Effects of Averaging to Reject Unwanted Signals in Digital Sampling Oscilloscopes

Charles Bishop
Catherine Kung
Systems Test Group
Teradyne, Inc.
North Reading, MA USA

Abstract— **DSOs (Digital Sampling Oscilloscopes) generally allow the use of averaging to increase vertical resolution and lower uncorrelated noise. While averaging is a useful tool, it is important to remember that it is a type of filtering. Applying averaging successfully is easier if the user understands the characteristics of the filters being applied.**

I. INTRODUCTION

A Digital Sampling Oscilloscope (DSO) is used in Automatic Test Equipment to look at a waveform at discrete sampling instances, so that the waveform can be displayed for visual interpretation or digitized for post-processing analysis. Very often a DSO is used in combination with other test equipment, or integrated into a larger test system, where unwanted signals might couple into the waveform under test. Users of a DSO often want to improve the quality of their measurement and one approach for doing so is by averaging.

There are two general use-cases for averaging in a DSO. The first, successive sample averaging, takes a single acquisition and averages between its samples. The second, successive capture averaging, combines the corresponding samples of multiple captures to create a single capture.

II. THEORY AND BENEFITS OF COMMON TYPES OF AVERAGING

A. Successive Sample Averaging

Successive sample averaging is also called boxcar filtering or moving average filtering. In an implementation of this type of averaging each output sample represents the average value of M consecutive input samples. Figure 1 illustrates the concept for a 3-sample average. In this example the input and output sample rates are equal. This type of averaging removes noise by decreasing a DSO's bandwidth. It applies an LPF function with a 3dB point approximated by

$$0.433 * s/M \qquad (1)$$

where M is the number of samples to be averaged, and s is the sample rate in samples per second. This type of filter will have very sharp nulls at frequencies corresponding to signals whose periods are integer sub-multiples of M/s. The possible noise reduction is roughly proportional to the square root of the number of points, so a 25 point filter would reduce high-frequency noise amplitude by a factor of 5. In DSOs this type of averaging is often used to implement a selectable bandwidth function.

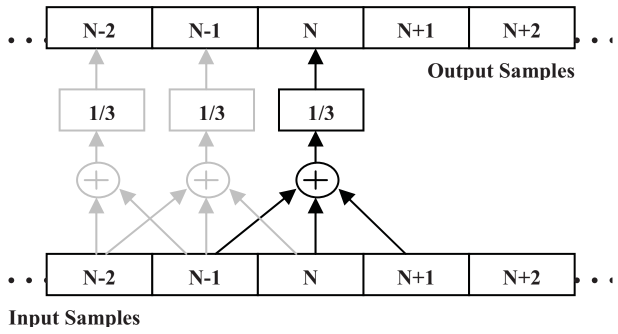

Figure 1. Successive Sample Averaging

Successive sample averaging may also be employed as part of a sample decimation scheme. Sample decimation is employed to improve vertical resolution at lower sample rates. Rather than sample at the user-selected rate, a DSO may sample at M times that rate, and then average together M input samples to produce each output sample at the lower rate.

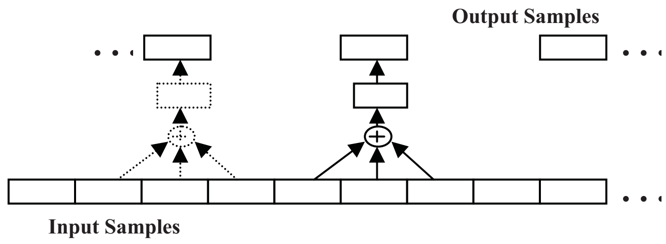

Figure 2. Decimation Based on Successive Sample Averaging

In this case the resolution improvement, for small M, is proportional to M. As rule of thumb, 1 bit of resolution improvement may be achieved for each factor of four in M. Averaging by 16 would therefore improve resolution by about 2 bits. In general, too, the user should not expect resolution improvement beyond 4-6 additional bits. Note also that the improvement depends on a dynamic signal. Resolution will always be improved on those portions of a signal that slew through multiple code counts between samples, but steady-state signals will see improvement only if there is noise present with amplitude greater than 1 or 2 ADC LSBs. Fortunately, for real-world signals this is almost always the case.

In figure 3 a half-cycle of a sine wave is shown digitized to 4-bit resolution. Quantization is evident on all parts of the waveform. In figure 4 the same waveform has been decimated and averaged by a factor of 16. Quantization is almost completely removed from the fastest-slewing portions of the waveform, but is still evident at the peak, where the slew rate is least.

Improvement in resolution implies but does not guarantee improvement in accuracy. Inaccuracies due to noise and quantization can be averaged away, but errors arising from measure path non-linearity, converter INL, or long-period errors such as thermal drift will remain. When capturing signals in a noisy environment averaging over thousands of samples may be required, but the result will not have a greater basic accuracy improvement.

