ELECTRONOTES 223

Newsletter of the Musical Engineering Group

1016 Hanshaw Road, Ithaca, New York 14850

Volume 23, Number 223 November 2014

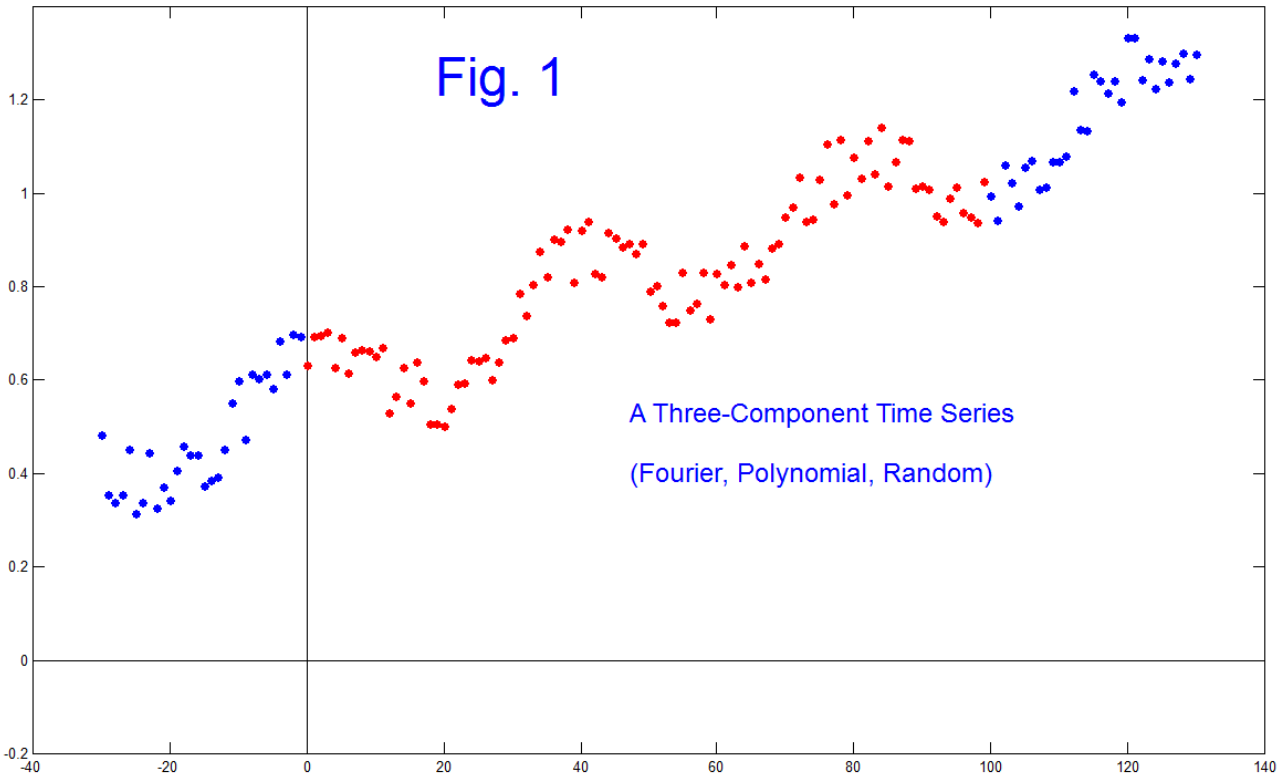
TIME-SERIES SMOOTHING - A REVIEW

-by Bernie Hutchins

INTRODUCTION

Here we have in mind a sequence of samples which correspond to a digital "signal" for which we shall be assuming the independent variable is time (although it could be something else) and that the samples are equally-spaced. In other words, a normal type of digital series. This could be something like physical measurements, perhaps of temperature, or perhaps something like stock-market values. Often times the series seems to be or IS noisy. The noise could well hide some underlying simple signal. Our goal might well be to remove some of the noise. Often (while doing so is highly problematic) we may well also have a goal of discovering and quantifying an underlying pattern, and not unusually, a goal of predicting into the future.

Fig. 1 shows the general sort of discrete series (signal) we have in mind. It was in fact generated by the sum of an upward ramp, a sinusoidal component, and a random sequence. The exact values are not important, although we will note that the individually described components are believable based on just the visual evidence. It might well be that this would not be the case – we might see very little except noise. Further here we will suppose that while all 161 points on the graph are potential data, we might only have available the 100 red points to work with. This "windowing" suggests the equivalent of a local "dc level" in the limited number of points. It may not be appreciated, until analyzed, that the data is generated from mathematical animals of three different classes: a Fourier component, a polynomial (the ramp), and a non-deterministic component.



{ It is not always noted that a constantly rising function – a ramp or "trend" - is polynomial and not Fourier. Like any polynomial, of which the ramp is first-order, it cannot be a model, long term, for anything that is physical, because all polynomials (except 0-order) run to infinity [1]. A polynomial may be useful locally (as in interpolation [2]). }

While we are entirely certain about how Fig. 1 was generated (we made it), and have admitted to three quite different processes that are added together, it is not clear that we could separate these again, or what the goal of doing so might be. In general, we may think of using some sort of filtering, statistical procedure, or other analysis to find and characterize "parameters" of existing data sequences (often time-series) which may be multi-component and noisy. Here we are interested in first applying our analysis tools to known, artificially-created sequences. This is done in an effort to see what sort of components can be isolated (the ability of available "tools"), and what sort of processing might produce spurious results.

In the case of Fig. 1, we can rather clearly and convincingly argue that the components claimed are right there to be seen, without any special processing. We might be interested in "testing" a process to be tried in a case where we believe (we think) we see something but want to see it better. Or perhaps we don't see much more than a hint, and want to enhance and/or suppress particular elements (like Fourier components) which we can define fairly well. Perhaps we want to reduce the noise. Or perhaps we want to remove the linear trend (the ramp). Perhaps we want to enhance what appears to be a periodic component. All this we propose to approach with extreme care – less we fool ourselves.

In Fig. 1 we have already suggested that we may not be processing with all 161 points of the data shown (blue as well as red). It may be the case that we don't even have the blue points. Or perhaps we are strictly reserving certain portions of the data as a test if we actually form a model from the rest of it. Or perhaps we want to test the processing as done on various portion sizes. In many cases, data points are not all considered equally reliable, so we are always cognizant of advantages and pitfalls of disregarding portions. Engineers in general offer no hard and fast rules in selecting data segments.

TIME DOMAIN – Basic View

Electrical engineers are very much aware of the usefulness, if not the absolute need, to consider data in both the time domain and the corresponding frequency domain. For some cases, something like music for example, the frequency domain seems primary, and ordinary frequency-domain filtering seems the most usable point of approach. Indeed, when filtering is mentioned, we almost always think in terms of a "frequency response" first. Time domain responses seem secondary. (Ohm's Acoustic Law: the ear is "phase deaf".)

Here we are specifically addressing "time-series" right in the title. By adding the term "smoothing" we pretty much suggest that something akin to low-pass filtering is to be brought into play. In fact, some would assume that some flavor of a moving-average is to be employed for smoothing. We can do this, and even most non-engineers will understand exactly what is being done with no need to even mention a frequency response. Everyone understands a basic averaging process – while perhaps missing the "moving" part! We will of course be comparing other filterings we propose to the moving-average. What we shall try to avoid is being surprised by the consequences of these filtering as reflected in a corresponding frequency response. Nor should we suppose that a "fix" for any discovered artifact or spurious result is to be approached only through time-domain analysis. After all, much is already known about digital filter design [3-5].

We can apply the idea of a moving average to the data of Fig. 1. We do have the duty to select a length for the moving average. Typically for example, we might replace the current value x(n) by some average y(n) such as:

$$y(n) = [\, x(n-2) + x(n-1) + x(n) + x(n+1) + x(n+2) \,] / 5 \qquad (1)$$

a length-5 moving average. This is a "zero-phase" FIR (Finite Impulse Response) filter, and could be any length with proper variations implemented. Here we say that the current value of the average is found by multiplying the center value of x, the two previous values, and the two following values, each by 1/5, and adding. A more general FIR filter would have a general length and not all weights (coefficients of the impulse response) would all be equal, although generally a symmetry of weights about the center is often preferred.

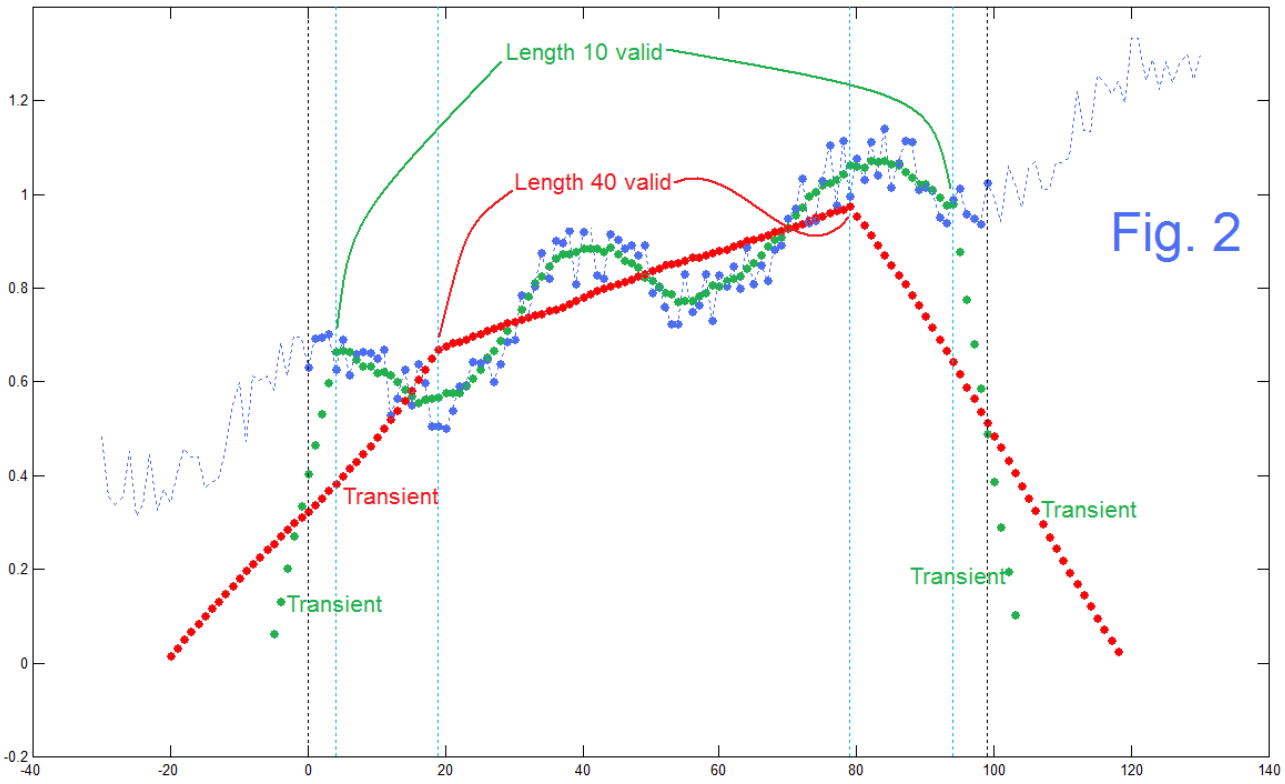


Fig. 2 illustrates the application of two different moving average filters to the data of Fig. 1. Here we have assumed the original data (blue dots and dashed blue) is available only for the range of 0 to 99 (the blue dots). Outside this range, we do not know the values and no guesses about the data points are made. For purposes of averaging, the data points on the ends are initially considered to be zero. Often non-zero guesses are made – but they are just guesses in general - although we might be just holding back the dashed blue portions for later testing. Further two different lengths are used for the moving average: length-10 (green dots) and length-40 (red dots). Clearly the moving averages have "transients" ramping up and down in an unrealistic manner. We might well argue that the output points in the middle (n=4 to 94 for the green, n=19 to79 for the red, as marked by the dashed blue vertical lines) are valid. In general, two points of view have merit: (1) don't do any smoothing and if you really must (2) then stay away from the transients.

A moderate level of study of Fig. 2 shows that the moving averaging may have done some good. Assuming that the random noise is <u>really</u> noise (not an information-bearing signal), the moving average seems to have reduced this noise. In this regard, consider the green dots (length-10) between n=4 and n=94, relative to the corresponding blue dots. If we then look at the red dots between n=19 and n=79, we might claim slightly better noise reduction, but probably not enough to sacrifice 30 more output points. <u>The astounding difference between the red and the green is the rejection of the periodic component</u>. This, more or less leaves only the ramp. Perhaps this is what we want – perhaps not.

To understand the rejection of the periodic component we can stay in the time domain and observe that if something is periodic with length 40, a length-40 moving average will remove it. As the data advances one step, a sample drops off the far end and is replaced by a sample 40 steps behind, which has the same value – no change. So it looks a bit like we might want to consider this in the frequency domain. Indeed, the frequency response of the length-40 moving average rejects a frequency of 1/40. But this view would be too simple and even misleading. Something far more interesting is going on.

FREQUENCY DOMAIN – Basic View

The frequency response $H(e^{j\omega})$ of a filter with impulse response $h(n)$ is given by

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h(n)e^{-jn\omega} \qquad (2a)$$

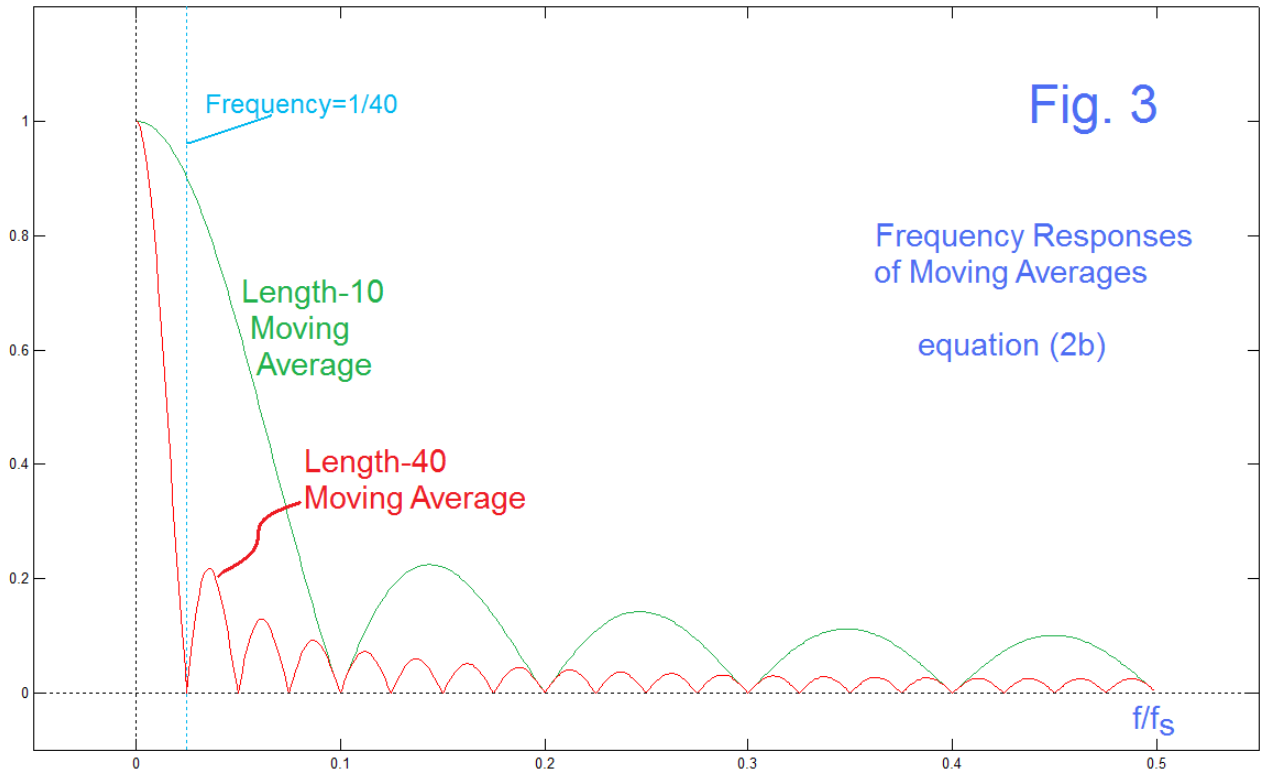
in all cases – the DTFT or "Discrete-Time Fourier Transform". For a length-N moving average h(n) = 1/N for values n=0 to n=N-1 (or for some other N consecutive values). The magnitude of this frequency response is:

$$|H(\omega)| = \frac{\sin(\frac{N\omega}{2})}{\sin(\frac{\omega}{2})} \qquad (2b)$$

and this is NOT a "sync" function because sync is not periodic, and equation (2b) is periodic because of the sine in the denominator. This periodicity is nothing more than a consequence of the discrete nature of the time domain signal h(n). [A sinc would have the form sin(x)/x.] Equation (2b) is usefully called a "periodic sinc" or revealingly, an "aliased sync" and sometimes a Dirichlet kernel. Because it is periodic with a period equal to the sampling frequency, we often just plot it on an interval of 0 to half the sampling frequency, and this is often normalized so that the sampling frequency is 1 Hz or 2π radians. We will be plotting frequency responses on 0 to 1/2.

We are likely familiar with the "dual" notions of functions in Fourier transform pairs, and know that a rectangle in frequency should be a sinc in time, and vice versa. This is true, but we also need to keep in mind when signals are discrete, at which point we have sampled rectangles and periodic syncs. Equations (2a) and (2b) are examples.

