ELECTRONOTES 226

Newsletter of the Musical Engineering Group

1016 Hanshaw Road, Ithaca, New York 14850

Volume 23, Number 226 January 2016

LINEAR AND EXPONENTIAL RAMPS

-by Bernie Hutchins

INTRODUCTION

A major element of what we call “signal analysis” is “Fourier analysis”, but certainly not exclusively that. We of course need to know what we mean by “signal”, which is largely by examples and by inference. On the other hand, it is likely less ambiguous what we mean by analysis – we usually mean we are extracting (detecting?) some information contained in the signal. An associated issue (as to what actual information is contained) is how the information got into the signal in the first place.

Two cases are seen here. Perhaps the information was imposed on the signal to form a familiar communications “channel”. That is commonly called “modulation”. We have the notions of original and recovered information in such schemes. On the other hand, the information could be naturally “incorporated” by some (possibly obscure) process. Most likely, in this case, the information is unknown in its details, and often, even in its certainty of inclusion. [Here we risk confusion in that the term “signal” is sometimes used to refer to the information **inside**. That is, the signal is, in this instance, perhaps better referred to as “the data”. For example, some look for a “temperature signal” to be extracted (somehow) from data that represents the width of tree rings centuries before any proper thermometers were known.] In the case of an unclear notion of information (at least for testing), the analysis or detection is not judged by the success we might find in “demodulation” in a communications channel. We are in fact too often lacking a proper notion of a fully right answer. The extracted information would need to be vetted against what is physically, likely correct, and how well it compares to other evidence – and how replicable the result is.

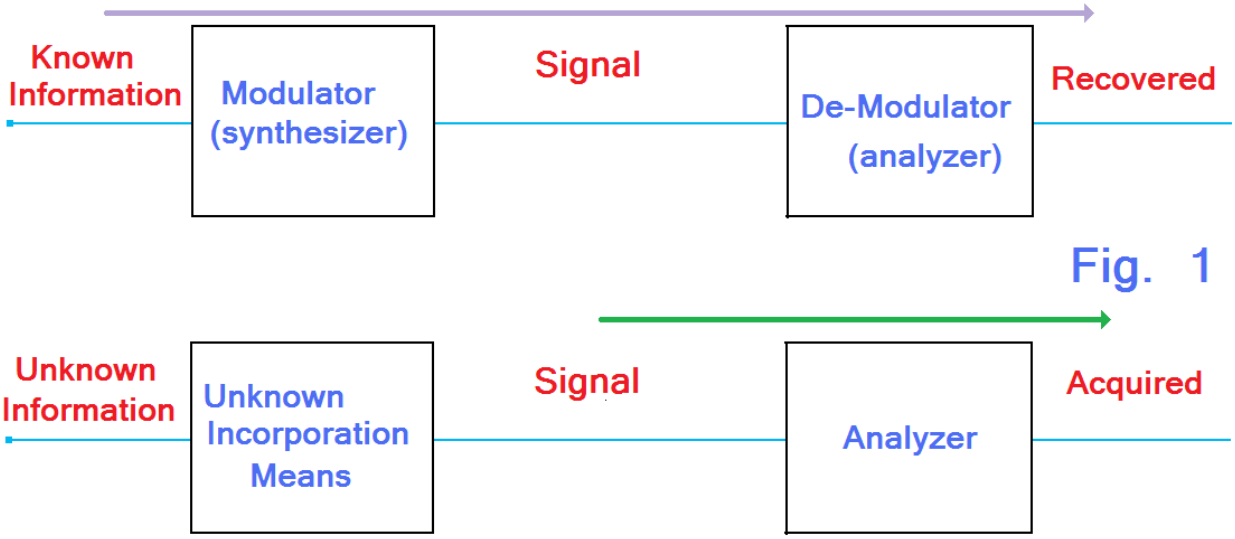


Fig. 1

Fig. 1 is a comparison of the two views. We do not intend to pursue the upper path in its sense of a communications channel but rather in the sense of the generator of a test procedure. Given input data (perhaps as parameters) we "synthesize" a well-defined test signal, and thereafter test the analyzer to see of the known test input is recovered (longer purple arrow). This is exactly what we do when we ask (perhaps in a teaching situation), what the DFT (Discrete Fourier Transform, or FFT: Fast Fourier Transform) of a sequence of time domain sinewave samples would look like in its frequency transformed ("dual") domain . These initial trials test and calibrate our procedures. In the lower portion of Fig. 1 we assume we don't know the input information or the means by which it influences the generation of the signal samples. We just have the signal (green arrow). Our analyzer then produces some output as "acquired". The hope is that this is useful in learning about the desired information (data – perhaps non-Fourier) in the signal.

WHAT ARE WE LOOKING FOR?

If we claim to be doing "signal analysis" it is most likely that others will assume we mean some form of Fourier analysis [1-3] or possibly something like "wavelet analysis" [4] (somewhat similar) or related mathematical modeling (such as polynomial representations [5], system identification such as Prony's method [6,7] or some lesser known functional usages (like Kautz functions) [8]. We mean to suggest that these and others should be considered, with the specific choices being a matter of determining what we are looking for.

Reasonable assumptions are probably that the information in the signal is not obvious just by observation. That is, likely the signal has noise (perhaps a lot), is finite length (else what!), and we don't know very well where the information is in the signal (beginning,

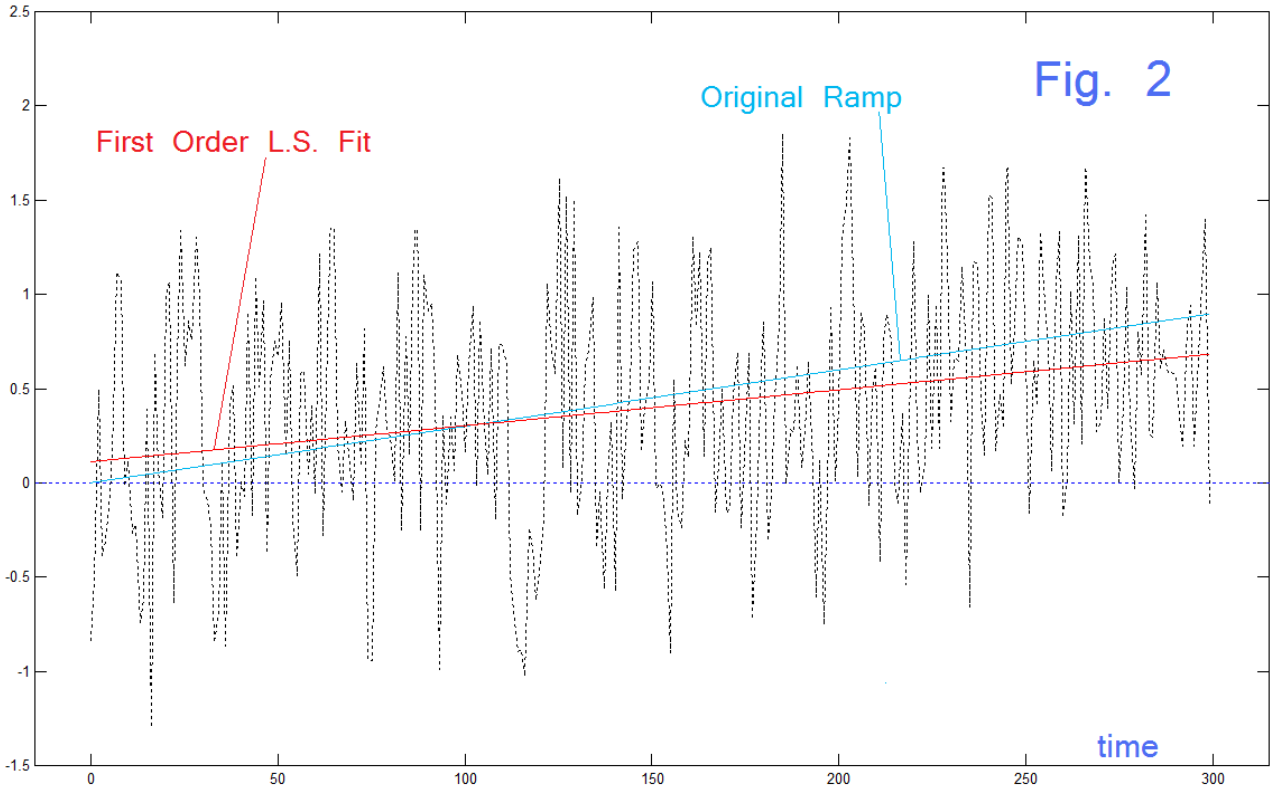
middle, end, everywhere?). In our examples here, however, we perhaps exaggerate the information for a better illustration. For instance, if we have a random signal with a linear trend (ramp upward) we may construct a signal visibly rising, or if there is a periodic component, we may make this oversized. We can easily, subsequently, investigate what happens if the information is better hidden.