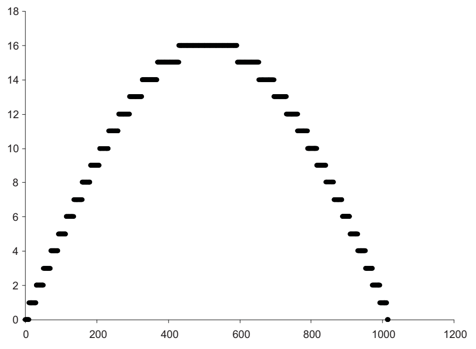

Figure 3. 1024 Samples Quantized at 4 bits

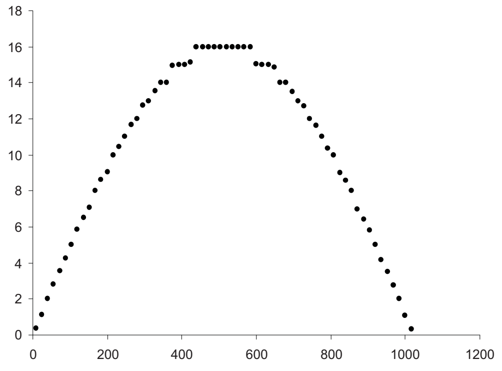

Figure 4. Decimation by 16

Decimation with averaging has another benefit in that it reduces aliasing. By over-sampling at M times the requested rate, the Nyquist frequency is also raised by a factor of M.

B. *Successive Capture Averaging*

Most DSOs can employ averaging in a second way that takes advantage of the repetitive nature of many signals under test. With successive capture averaging, M complete waveforms are digitized, and then corresponding samples, one from each set, are averaged together to produce each sample of the output waveform. This technique rejects signals that are not correlated to the trigger without band-limiting the waveform.

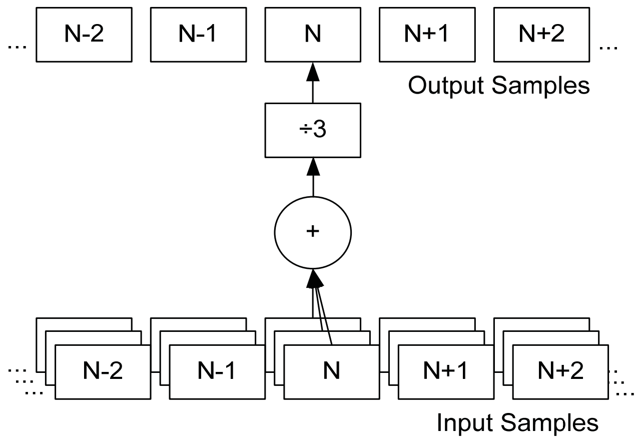

Figure 5. Successive Capture Averaging

While successive capture averaging does not decrease the bandwidth of the DSO, it does require that M captures be completed prior to the first valid output. This can have a pronounced impact on throughput if too large a value is used for M.

The latency associated with successive-capture averaging can be most obvious when the result of a continuous series of captures is being streamed to a display. This is the typical use-case for a DSO in non-ATE applications. An implementation that only updated the display every M captures would produce a very intermittent display. For this reason most DSOs incorporate some sort of continuous averaging algorithm for use with the display. This can take the form of arithmetic averaging but for reasons of calculation efficiency and display response an exponential algorithm is often used instead.

III. EXAMPLES

The effects of averaging on a signal can be viewed in either the time domain or the frequency domain. For examining the time domain, a unit step is a very useful test signal. In the frequency domain it is helpful to look at an FFT (Fast Fourier Transform) of the impulse response.

A. *Frequency Response of Successive Sample Averaging*

Figure 6 compares the frequency response of 64-point and 256-point successive sample averages. In each case the impulse response has been calculated over 4096 time domain samples. An FFT of the response has then been plotted. The sample frequency is assumed to be 1GS/s. Per equation (1) we would expect the 3dB points to be 6.76MHz and 1.69MHz respectively, and they are.

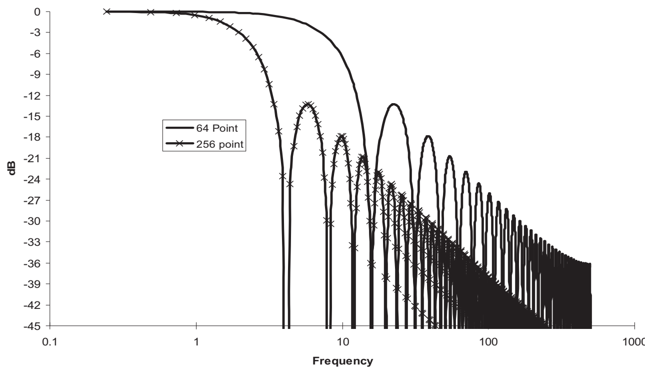

Figure 6. Filter Bandwidth of Successive Sample Averaging

Figure 6 shows the dramatic nulls that exist in the stop-band of averaging filters. These null points are at the frequency s/M and its harmonics. The figure also shows that at each point in the stop-band the 256-point average has about 12dB greater attenuation than the 64-point.