Fig. 3 shows the magnitudes of the frequency responses for a length-10 moving average (green) and a length-40 moving average (red). Note that both frequency responses are 1 at frequency 0 (dc). Further, while we have plotted a magnitude we recognize that alternating lobes alternate in signs. The longer length-40, red has four

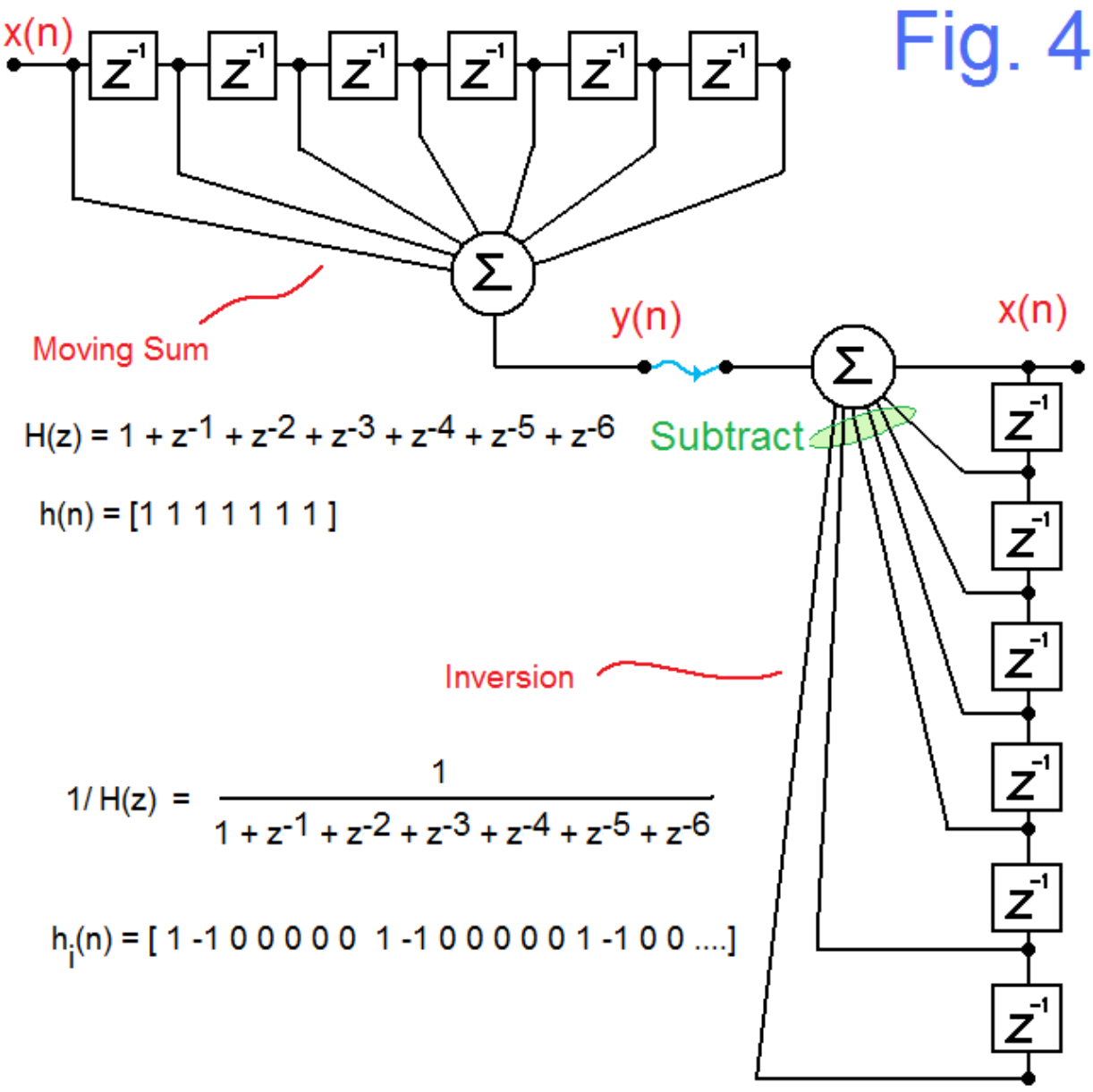


times as many lobes. But here the main item of interest is that the length-40 moving average has a null at the frequency 1/40 (dashed blue vertical line) while the shorter length 10 has a response there that is just above 0.9 (very little attenuation). So while we understood the rejection of the periodic component of period 40 as a time-domain phenomenon, we now see it in the frequency domain as well. *Or do we*?

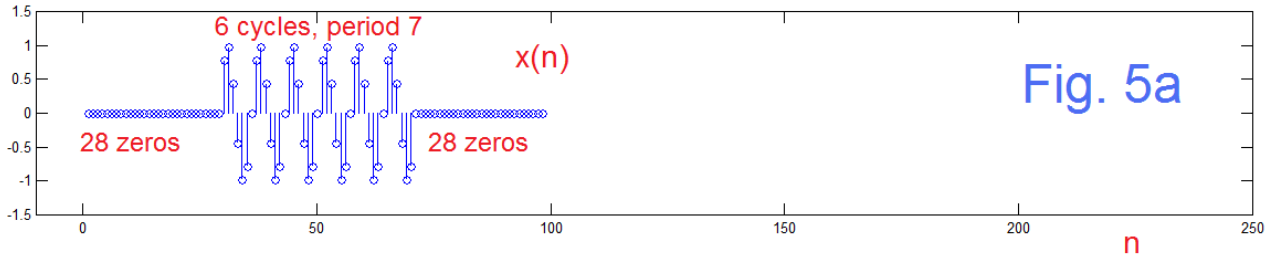
In some over-viewing sense, we see the rejection of the Fourier component as a result of the moving-average being fundamentally low-pass, and the longer moving average has a lower cutoff, which moves below a particular frequency of interest. Looking closer, we see not just smaller sidelobes, but an actual null at the frequency 1/40 for this case, but still generally low-pass. But it is also clear that if we choose a length somewhere around length-40 but not exactly length 40, some signal would get through. In fact, if the length were increased to 60 we see that the first sidelobe (at about 0.2 magnitude) will be on the frequency 1/40. So it is important to understand the entire response.

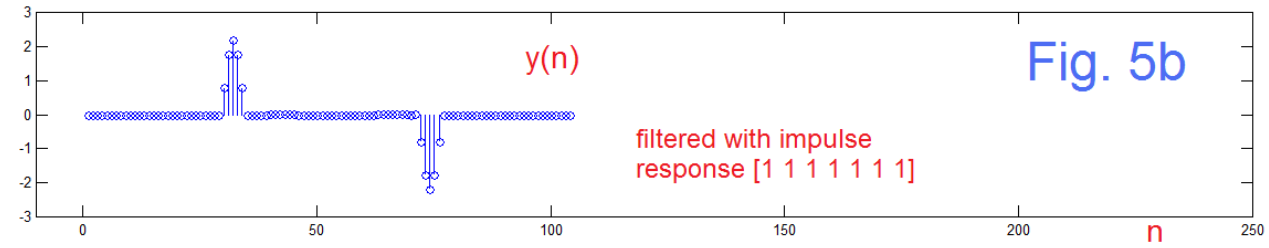
Even more caution would be afforded by considering that we have not actually tested the filter with an input component of 1/40. We only had a sinusoidal burst of about 2.5 cycles, and the Fourier transform of this is not a single frequency but a band centered about 1/40. Such testing (or use) of a filter with a limited duration sinusoidal is obviously general – we have no infinite duration signals of any sort. When we test a filter on the

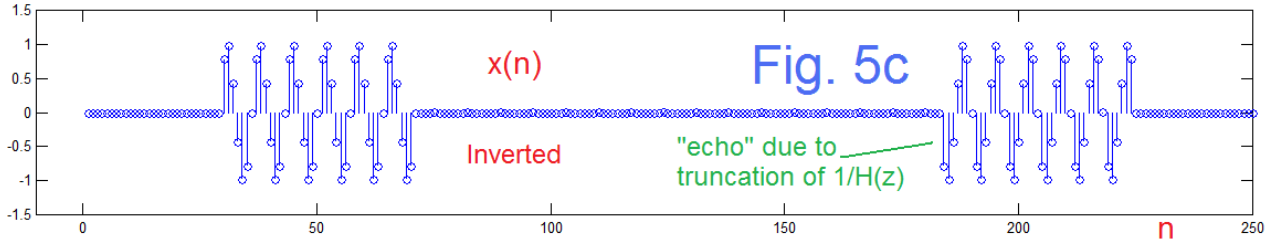
bench, we power up the filter and switch on a function generator as input, and switch it off after making an observation. We consider this, usually swept over a range of frequencies, to be an adequate measurement of the "steady state" frequency response, and ignore the start-up and ending "transients" as being either already finite, or decaying far too rapidly to measure. The steady-state portion of the length-10 output in Fig. 2 is from n=4 to n=94. The green dots outside that center range are transient. Here the transients are steep sloped because of the pedestal nature of the ramp component (as it is time-limited here). We can isolate the transient due to the sinusoidal component as follows, by borrowing Fig. 4 and Fig. 5 from a previous presentation [6]. This shows recovery of original data from a moving-average, and is important as it shows how the moving-average rearranges data rather than blocks it. Keep in mind that we are not inverting an average (one number!) but a moving average sequence.

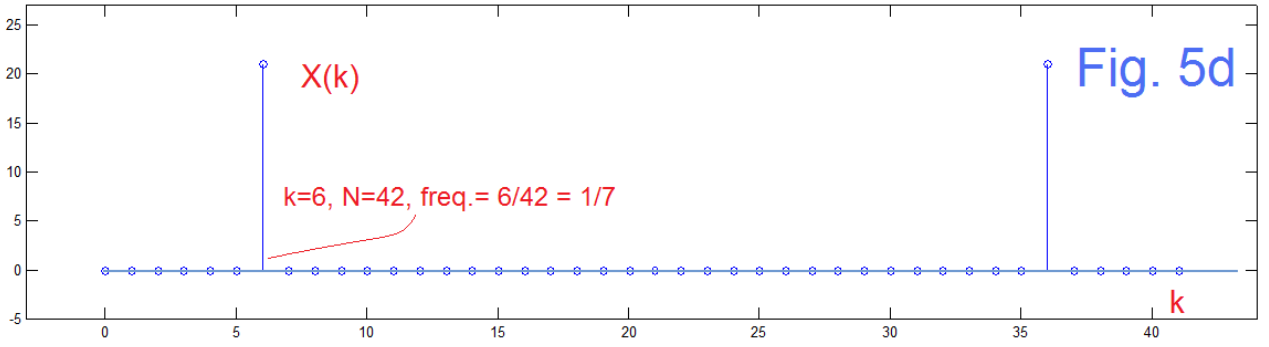


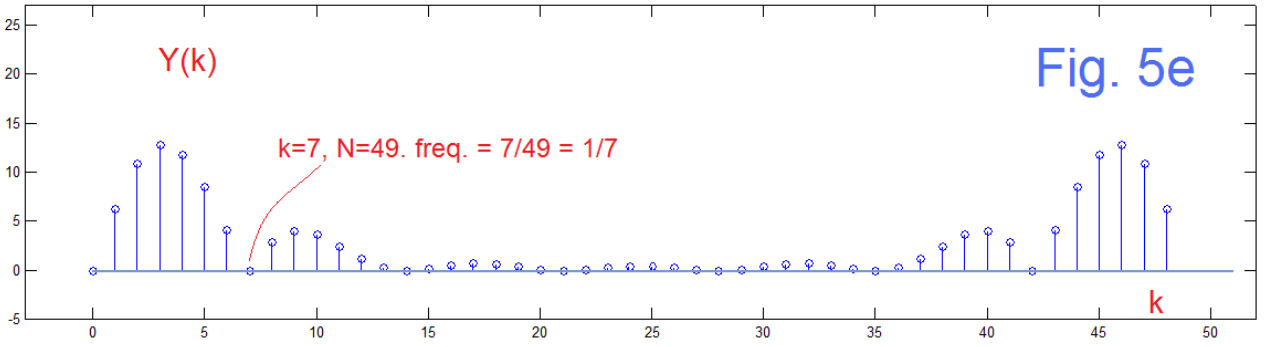
The FIR filter H(z) in Fig. 4 is a moving-sum for simplicity [for a moving-average, divide the output y(n) by 7]. As discussed in detail in [6], we can invert ("un-filter" or "equalize") y(n) back to x(n) using 1/H(z), which in a non-decaying IIR in this case. Nothing in the scheme would indicate that any signal x(n) is treated any differently from any other.











Indeed Fig. 5 which shows the sequences involved in Fig. 4 for a sinusoidal burst at a frequency where there is a steady-state null shows (perhaps surprisingly) that the inversion scheme still works. (Here we compare Fig. 5a and Fig. 5b to the blue and red dots of Fig. 2.) In both cases we see the center region of the sinusoidal sequence nulled out. Here the actual nature of the transients (Fig. 5b) are much more apparent. The transient in Fig. 5b begins on the left with the length-7 moving sum moving into the sequence [0.0000 0.7818 0.9749 0.4339 -0.4339 -0.9749 -0.7818 . . . .] so we get a sum reaching 2.19 and returning to zero and staying there until the sixth cycle at which times the transient reverses. That's all there is to y(n). The middle is "gone" in some sense. But……

It was the purpose of the app note [6] and two follow-up notes [7,8] to show how the moving-average could be inverted (reversed), and this is what the 1/H(z) of Fig. 4 and Fig. 5c is for. We see clearly that x(n) can be recovered exactly from y(n), and a rather complete study (and a "Recipe") is available [8]. Here we must remark on the "echo" that appears on the right side of Fig. 5c. We need not have this. It appears because we truncated the otherwise infinite-duration and non-decaying impulse response of 1/H(z). We waited until the original sequence was recovered and then a good bit more. Doing this is of the same nature as would be the testing of a filter on a workbench followed by shutting down the equipment. See reference [8] for more discussion.

Fig. 5d and Fig. 5e show the process in the frequency domain, using the FFT. Fig. 5d is NOT the FFT of the whole signal x(n) but only the FFT of the 42 samples corresponding to the six full cycles. As expected, the FFT shows a non-zero result only for k=6 and k=36, corresponding to a frequency 6/42=1/7. When we pass these six cycles through the length-7 moving-sum, we get the result y(n) which is the convolution of a length-42 x(n) with the length-7 moving-sum, which is length 49 allowing for the symmetric zero on the end. Taking the length-49 FFT of y(n), we find there is energy for all k except for k=7 and k=49-7=42 (and multiples of 7). This is the same frequency of 1/7, this time as 7/49.

So what happens is that the moving-sum has forced the energy out of its perfectly resolved case for the sinusoidal burst and into all other frequencies - it is now all in the transients. So what? (In cases where we are then inverting back from the transients, the inversion is still perfect.) However, with time-series smoothing, it is often the case that we try to (artificially) "pad" the ends of the time series (quite artificially) so as to make the end effects (the transients) more like what we would have had if we had more data beyond the ends. Still no problem as long as we do not modify the thus-obtained transients (even by ignoring them – i.e., cut them off to make the sequence its original length) and do not then proceed to further analysis/processing. Now, there is a close relationship between reconstruction from smoothed data and making inferences about what is "really" there but "hidden" by our limited availability of correct data. Since we know we can invert transients for an exact recovery, it follows unavoidably that if we mess with the post-filtered ends for modified transients, we are messing with what is really there. Handle with care.

FREQUENCY DOMAIN – Filter Design

We have seen that the moving-average is a simple and relatively poor low-pass filter. Still it seems to have merit when we consider just noise reduction. But we certainly know how to design filters that do a lot "better job", and arguments about computational advantage of a moving average can be quite silly, since we are often talking about a small handful of computations done in fractions of a second, one time. Perhaps we are smoothing data once for a paper to be published, not designing an audio player that runs continually. Further, most (sophisticated) modern filter design methods optimize some performance parameter mathematically in a well-defined way. As discussed in Appendix A, such methods tend to converge. A method that minimizes integrated squared error may be almost identical, in terms of network coefficients and frequency response, to one that minimizes maximum error (for one example); given sufficient, and comparable resources (like similar lengths).

The other thing about filter design relates to the well known uncertainty principle. That is if we want a filter that has a feature characterized over a certain (small) interval of frequency, we are not going to get it unless we have an impulse response of sufficient length. Fig. 3 is a good example. The low-pass passband for the length-10 filter is 0.1. In order to get a smaller passband of 0.025, we needed a length-40 filter. This is well known. What is perhaps not so well appreciated is that because of thus uncertainty relationship, no matter how clever the designer is, if you do not invest in a longer filter, your improvements in performance specifications with respect to cutoff rates will be minor.

We could choose a good number of design methods to illustrate here. The point we want to study is whether or not choosing a sophisticated design procedure, to replace moving average, pays off (not much) and/or if it introduces artifacts to interpret (likely does). In other sections we use the "error criterion" of minimizing the integrated squared error in the frequency domain (Matlab ***firls***). This is popular, as is the neat design of minimizing the maximum error, thereby achieving an equal ripple [variously called Parks-McClellan (PM), min/max, equi-ripple, or Chebyshev error] and found in Matlab as ***firpm*** or ***remez***. Since our basic design goal is likely a rectangular low-pass, the impulse responses tend to be sinc-like. The PM impulse responses are sinc-like but may have a pair of taps out on the ends that kind of jump up. This is a consequence of the comb-filter-like nature of the equalized ripple. Here, using PM will be useful as showing the essential points about the uncertainty relationship, and to remind readers of this useful design in the "tool-box".

Fig 6 shows three different PM designs, impulse response on the left, and frequency responses on the right. The top two designs are narrow-banded to correspond exactly with the moving-averages of Fig. 3. The first design is length 10 and the second and third are length 40. All three have required "don't care" regions and band weightings that need not concern us right now.