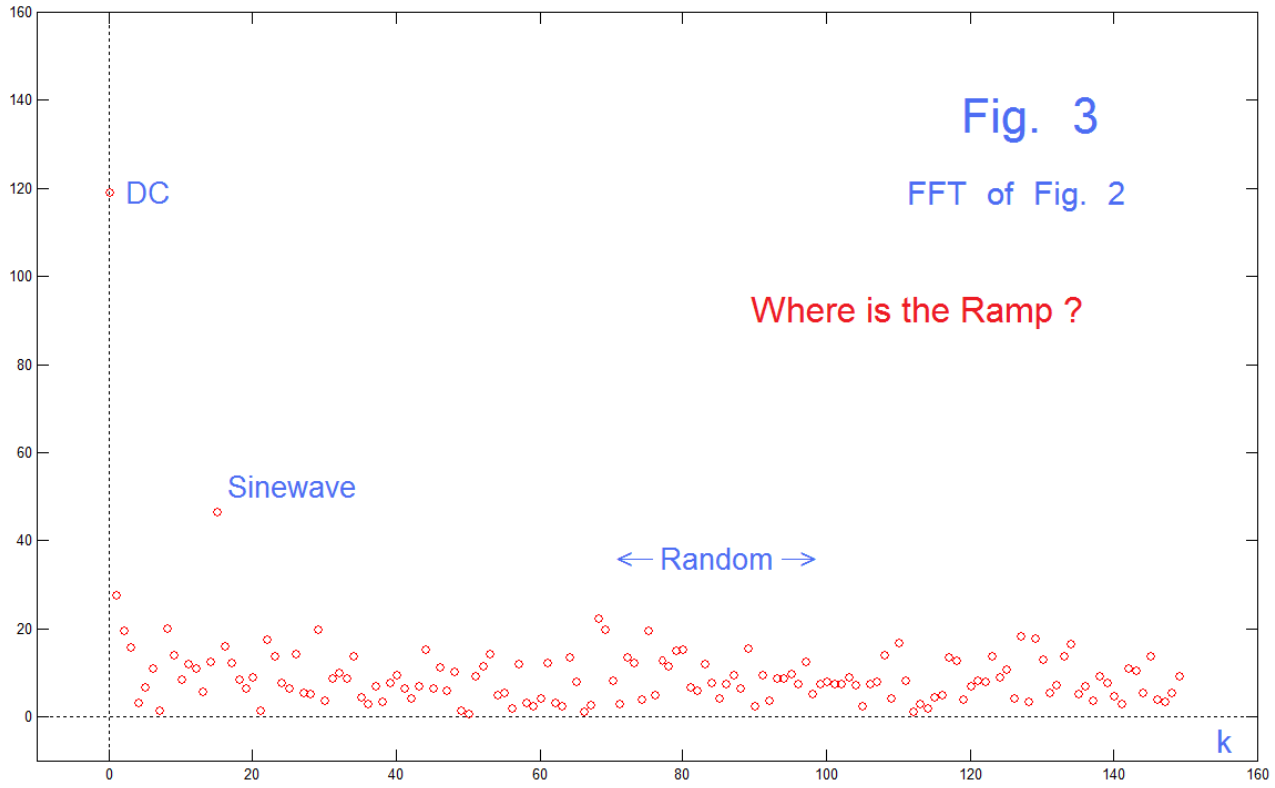
LINEAR RAMP: A THREE FEATURE EXAMPLE

Fig. 2 shows a signal composed of three components as generated by the three Matlab lines:

```
xr=2*(rand(1,300)-1/2);
xt=3*[0:0.001:0.299];
xs=0.4*sin(2*pi*[0:299]/20);
```

This is a random component xr (uniform distribution) between -1 and +1, a ramping "trend" xt of slope 0.003, and a sinusoidal xs of amplitude 0.4 with a total of 15 cycles (300/20). As suggested, just using the original black data, you can pretty well see the ramp, and can almost see the periodic component (15 wiggles). But initially suppose we only have the 300 black data points. If we resort to the FFT of Fig. 2, what can we expect? This is seen in Fig. 3. Keep in mind that we are looking at just one example for the random component here. Each run looks different.

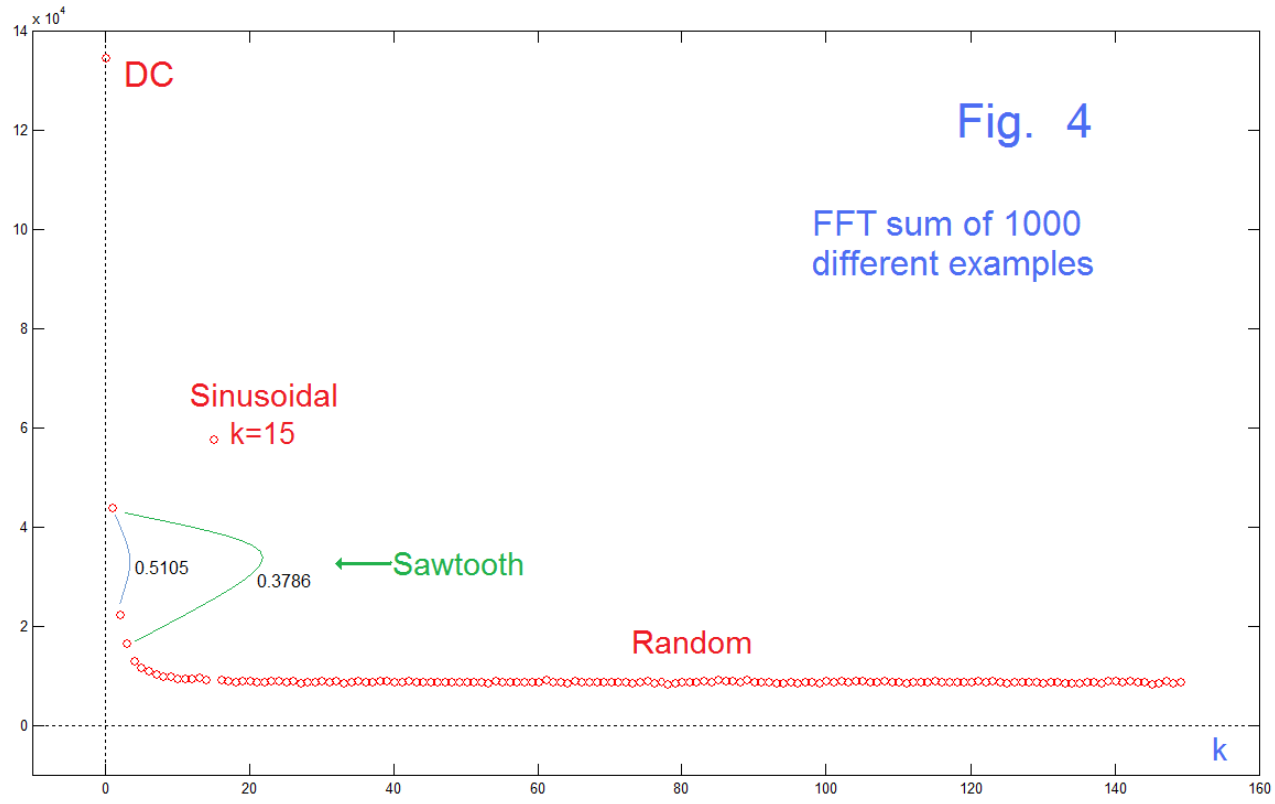




We see in Fig. 3 a DC term at k=0. Further, this term is somewhat larger than we might expect from a random signal, but is fully expected from a glance at Fig. 2. (We do see that it tips upward.) Further, here we see reasonable evidence of a significant component at k=15. [We did choose to generate this so that the 300 samples had exactly15 cycles.] If we had not done this, we could expect a number of adjacent FFT bins of significance. So-called FFT "leakage" [9]. The remainder of the spectrum (note the signal was real so we only need to show the first half of the FFT) is ragged, but might be flat if we averaged a bit (below, and previous discussed [10]). Note that except for the large DC term, we don't see much to suggest the ramp. But any imbalance of polarity (not just a ramp) could be the cause or a large DC term if we guess based only on the FFT.

So – what should the FFT of a ramp look like! Well, first of all, if this is data corresponding to something real (physical), we can't have a linear trend! For a limited period of time, it can ramp up or down, but not forever. All Fourier components we deal with go on forever in time (or are periodic). So we can't have a linear trend, (which ramps to infinity) and expect to find it in Fourier Analysis. For a FFT (or a Fourier Series) we would have periodic components. Thus the ramp (blue on Fig. 2) is one cycle of what we would call a sawtooth with period equal to the full length of the signal (300 samples here). And we have evidence of this in the FFT, as we shall see.

Since it is difficult to see through all the noise, we know the trick of averaging (or just adding) a whole bunch of equivalent random examples. Fig. 4 shows the average of 1000 trials like the one that produced Fig. 3.



As expected, we show the raggedness of the random amplitudes of the spectral components has been averaged away (say in the upper 90% of the FFT). We see as well the strong DC (k=0) component and the periodic component (k=15). What we see easily here, which was hard to see in Fig. 3, is the strong components beginning at k=1 and rolling off quite rapidly. If we had an actual sawtooth, its Fourier Series would fall off as 1/k. Here we see the k=2 as being 0.5105 relative to k=1, and k=3 falling to 0.3786 relative to k=1 – roughly the Fourier Series result. But the FFT results fall only to the average of the random components. This is of course not just purely sawtooth – we had the random and periodic components too. But, hopefully, it is convincing that this is the FFT's response to the linear trend.