Choosing the width of a successive sample average involves trade-offs. In the example above, averaging across 256 points results in about 12dB greater noise suppression than averaging across 64 points, but it also reduces the capture bandwidth by a factor of four.

B. *Successive Capture Averaging*

Successive capture averaging doesn't affect the bandwidth of a DSO, but it does affect noise amplitude and we'd like some way of evaluating the relative effectiveness of different amounts of averaging. One good approach is to view the noise as an entirely random process and then look at the effect of averaging a random series. In figure 7 a 10V signal with +/-1V of noise is subjected to repeated sampling and averaging. This would represent the variability of a single point within a larger capture. Averaging across 500 captures is sufficient to reduce the noise to 50mV, but after that the noise does not converge as quickly to 0V. You might suspect that these averages were done with single-precision floating point, and that the accumulated rounding error across such a large sample set is dominating the total error. In fact, the reason is that the random number generator used in this analysis appears not to be completely random. It uses a seed value perhaps borrowed from the processor's real time clock. The result is that there are long-period biases in the random function that are very hard to average out. I've kept the error in the example because it illustrates an important fact; noise is very broadband. Some noise will always fall within the pass-band of an averaging function, no matter how many points are used.

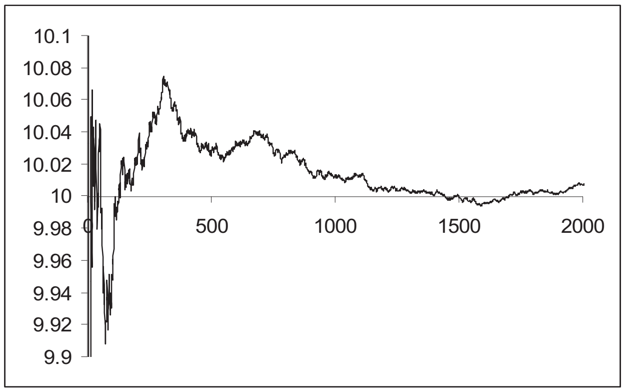

Figure 7. Convergence of Successive Capture Averaging

Figure 8 shows a closer view of the error. Given the long-period content in this pseudo-random noise, there's little point in averaging more than about a thousand captures.

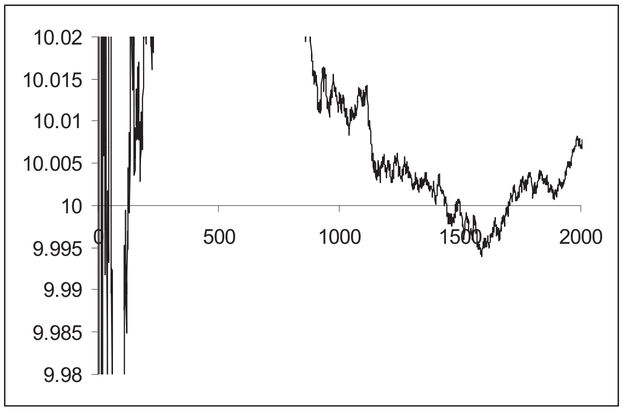

Figure 8. Convergence of Successive Capture Averaging

Another way of looking at successive capture averaging is in terms of impulse response. In this case we think of the noise in terms of a one-time interfering signal. Figure 9 shows the decaying response of a 1V impulse after repeated averaging. At 1000 averages the remaining amplitude is 1mV.

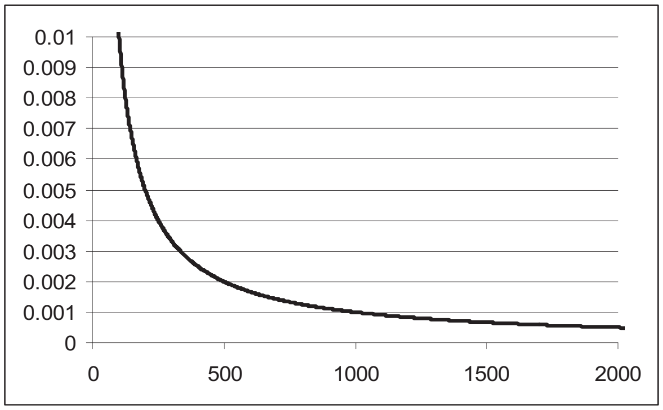

Figure 9. Impulse Response of Successive Capture Averaging

It's important to remember that successive capture averaging only attenuates signals that are un-correlated to the trigger.