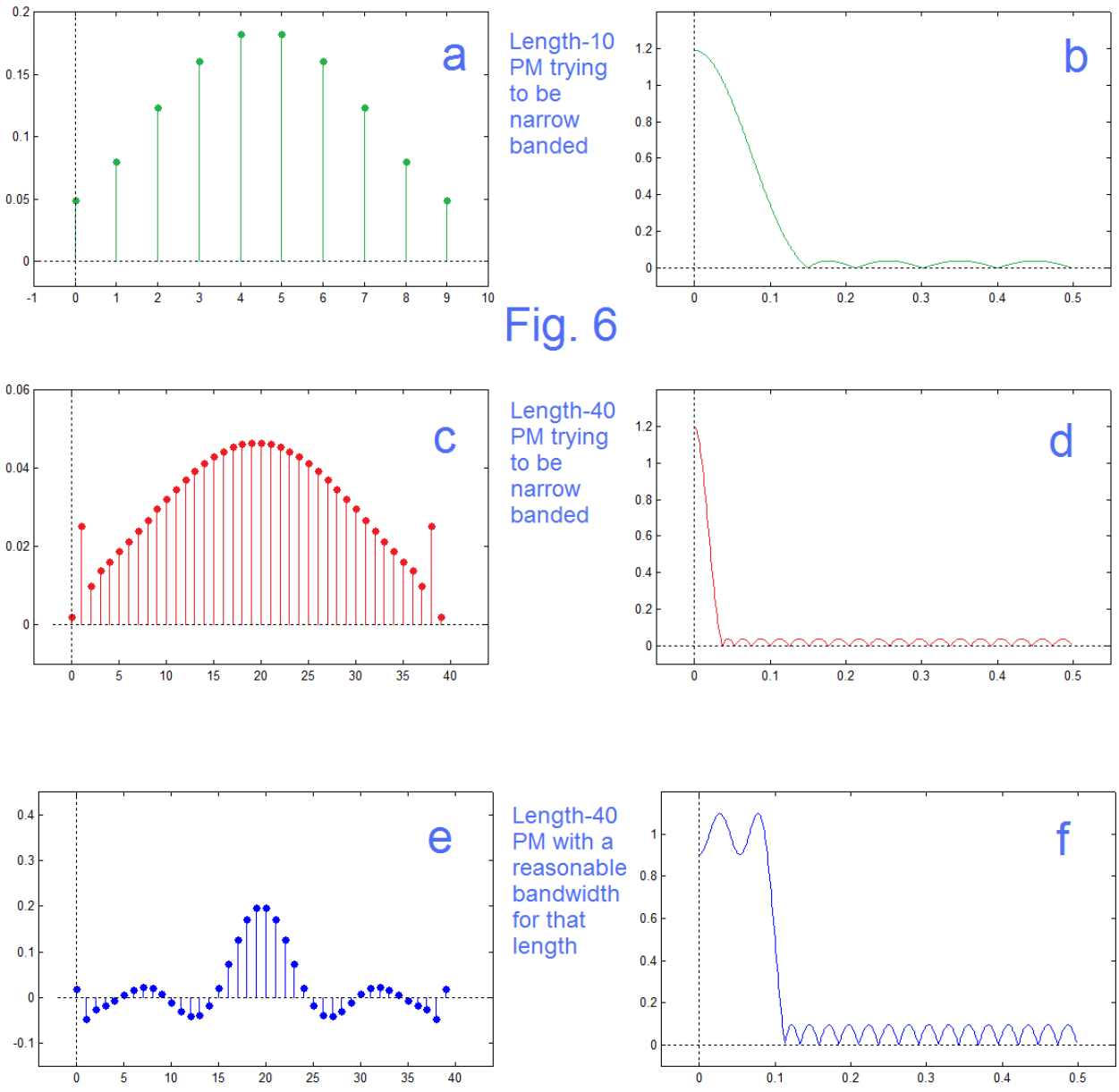


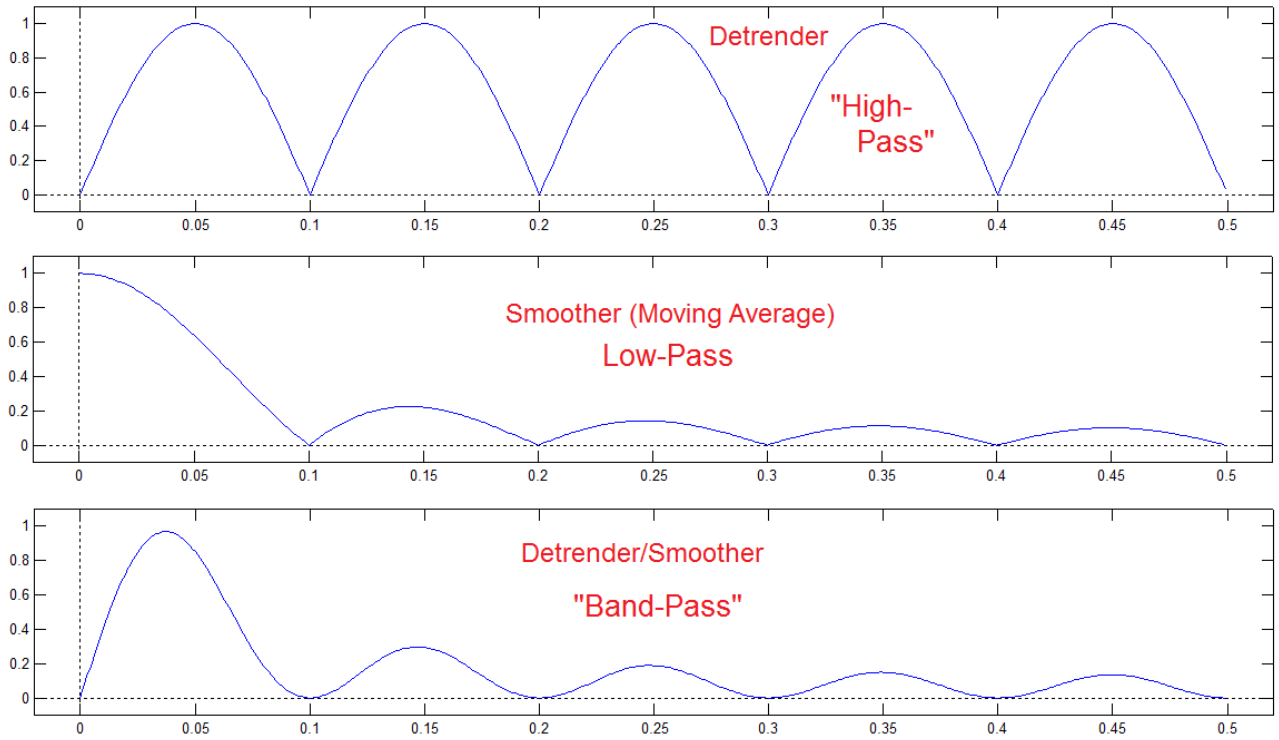
Fig. 6

Neither of the top two designs is particularly impressive if we were expecting “magic” instead of an optimization exactly according to what a particular theory guarantees. Indeed the frequency responses (green and red) do resemble the corresponding ones in Fig. 3, except the stopbands are improved through the equalization of the ripple and an additional weighting of the stopband error. That is, we monkey with the parameters to push the design until it pushes back. Note that the impulse responses are rounded or ripple as compared to the perfectly rectangular moving averages (not shown). What this is telling us is that insisting on narrow passbands and short (relatively short) lengths is not a game we can win – elegant methods or not. For comparison, by allowing a wider passband, the length-40 PM can be quite attractive (Fig. 6e and Fig. 6f - blue), which shows a more typical PM equiripple result. For certain, we could achieve a much better response with the narrow passbands by increasing the lengths of the filters well beyond 40. But this would have made filtering such as Fig. 2 all transient as the convolved length expands.

FILTERS PLAYING TRICKS

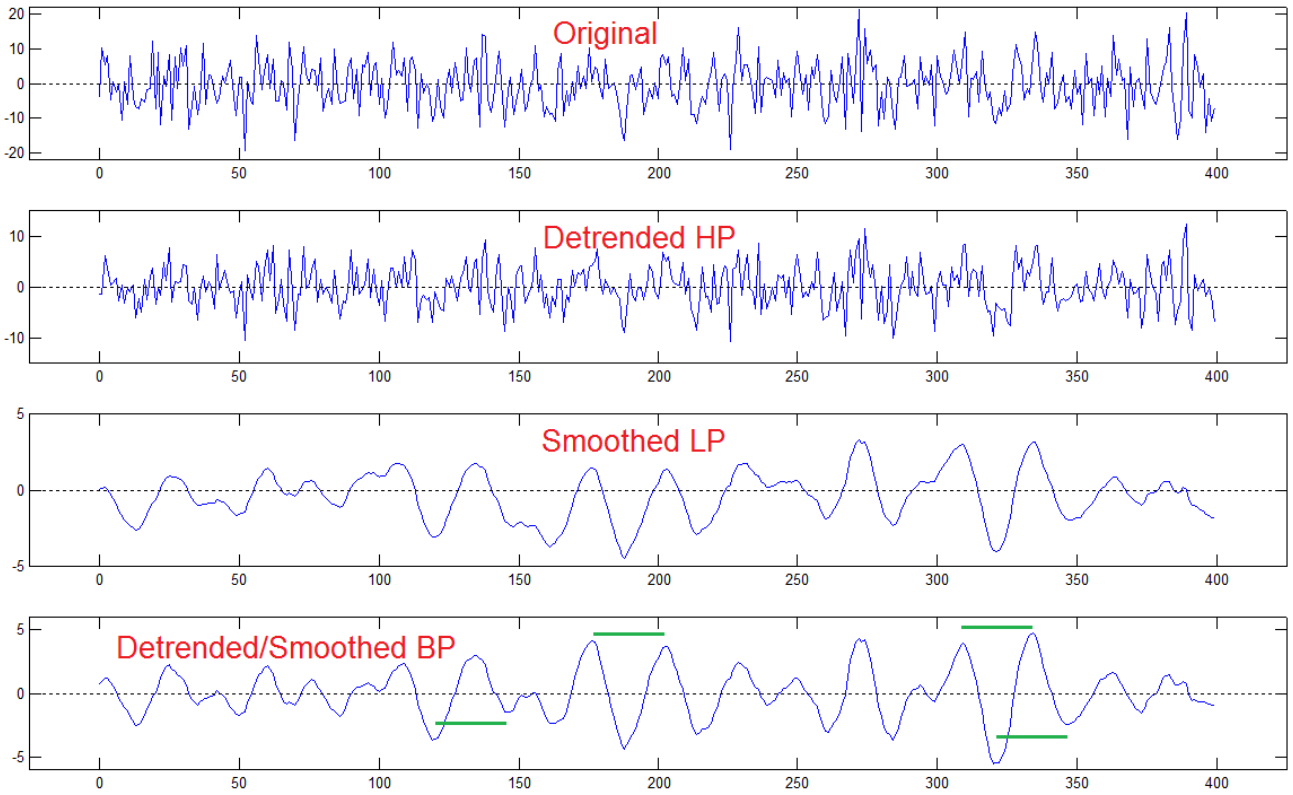
On the one hand, we tend to think of filters as honest and friendly! On the other, we know they can fool us. If we look at a filter output and see something that was not apparent in the input, we may well initially take it to be the case that the filter "dug" this out for us. For example, if we have a noisy signal and a very sharp band-pass filter, we often see a sinusoidal component corresponding to the center frequency of the band-pass at the output (colored noise). The sense in which this is "real" or not can be complicated, if not philosophical. Indeed, we have seen that a result can be downright spurious [9] as a result of detrending (high-pass removal of a linear trend or ramp) followed by moving average smoothing (low-pass), combining to a band-pass bump, as shown in Fig. 7 and Fig. 8.

Fig. 7



In Fig. 7 we employ two filtering: detrending at the top, moving-average smoothing (middle) and both in series (bottom). The middle moving-average is familiar from the above material and we recognized it would be basically low-pass, have a relatively narrow low-pass band, and significant sidelobes. The impulse response of the smoother is length 10, being (1/10)[1 1 1 1 1 1 1 1 1 1]. The purpose of the detrender is simply to remove any linear trend. It's impulse response is (1/2)[1 0 0 0 0 0 0 0 0 0 -1] and has the comb-filter response in the top panel of Fig. 7. Combined (bottom panel), roughly speaking it turns the first lobe of the moving average into a band-pass. Probably we were not thinking of it in those terms.

Fig. 8



From the bottom panel of Fig. 7 we see that the band-pass peak is at roughly 0.04 or (1/25) favoring signals with periodicity 25. To see what this means, we can look at the correlations of two white noise segments as originally generated and then subject to various filtering (Fig. 8). The details of what we have done (straightforward but with some necessary intermediate steps), and more examples, can be found in the reference [9].

We started with two different length-1000 white noise signals and the top panel of Fig. 8 shows the center of their correlation. Nothing to see. The signals are not correlated. There is very little difference with the similar correlation of the detrended signals (second panel) which we also might have expected as there is little in the way of low-frequency trends.

Smoothing of the white noise signals, on the other hand, has a much larger effect. We see correlated components corresponding to low frequencies (third panel of Fig. 8). This is principally due to the first low-pass lobe. This first lobe is not that dissimilar to the band-pass, and we see some indications of a, roughly, length-25 periodicity. But the case of detrending and smoothing (bottom panel of Fig. 8) is quite convincing. Four of the many possible length-25 segments are shown with green bars.

It is still a somewhat philosophical question as to whether or not the correlation was “in” to original data. But it is unquestionable that we have enhanced what would at best be a very weak correlation, and the result obtained is determined by our choices of filter.

Above with detrending and smoothing we were not so much thinking of frequency-domain filtering as we were time-domain manipulations for “better seeing”. At times, we do think directly of frequency-domain filters. From such a perspective, we might hope a particular filter blocks the “bad stuff” and lets the “good stuff” through – whatever we take

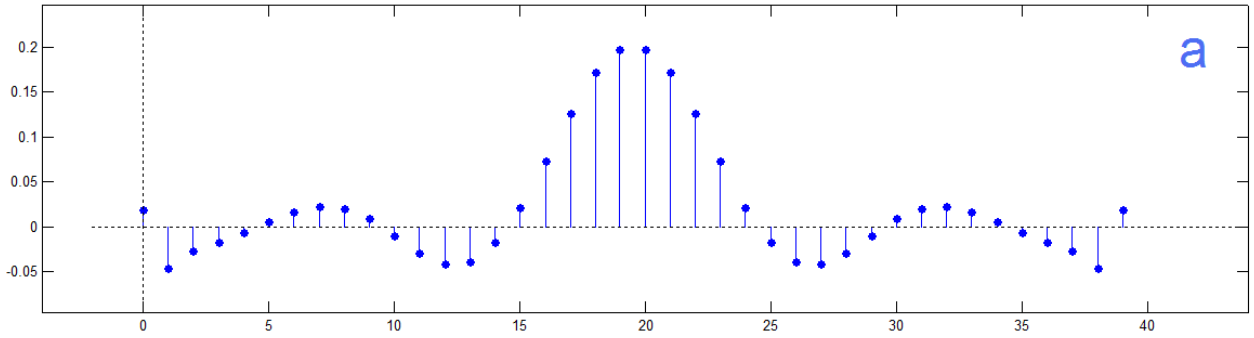
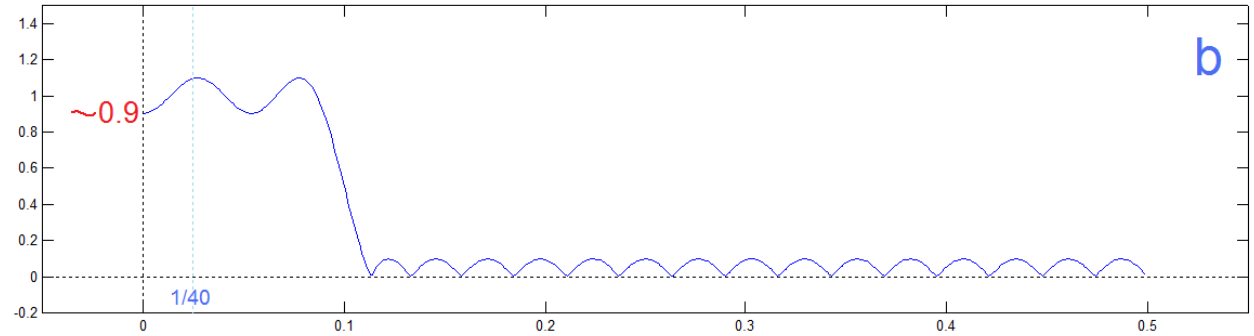
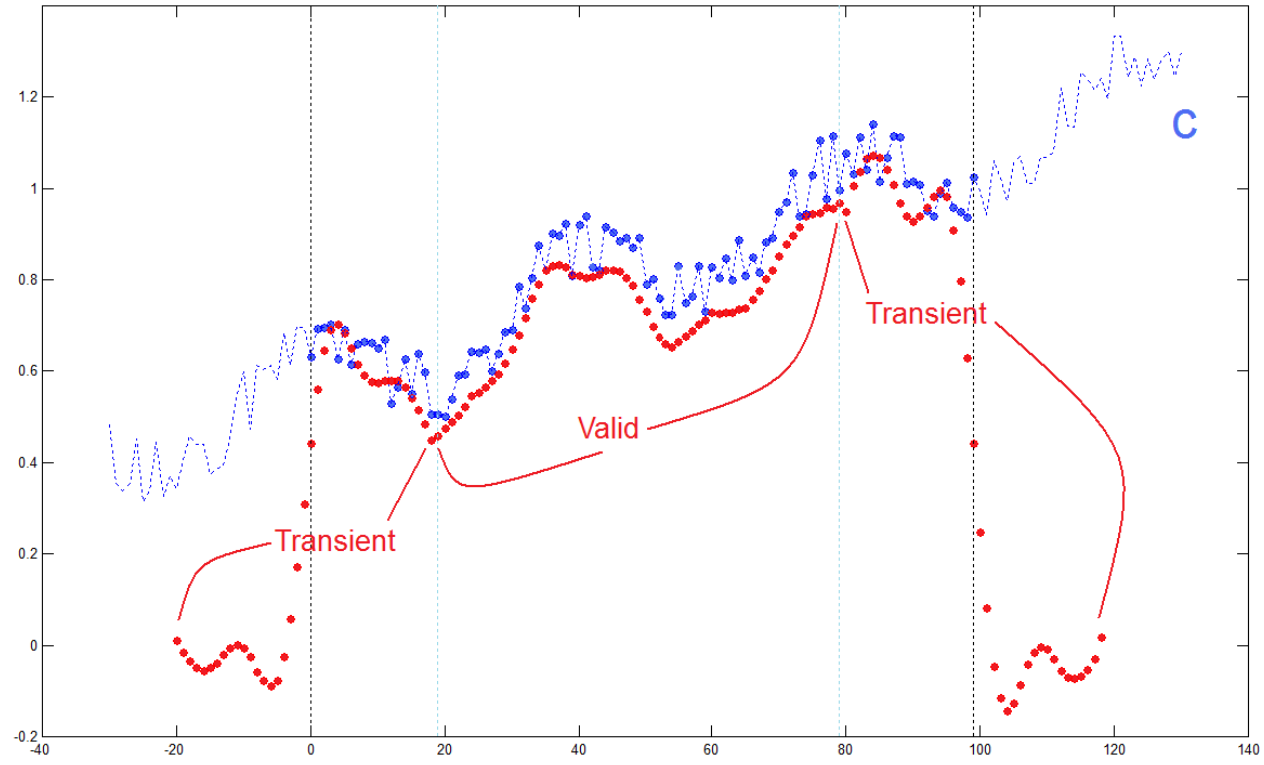