We have thus dealt with the linear trend, as the FFT sees it as a sawtooth. The example in Fig. 2 shows two different ramps. The blue one is the one (going through 0 at time 0) that we used to generate the data (the Matlab code for xt). Remember that we used this to make the test sequence. The red line is recovered from the data. This was found as a least-squares fit to all 300 points. Because of the random component, each run is different. Please note that we chose a particularly bad example. Many examples would not cross over. The red recovered result would be more parallel to and be a closer fit to the blue. We wanted to show an example that did not mislead the reader into supposing a generally more favorable result. So how do we actually get the red line?

The least-squares fit to noisy data (a “Scatter-Plot”) is often called a “linear regression” in signal processing and in statistics. If one Googles “Least Squares Linear Regression” one soon finds many tutorials and videos. Too many of these are a superficial discussion of what one is trying to do (fit a line minimizing squared error) and/or an over-detailed discussion (cook book) of how to compute the slope and intercept of the regression line. Presumably, most or all of these are correct and equivalent. If you just want the result, perhaps you don’t need to know why this works.

In fact, many engineers are annoyed by the sometimes stereotypical view of an engineer as a person who knows how to look in the right handbook for the right formula, plug in the right numbers, and do the math (arithmetic!) correctly. Thereafter, the bridge does not fall down! Many engineers have an admirable preference for a colleague who derives or re-derives procedures right at the “chalkboard” and then checks the result against all available supporting evidence. This keeps reliance on memory (or a cache of references) in a reasonable perspective.

Here we will suggest the reader concentrate on the notion of attempting to fit a straight line to more than just two points – impossible to do exactly. Instead, we resort to minimizing the total squared error. This minimization process is the well-understood procedure of solving the co-called “Normal Equations”. That is, we use the calculus procedure of minimizing by setting partial derivatives equal to zero. This really is straightforward (see the two-page App Note [11] if you have forgotten this.) That’s it. We have more equations than unknowns, and the procedure is basically the same for many more points and higher order regression curves (polynomials beyond a straight line). This becomes familiar as the “pseudo-inverse” or “least squares inverse” of the matrix representing all the equations. The use of the over-determined equations was a fundamental feature of (frequency domain) “Generalized Frequency Sampling” [12] and has been brought in frequently recently [13a, 13b, 13c, and 13d, 14]. In the present example, the setup is exceptionally simple. Plenty of examples of applications and code.

Specifically here we assume a straight line x(t) = b + mt where we seek the values of b (axis intercept) and m (slope). If we have two points, say x(1) and x(2) we write two equations:

$$\begin{aligned} b + \quad m &= x(1) \\ b + 2m &= x(2) \end{aligned} \tag{1}$$

which are easily solved for b and m. If we add a third point, we now have three equations:

$$\begin{aligned} b + \quad m &= x(1) \\ b + 2m &= x(2) \\ b + 3m &= x(3) \end{aligned} \tag{2}$$

and these we can't solve exactly. Here is where the classic textbook case of least-squares fitting comes in [11, 14] in a convincing manner. So what could prevent us from writing 300 equations for 300 data points?:

$$
\begin{aligned}
b + m &= x(1) \\
b + 2m &= x(2) \\
b + 3m &= x(3) \\
&\cdots\cdots\cdots \\
b + 298m &= x(298) \\
b + 299m &= x(299) \\
b + 300m &= x(300)
\end{aligned}
\tag{3}
$$

In theory, we could handle this in exactly the same manner we did three points. In practice, even going to just 4 or 5 points presents pages and pages of algebra for which one is unlikely to avoid errors. Thus we look for the general solution, as recently reviewed [14] or as found in many texts [15], in terms of the matrix representing the equation set.

$$
\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ & \\ & \\ & \\ \cdot & \\ \quad\cdot & \\ \quad\quad\cdot & \\ 1 & 299 \\ 1 & 300 \end{bmatrix}
\begin{bmatrix} b \\ m \end{bmatrix}
=
\begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ \\ \\ \\ \cdot \\ \cdot \\ \cdot \\ x(299) \\ x(300) \end{bmatrix}
\tag{4}
$$

$$
\mathbf{M} \qquad \mathbf{p} \qquad \mathbf{x}
$$

Here we have the 300 by 2 matrix M (tall and skinny – certainly not square) and the parameter vector p: the two unknowns b and m. All the 300 values of x are known. Thus the job is to invert the matrix equation for p. We can't possibly solve this by hand. If we have Matlab, we can use one of three methods. The simplest would be to just ignore the equation above, and use the built-in ***polyfit*** as:

```
p=polyfit([0:299],x,1)
```

which fits a polynomial of order 1 to the 300 points of x. It just does that. (p = [b, m] here). The second Matlab function uses the pseudo-inverse ***pinv*** as:

p=pinv(M)*x' (5)

The third method which is illustrated here for Matlab but which is more general is:

p=inv(M'*M)*M'*x' (6)

which only requires a square matrix inversion. Here the single quote is a transpose. Frankly, you decide if you need the transpose by guessing and seeing if you get an error. The same is possibly true to the three Ms in the third method. Keeping in mind also that the matrix products must "conform" it is generally possible to just fudge around a bit until it obviously gives a proper result. Without the Matlab code, the matrix solution, in common matrix notation, is:

$$p = (M^tM)^{-1}M^tx \quad (7)$$

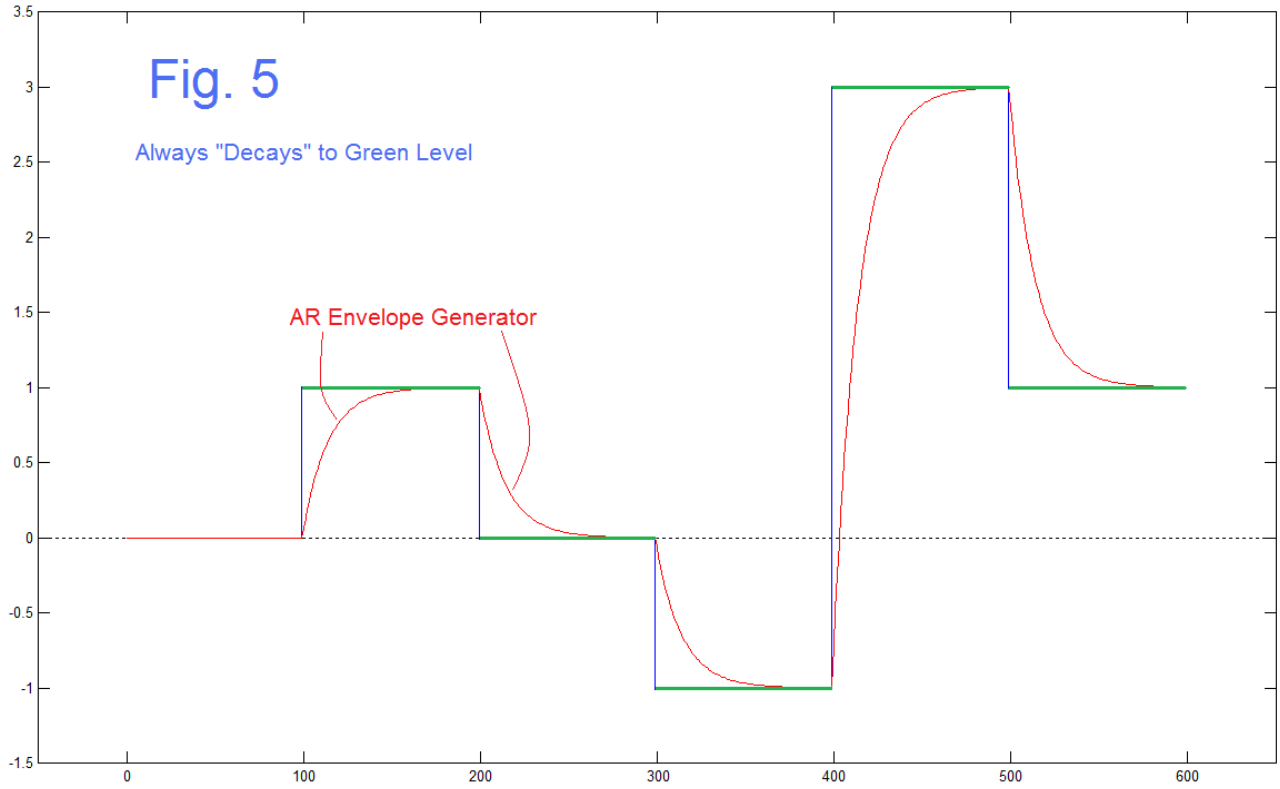
where the ^{t} signifies transpose and $^{-1}$ signifies a matrix inversion. (The data x may or may not need to be transposed depending on the program.)

THE EXPONENTIAL RAMP

Above for the linear ramp we have cautioned that it can't correspond to anything "real" because we can't have it running on forever; because it ramps to infinity in both directions. This may be an issue we barely mention, let alone worry about. We know the ramp only applies locally; or not unusually, it is part of something that resets periodically (a sawtooth).

