all t. By Taylor's theorem [**?**, p. 119], we have

$$h(t_0 + \eta) = h(t_0) + \eta h'(t_0) + \frac{1}{2}\eta^2 h''(t_0 + \lambda\eta), \tag{6}$$

for some $\lambda \in [0, 1]$, where $h'(t_0)$ denotes the time derivative of $h(t)$ evaluated at $t = t_0$, and $h''(t_0)$ is the second derivative at t_0.

The linear interpolation error is

$$e(t) \triangleq h(t) - \hat{h}(t), \tag{7}$$

where $t = t_0 + \eta$, $t_0 = \lfloor t \rfloor$, $\eta = t - t_0$, and $\hat{h}(t)$ is the interpolated value given by

$$\hat{h}(t) \triangleq \overline{\eta} h(t_0) + \eta h(t_1), \tag{8}$$

where $\overline{\eta} \triangleq 1 - \eta$ and $t_1 \triangleq t_0 + 1$. Thus t_0 and t_1 are successive time instants for which samples of $h(t)$ are available, and $\eta \in [0, 1)$ is the linear interpolation factor.

By definition, $e(t_0) = e(t_1) = 0$. That is, the interpolation error is zero at the known samples. Let t_e denote any point at which $|e(t)|$ reaches a maximum over the interval (t_0, t_1). Then we have

$$|e(t_e)| \geq |e(t)|, \quad \forall t \in [t_0, t_1].$$

Without loss of generality, assume $t_e \leq t_0 + 1/2$. (Otherwise, replace t_0 with t_1 in the following.) Since both $h(t)$ and $\hat{h}(t)$ are twice differentiable for all $t \in (t_0, t_1)$, then so is $e(t)$, and therefore $e'(t_e) = 0$. Expressing $e(t_0) = 0$ as a Taylor expansion of $e(t)$ about $t = t_e$, we obtain

$$0 = e(t_0) = e(t_e) + (t_0 - t_e)e'(t_e) + \frac{(t_0 - t_e)^2}{2!}e''(\xi) = e(t_e) + \frac{(t_0 - t_e)^2}{2}e''(\xi)$$

for some $\xi \in (t_0, t_1)$. Solving for $e(t_e)$ gives

$$e(t_e) = -\frac{(t_0 - t_e)^2}{2}e''(\xi)$$

Defining

$$M_2 \triangleq \max_t |h''(t)| = \max_t |e''(t)|,$$

where the maximum is taken over $t \in (t_0, t_1)$, and noting that $|t_0 - t_e| \leq 1/2$, we obtain the upper bound[20]

$$|e(t_e)| \leq \frac{M_2}{8} \tag{9}$$

[20]Thanks to Magnus Lundin for contributing this formulation, which sharpens the bound derived in [**?**].

3.6.3 Application to the Ideal Lowpass Filter

For the ideal lowpass filter, we have

$$h(t) = \operatorname{sinc}(\omega_L t/\pi) \overset{\Delta}{=} \frac{\sin(\omega_L t)}{\omega_L t} = \frac{1}{\omega_L} \int_0^{\omega_L} \cos(\omega t), \tag{10}$$

where $\omega_L = \pi/L$, and $L = 2^{n_l}$ is the number of table entries per zero-crossing. Note that the right-most form in Eq. (10) is simply the inverse Fourier transform of the ideal lowpass-filter frequency response. Twice differentiating with respect to t, we obtain

$$h''(t) = -\frac{1}{\omega_L} \int_0^{\omega_L} \omega^2 \cos(\omega t), \tag{11}$$

from which it follows that the maximum magnitude is

$$M_2 = \frac{\omega_L^2}{3} = \frac{\pi^2}{3L^2}. \tag{12}$$

Note that this bound is attained at $t = 0$. Substituting Eq. (12) into Eq. (9), we obtain the error bound

$$|e(t_e)| \leq \frac{\pi^2}{24L^2} < \frac{0.412}{L^2} = 0.412 \cdot 2^{-2n_l}. \tag{13}$$

Thus for the ideal lowpass filter $h(t) = \operatorname{sinc}(t/L)$, the pointwise error in the interpolated lookup of $h(t)$ is bounded by $0.412/L^2$. This means that n_l must be about half the coefficient word-length n_c used for the filter coefficients. For example, if $h(t)$ is quantized to 16 bits, L must be on the order of $2^{16/2} = 256$. In contrast, we will show that without linear interpolation, n_l must increase proportional to n_c for n_c-bit samples of $h(t)$. In the 16-bit case, this gives $L \sim 2^{16} = 65536$. The use of linear interpolation of the filter coefficients reduces the memory requirements considerably.