Fig. 9





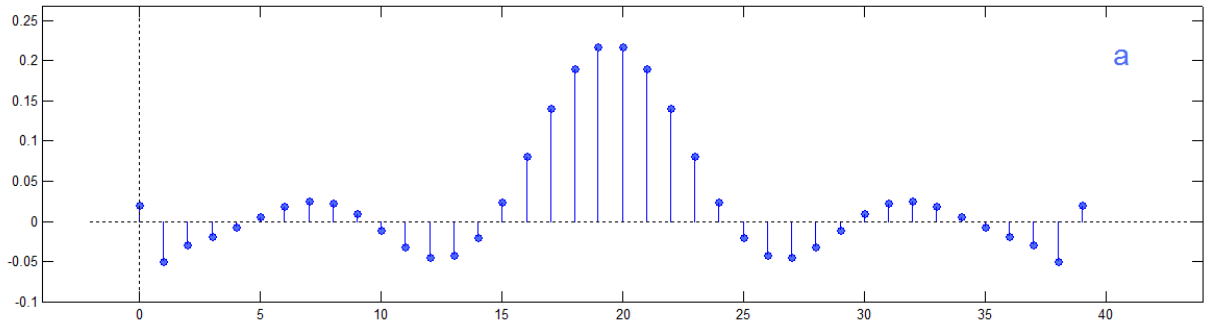
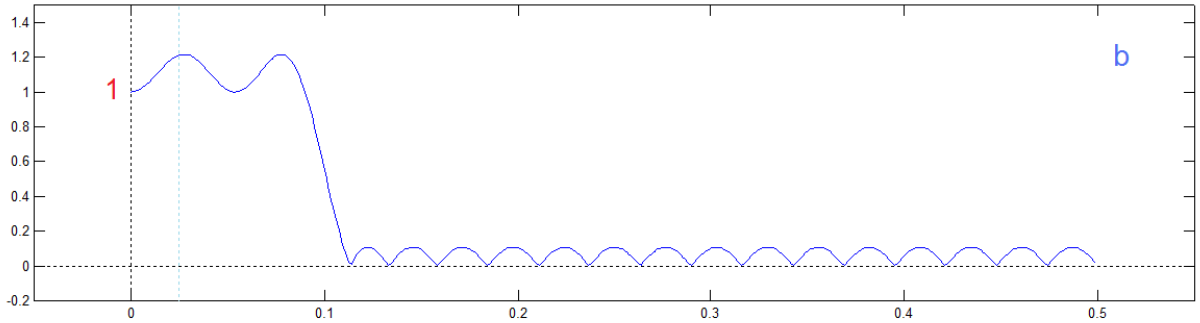
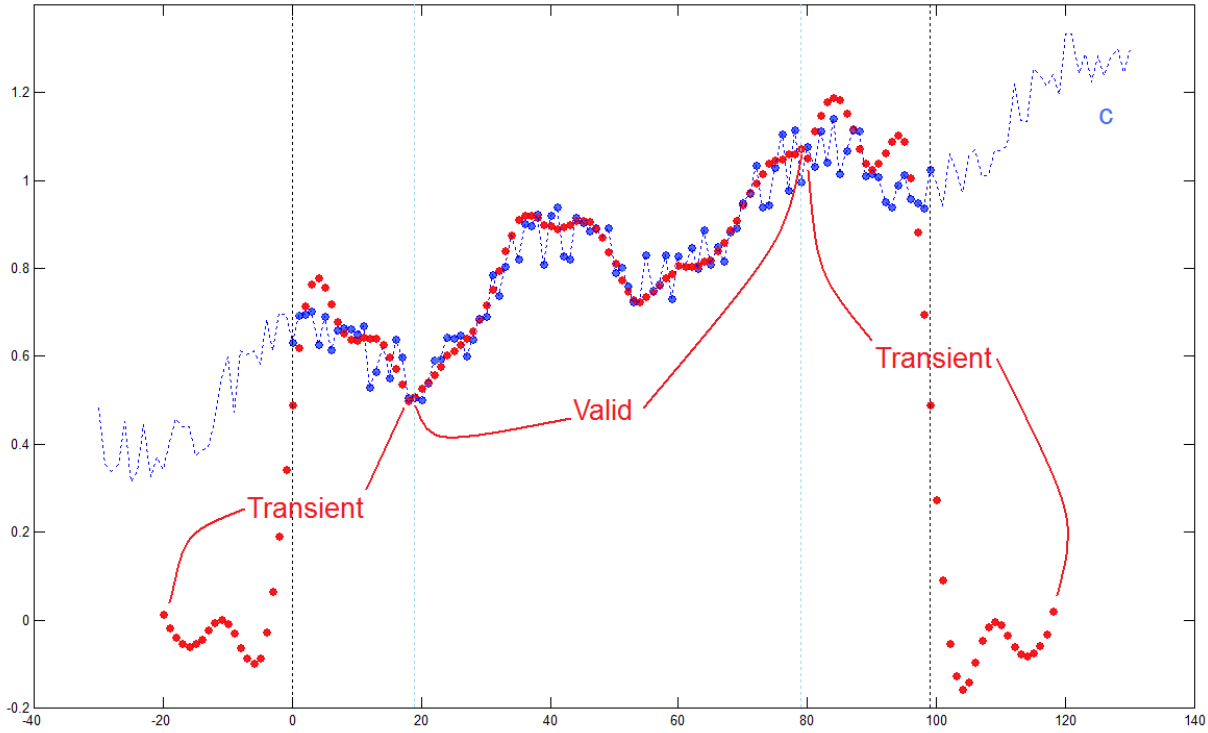


Fig. 10





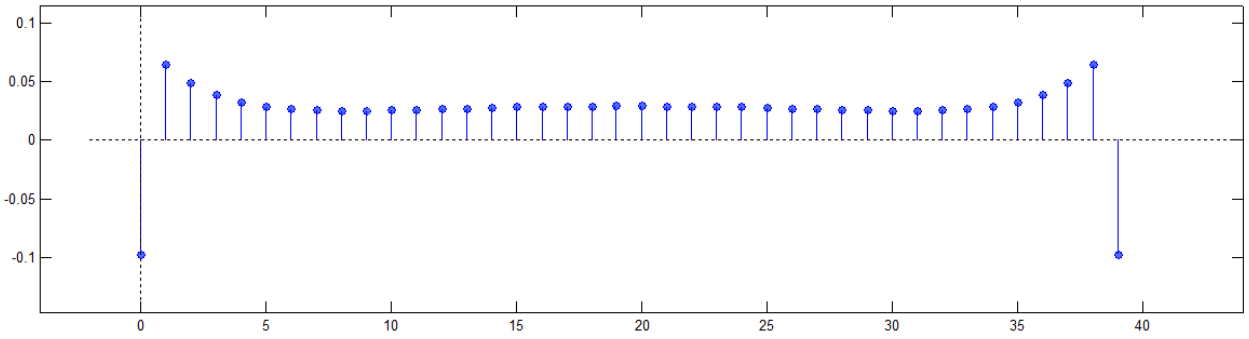
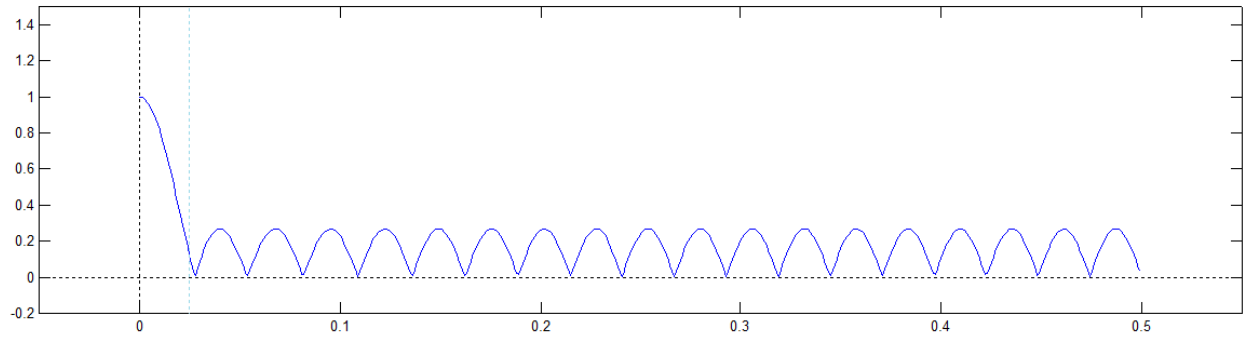
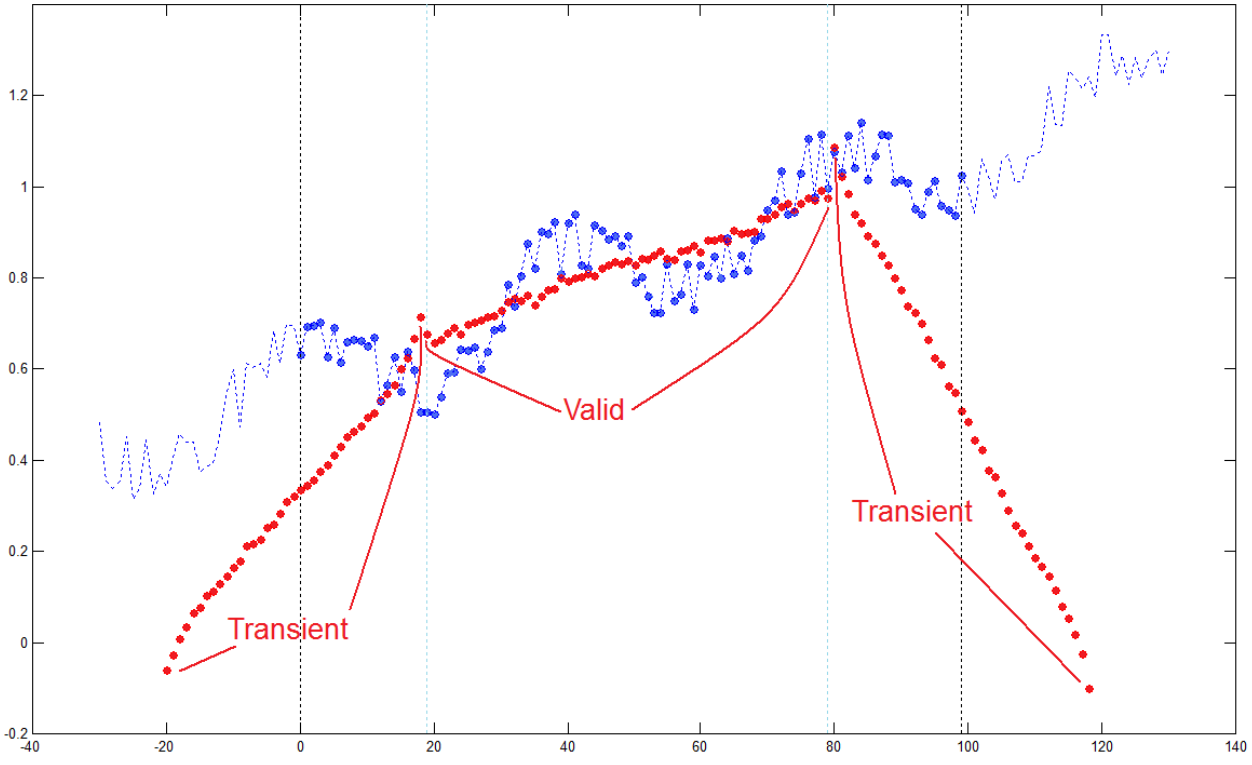

0.1
0.05
0
-0.05
-0.1
0
5
10
15
20
25
30
35
40

Fig. 11


1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
0.1
0.2
0.3
0.4
0.5


1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-40
-20
0
20
40
60
80
100
120
140
Valid
Transient
Transient

that to mean. And we would not like the filter to make its "own stuff". In addition, by embracing a sophisticated filter design method such as PM or minimum integrated squared error (Matlab's ***firls***) we are changing the initial viewpoint. We design a "good filter" and see how it handles our data. This, generally frequency-domain approach, probably does not guarantee performance results in an obvious way – we have to look.

Fig. 9 shows a straightforward PM design applied to the data of Fig. 1. This is a length-40 FIR design with a low-pass cutoff of 1/10. (Note that a low-pass cutoff of 1/2 would mean nothing was rejected.) As with moving average, we see the removal of much of the noise. Further, we have large transient regions and indeed only the center 60 points are in a steady-state ("Valid") position. We see the upward ramp and the sinusoidal component. It is clear that a sinusoid at frequency 1/40 should pretty well get through the low-pass with cutoff at 1/10. So what is most notable here is the fact that the output is lower than expected. This we understand in terms of the FIR design minimizing a maximum error, but not requiring a particular "dc response" to be 1. Indeed we see from Fig. 9b that the response at dc is only about 90%. This is very evident here, and easily fixed (Fig. 10) by simply rescaling the impulse response upward (or by other means). So we understand what went wrong in Fig. 9. (Perhaps more of a problem would be a case where the dc response was only slightly different from 1 (perhaps 0.98 or 1.02) in which case the error might not "shout" at us.) Fig. 11 shows the effect of lowering the cutoff below 1/40 which tends to reject the periodic component. Of course, since the input signals are time limited, the frequency response view used here is useful, but not an exact explanation.

By using the controlled design of an FIR filter, we can better say what we are attempting to accomplish. This, keeping in mind the limitations of (erroneously) supposing we understand the Fourier content of our time-limited signals, is generally helpful.

CHOOSING MOVING-AVERGE LENGTH FOR A SPECIFIC PURPOSE

It is a usual case that we invoke moving average with the almost casual understanding that we are doing it primarily to reject a certain periodicity while at the same time enhancing an underlying low-frequency (and rejecting noise). A good example might be the filtering of monthly climate temperature data. Given that this data might be noisy even averaged over the entire month, over how many months might we want to average it to reject the yearly seasons? Of course, we average over 12 months (neglecting the slight variations of the number of days per month). We have recently looked at this in some detail [10].

Fig. 12 is a block representation of some moving-average choices. This is shown for the choice of January of one year to January of the next year. Suppose we try a moving-

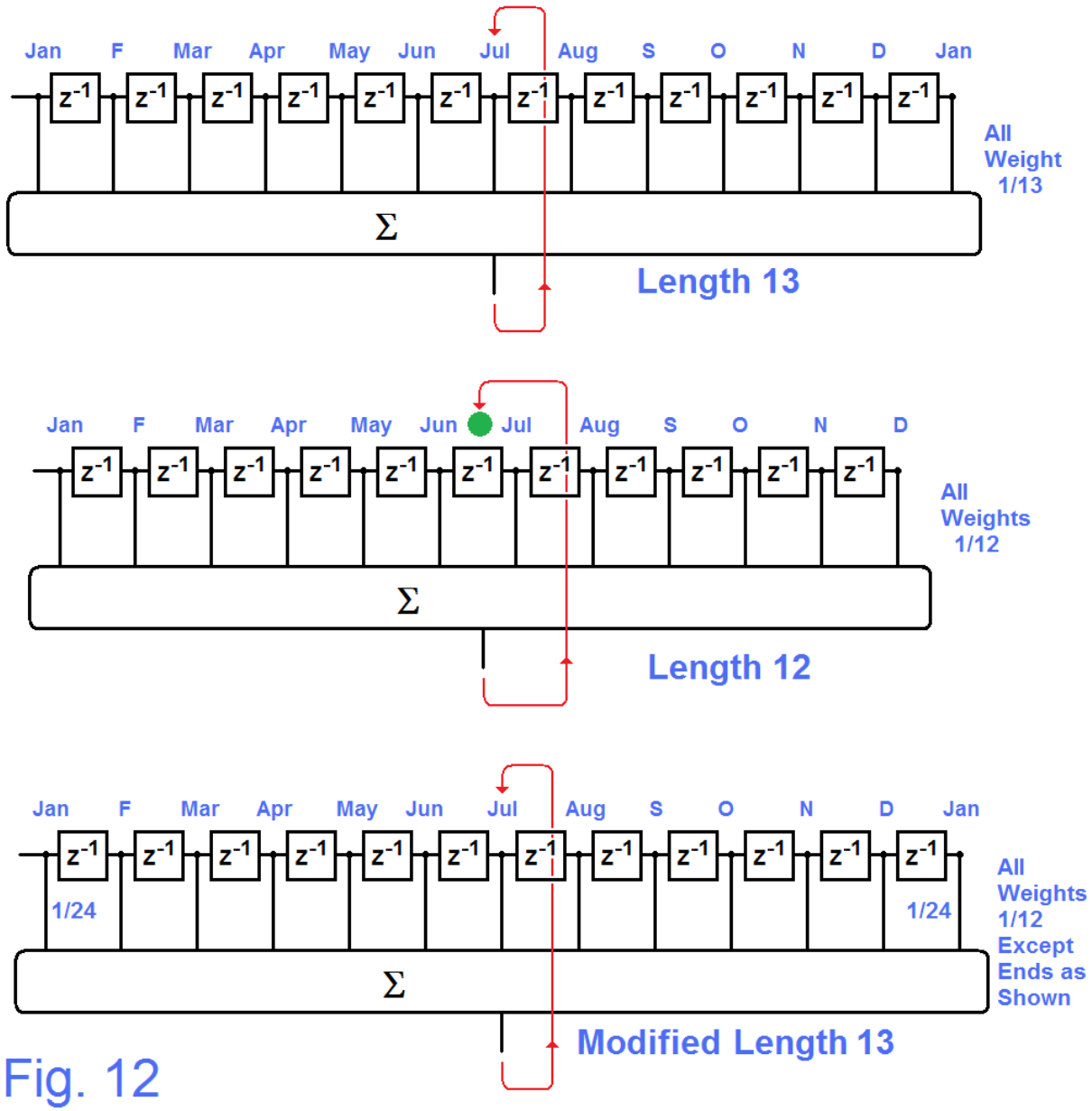
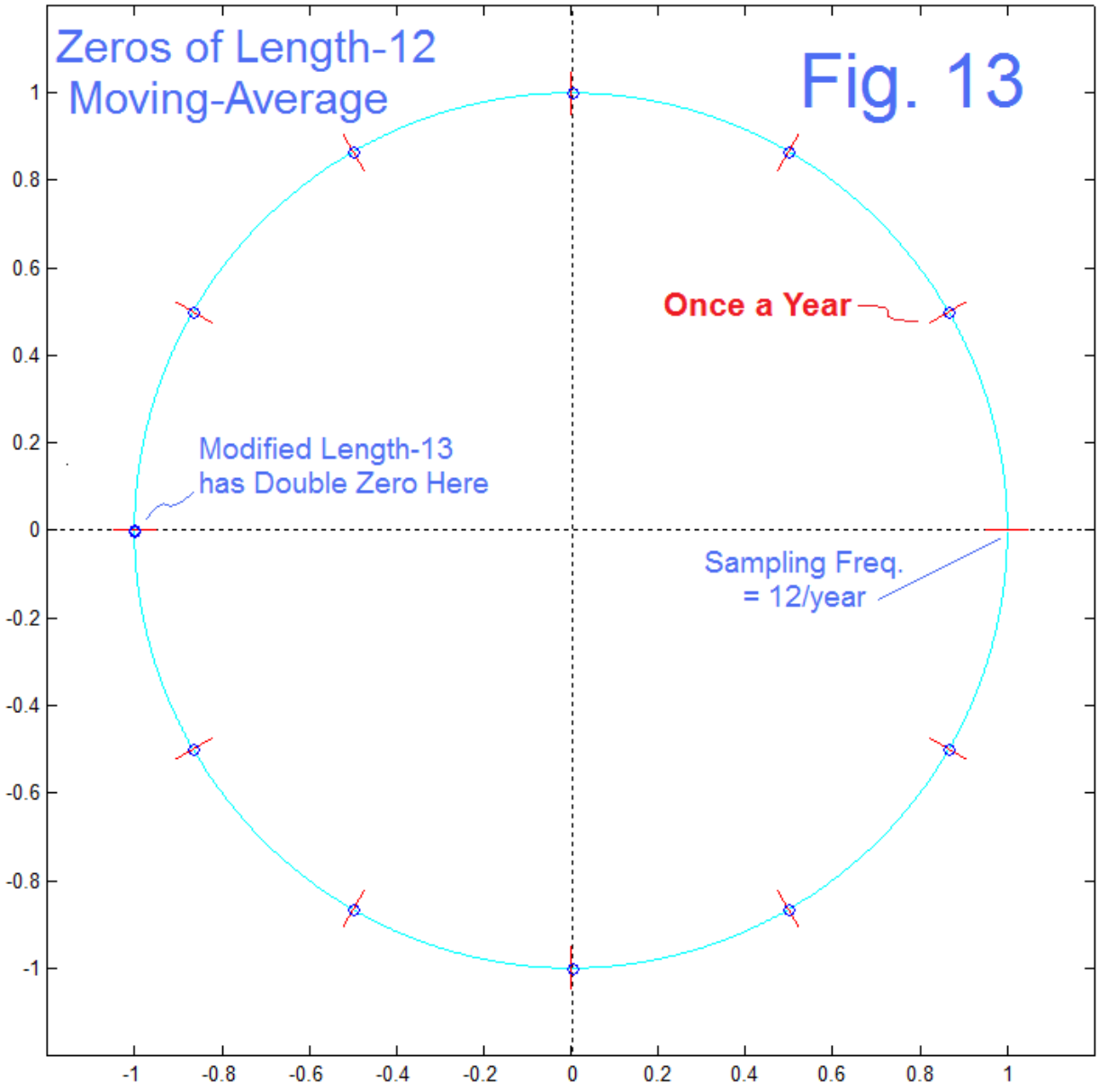