The exponential ramp here is understood to be the case where there is some system that ramps toward and exponentially approaches some target level. This is familiar as the well-known series RC [16] such as we use in electronic music envelope generators. This "ramp" has a curved shape, and is fundamentally different from a linear ramp.

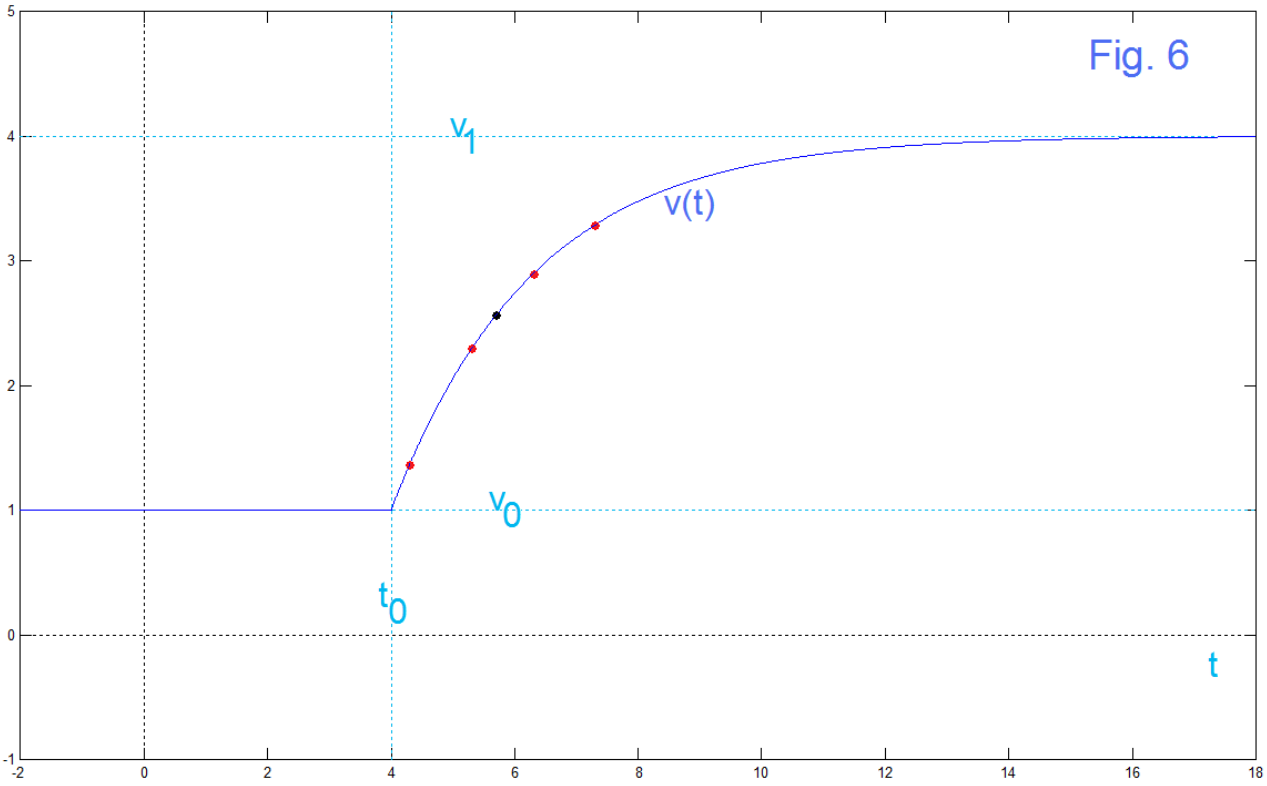
Fig. 5 reminds us how a first-order RC low-pass follows a series of steps at its input. The green steps with the blue transitions represent the input while the red waveform is the output [16]. The first two steps correspond to an "AR" (attack/release) envelope generator as used in electronic music. The first misconception to address here is that the network somehow knows the direction of the step (up or down) and responds differently to that polarity, being convex if charging up, and concave if charging downward. This would be illogical in the first place. Further, we see that consistently the red waveform moves rapidly in the direction of the step levels (green) and then slows asymptotically. It consistently "discharges" exponentially to the green.



A second thing to note is that the target level can be negative (300 to 400), that it may cross zero (400 to 500), and need not discharge to zero (500 to 600), but see below as well. Here we have also chosen a time constant so that the red curve gets well within the plotting width of the target green levels within the 100 time intervals allotted. In theory, it never gets there of course. So, for example, the discharge from 1 (apparently) toward 0, from 200 to 300, did not actually start at 1, but was slightly less. The overall view suggested by Fig. 5 is that the system "chases" the changing step levels. So there is no notion (as there was with the linear ramp) that it could run away positively or negatively.

Less conveniently, we have to ask if we could use the data of the curve (red) to obtain the parameters of the segments. With linear segments, we looked to obtain the slope and intercept for a segment. How do things change with exponential segments?

Fig. 6 shows the case of just one segment. Here we assume that the input is v_0 to the left of t_0, and becomes v_1 right of t_0. This is an offset step. It is probably safest to assume that left of t_0 the input has been v_0 for a very long time. The network has a time constant $\boldsymbol{\tau}$. Thus we have four unknowns, v_0, v_1, t_0, and $\boldsymbol{\tau}$. In fact, you could of course read the first three unknowns (actually, all four) from the graph, but we are postulating that you just have the data. In fact, we show in Fig. 6 four red samples between 4 and 8, perhaps with the idea that four data points should support a solution for four unknowns. Thus, suppose we only had the four red points. Find the exponential. Not so easy! We will in a moment write down the general equation for the exponential, so could write four instances of this equation. But these won't be linear equations of course.



Here we have assumed that the level v_0 is well established. This serves as a baseline for what follows, which is a charging by a step of amplitude $v_1 - v_0$. and it is not difficult to just write by inspection:

$$v(t) = v_0 + (v_1 - v_0)\left(1 - e^{-\frac{t-t_0}{\tau}}\right) \tag{8}$$

If we knew the three values v_0, v_1, and t_0 we can solve this for $\boldsymbol{\tau}$. Solving for the point corresponding to the black dot [v_0=1, v_1=4, t_0=2, t=5.7, v(t)=2.5674] , we get $\boldsymbol{\tau}$ =2.3, exactly the value we used to generate the curve in Fig. 6, using:

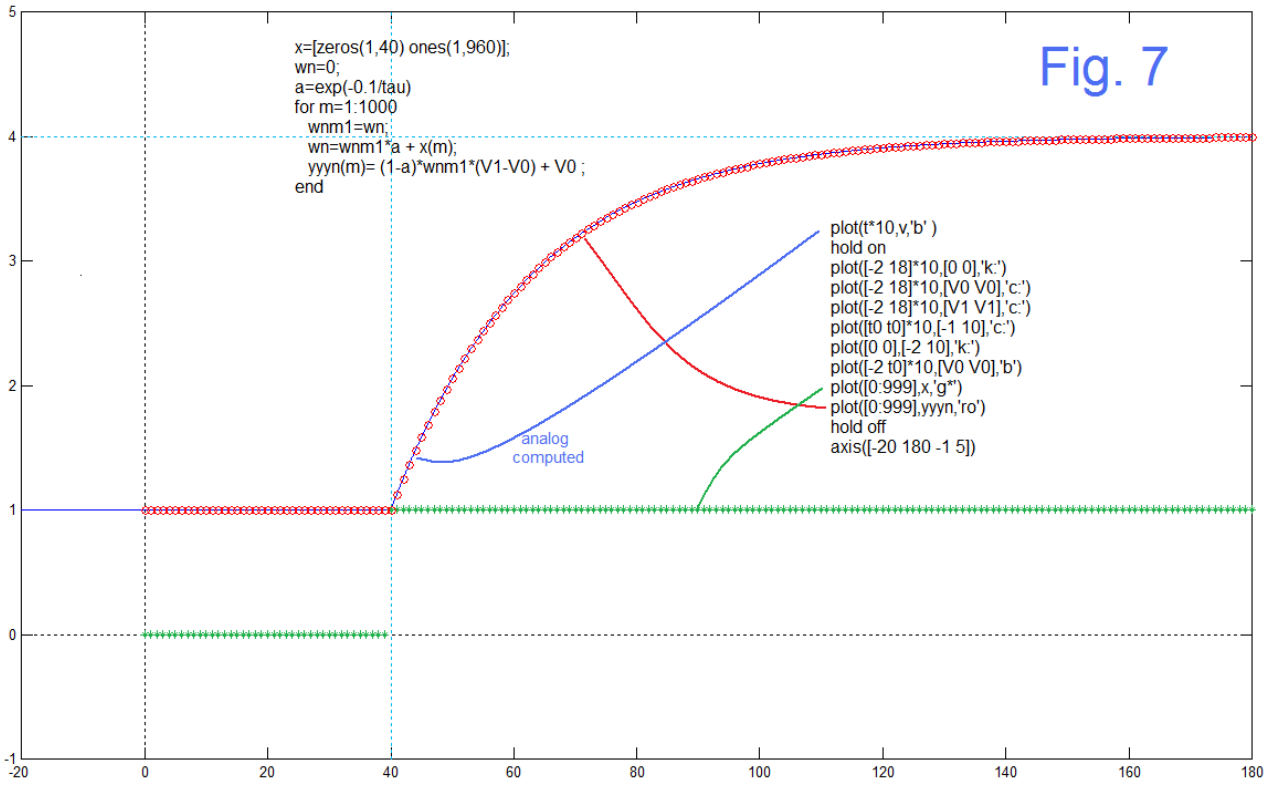
$$\tau = \frac{t_0 - t}{\ln\left[\ 1 + \frac{v_0 - v(t)}{v_1 - v_0}\ \right]} \tag{9}$$

Here we note that we have estimated the unknowns v_0, v_1, and t_0. We have enough of the curves to see the two voltage levels, and the upward transition at t_0 is sharpest at t_0. Thus we can solve for $\boldsymbol{\tau}$. In fact, we could also without much difficulty estimate $\boldsymbol{\tau}$ from the 1/e time of the charge, which is about 63% of the amplitude of the step. This would be just beyond t=6 on the curve.