It's also important to keep in mind a particular case; signals which trigger alternately on both the rising and falling edge. If too little trigger hysteresis is employed when capturing a noisy signal the result can be a set of captures where the signal of interest has two distinct and opposite phases. The result of averaging these captures will be to attenuate the desired signal, and in the extreme case to cancel it completely.

C. Continuous Successive Capture Averaging

One important use of DSOs is to observe changing signals interactively. In this case the instrument performs one capture after another and continuously sends the result to a display. This can present challenges to the design of a successive capture averaging algorithm. First of all, there is the need to maintain high throughput. The display should update quickly enough that the user can perceive dynamic changes in the signal and not be distracted by the update process itself. Secondly there is the need to conserve hardware resources. All averaging algorithms to this point could be implemented by capturing into an accumulator, and then performing a division to get a result. For continuously updating a display we would like to update the display after each capture, rather than to wait until we've accumulated M capture sets. This could mean that we need to store M complete captures in a FIFO array.

Alternatively, we can use an exponential averaging function to approximate arithmetic averaging. With this algorithm we can use a single accumulator array and we still get a valid result after every capture. With exponential averaging we accumulate M samples just as with arithmetic averaging, but each time we add a new sample to the accumulator we subtract the previous average value and then divide by M to obtain a new average value.

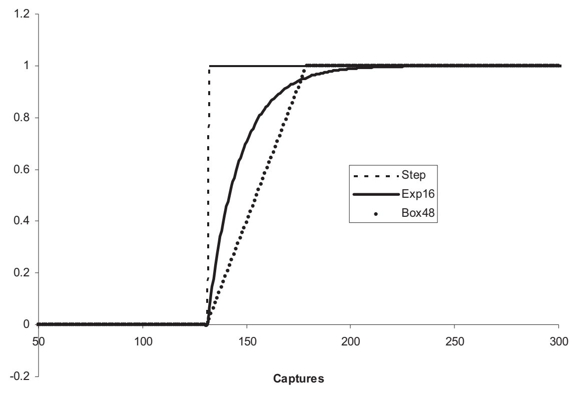

Figure 10. Exponential Averaging

Figure 10 compares the time response, in captures, of a 48-capture arithmetic average with 16-capture exponential average. The real-world case is where the display is responding to a sudden change in signal. The arithmetic average reaches the final value slightly more quickly, but the exponential average has a quicker initial response and a more "natural" settling characteristic. The result is a display that is seen by the user as more responsive. The figure illustrates another advantage of the exponential algorithm. It requires fewer captures to achieve similar results. In terms of rise time or 3dB point, an exponential average is comparable to an arithmetic average of three times the number of points. In terms of stop-band attenuation it requires half the number of points.

D. Teradyne's Ai-760 DSO

An example of an instrument that employs many of these averaging techniques is Teradyne's Ai-760 Digital Sampling Oscilloscope. In addition to its other features this DSO has the ability to perform post-processing of DSO captures, including application of the FFT (Fast Fourier Transform) using the Spectrum function in its Measurement Library.

1) Successive Sample Averaging With Decimation

The instrument has a High-Resolution mode where the instrument samples at an integer multiple of the user's requested sample rate. It then averages together the extra samples to improve accuracy, lower noise, and improve anti-aliasing. This averaging is done in firmware in real time, so there is no throughput penalty.

2) Successive Capture Averaging.

The instrument allows the user to average together multiple captures. This feature is supported, not only for real time sampling mode, but also for equivalent time sampling, where dithering of the timebase is used to fill in the gaps between samples. Special attention has been paid to mathematical precision, so that averages across several thousand captures can be performed without degradation by accumulated errors.

3) Continuous Successive Capture Averaging

The instrument supports a GUI display and can update it in real time, even when performing successive capture averaging. The instrument is designed to employ both arithmetic and exponential averaging in support of the GUI.

IV. Summary

DSOs employ averaging in a variety of ways. Averaging suppresses noise, increases accuracy and resolution, and when used in conjunction with decimation improves anti-aliasing performance.

Averaging across successive samples can provide effective noise reduction and resolution improvement, particularly where a low-pass characteristic is desired.

Averaging across successive captures is a powerful tool in that it suppresses noise while preserving bandwidth.

However it's applied, averaging always presents the user with a case of diminishing returns. Some amount of averaging improves a signal, but more is not always better. Averaging can't improve the resolution of an instrument by more than a few bits, and too much averaging can impact test times. An understanding of how averaging is used in a DSO can help the user to know when it will produce benefits, and when it won't.