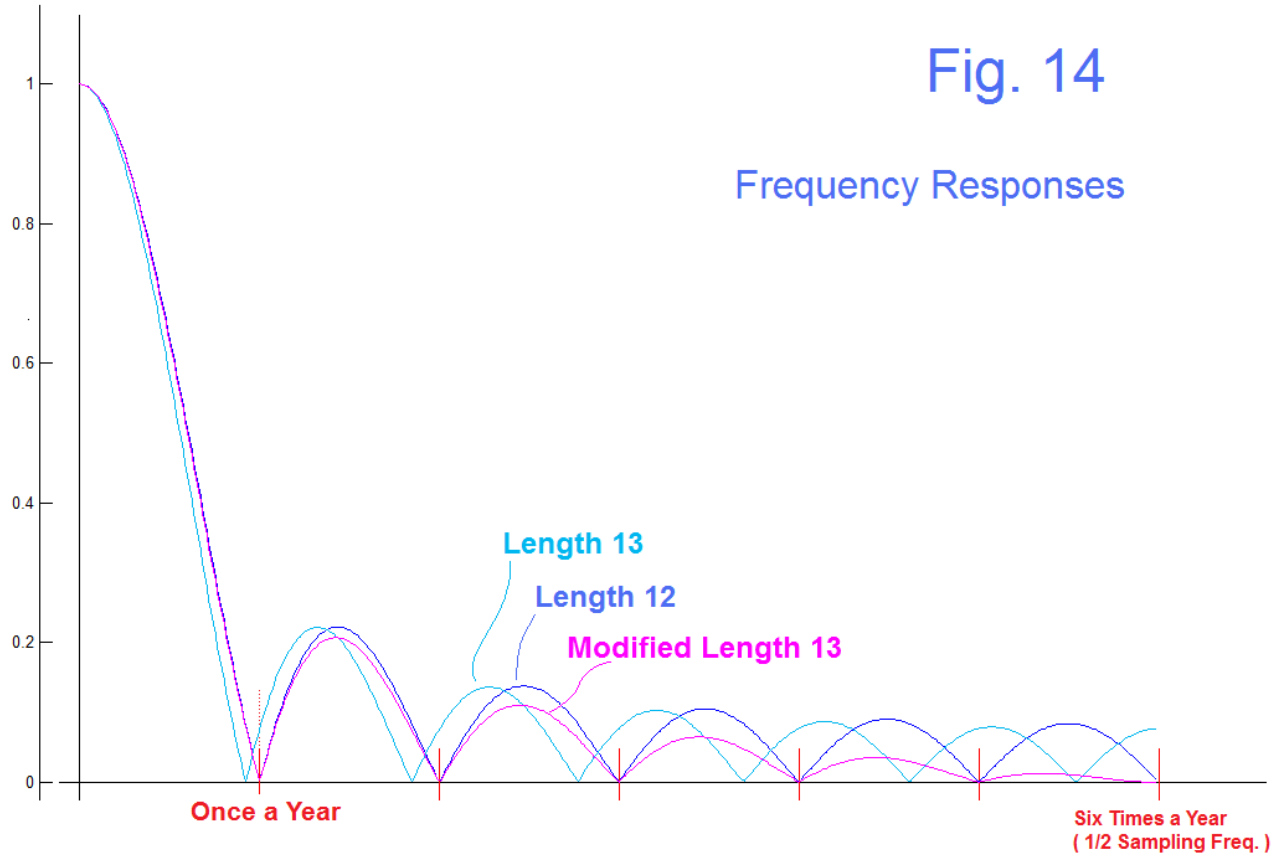
Fig. 12

average of length 13 and include both the first and the last Januaries (top network). Not likely a good idea. It has the single advantage of having an output that is exactly placed in July. In contrast, a length-12 moving average which represents each month only once (from January to December here) places the output (the center of the FIR) between June and July (middle network of Fig. 12). In all likelihood, this is something we consider and adjust for with just a moment's consideration. To the extent that this half-month shift may cause problems, we may want to consider a modified version of the length-13 moving average. Quite intuitively, we would keep the same month on the ends but weight the ends half their original value (1/24 instead of 1/12) as in the bottom network of Fig. 12.

Fig. 13 shows what happens in the z-plane. To begin with, we have not shown above where the zeros of the moving average are. For an N-point moving average, they are at N-1 points on the unit circle at N equally spaced points, excluding z=1, as in the figure.



Thus the length-12 moving-average has 11 zeros spaced 30° apart, with no zero at z=1. In the modified length-13 moving average, we convolve the length-12 with a length-2 moving average. The length-2 has a zero at z=-1, and this is a second zero at that location. Nothing wrong with that really – as we show in Fig. 14. Let's be sure we understand the weights. The length-12 has weights h_{12}=(1/12)[1 1 1 1 1 1 1 1 1 1 1 1] while length-13 has weights h_{13}=(1/13)[1 1 1 1 1 1 1 1 1 1 1 1 1] and the length-13 modified is h_{13mod}=(1/12)[0.5 1 1 1 1 1 1 1 1 1 1 1 0.5]. We see in Fig. 14 that the length-12 is exactly as we have seen above, as is the length-13, which is fine <u>except</u> it does not notch out the seasonal variation (once a year). The zeros of the modified length-13 are the same as the length-12, and null the seasonal variation. The extra zero at z=-1 actually improves the stopband, and note that it is actually rounded downward (second order zero) at z=-1.



This modified length-13 looks like a generally good idea. It does not require a phase shift of half a delay, and is a better low-pass. It is just slightly longer. For a job like we specified (reject the seasonal variations but pass the low-frequency) it can't be beaten. The remaining consideration would be what the filter might do to other components (other than 0 frequency and 1/12 frequency), if they are an issue.

Oh, we do need to discuss the transients!

END EFFECTS

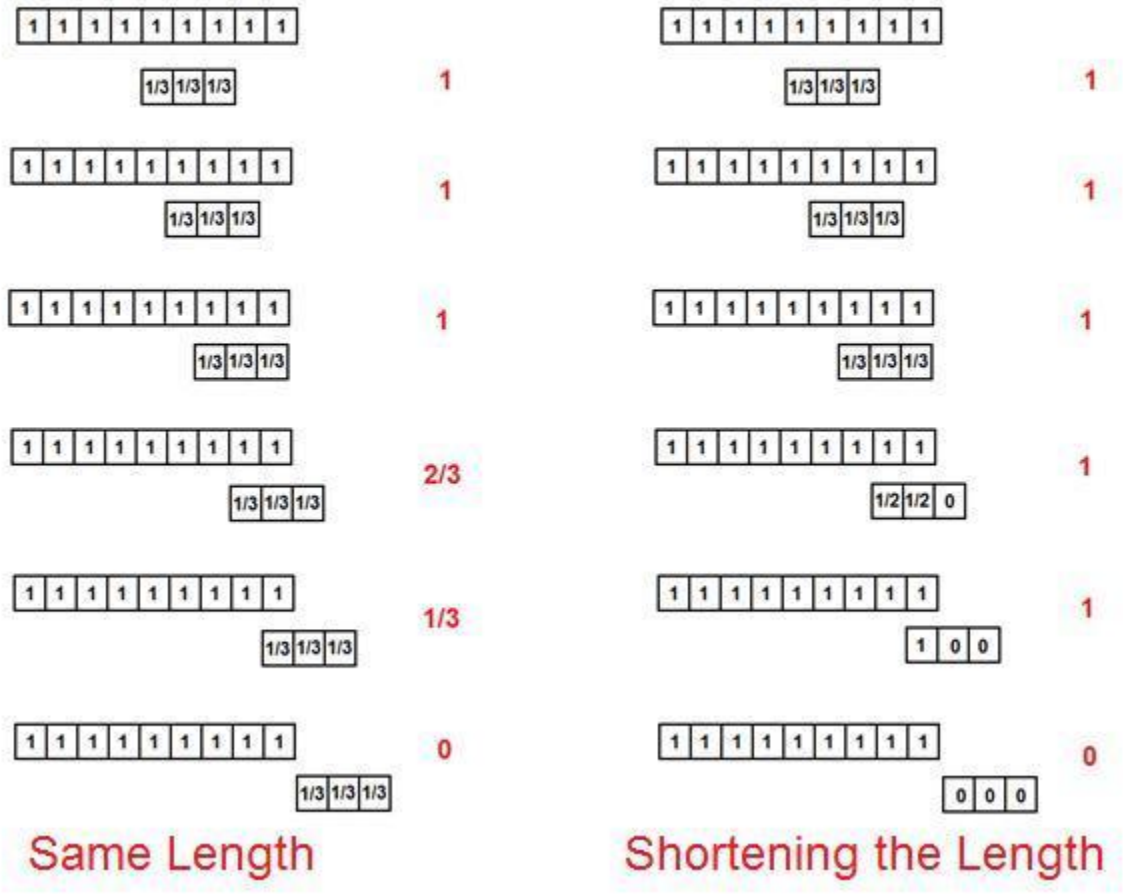
Here we get to talking about a major issue – how do we deal with the ends. Possibly the only truly rational way is to just ignore the ends in the sense that for the points of the convolution where we lack sufficient points to match the FIR filter length, we disregard the ends as having NO meaning. Because this might be discarding a substantial portion of the output (as in Fig. 2) and because it may be ends that are of most interest, people are tempted not to take this first option. Oftentimes something called "padding" is used where data that we believe is a good approximation to what we are missing is attached to the ends. Some notion of what we might be looking for is in Fig. 2 where we had more data but chose to ignore it. It is probably the rare case where we do really have access to additional

real data points (as we have in Fig. 2). Instead we want something that we believe is a good approximation, or at least misleads the least amount. But make no mistake: nothing we substitute, no matter how sophisticated our filter design, is other than a guess.

So instead, let's consider something simpler first [11]. In some cases, we chose a length for a moving average (or other FIR filter) with a specific goal such as the rejection of a particular frequency while generally smoothing out noise (section just above). In other cases, we choose a smoothing length with no goal other than noise rejection. In such a case we might have chosen length 40, but would the result have been significantly different if we had chosen length 47 or length 38 perhaps? If the length is so long that it is limiting our access to the ends, why not change it. Changing from length 40 to length 30 would give us 10 extra valid outputs on each end. This is obvious.

If one really likes the smoothing of the length 40, what if we considered changing the length only for the ends, decreasing by one sample at a time as we approach the end. Fig. 15 shows a simple length-3 implementation of this idea.

Fig. 15



The left side of Fig. 15 shows the usual implementation of a length-3 moving average, as applied to the tailing end of a sequence of all 1's. Indeed the average drops from 1 to 2/3, then to 1/3, and then to 0. IF we had reason to believe that the sequence continues beyond the end, we might have padded the end with extra 1's. Otherwise that nasty end transient down slope is necessary. Using the notion of changing the length (right side) we can hold the correct average. Note that in a general case the sequence of 1's would not be exactly 1's but would be noisy. There would be more noise in the shorter lengths. But this understood, this looks like a better choice.

The reader may well complain that when we change the length, we change the frequency response of the averager. Surprisingly, this is not true – we DESTROY the very idea of a frequency response – the system is no longer LTI (Linear Time Invariant) and the frequency response is not defined. We might still be justified in describing how the data is manipulated, to some advantage, in the time domain. Of course, if we were only shortening the length slightly, we might find this change of length totally acceptable, but the general case is probably that we want the length to go all the way to zero to get to the exact end of the data. Related to this point is the fact that we can change the length of the moving average, overall, fairly easy, and this may be useful [7,8].

As we have said above, the notion of padding is evident from looking at the ignored data of Fig. 2. Let's confess how these data were generated.

```
n=-30:130;
y=0.5 + n/200 + 0.1*cos(2*pi*n/40)+0.15*rand(1,161);
```

We don't know this in general, and while we might have exactly predictable code for the first three terms, we are completely (generally) unable to know in advance what the rand function will return. So let's restrict ourselves to the first three.

In general, if we are asked to guess the next value of a sequence, we are advised, in the absence of any other information, to choose 0. This is just truncation – no padding at all. Perhaps the next best guess is the last sample value at the end (a zero-order hold), and you probably won't get much argument if you prefer the last output (the last moving average result) to pad.

Another choices could be to mirror the end data. That is, if the data ends with x(N) you make x(N+1)=x(N-1), and in general x(N+n) = X(N-n). This might work well if the last data point actually happened to occur at an extreme value that was reversing. Yet another popular padding is to "reflect" the data through the last point. If the last point is again X(N), you make X(N+1) = X(N) +[X(N) – X(N-1)] = 2x(N)-x(N-1), and in general X(N+n)= 2X(N)-x(N-n). For an example, suppose your data points were 1 2 3 3 4 5 3 4 5 6 7. With a 5 point reflection, the data becomes 1 2 3 3 4 5 3 6 5 6 7 8 9 8 11 9. The green points are reflected here. This should work well with a linear trend.

It is impossible to defend the decision to make up data and then pretend that it means anything. Making up well defined data is a wonderfully useful way to test some sort of processing. But there is no magic: none offered here – none available.

SUMMARY

We have noted that time-series smoothing is basically an FIR low-pass, probably most often intended to remove noise by averaging it out. In that sense, we might contend that any old low-pass would do. However, most such low-pass filters have stopband zeros, and it is not uncommon to place zeros to strongly reject a frequency already in the stopband, so the general low-pass itself may be secondary to the exact nulling of some specific frequency or frequencies.

In all cases it is useful to consider the smoother as it behaves in the frequency domain as well as the time domain. But because it will frequently be the case that time series are relatively short portions of data (not like, say, continuous music) and our FIR filters need some minimal length, much of the output may well be "transient" and inspire various measures of dealing with bogus data on the ends. None of these schemes is a true fix, and care in their use is required. Ad hock tweaks and special pleadings should make us uncomfortable. Unprocessed data has a certain fundamental honesty.

Because the time series is by definition modified by any filter, we need to be on the alert for any spurious generation of features that are the result of the processing. Whenever we see something come out of the smoothing filter that looks new or "dug out" we must be very careful to be sure that it is not a joke the filter is playing on us. An extensive examination of the filter with a wide range of artificial data is suggested.

There are more items that we can also discuss here, and these we include in the several Appendices below:

REFERENCES FOR MAIN TEXT

All references to Electronotes Publications

[1] "Models – Good, And Bad: And The (Mis-)Use Of Engineering Ideas In Them," ***Electronotes,*** Volume 22, Number 211 July 2012
http://electronotes.netfirms.com/EN211.pdf

[2] "Polynomial Fitting for Sample-rate Changing at Rational and Irrational Frequency Ratios," Electronotes Application Note No. 317, January 1992**;**
http://electronotes.netfirms.com/AN317.PDF
"Interpolation, Decimation, and Prediction of Digital Signals," ***Electronotes***, Vol. 15, No. 164-167 (Special Issue F) July 1986.

[3] “Basic Elements of Digital Signal Processing: Filter Elements – Part 1” ***Electronotes***, Vol. 20, No. 197, April 2001
http://electronotes.netfirms.com/EN197.pdf

[4] “Basic Elements of Digital Signal Processing: Filter Elements – Part 2” ***Electronotes***, Vol. 20, No. 198, June 2001
http://electronotes.netfirms.com/EN198.pdf

[5] “Basic Elements of Digital Signal Processing: Filter Elements – Part 3” ***Electronotes***, Vol. 20, No. 199, Sept. 2001
http://electronotes.netfirms.com/EN199.pdf

[6] “Inverting a Moving Average Operation,” Electronotes Application Note No. 407 March 30, 2014
http://electronotes.netfirms.com/AN407.pdf

[7] “Examples Of Inverting Of Smoothers,” Electronotes Application Note No. 408 April 5, 2014
http://electronotes.netfirms.com/AN408.pdf

[8] “Changing The Moving Average Length – A Recipe,” Electronotes Application Note No. 409, April 6, 2014
http://electronotes.netfirms.com/AN409.pdf

[9] “Spurious Correlations Due To Filtering (Of Noise).” Electronotes Application Note No. 403, Jan 27, 2014
http://electronotes.netfirms.com/AN403.pdf

[10] “Yearly Moving Averages As FIR Filters,” Electronotes Application Note No. 401 Dec 22, 2013
http://electronotes.netfirms.com/AN401.pdf

[11] “Averaging - And Endpoint Garbage,” Electronotes Application Note No. 395, March 30, 2013
http://electronotes.netfirms.com/AN395.pdf

APPENDIX A – DESIGNS CONVERGE

This is not really a surprise. When we compare two different digital filter design procedures and manipulate the specifications so that they approach each other in one domain (usually the frequency domain) they converge in the time domain as well. After all, the DTFT tells us:

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h(n)e^{-jn\omega}$$

which means that there is a unique pairing of some $H(e^{j\omega})$ and a corresponding $h(n)$. That is, the better we define $H(e^{j\omega})$, refining the specifications, the more the method is obliged to give a particular result. Here we will give two examples.

The familiar moving average is a trivial "design" in that the impulse response is just a rectangular sequence and the frequency response is a periodic sync. It is carved in stone. Fig. A1 shows a length 15 response [all h(n) are 1/15=0.0666…]. Naturally we might suppose that if we removed the restriction of all the h(n) being the same, we could do better. Of course we can, but what happens if we retain the restriction on the length (keeping it at 15) and insist on a similar sharpness of the frequency response. Here we will use as the second design method the least integrated square error (Matlab ***firls***) and more or less by trial and error try to get a first sharp lobe. Thus we follow the pink circular path, with the two upper panels being automatic, and then the specifications of the ***firls*** design are adjusted to get a good agreement in the two frequency responses (two right panels). Finally we take a glance at the ***firls*** impulse response. The lower two panels of Fig. A1 show the result. What a disappointment! There is barely any improvement in the frequency response, and the impulse response looks as though it wasn't trying very hard! It just curls down a bit. However, the ***firls*** filter is doing the best it can, and doing exactly what we asked it to do – establish a sharp cutoff with short length. It is just that the best may not have been anywhere near as good as we were hoping for or expecting. This is a very important observation.