The error bounds obtained for the ideal lowpass filter are typically accurate also for lowpass filters used in practice. This is because the error bound is a function of M_2, the maximum curvature of the impulse response $h(t)$, and most lowpass designs will have a value of M_2 very close to that of the ideal case. The maximum curvature is determined primarily by the bandwidth of the filter since, generalizing equations Eq. (10) and Eq. (11),

$$h''(0) = -\frac{1}{\pi} \int_0^{\pi} \omega^2 H(\omega),$$

which is just the second moment of the lowpass-filter frequency response $H(\omega)$ (which is real for symmetric FIR filters obtained by symmetrically windowing the ideal sinc function [?]). A lowpass-filter design will move the cut-off frequency slightly below that of the ideal lowpass filter in order to provide a "transition band" which allows the filter response to give sufficient rejection at the ideal cut-off frequency which is where aliasing begins. Therefore, in a well designed practical lowpass filter, the error bound M_2 should be lower than in the ideal case.

3.6.4 Relation of Interpolation Error to Quantization Error

If $h(t) \in [-1, 1 - 2^{-n_c}]$ is approximated by $h_q(t)$ which is represented in two's complement fixed-point arithmetic, then

$$h_q(t_0) = -b_0 + \sum_{i=1}^{n_c - 1} b_i 2^{-i},$$

where $b_i \in \{0,1\}$ is the ith bit, and the worst-case rounding error is

$$|h(t) - h_q(t)| \le 2^{-n_c}.$$

Letting $h_q(t_i) = h(t_i) + \epsilon_i$, where $|\epsilon_i| \le 2^{-n_c}$, the interpolated look-up becomes

$$\hat{h}_q(t_0 + \eta) = \overline{\eta} h_q(t_0) + \eta h_q(t_1) = \hat{h}(t_0 + \eta) + \overline{\eta}\epsilon_0 + \eta\epsilon_1.$$

Thus the error in the interpolated lookup between quantized filter coefficients is bounded by

$$|e_q(t)| \le \frac{M_2}{8} + 2^{-n_c},$$

which, in the case of $h(t) = \text{sinc}(t/L)$, can be written

$$|e_q(t)| < \frac{0.412}{L^2} + 2^{-n_c} = 0.412 \cdot 2^{-2n_l} + 2^{-n_c}.$$

If $L = 2^{n_c/2}$, then $|e_q(t)| < 1.5 \cdot 2^{-n_c}$.

3.6.5 Error in the Absence of Interpolation

For comparison purposes, we derive the error incurred when no interpolation of the filter table is performed. In this case, assuming rounding to the nearest table entry, we have

$$\begin{array}{rcl} t &=& t_0 + \eta, \qquad |\eta| \le \frac{1}{2} \\ \hat{h}(t) &=& h(t_0) \\ e(t) &=& h(t) - h(t_0) \\ &=& \eta h'(t_0) + \frac{1}{2}\eta^2 h''(t_0 + \lambda\eta) \\ |e(t)| &\le& \frac{M_1}{2} + \frac{M_2}{8}, \end{array}$$

where $M_1 \stackrel{\Delta}{=} \max_t |h'(t)|$. For the ideal lowpass, we have

$$h'(t) = -\frac{1}{\omega_L}\int_0^{\omega_L} \omega \sin(\omega t) d\omega = \frac{\omega_L t \cos(\omega_L t) - \sin(\omega_L t)}{\omega_L t^2}.$$

Note that $h'(L) = -1/L$ and $|h'(t)| < \omega_L/2 = \pi/2L$. Thus $M_1 = a/L$ where $1 \le a < \pi/2$. The no-interpolation error bound is then

$$|h'(t)| \le \frac{a}{2L} + \frac{\pi^2}{24L^2} < \frac{0.7854}{L} + \frac{0.4113}{L^2}.$$

3.6.6 Choice of Interpolation Resolution

We now consider the error due to finite precision in the linear interpolation between stored filter coefficients. We will find that the number of bits n_η in the interpolation factor should be about half the filter coefficient word-length n_c.