Now suppose we have four data points (like the red dots in Fig. 6). That is, we know v(t) for four values of t (4.3, 5.3, 6.3, and 7.3). We have, presumably, taken these within

what seems to be a charging region. It would not matter in theory if we took them in a relatively flatter region, but we prefer some wider range of v(t) to avoid numerical problems. So we plug these in as four instances of equation (8) – and we are OUT OF LUCK. These are not linear equations.

So, in this case, we make the attempt at an exponential model – i.e., Prony's method [6, 7]. Surprisingly (I was surprised) this doesn't work in general. We are still out of luck. Above (Fig. 5) we emphasized the general "equivalence" of the array of exponential curves, and this remains the case. Except the first two curves (really the second curve only) is untypical, and yields to Prony. The problem is that the other curves are DRIVEN. The input is NOT zero. This we shall presently look at in some detail. Before doing that however, we note that we can rather exactly compute a discrete version of Fig. 6 as Fig. 7.



DETERMINING FEEDBACK FROM DATA

First, in the case of an exponential ramp decaying to zero, we know that the analog case is (Fig. 8A), in response to an impulse,

$$V_{out}(t) = e^{-t/RC} \qquad (10)$$

so that for a time interval T, the output decreases by a factor of:

$$a = e^{-T/RC} \qquad (11)$$

This in turn suggests two different digital equivalents for the RC circuit, Fig. 8B which is exactly equivalent to Fig. 8A for the impulse case, and Fig. 8C which is equivalent for the case of a step, the DC gain of network C being 1/(1-**a**) from:

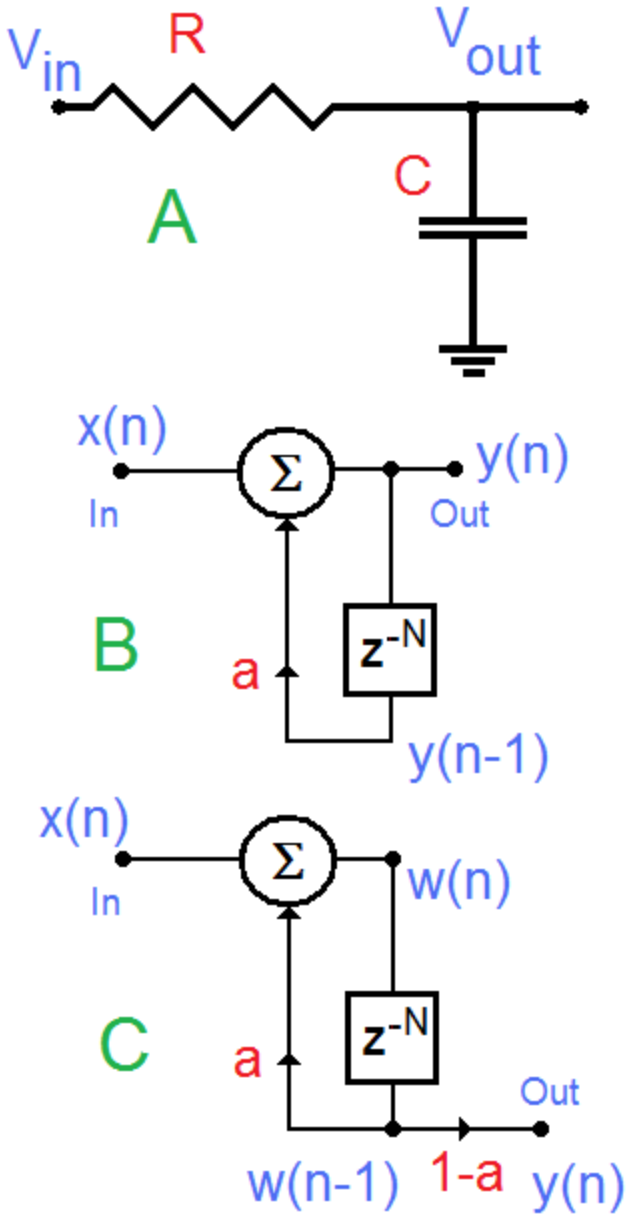
$$H(z) = \frac{1}{1 - az^{-1}} \qquad (12a)$$

so that:

$$H(z = 1) = \frac{1}{1 - a} \qquad (12b)$$

Although the only differences between Fig. 8B and Fig. 8C are a delay of 1 and a gain of (1-**a**), this is necessary to allow for the discrete nature of the digital networks. Note that the continuous-time network (Fig. 8A) will work for either the impulse (a Dirac delta) or a step. In the case of the Dirac delta, we regard the input V_{in} as being zero (grounded) except at time t=0 a charge is instantaneously applied to the capacitor to make the output equal to 1. Thereafter, the output decays exponentially as in equation (10).

Fig. 8



Solving for the decay factor a is a trivial example of Prony's Method [6] and can be seen from equation (10):

$$V_{out}[(n+1)T] = e^{-(n+1)T/RC} = e^{-nT/RC}e^{-T/RC} = \quad V_{out}(nT)e^{-T/RC} = a\,V_{out}(nT) \qquad (13)$$

for which the discrete time-version is (Fig. 8B):

$$y(n) \;=\; a\,y(n-1) \qquad (14)$$

In consequence, any two consecutive values taken during a discharge phase constitute one equation in one unknown (the unknown **a**). The second part of Prony's method is a simple use of initial conditions. For example, from equation (14) we have:

$$a = \frac{y(1)}{y(0)} \tag{15}$$

so:

$$y(n) = y(0)a^n \tag{16}$$

Above (around Fig. 5) we have made the point that all exponential ramping between step levels is largely equivalent. So – can't we apply an equation such as (14) to <u>any</u> output segment. No we can't – only to segments discharging to zero! This we will elaborate on just below and attempt to get around after that.