Fig. A2 is a second example that is perhaps equally astounding. Here we start with an IIR filter, an 8th-order Butterworth in fact (Matlab's ***butter***), and will end up comparing this to a FIR filter designed by ***firls***. Again we will obtain a convergence of the designs for the same length and similar cutoff properties. The full code is listed as ***buttervsfirls.m***. Of course, butter gives us a non-symmetric IIR, but the result can be made into a reasonable FIR design method, as follows:

We start with ***[b,a]=butter(8,2*0.125)*** which gives us the nine numerator coefficients (b) and the nine denominator coefficients (a) of a 8th-order Butterworth low-pass. [Note that Matlab's filter design functions assume a sampling frequency of 2 (!) so it is the author's habit to specify the usual desired response on the range 0 to 0.5 and double it.] The IIR filter coefficients are:

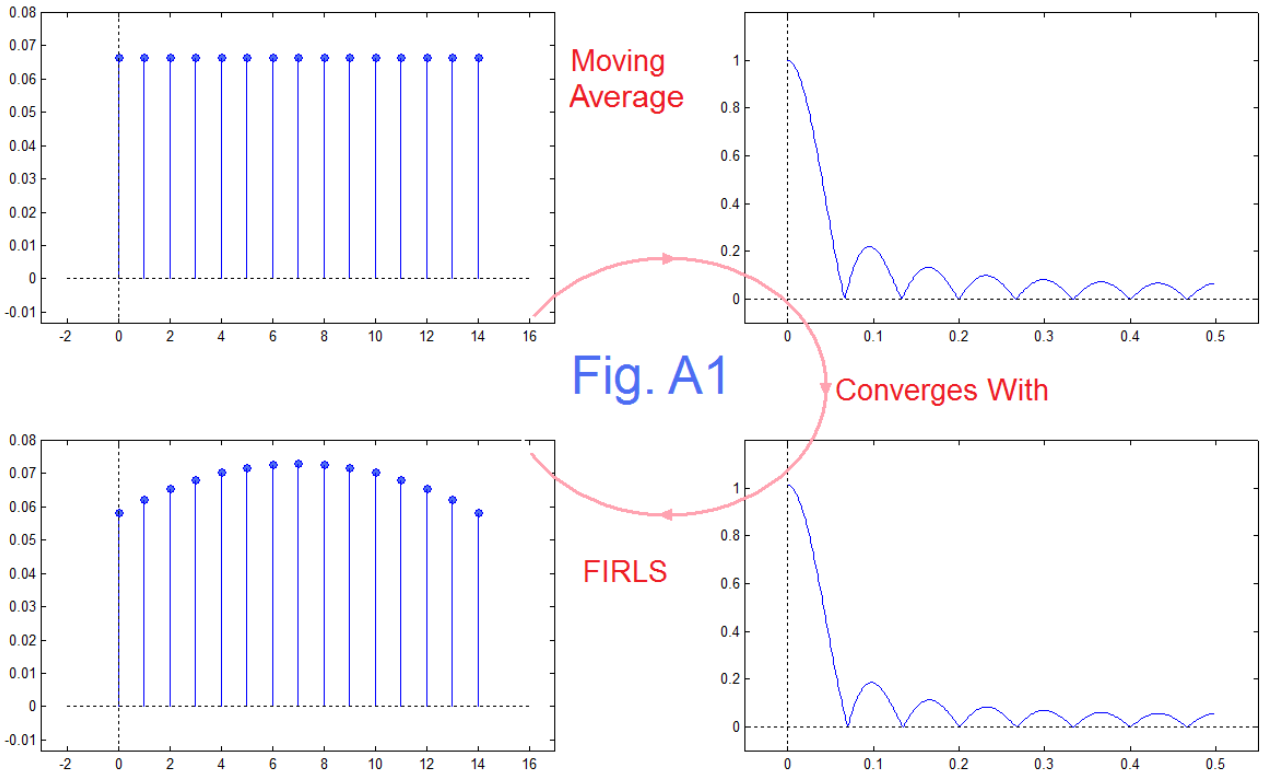

Moving
Average
Converges With
FIRLS

Fig. A1

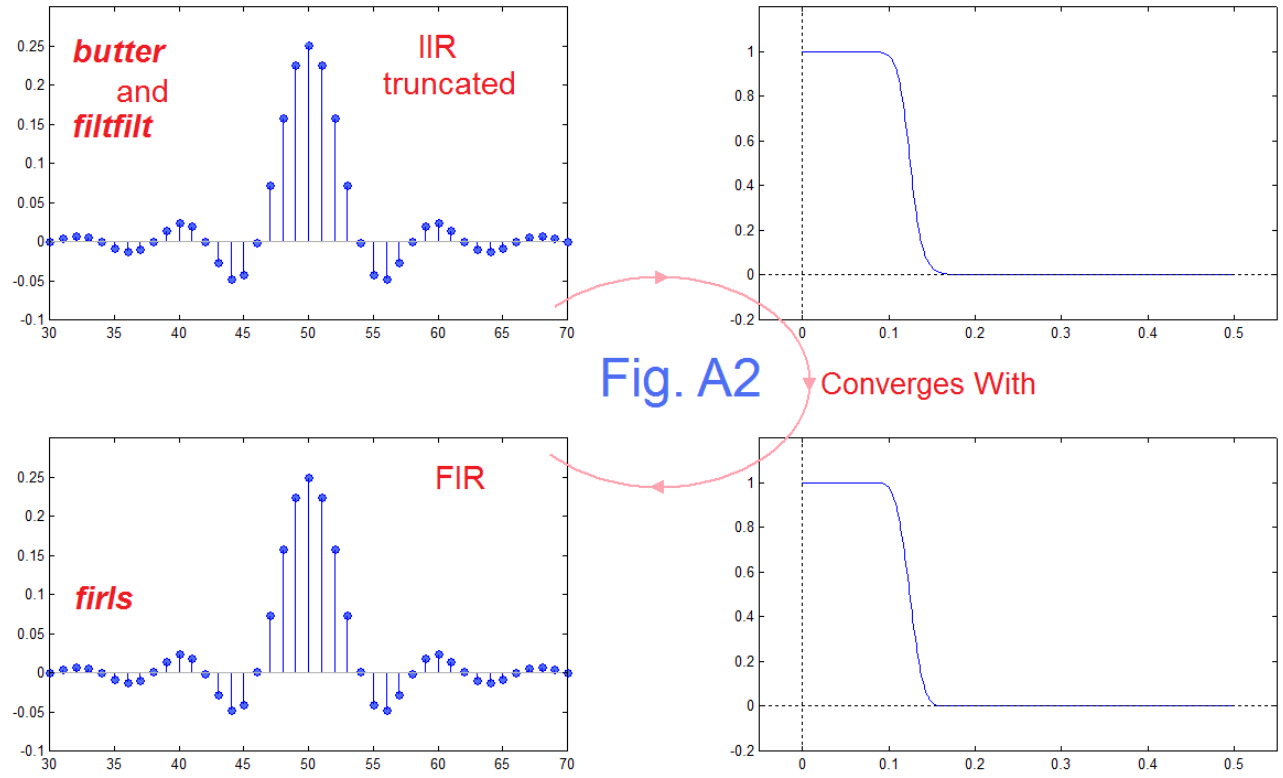

butter
and
filtfilt
IIR
truncated
Converges With
FIR
firls

Fig. A2

b = 0.0001 0.0009 0.0030 0.0060 0.0076 0.0060 0.0030 0.0009 0.0001

a = 1.0000 -3.9838 7.5362 -8.5998 6.4002 -3.1560 1.0017 -0.1863 0.0155

Perhaps more insight as to what ***butter*** does is found by computing:

b/b(1)= 1 8 28 56 70 56 28 8 1

which shows the familiar binomial coefficients for a numerator that results from using the Bilinear-z-Transform IIR design to an analog prototype. Unlike with an FIR filter, these coefficients here are not the impulse response (which is infinite duration starting at n=0 by the way, although it has a limited range of significant values). We could use Matlab's ***filter*** applied to a unit pulse such as [1 0 0 0 0 0 0 ….] to see the IIR impulse response. Instead we use Matlab's ***filtfilt*** (with a unit pulse in some middle region) which first filters and then filters the output a second time with time reversed. This effectively squares the magnitude response and makes the impulse response symmetric. We then truncate the resulting impulse response on both sides, and this is reasonable since the response falls off rapidly. In this sense, the use of ***butter*** followed by ***filtfilt*** applied to an impulse in the middle region, followed by truncation, can be considered to be an FIR design method. This is how we got the filter in the top portion of Fig. A2. We have truncated it at 30 and 70 to avoid clutter. Note that the impulse response rolls-off rapidly on both sides.

We would suspect that the FIR obtained in this manner is not optimal although the IIR Butterworth is optimal in a known way. Could we do better, again keeping the length the same, and insisting on a similar cutoff and stopband? Again, more or less trial-and-error we use ***firls***, and we find ourselves in the same sort of corner as with the moving average. Comparison of the two panels at the top to the bottom panels of Fig. A2 is convincing that the designs again converge. If you insist, the ***firls*** design has an ever so slightly better corner to my eye.

Once again, the methods, radically different in this case, converge.

The code for Fig. A2 is shown below:

```
[b,a]=butter(8,2*.125)
hff=filtfilt(b,a,[zeros(1,50),1,zeros(1,49)]);
h=firls(100,2*[0 .075 .175 0.5],[1 1 0 0]);
figure(1)
subplot(221)
stem([0:99],hff)
axis([30 70 -0.1 .3])
```

```
subplot(222)
plot([0:.001:.499],abs(freqz(hff,1,500)))
hold on
plot([0 0],[-1 2],'k:')
plot([-.1 .6],[0 0],'k:')
hold off
axis([-.05 .55 -.2 1.2])

subplot(223)
stem([0:100],h)
axis([30 70 -.1 .3])

subplot(224)
plot([0:.001:.499],abs(freqz(h,1,500)))
hold on
plot([0 0],[-1 2],'k:')
plot([-.1 .6],[0 0],'k:')
hold off
axis([-.05 .55 -.2 1.2])
```

APPENDIX B ALTERNATIVES TO MOVING AVERAGE

B1 Sharpening

A method of starting with one FIR filter and then using it twice of three times was an idea of Kaiser, Hamming, and likely Tukey as well. It should be better known. Originally thought of as contributing to computational efficiency (like with hardware) it is now seen as a clever general method. The method starts with a FIR core filter. For example, it might be a length-3 moving average: $h_1(n)$ = [1/3 1/3 1/3]. We know that by cascading this with itself, we get a longer filter with weights $h_2(n)$ = [1/9 2/9 3/9 2/9 1/9] which has a better stopband but a less good passband (for general filters at least). The method here, as discussed in the references below, involves putting a "pre-processing" stage which also involves $h_1(n)$ in front of the cascade, as in Fig. B1. Fig. B1 shows as well a specific example where $h_1(n)$ is in fact [1/3 1/3 1/3]. The overall filter then becomes $H(z) = [3z^{-1} – 2H(z)]H^2(z)$.

We have sketched out, step-by-step, the development of the impulse response. Note that the final cascade increases the length from 3 to 5, and the pre-processor on the left makes it length 7. In the frequency responses, we see the original length-3 moving average (black) and as expected, squaring the moving average improves the stopband,

green, (the stopband zeros become 2nd order and round at the zero - frequency 1/3). We see that this sharpening trick works well (red) in giving a broader passband and still a degree of flattening in the stopband. Keep in mind that the red filter is length 7 – it was not free.

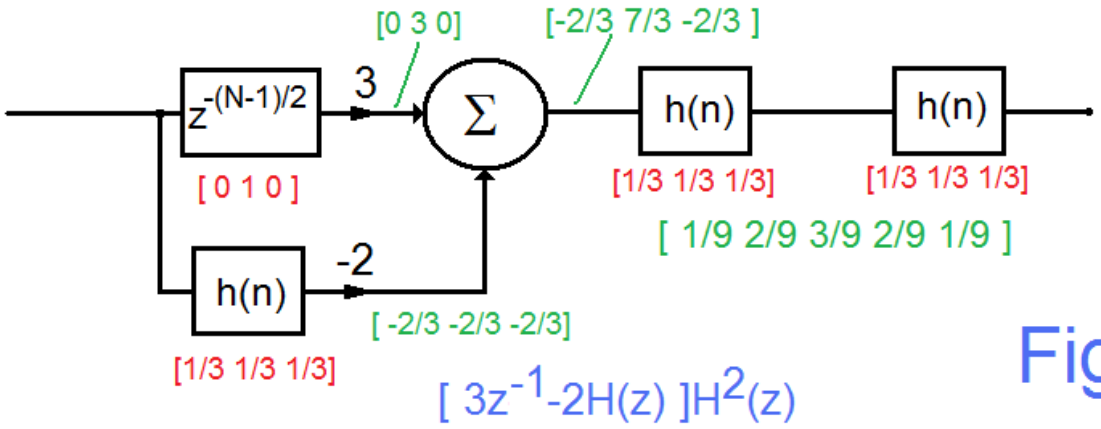
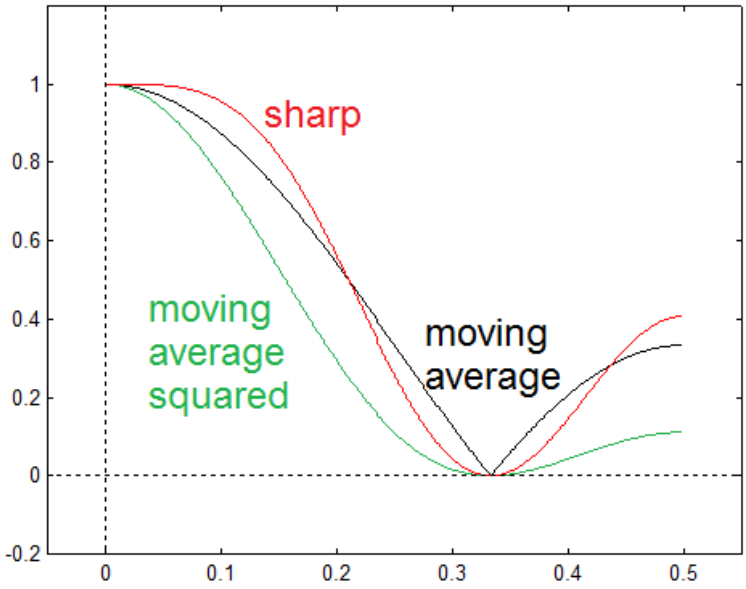


Fig. B1

$$[\ 3z^{-1}-2H(z)\ ]H^2(z)$$

$$[\ -2/27\ \ 1/9\ \ 2/9\ \ 13/27\ \ 2/9\ \ 1/9\ \ -2/27]$$



REFERENCES FOR B1

J. Kaiser and R. Hamming, "Sharpening the response of a symmetric non-recursive filter by multiple use of the same filter," ***IEEE Trans. Acoust. Speech & Sig. Proc***., Vol. ASSP-25, No. 5, (1977) pp 415-422; R.W. Hamming, ***Digital Filters***. 3rd Ed., Dover (1989), "New Filters from Old Ones: Sharpening a Filter" Section 6.6, pp 140-146 (very good explanation); Donadio, M., "Lost Knowledge Refound: Sharpened FIR Filters," ***IEEE Sig. Process. Mag.***, Vol. 20, No 5, Sept 2003; Richard Lyons, ***Understanding Digital Signal Processing***, 3rd Ed., Prentice-Hall (2011) "Sharpening FIR Filters" Section 13.13 pp 726-728; and this online: http://homedir.jct.ac.il/~shlomoe/Public/filt_sharp.pdf

B2 Triple Running Mean – Sort of Gaussian

The climate science world has a lot of use for time-series smoothing, often of temperature data. Principally this is for the removal of "noise" and sometimes for rejection of a particular periodic component (such as 12 month periodicity due to seasons). Climate science involves (or should) a wide range of technical specialties, of which signal processing is one.

For the example here, one method of dealing with artifacts of a moving average (sometimes called a "running mean") is the "Triple Running Mean" (TRM). This is a method typical of those who notice problems and invent solutions. Such solutions typically do "work" although it may not be optimal or even be necessary in all cases.

One readily-available presentation of the idea here is:

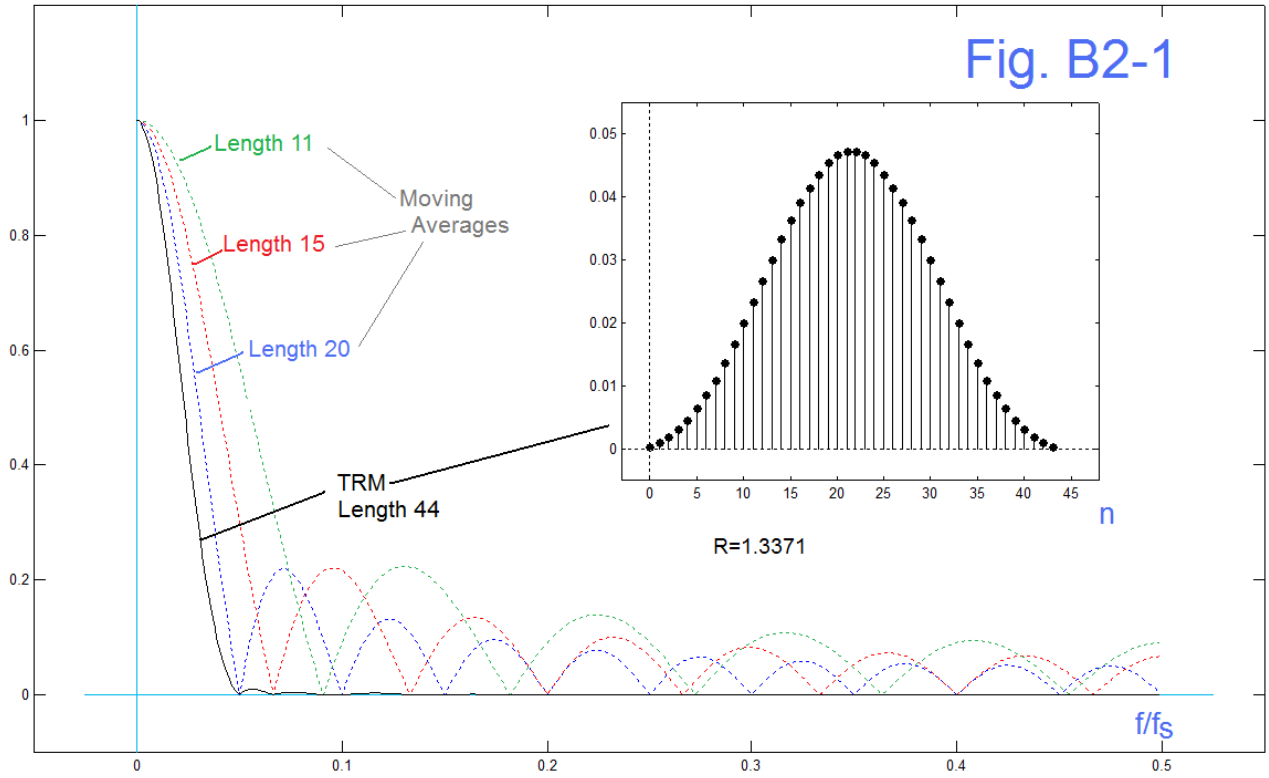
http://climategrog.wordpress.com/2013/05/19/triple-running-mean-filters/

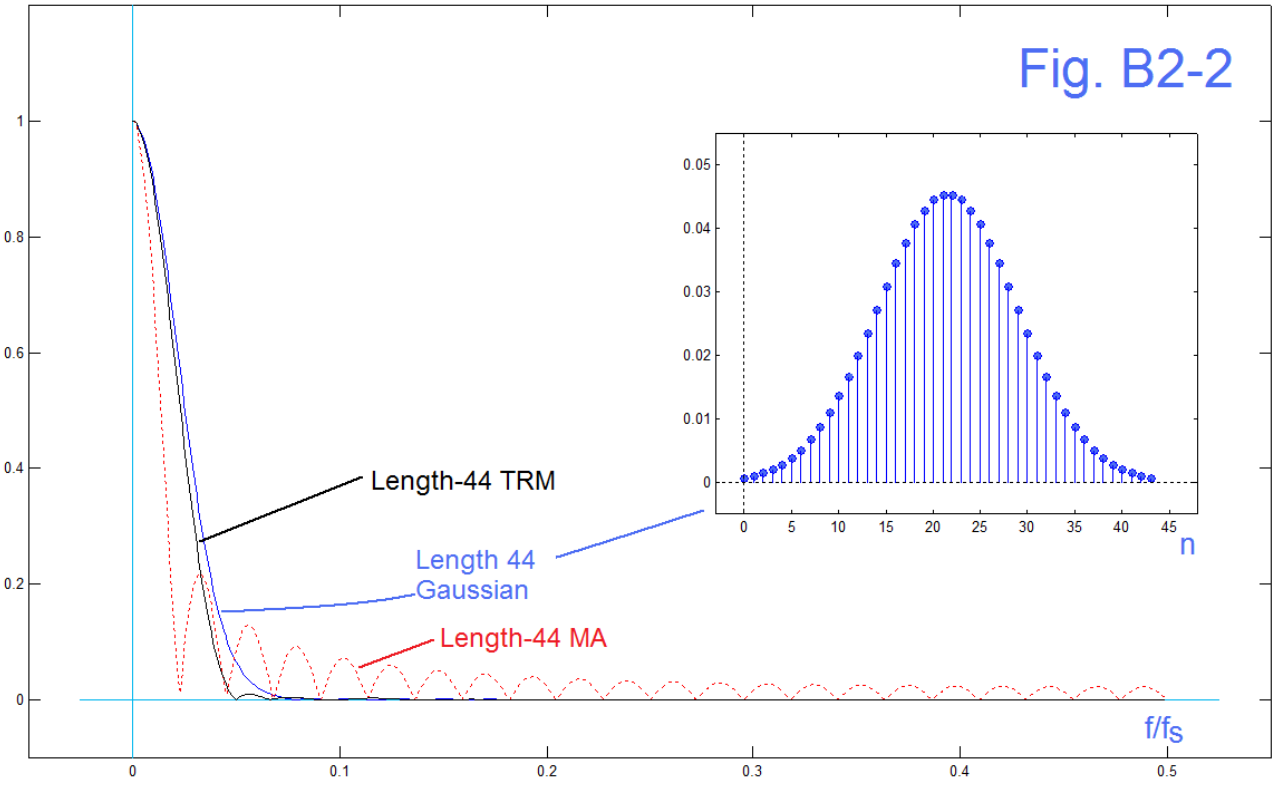
Here Greg Goodman (and others) have argued, correctly, that the frequency response of the running mean (moving average) has significant stopband sidelobes, and further that they alternate in sign. In a stopband, any input components are attenuated, possibly inverted, and at isolated points, completely cancelled. This makes a difference IF there is a significant signal component at a non-zero stopband response frequency of interest. For example, for the green curve (length 10) of Fig. 3 if we have a low frequency of 0.1 which we want to reject, a frequency of 0.15 which we want to keep, and a flat broadband noise, then we see that the moving-average is helpful, but it will invert the component at 0.15. Keep in mind that frequency response plots are generally magnitude and successive lobes alternate in sign.

