Simultaneous convolution and downsample

Brett Viren

March 22, 2024

1 Problem statement

We are given a signal of interest a characterized by its sample period and number of samples $s_a : (T_a, N_a)$ and a filter signal b with sampling $s_b : (T_b, N_b)$. We restrict the problem to the case where the ratio of sample periods, $R = T_a/T_b = L/M,\ \{L, M\} \in \mathbb{N}$ can be expressed as an integer fraction. With no loss of generality we take the case where $T_a \leq T_b,\ R < 1$. That is, we give mere semantics in defining "the filter" as being the signal that is sampled more slowly than the defined "signal of interest". We then wish to apply the filter to the signal of interest such that the result is sampled with the slower period T_b.

2 Common sampling period

In the special case that $T \equiv T_a = T_b$ there is of course no downsampling required. The problem reduces to performing only the filtering through convolution. As we do not know the extent over which each signal is nonzero we should apply linear convolution to avoid any "wrap-around" effects that a cyclic convolution may cause. This requires *extending* both signals with zero-valued interval domain samples to a length at least $N_a + N_b - 1$. We will choose to add an additional sample and extend by $N_c = N_a + N_b$ and recognize that this is equivalent to extending the signals by the sum of their durations.

$$s_a^e : (T, N_c) = extend(s_a, N_c) \tag{1}$$

$$s_b^e : (T, N_c) = extend(s_b, N_c) \tag{2}$$

The convolution is most efficiently performed as a multiplication in the frequency domain through application of discrete Fourier transform (DFT) $S \equiv \mathcal{F}\{s\}$ according to:

$$c = a \circledast b \leftrightarrow C = AB \tag{3}$$

$$A = \mathcal{F}\{a\} \leftrightarrow a = \mathcal{F}^{-1}\{A\} \tag{4}$$

Where we denote a signal in the (time) interval domain with a lower case letter and in the frequency domain with an upper case letter. Multiplication of two signals such as AB is understood to be performed in a sample-wise basis.

To apply this isometry requires the two signals to have equal frequency sample bandwidth $\delta F = \frac{1}{NT}$. By construction, they have the same sample period and after extension the same size and thus the same sample bandwidth.

The procedure now applies the convolution

$$s_c^e = conv(s_a^e, s_b^e) \tag{5}$$

Where the $conv(a, b)$ function is assumed to be implemented as

$$S_a = \mathcal{F}\{s_a\} \tag{6}$$
$$S_b = \mathcal{F}\{s_b\} \tag{7}$$
$$S_c = S_a S_b \tag{8}$$
$$s_c = \mathcal{F}^{-1}\{S_c\} \tag{9}$$

At this point the signal $s_c^e : (T, N_c)$ may be truncated to size N_a or N_b or etc as required by downstream algorithms.

3 Upsample-filter-downsample

When $T_a \neq T_b$, the above procedure can not be applied. We may resample one (or in principle both) signals in order that they have common sample bandwidth $\delta F_a = \delta F_b$. We should not downsample the "faster" signal s_a as there is no available anti-alias filter (after all, that is what s_b is for). On the other hand, we may safely upsample the s_b to sample period T_a.

$$s_b^u : (T_a, N_b/R) = interp(s_b, 1/R) \tag{10}$$

We assume the $interp(s, r)$ function interpolates $s : (T, N) \rightarrow s^u : (T/r, rN)$ and is implemented according to the procedure:

$$S : (T, N) = \mathcal{F}\{s\} \tag{11}$$
$$S^u : (T/r, rN) = resize(S, r) \tag{12}$$
$$s^u : (T/r, rN) = \mathcal{F}^{-1}\{S^u\} \tag{13}$$

Where the $resize(S, r)$ function will remove samples from or insert samples to signal $S : (T, N)$ so that the new signal S^u is sampled as $(T/r, rN)$, keeping $\delta F = \frac{1}{NT}$ constant. When upsampling $r > 1$, zero-valued samples are inserted above the original Nyquist frequency $\frac{1}{2NT}$. When downsampling $r < 1$, all existing samples above the new, downsampled Nyquist frequency $\frac{r}{2NT}$ are removed.