3.6.7 Quantized Interpolation Error Bound

The quantized interpolation factor and its complement are representable as

$$\begin{aligned}\eta_q &= \eta + \nu \\ \overline{\eta}_q &= \overline{\eta} - \nu\end{aligned}$$

where, since $\eta, \overline{\eta}$ are unsigned, $|\nu| \leq 2^{-(n_\eta+1)}$. The interpolated coefficient look-up then gives

$$\begin{aligned}\hat{h}_{qq}(t) &= (\overline{\eta} - \nu)[h(t_0) + \epsilon_0] + (\eta + \nu)[h(t_1) + \epsilon_1] \\ &= \hat{h}(t) + \overline{\eta}\epsilon_0 + \eta\epsilon_1 + \nu[h(t_1) - h(t_0)],\end{aligned}$$

where second-order errors $\nu\epsilon_0$ and $\nu\epsilon_1$ are dropped. Since $|h(t_1) - h(t_0)| \leq M_1$, we obtain the error bound

$$|e_{qq}(t)| \leq 2^{-n_c} + 2^{-(n_\eta+1)} M_1 + \frac{1}{8} M_2. \tag{14}$$

The three terms in Eq. (14) are caused by coefficient quantization, interpolation quantization, and linear-approximation error, respectively.

3.6.8 Ideal Lowpass Filter

For the ideal lowpass, the error bound is

$$|e_{qq}(t)| \leq 2^{-n_c} + a 2^{-(n_l+n_\eta+1)} + \frac{\pi^2}{8} 2^{-n_l}.$$

Let $n_l = 1 + n_c/2$ and require that the added error is at most $\frac{1}{2}2^{-n_c}$. Then we arrive at the requirement

$$n_\eta \geq \frac{n_c}{2}.$$

3.7 Conclusions

A digital resampling method has been described which is convenient for bandlimited interpolation at arbitrary times and for smoothly varying sampling rates, and which is attractive for hardware implementation. We have presented the case which assumes uniform sampling of the input signal; however, extensions to variable sampling rates and isolated-point evaluation are straightforward.

A quantization error analysis led to the conclusion that for n_c-bit filter coefficients, the number of impulse-response samples stored in the filter lookup table should be on the order of $2^{n_c/2}$ times the number of "zero-crossings" in the impulse response, and the number of bits in the interpolation between impulse-response samples should be about $n_c/2$. With these choices, the linear interpolation error and the error due to quantized interpolation factors are each about equal to the coefficient quantization error. A signal resampler designed according to these rules will typically be limited primarily by the lowpass filter *design*, rather than by quantization effects.

We note that the error analysis presented here is pessimistic in the sense that it assumes worst-case input signal conditions (e.g., a sinusoid at half the sampling rate or white noise). A different type of error analysis is possible by treating the filter coefficients as exact but subject to time jitter. In this approach, the error can be expressed in terms of the input signal Taylor series expansion, and consequently in terms of the input signal bandwidth (or maximum slope). Such an analysis reveals that for most practical signals, the quantization error is considerably less than the levels derived here.

3.8 Appendix: Periodic Sinc Interpolation

It is interesting to note that all *periodic* sampled signals can be sinc-interpolated *exactly* using the following formula [**?**]:

$$x(t) = \frac{\sin(\pi t)}{2N} \sum_{n=-L}^{M-1} x_n (-1)^n \left[(-1)^{N+1} \tan\left(\pi \frac{t-n}{2N}\right) + \cot\left(\pi \frac{t-n}{2N}\right)\right], \quad N = L + M$$

where the sampling rate is normalized to be $T = 1$, and the period is $N = L + M$ samples.