In the TRM scheme, the data is run through a first (longest) running mean, and we know exactly what this does. Then the original length is divided by some number R>1 (and rounded) for a shorter length to be used in a second pass, and the output of the first running mean is put through this second filter. The data is then run through yet a third filter, where again the length is further divided by (the same) R. The value of R is variously said (without much derivation) to be 1.4303, 1.3371, or 1.2067. Figure B2 shows an example where the lengths are 20, 15, and 11 (using R=1.3371). Note that the length of the overall impulse response of course keeps getting longer. More importantly, the shape of the impulse response becomes tapered. A double running mean would be trapezoidal, and the TRM looks roughly Gaussian. We see from Fig. B2-1 that in the stopband, sidelobes are relatively small. This was the point.

The result here for the TRM needs to be compared to (1) a true Gaussian and (2) to the immense range of other possible filters. Thus here we have taken the first steps to attempt

to have a comparable Gaussian, as seen in Fig. B2-2 The second question involves the vast number of alternatives above and below and is not addressed directly, except to note that the alternative approaches are somewhat better developed and defined.



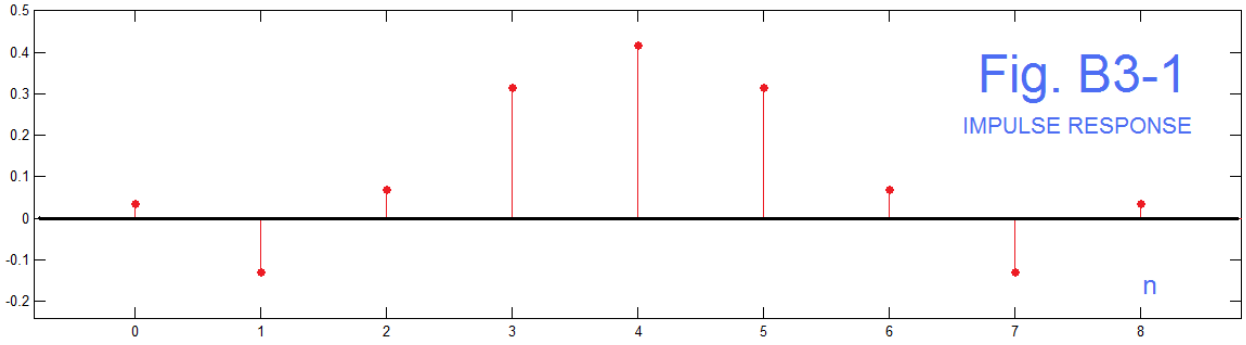


In Fig. B2-2 we have attempted to fit a true Gaussian (here the standard deviation was about 55). The true Gaussian in this case, in fact, does not look as good as the TRM. We show for comparison here (which we can also compare to Fig. B2-1, the length 44 MA. Note that with regard to "sharpness" both the Gaussian and the TRM are less sharp than the MA, as we would expect since tapering effectively narrows the time response and widens the frequency response. It can also be pointed out that with the TRM, as with the Kaiser/Hamming sharpening, one attractive point related to hardware efficiency (MS is easy) but this is probably not an issue with smoothing data (just a few times) for presentations and that sort of applications.

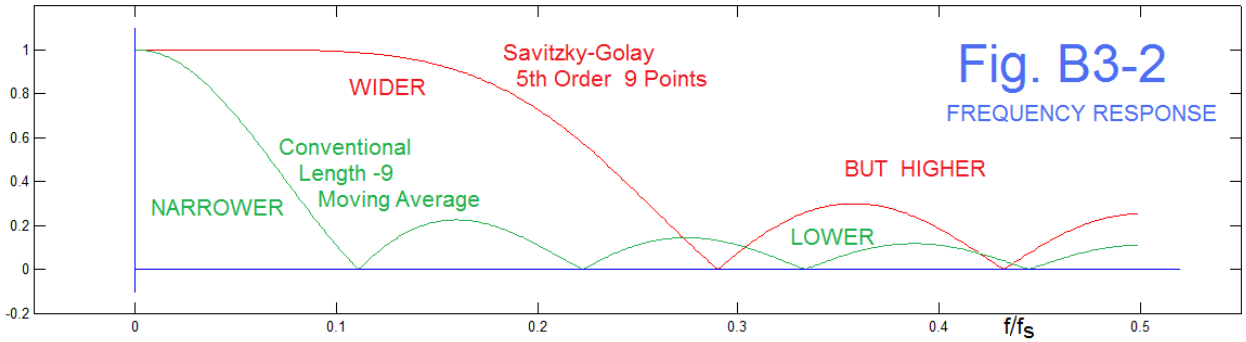
B3 Savitzky-Golay

Another popular smoother is the Savitzky-Golay low-pass which we have looked at in three places:

"Savitzky-Golay Smoothing," Electronotes Application Note No. 404, Feb 13, 2014
http://electronotes.netfirms.com/AN404.pdf
"Examples of Inverting of Smoothers," Electronotes Application Note No. 408, April 5, 2014
http://electronotes.netfirms.com/AN408.pdf
"Savitzky-Golay = Maximally Flat?", Electronotes Webnote 18, 2/15/2014
http://electronotes.netfirms.com/ENWN18.pdf



Savitzky-Golay



There is extensive coverage of this design option in AN-404, so here we will principally be just offering Fig. B3 as a reminder of what it does. Further it is popular (apparently) as part of several statistical software packages, so is deserving of consideration. Note that the main interest is the low-pass passband which is well defined (and flat), so is very useful where we are interested in more than just a "DC response". See also Appendix B5 below. This is just least-squared polynomial fitting.

B4 – Lanczos Filtering

Cornelius Lanczos 1893-1974 was a pretty good physicist who also has his name attached to two important topics is signal processing, the Lanczos Inverse and the Lanczos Filter, and is one of the names that come up when one traces the origins of the FFT back before Cooley/Tukey (even to Gauss).

The filter is quite simple, being the product of two sinc functions, one of which is time limited (and thus time limits the other in the multiplication)

$$h_L(x) = sinc(x)sinc\left(\frac{x}{a}\right)$$

where x is limited to the range –a to +a. We are very much accustomed to dealing with a time-limited sinc function where the "window" is a rectangular function, or perhaps a Hamming window or one if its relative. Here the window is a sinc lobe, and we consider it to be whichever sinc of the product is wider. Here we have not looked closely at this, and our purpose is to keep the correct perspective (a sinc-windowed sinc). Further, somewhat like the case of the Sovitzky-Golay we need to pay attention to this as a part of many statistical software packages. Worthy of more study. For example, here:

http://journals.ametsoc.org/doi/pdf/10.1175/1520-0450%281979%29018%3C1016%3ALFIOAT%3E2.0.CO%3B2

B5 - LOESS and LOWESS Smoothing

As noted in various places above, we can become confused in terminology, particularly as the same things are called by different terms in signal processing on the one hand and in statistical modeling on the other. The terms LOESS and LOWESS are examples of "interference in learning" (not unlike the case of DTFT and DFT!). The terms refer to "Local" (the LO) and to "Scatterplot Smoothing" (the SS) with something like "Weight" in between. A "scatterplot" is really just a plot – usually a noisy (measurements with errors) plot seeking a correlation or "regression" between two variables.

All this is more statistics than signal processing. It seem to this author that we are involved here with looking for a least-squares fit of a regression line to data, and that "local" means that we propose a fit to a specific range of data points, weighted in a "span" about some center; and finding this, move on to another range. It looks very much like the polynomial fitting and least-squares models we are very familiar with. I could be wrong in the statistical terminology and intent, but the signal processing is nonetheless easy to follow from out viewpoint. We have been doing this for years.

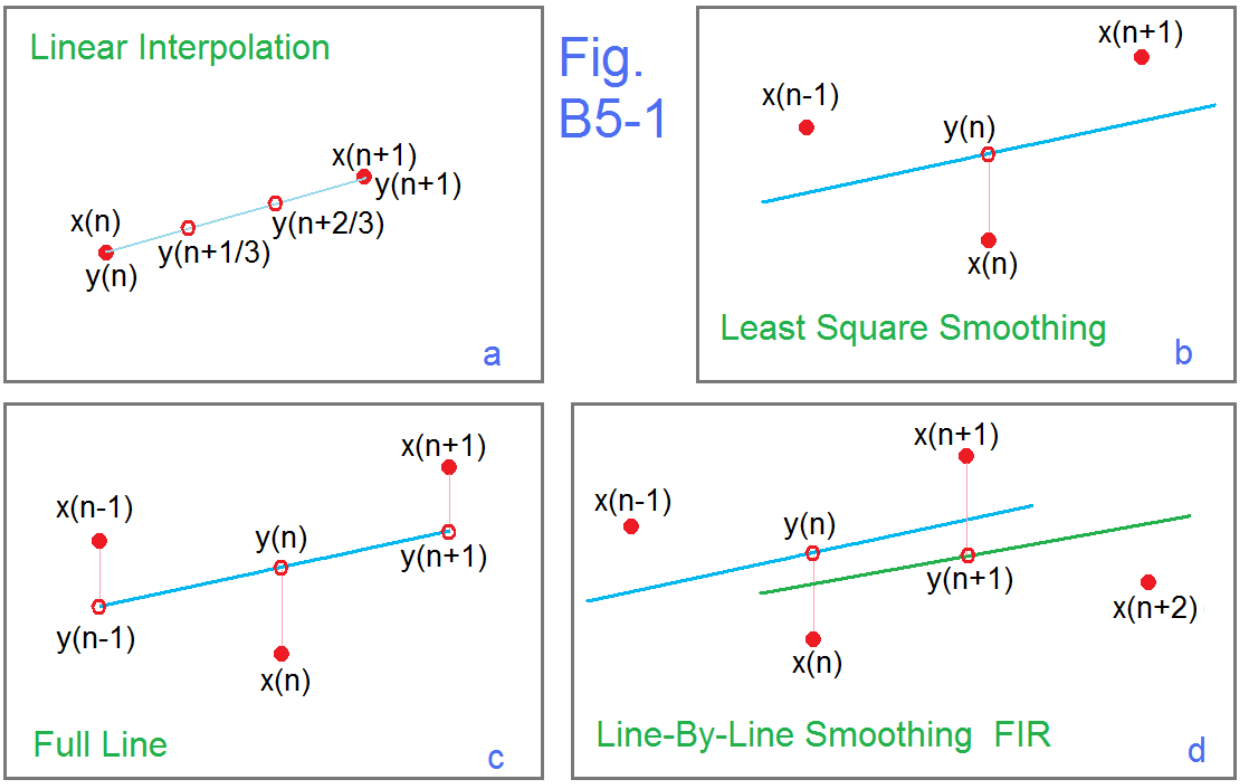


Fig. B5-1 shows the necessary concept, illustrated here for the familiar straight-line fit to data points. In signal processing, much higher order polynomials are often used. [One source says LOWESS is for first-order, LOESS is quadratic (2nd-Order). So perhaps the statistical models stop there.] The ideas here apply to interpolation and to smoothing. The outstanding finding about these applications is that the implementation (astonishingly!) is just by rather ordinary FIR filters. No "on-the-fly" curve fitting is required. See for example: Section 4b of "Basic Elements of Digital Signal Processing: Filter Elements – Part 2" ***Electronotes***, Vol. 20, No. 198, June 2001 http://electronotes.netfirms.com/EN198.pdf

Fig. B5-1a shows linear interpolation. This is the familiar case where a straight line is exactly fit to consecutive points, and points in between are then selected (1/3 and 2/3 of the way across in the drawing). This is a well-known FIR procedure and thus represents a choice for filter design.

The remaining three panels of Fig. B5-1 show a fitting of straight lines to three points – thus a least square fitting. We could have had other fits: perhaps a straight line to 7 points or a cubic to 17 points, order 7 to 31 points, etc. All of these procedures give FIR filters in the case (such as Fig. B5-1d) where the fit is done for each iteration. An old app note illustrated the procedure in just two pages: "Time domain Least Squared Low-Pass Filters," Electronotes App. Note No. 318, Feb. 1992: http://electronotes.netfirms.com/AN318.PDF Much to our surprise originally, this example resulted in a length-3 MA.

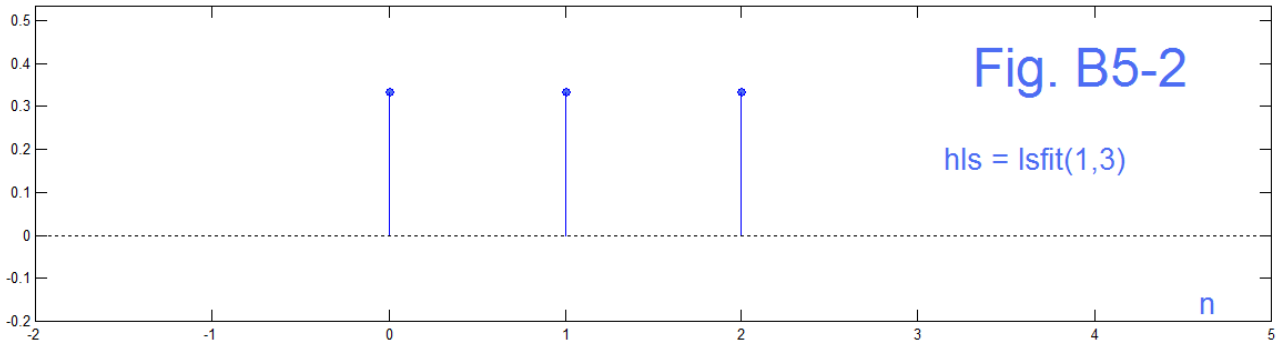
Fig. B5-1b shows the procedure of fitting a straight line to three points. Unless the three points are already on a straight line, the best least squares fit won't go through any of the points. The line will in fact always be parallel to a line between the endpoints, and situated so that the middle point is on the opposite side of the line from the endpoints. The line is closer to the endpoints than to the middle. The thing to note here is that we suppose that the line is "better" in some sense than the data. Thus the point on the line (as found directly above or below the midpoint) is a superior choice for the midpoint. The open circle is the output of the process.

We envision the procedure of fitting the line not as a one-time calculation for all the data points we have, but rather an ongoing situation for local data points. (Cases, where we are only looking at one overall data set, are not at all uncommon.) When it comes to moving subsets of a running data set, one choice (probably not a good one) would be to replace the endpoints as well as the midpoint with values from the least square line. This (Fig. B5-1c) would seem acceptable if the calculation were one-time using all the data. But here we are talking more about a "running" smoothing operation where we always use data points about some center, output a center point replacement, and then move to a new line (Fig. B5-1d – replacing the blue line with the green one, etc.). If this did not lead to an FIR implementation, we would perhaps be less enthusiastic.

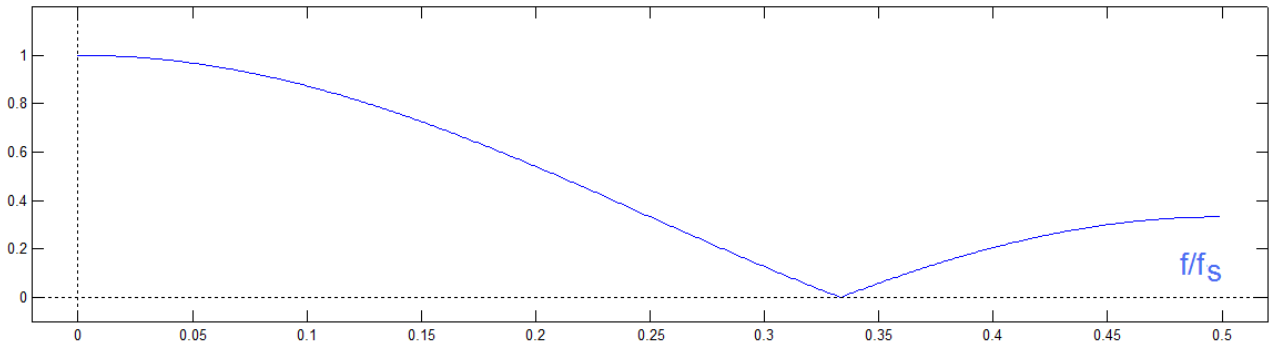
How easy is this to do? Often times when we seek a filter design we are asking for an impulse response, and if we ask for the impulse response of a least-squares curve fitter, the seemingly "wise-guy" answer is that it is the response of a least-squares curve fitter to an impulse. So if we have a function that computes the least squares fit (Matlab's ***polyfit***), and one that plots the resulting polynomial (Matlab's ***polyval*** – used at one point in this case), we have enough. A simple function ***hls = lsfit(order,numberpoints)*** is printed below, and the code in green is the core – the rest being display.