The frequency sample bandwidth of the upsampled filter signal now matches that of the original signal of interest. The previously described *common sampling period* procedure may now be applied to the original signal of interest $s_a : (T_a, N_a)$ and the upsampled filter signal $s_b^u : (T_a, N_b/R)$. This results in a signal $s_c^u : (T_a, N_c = N_a + N_b/R)$. To complete the problem, this signal is downsampled

$$s_c : (T_b, N = RN_a + N_b) = interp(s_c^u, R) \tag{14}$$

In summary, the high level procedure is

$$s_b^u = interp(s_b, 1/R) \tag{15}$$
$$N_c = N_a + N_b/R \tag{16}$$
$$s_a^e = extend(s_a, N_c) \tag{17}$$
$$s_b^{eu} = extend(s_b^u, N_c) \tag{18}$$
$$s_c^{eu} = conv(s_a^e, S_b^{eu}) \tag{19}$$
$$s_c^e = interp(s_c^{eu}, R) \tag{20}$$

4 Single-step

In the high-level procedure summary of the *upsample-filter-downsample* it is obvious that the final two steps can be rewritten as the function composition $interp(conv(a,b),r) \rightarrow (interp{\cdot}conv)(a,b,r)$. This allows the $resize()$ function that makes up the payload of *interp* to be applied directly to the intermediate spectrum calculated after the multiplication inside *conv*.

The two earlier calls to $extend()$ do not fall to such simple optimization as they operate in the interval domain, by construction. However, we may perform the extension required for the eventual $conv()$ prior to the interpolation by extending the original signals by suitable amount.

Recalling how the extension in size for signals of common sample period is equivalent to extension in duration, we extend each signal of differing sample period based on their duration.

$$d_c = d_a + d_b = T_a N_a + T_b N_b \tag{21}$$
$$s_a^e = extend(s_a, d_c/T_a) \tag{22}$$
$$s_b^e = extend(s_b, d_c/T_b) \tag{23}$$

The extend signals $s_a^e : (T_a, N_a + N_b/R)$ and $s_b^e : (T_b, N_a R + N_b)$ may each be transformed to the frequency domain. As they have the same duration, they have the same sample bandwith and the order of the frequency domain multiplication and $resize()$ are commutative.

$$S_a^{ed} = resize(S_a^e, R) \tag{24}$$
$$S_c^{ed} = S_a^{ed} S_b^e \tag{25}$$
$$s_c^{ed} = \mathcal{F}^{-1}\{S_c^{ed}\} \tag{26}$$

This final signal is $s_c^{ed} : (T_b, N_a R + N_b)$. A total of one forward $\mathcal{F}$ for each original, extended signal and one inverse $\mathcal{F}^{-1}$ for the final signal has been applied.

5 Prime factoring constraint

With the two applications of $extend()$ performed on the initial interval-domain signals the problem is reduced to one application of $\mathcal{F}$ at size $N_a + N_b/R$ and another at size $N_a R + N_b$ followed by $\mathcal{F}^{-1}$ also at size $N_a R + N_b$. For a DFT to be implemented as via FFT the size must consist of small prime factors. Any prime factors larger than, typically 5 or 7 will incur performance penalties worse than $\mathcal{O}(n \log n)$.

For a given DFT size N one may always find a size $N + n$ that satisfies this small-prime-factor requirement. In many cases the extra computation cost of a somewhat larger size is offset by the speed provided by FFT.

When applied in the context of the above procedure, care is needed to assure that the additional extension retains a common total duration for the signal of interest and the filter signal. Simplifying notation, after initial extension we have $s_a : (T_a, N_a)$ and $s_b : (T_b, N_b)$ and wish to further extend to $s_a^t : (T_a, N_a + n_a)$ and $s_b^t : (T_b, N_b + n_b)$ such that $N_a + n_a$ and $N_b + n_b$ are factored by (in general, a different) set of small primes. For any n_a and n_b the common total duration requires $\frac{N_b + n_b}{N_a + n_a} = R = \frac{N_b}{N_a}$ or $n_a/n_b = N_a/N_b$. In general it may not be easy to guarantee a simultaneously satisfying n_a and n_b. Though, in some practical cases, the size $N_b + n_a$ will see multiple downstream uses. In such cases, optimizing on this number and leaving the signal in the frequency domain can bring additional optimization.