The first step in the derivation is the exact general formula

$$\begin{aligned} x(t) &= \sum_{n=-\infty}^{\infty} x_n \frac{\sin[\pi(t-n)]}{\pi(t-n)} && (15) \\ &= \frac{\sin(\pi t)}{\pi} \sum_{n=-\infty}^{\infty} x_n \frac{(-1)^n}{t-n} && (16) \end{aligned}$$

which follows immediately from the identity $\sin[\pi(t-n)] = (-1)^n \sin(\pi t)$. This form can be used to develop a table-based sinc interpolation algorithm in which the function $1/t$ is sampled, windowed, and stored in a table over a small range of t. (Reverting to the weighted sinc table is advisable near an argument of zero where there is a pole-zero cancellation in the definition of sinc, i.e., when $|t-n| \ll 1$.) Note that when t crosses 2, the $1/t$ table can be implemented as $(1/2)(1/(t/2))$. In other words, the table between $t = 2$ and $t = 4$ can be computed from the table between $t = 1$ and $t = 2$ using a simple one-bit right-shift on the table address and the table output. If this trick is used, the table window must be applied separately, but there ways to synthesize simple windows (e.g., the Hanning or Hamming windows which consist of a single sinusoidal component) using waveform synthesis techniques, avoiding a separate table for the interpolated window function.

3.9 Appendix: Relation between Sinc and Lagrange Interpolation

Lagrange interpolation is a well known, classical technique for interpolation [**?**]. It is also called Waring-Lagrange interpolation, since Waring actually published it 16 years before Lagrange [**?**, p. 323]. Given a set of $n+1$ known samples $f(x_k)$, $k = 0, 1, 2, \ldots, n$, the problem is to find the unique order n polynomial $y(x)$ which interpolates the samples. The solution can be expressed as a linear combination of elementary nth order polynomials:

$$y(x) = \sum_{k=0}^{n} l_k(x) f(x_k) \tag{17}$$

where

$$l_k(x) \triangleq \frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} \tag{18}$$

From the numerator of the above definition, we see that $l_k(x)$ is an order n polynomial having zeros at all of the samples except the kth. The denominator is simply the constant which normalizes its value to 1 at x_k. Thus, we have

$$l_k(x_j) = \delta_{kj} \triangleq \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases} \tag{19}$$

In other words, the polynomial l_k is the kth *basis polynomial* for constructing a polynomial interpolation of order n over the $n+1$ sample points x_k.

In the case of an *infinite* number of *equally spaced* samples, with spacing $x_{k+1} - x_k = \Delta$, the Lagrangian basis polynomials converge to shifts of the *sinc function,* i.e.,

$$l_k(x) = \text{sinc}\left(\frac{x - k\Delta}{\Delta}\right), \quad k = \ldots, -2, -1, 0, 1, 2, \ldots \tag{20}$$

where

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

A simple argument is based on the fact that any analytic function is determined by its zeros and its value at one point. Since $\text{sinc}(x)$ is zero on all the integers except 0, and since $\text{sinc}(0) = 1$, it must coincide with the infinite-order Lagrangian basis polynomial for the sample at $x = 0$ which also has its zeros on the nonzero integers and equals 1 at $x = 0$.

The equivalence of sinc interpolation to Lagrange interpolation was apparently first published by the mathematician Borel in 1899, and has been rediscovered many times since [**?**, p. 325].

A direct proof can be based on the equivalance between Lagrange interpolation and windowed-sinc interpolation using a "scaled binomial window" [**?**, **?**]. That is, for a fractional sample delay of D samples, multiply the shifted-by-D, sampled, sinc function

$$h_s(n) = \text{sinc}(n - D) = \frac{\sin[\pi(n - D)]}{\pi(n - D)}$$

by a binomial window

$$w(n) = \left(\begin{array}{c} N \\ n \end{array}\right), \quad n = 0, 1, 2, \ldots N$$

and normalize by [**?**]

$$C(D) = (-1)^N \frac{\pi(N+1)}{\sin(\pi D)} \left(\begin{array}{c} D \\ N+1 \end{array}\right),$$

which scales the interpolating filter to have a unit L_2 norm, to obtain the Nth-order Lagrange interpolating filter

$$h_D(n) = C(D)w(n)h_s(n), \quad n = 0, 1, 2, \ldots, N$$

Since the binomial window converges to the Gaussian window as $N \to \infty$, and since the window gets wider and wider, approaching a unit constant in the limit, the convergence of Lagrange to sinc interpolation can be seen.

A more recent alternate proof appears in [**?**].