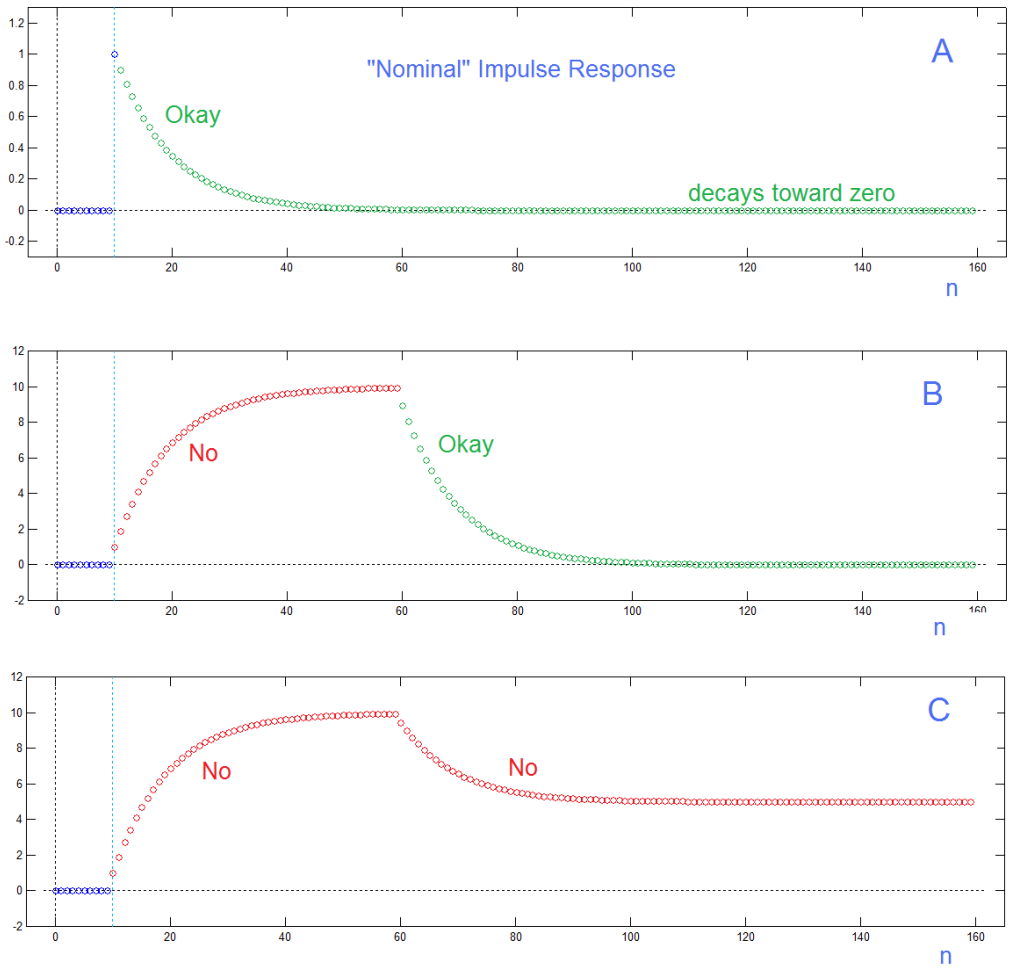


Fig. 9

Upper Output

Impulse
Input = 1 at n=10
0 else

Rectangle
Input= 1 n=10 to 59
0 else

Stair - Down
Input= 1 n=10 to 59
1/2 n=60 to 159
0 else

Fig. 9 shows the response of Fig. 8B (the digital network with output at top) for three different inputs. Here we have set **a**=0.9, which would correspond to sampling interval T = -ln(**a**)RC = 0.1054RC [equation (11)], in the analog case. This would mean that 9.49

(1/0.1054) samples would be the RC time constant and indeed, counting down about 9 to 10 samples gets us down to about 1/e (37%). Of the three plots in Fig. 9, the top one (Impulse) is the simplest. We can use any pairing of points after the impulse to solve for **a**. [The equations are easily modified if for some reason we prefer non-consecutive points.] Thus the entire discharge is shown in green (okay to use). In contrast, the rectangle (middle plot of Fig. 9) is not usable in the red charging portion (however – stay tuned); although after the rectangle ends, it is perfectly good (green) and is identical to the impulse. Continuing to the bottom plot, neither portion (both red) of the stair down is usable (again – stay tuned). Based on the rectangle case, we are not surprised that the first (upper) step does not work. So what is new in the bottom plot is that the step to 1/2 discharge does not work either. This is because the discharge is not to zero – the system is sill "driven".

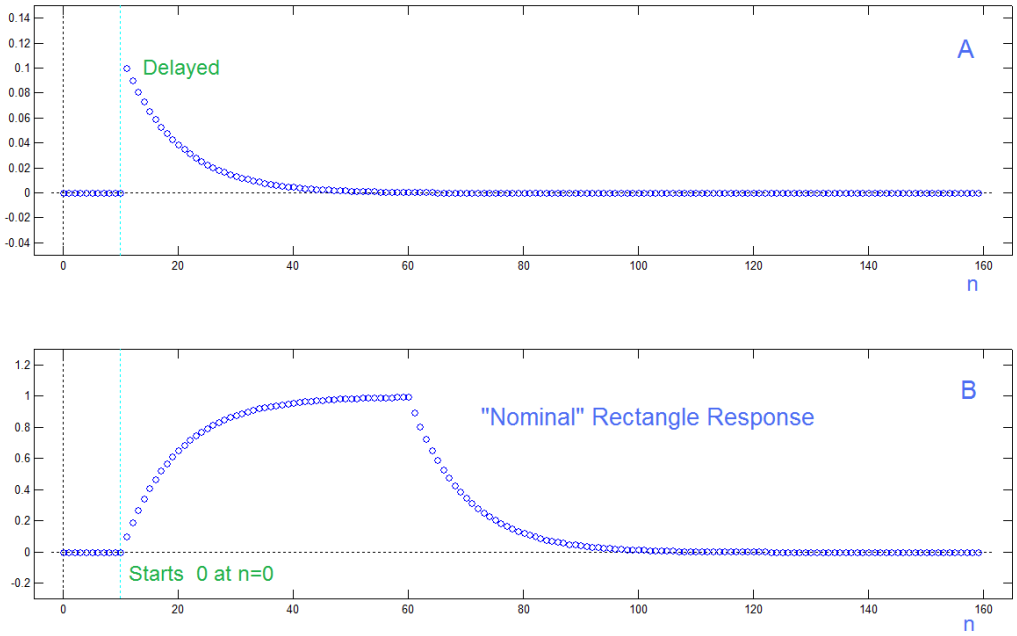


Fig. 10

Lower (Delayed) Output

Impulse

Rectangle

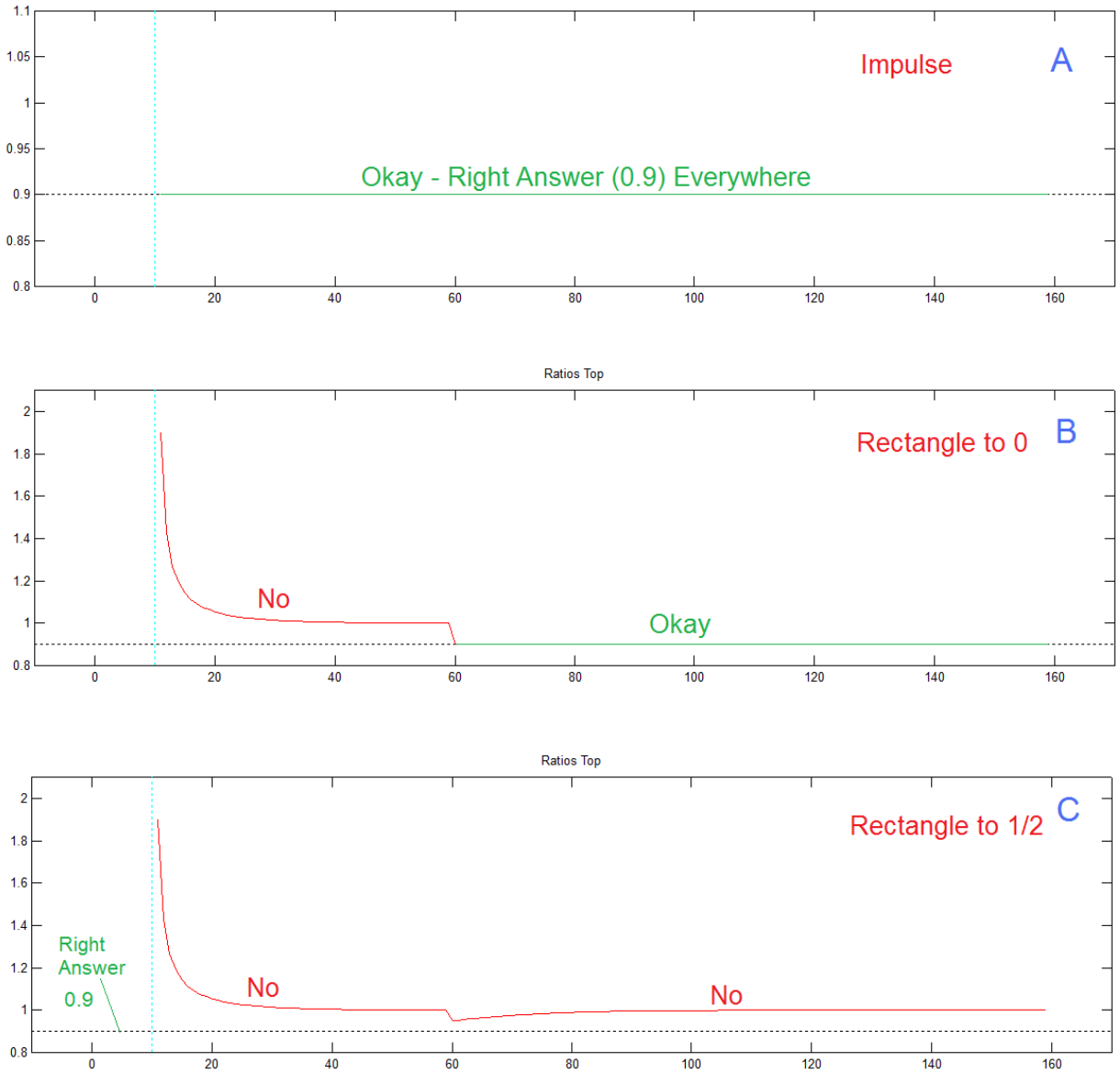
With regard to the determination of the multiplier **a** from the data, Fig. 10 here is an aside. This shows the corresponding curves for the lower (delayed), Fig. 8C. They look pretty much like the top two plots of Fig. 9. As mentioned, the difference is a delay of 1 sample, and a gain of 1-**a** (which is 0.1 here. The only really interesting thing here is that now, the rectangle response (or step) agrees with the analog RC circuit exactly.

Back to the issue of finding the parameters of the exponential ramp from the data, we have said that it seemed necessary that the network be undriven – thus discharging to zero. Yet the reader has noted that we said that what is apparently a convex charge is really a discharge to a non-zero target level. We could easily propose (for example) subtracting the middle sequence of Fig. 9 from 10 so that it looks like a concave decay. Indeed this would obviously work. In fact, we can propose adjustments to scales and offsets so that any segment could potentially work. We expect this is somewhat obvious. What are the details?