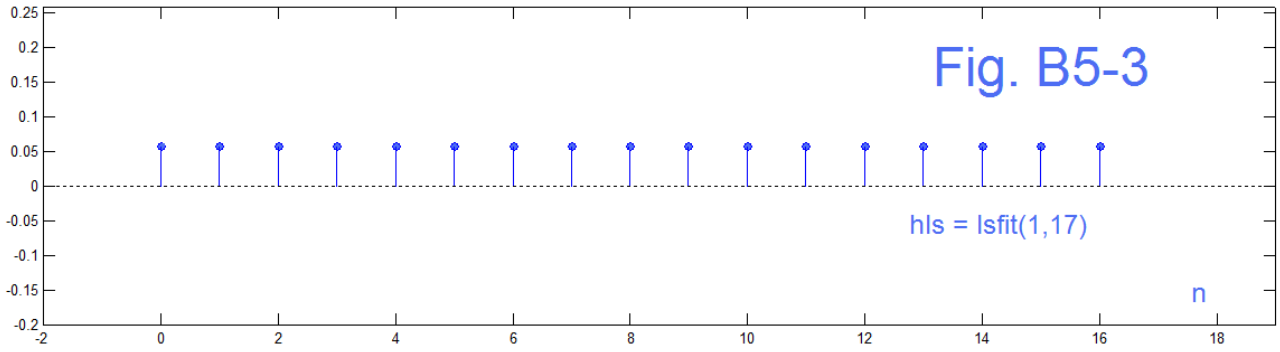
When we run the code for order 1 and length 3, we indeed get a length-3 MA (Fig. B5-2) as was done by hand in AN-318. Making the length 17, we get a length-17 MA as in Fig. B5-3. Making the order 2 and keeping 17 points, gives the curved impulse response of Fig. B5-4. Perhaps this is where LOWESS changed to LOESS and we need to stop. But the program ***lsfit*** can keep going. What if we try ***hls = lsfit(7,26)***. The result is seen in Fig. B5-5. Interesting, it resembles Sovitzky-Golay. In fact, it is identical! Running ***hsg=sg(7,26)*** from AN-404 in fact gives Fig. B5-6. The outputs hls and hsg are identical.

```
function h=lsfit(M,N)
% M = order
% N = number of points
h=[]
for n=1:N
   x=zeros(1,N);
   x(n)=1;
   m=-(N-1)/2:(N-1)/2;
   P=polyfit(m,x,M);
   h(n)=polyval(P,0);
end
%
figure(3)
subplot(211)
stem([0:N-1],h)
mih= min(h);
if min(h)>0; mih=0;end
mih=mih-.1;
mxh= max(h)+.1;
axis([-5 N+5 mih mxh])
%
subplot(212)
plot([0:.001:.499],abs(freqz(h,1,500)))
hold on
plot([-.1 .6],[0 0],'k:')
plot([0 0],[-.2 1.5],'k:')
hold off
axis([-.05 .55 -.2 1.2])
```

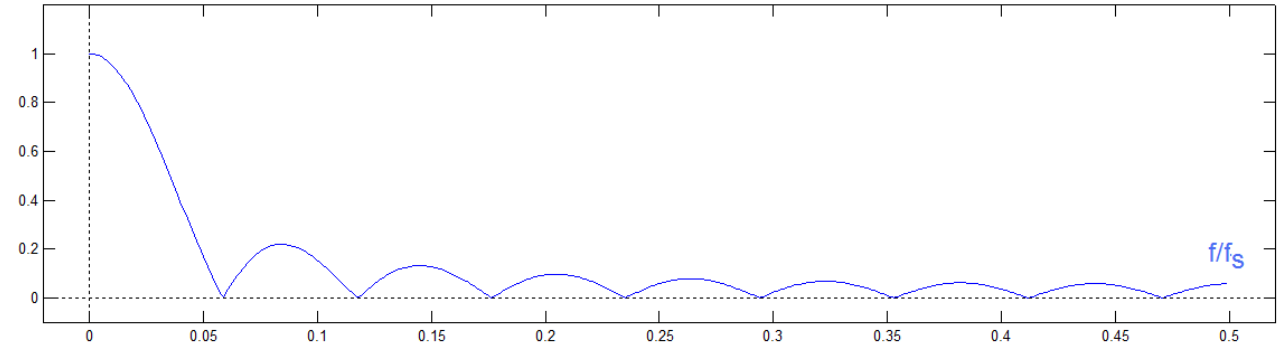


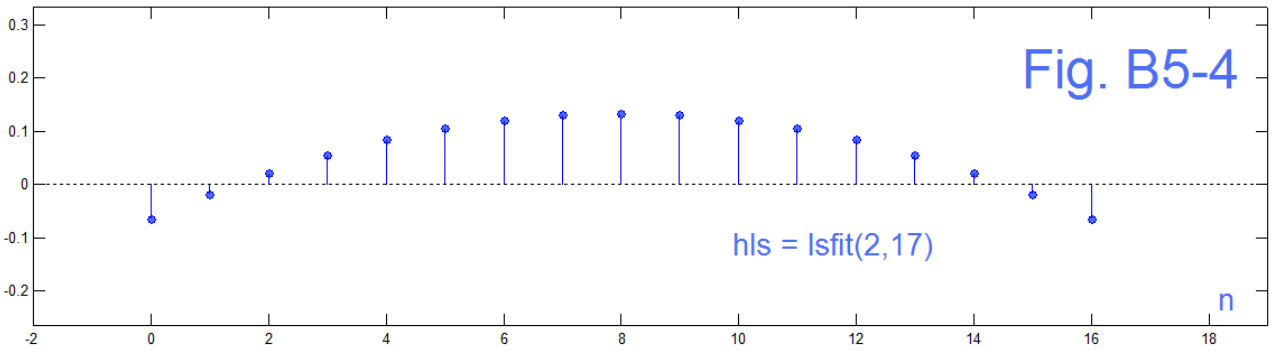
Become Length-3 Moving Average

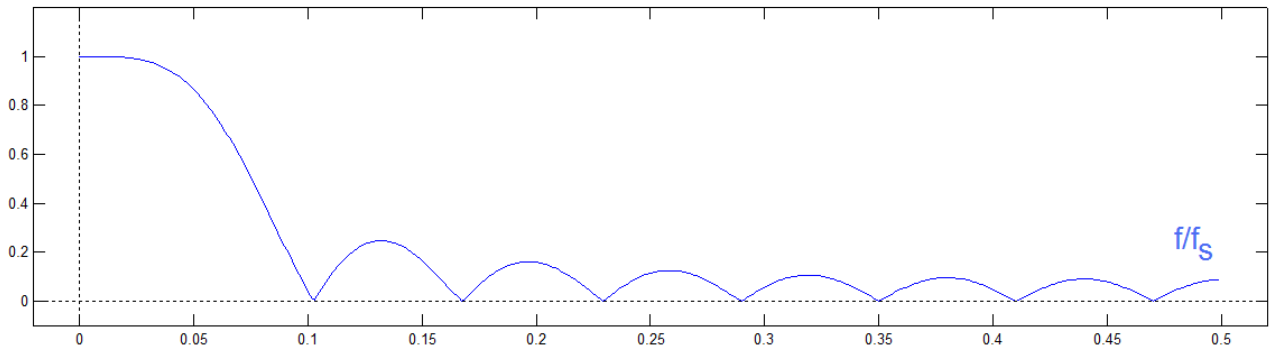


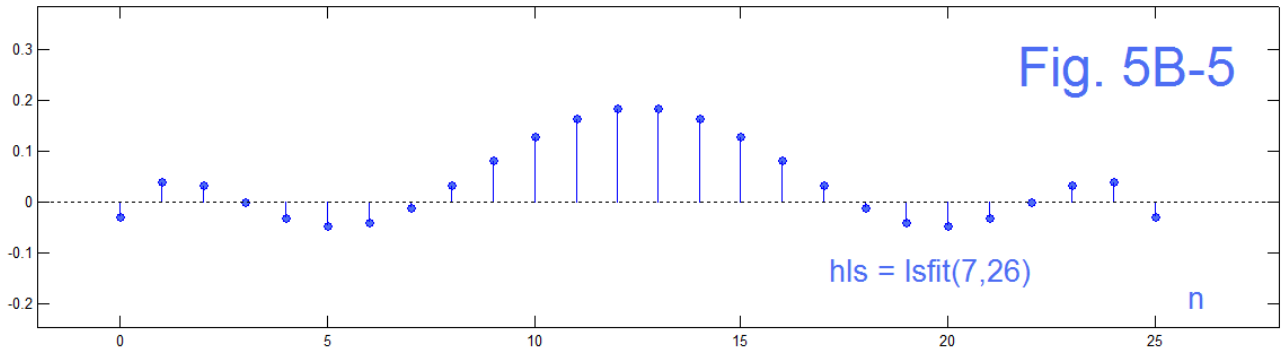


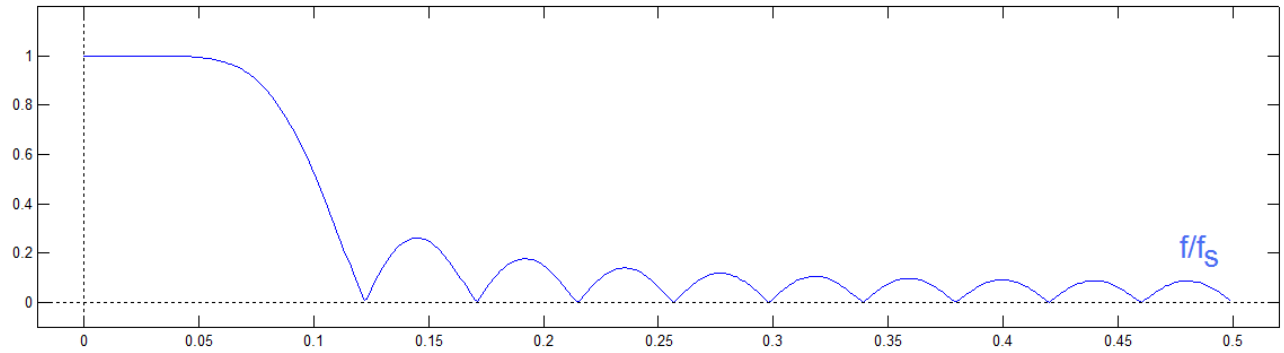
Becomes Length-17 Moving Average

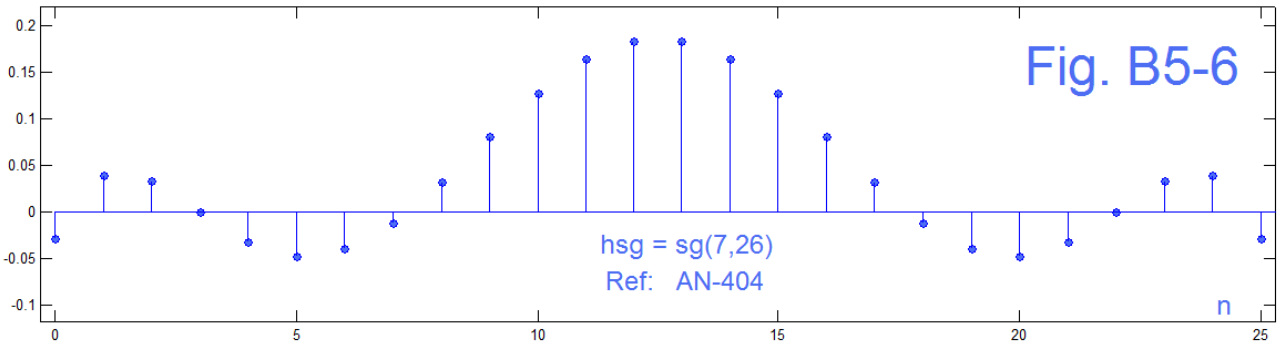




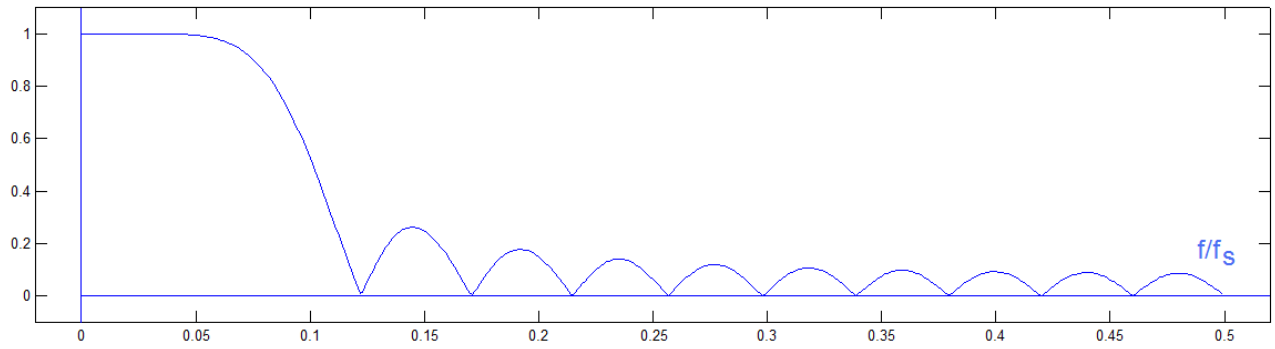






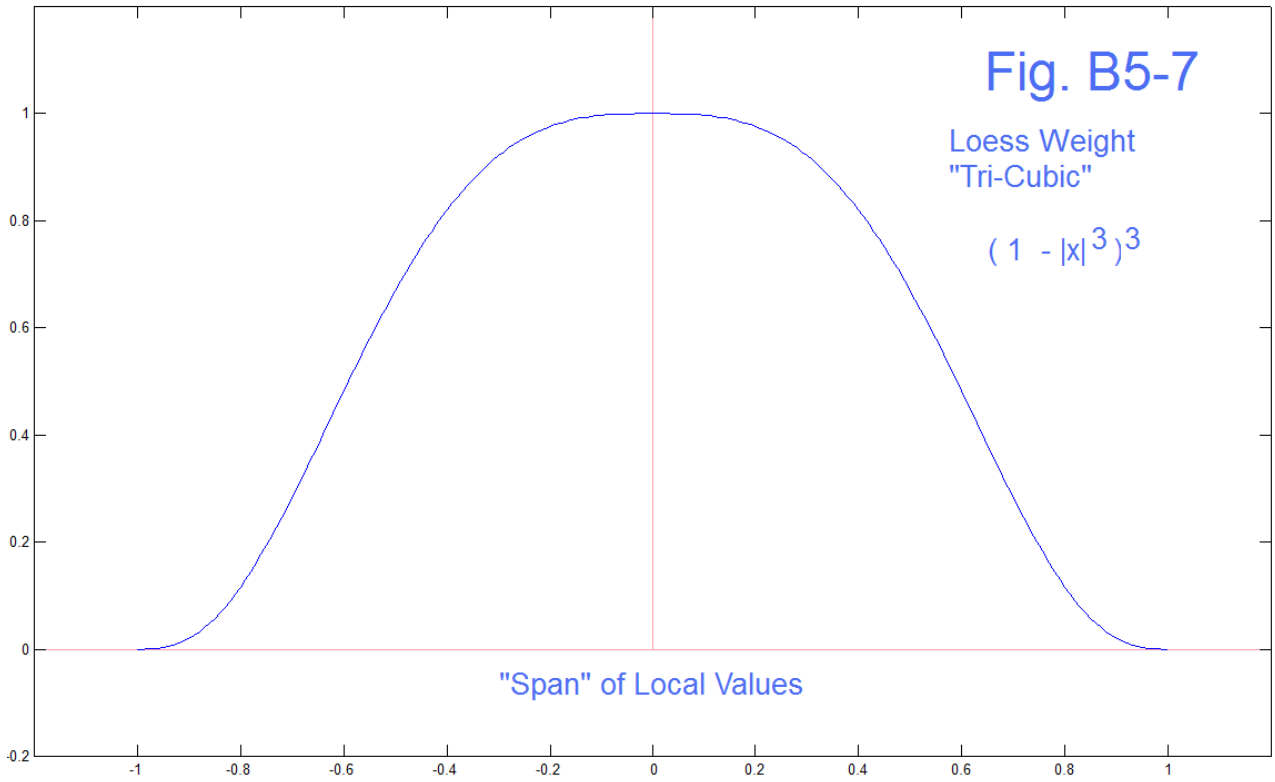


Sovitzky-Golay Gives Exact Same Result



Above we found (rediscovered AN-404 http://electronotes.netfirms.com/AN404.pdf) the close relationship between Savitzky-Golay and Least Square polynomial fitting – the difference resulting in what seems to be merely an alternative method of computing the filter coefficients. The only difference found was in where one method or the other "complained" when numerically stressed.

LOESS and LOWESS involve least squares polynomial fitting, but only to first and second order, and most importantly, the error is weighted. The weighting function used is called "tri-cubic" and is plotted in Fig. B5-7.



This looks like a window function, not totally unlike others we have seen (Hamming, Hanning, Gaussian, etc.) and the reason for this choice is not clear. Note that it is a little more like a rectangle (at least a trapezoid) than some others.

With ordinary least squares fitting, locally, the weight function would be a rectangle. Yet this is not an ordinary window, either in the rectangular case, or the tri-cubic in that it is the squared error (not the signal values) that are weighted. So the errors near the center of the "span" are weighted more than the ends. The "span" shown here from -1 to +1 is in practice scaled to include the range of actual samples considered to be "local". The weight function terminates on the ends, and thus does act as a window in setting the length. We note finally that some sort of alternative weight profile can be used on the ends to allow for the shorter span length.

We can do a very rough test of this idea by redoing the calculation of AN-318 where we found that the output was the length-3 moving average. Applying a span of 3, the weight for the endpoints would be calculated at ±1/2 and would be about 0.6699 (not 2/3 actually),

so why not do the calculation quite roughly by doubling the weight in the center. In AN-318 (flat weighting) we found the slope to be that of the endpoints: m = [x(2) – x(0)]/2 and the intercept to be b = (5/6)x(0)+ (1/3)x(1) – (1/6)x(2) so that the least square line mt+b evaluated at t=1 was x'(t) = [x(0) + x(1) + x(2)]/3, the length-3 moving average. Now, weighting the center point by 2 does not change the slope (this makes sense), but the new intercept is b = (3/4)x(0) + (1/2)x(1) - (1/4)x(2), which moves the line in the direction of x(1). The new weighting for the output point is:

$$x'(1) = x(0) / 4 + x(1) / 2 + x(2) / 4$$

The check on this is to try some examples, jiggle m and b by small amounts, and see that the (weighted) error goes up. It does.

FINAL COMMENTS

Several lessons are presented here. Perhaps the most important is that there is no "magic" formula or filter that fixes everything up just as it should be. The engineering, statistical, and scientific communities need to be aware of the severe limitations of their tools, particularly as they are too often inadequately tested and or produce their own artifacts. This is an ongoing problem. A fancy name is not enough.

Speaking of naming different procedures, we observe that engineers and mathematicians often don't even communicate within their own discipline. Worse signal-processing engineers and statisticians may well have similar stated goals but use tools that may be closely related or vastly different, with the terminology further getting in the way. While different procedures may converge (Appendices A and B6) this requires considerable attention.

There are a lot of software tools such as Matlab, C, and R. Relying on certain functions that are built into these packages is a great convenience, but too often also a trap. Accordingly we urge investigators to write their own code, and then test their code, and compare it with the packaged functions on the simplest possible examples. Too often assumptions are made and consequences ignored. Worse, individual scientists with little knowledge of statistics and/or signal processing (course "101" is not enough) "re-invent" processes, with naivety and with unwarranted confidence, and end up egg-on-face.

* * * * * * * *