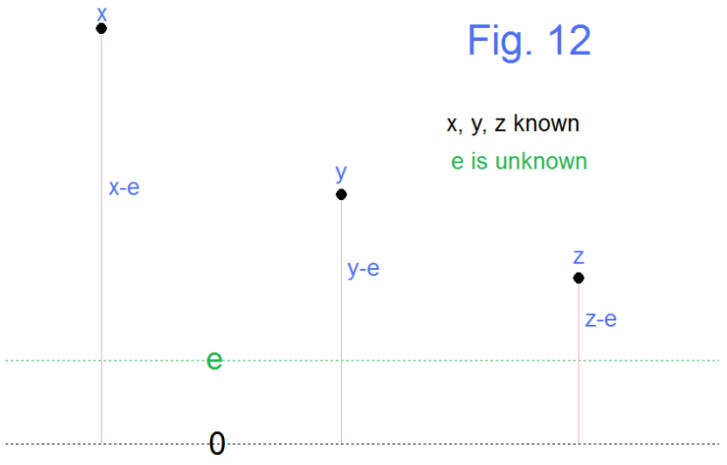
LOOKING FOR A MORE GENEAL SOLUTION

We have warned against directly using certain portions of output sequences if we want the right answer for the feedback factor **a**, and hinted that certain adjustments might make it possible to use additional segments. What does a wrong answer look like? Well this we can easily plot by continually monitoring the ratio between consecutive samples. As stated, for some portions (green in Fig. 9) we consistently get the right answer, which is 0.9 in our examples. In contrast, the red segments give us answers that are NOT 0.9. These results are shown in Fig. 11 and correspond exactly to Fig. 9. These serve to let us check for good areas. Perhaps more usefully, we may be able to use this idea to establish there is a different, non-zero reference level that will work, and then solve for **a**.

Fig. 11



Here we have a problem which is much easier that we might expect. It might look like we need to do a software search, but a closed-form solution is at hand.



Consider the case as shown in Fig. 12 where we have three consecutive samples x, y, and z which are known. We seek to determine if there is some constant level **e**, where **e** is not necessarily 0, such that the samples are exponentially converging to the target level **e**. Using the reference level **e**, we seek a ratio **a** that is the same for all three samples. Thus we want:

$$a = \frac{y-e}{x-e} = \frac{z-e}{y-e} \tag{17}$$

We can cross multiply the second and third terms in equation (17) and solve for **e** in terms of x, y, and z (happily the e^2 terms cancel), giving simply:

$$e = \frac{y^2 - xz}{2y-x-z} \tag{18}$$

giving the charging target level, and plugging **e** back into (17), we find the actual value of **a**. For example, if [x y z] = [5 3 2], as actually drawn in Fig. 12, **e** = 1 and **a** =1/2. If we have a convex set of samples, say [x y z] = [3 5 6], then **e**=7 (above the samples), and **a** is again 1/2.

This opens up the possibility of discovering the parameters of the exponential sequence of Fig. 9C (and Fig. 11C) which were not obtained originally because neither charge level was zero. After computing the sequence, we easily computed the ratios of consecutive samples, and this was neither correct, nor even constant (Fig. 11C). We can also compute, for consecutive samples, the target level **e** using equation (18) and the ratio **a** from equation (17). This, aside from certain errors where there is a transition (around n=60), shows an outstanding degree or success. Essentially Fig. 11C (complete failure to find **a**=0.9) now leads to Fig. 13B, near complete success. This whole finding is likely easy enough to see if we were to just cut and paste (scale and shift) various curves and describe the operations in terms of simple arithmetic.

Yet one more example is to create a “mixed bag” of exponential ramps. This we do by not just changing the target levels, but the exponential ratio factors. It might seem a bit hard to come up with an idea as to how the ratio factors might change. Yet readers of this newsletter are familiar with various “envelope generators” such as the ADSR generator

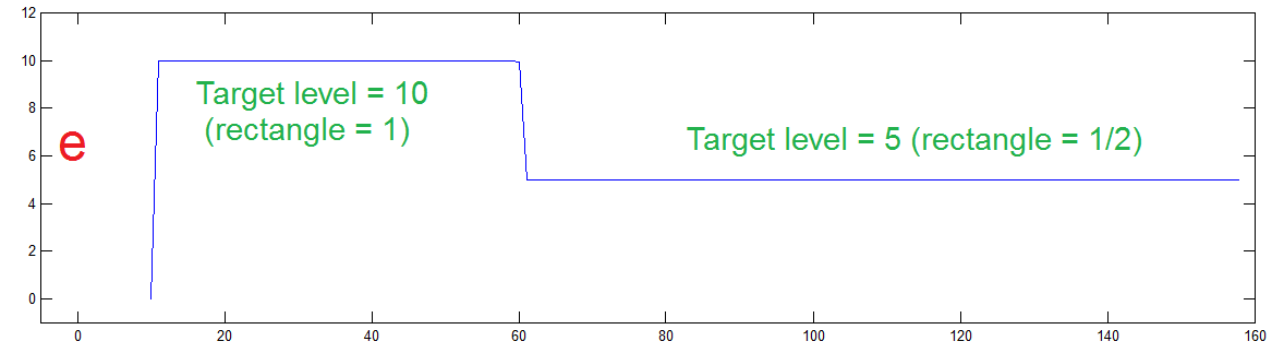
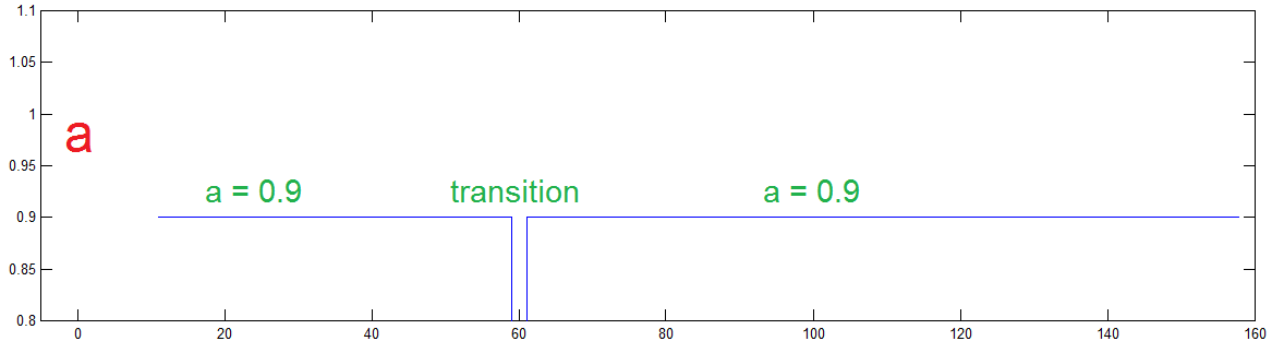


Fig. 13



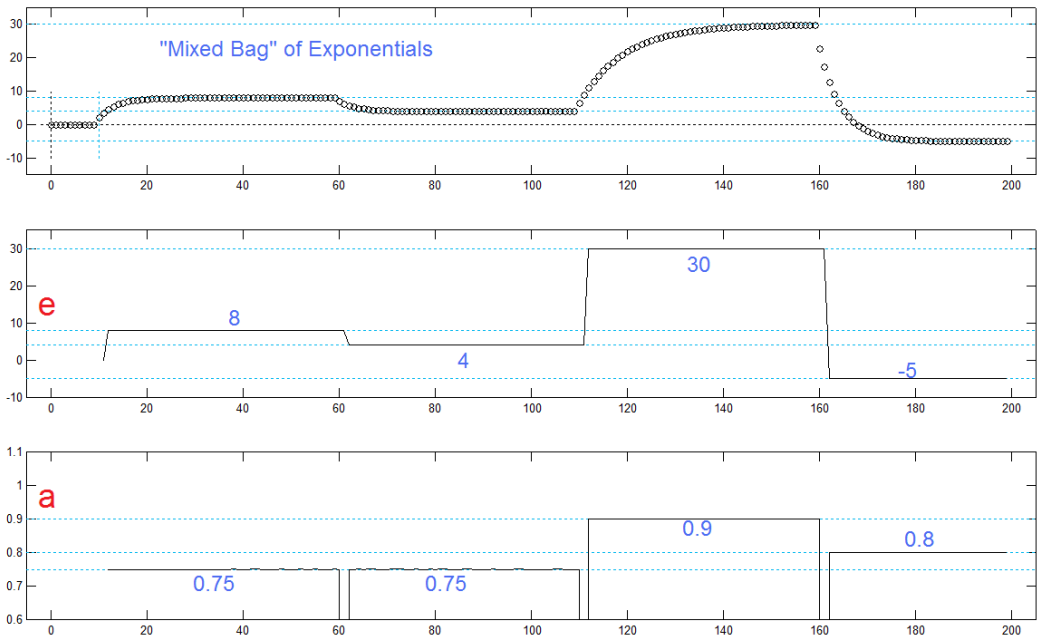


Fig. 14

(Attack, Decay, Sustain, Release) where the different segments have different target levels and RC time constants (controlled by switching or with diodes, etc.). So the result shown in Fig. 14 is actually relevant. Here we have the segments according to the table that follows:

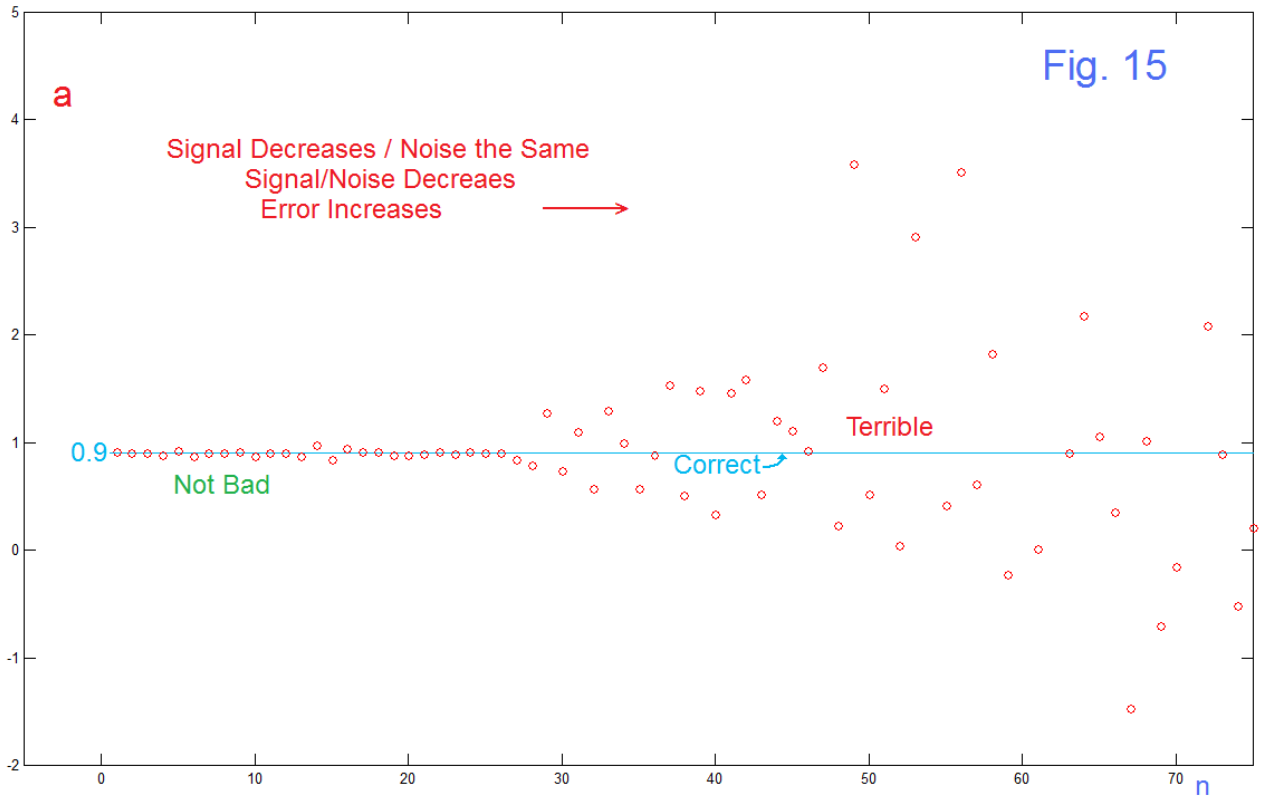
Interval	Input x	a	DC Gain:1/(1-a)	Ramps to:
1-10	0	0.75	4	0
11-60	2	0.75	4	8
61-110	1	0.75	4	4
111-160	3	0.90	10	30
161-299	-1	0.80	5	-5

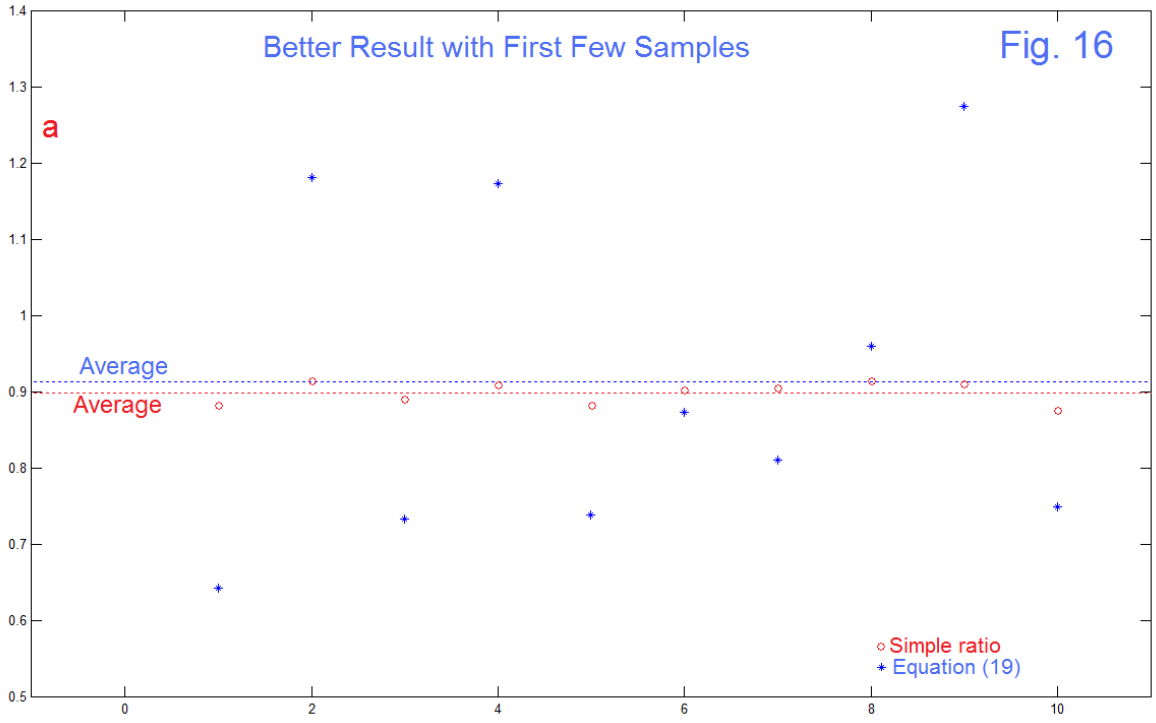
The same procedure [equations (18) and (17) applied to triplets of samples] gives us the right answers. We note happily that Fig. 14C shows the correct values of **a** are recovered (except at the transition regions). Also, the fourth segment as seen in Fig. 14A and Fig. 14B ramps to, and is recovered, as 30. Having done all the previous rampings with a value of **a**=0.9, this looked perfectly correct. But, at first, the input levels **x** of 2, 1, and -1 seemed like they should have been, correspondingly, to 20, 10, and -10. This surprise also looked correctly plotted (8, 4, and -5), and it was clear that we had not thought about the fact that the value of **a** was changing (that was the point of investigation after all). The "DC gain" (equation 12b) was the amplification of x. So, not only do we see faster convergences for lower values of **a**, but also lower gains.

NOISE IN EXPONENTIAL RAMPS

It is usually necessary to consider the effects of noise in the extraction of the parameters of a ramp. In the case of the linear ramp, we wanted to use as long a sequence as possible. The same is not true for an exponential ramp. The reason is that for the exponential decay of the signal amplitude, the signal gets smaller even as the noise remains the same, so the signal/noise ratio gets smaller and the errors in the parameters build up. This we saw in the case of the resonator [7].

We saw here that for the simple case of a first-order exponential decay, two consecutive samples (or any two samples if we do things right) we can extract the parameter as the ratio of the two samples. In the noise free case, we get the same (correct) answer no matter where we start. In the case of noise, the ratio will differ for different choices of sample pairs. If the noise is relatively small, the error may well be small enough. If the noise is relatively large, the errors can be large. We also know that we can mitigate the errors by averaging many pairs. We expect errors to tend to cancel. This means that we would in general prefer to use a long sequence and get a lot of answers to average. Except that we can't necessarily take a very long signal without including portions at the far end where the signal/noise ratio is very bad. Fig. 15 shows our example network where **a** = 0.9 (Fig. 9A) and where the signal is large at the beginning (left) the calculated value of **a** is close to 0.9. Every run of this program will differ, but we see the errors rapidly building





up as we move to the right. So if we imagine a series of values already reasonably close to 0.9, and we average them, we expect an even better result. However, we also recognize that we are going to eventually pay the price as we move into the region (like by iteration 30 in Fig. 15) where errors start to grow. Here we are pointing out the problem, not suggesting an optimum solution.

In Fig. 16 we want to show the averaging of far fewer calculations. At the same time, we can point out an alternative. Going back to equation (18) we pointed out that the value there calculated for the target level **e** could be plugged back into equation (17) to get the ratio **a**. Very true, easy, and at times essential. If we do plug this back in, **e** is in terms of knowns x, y, and z, and this gives us **a** in terms of just the known values of three consecutive samples x, y, and z. We did not do this explicitly. Sometimes we find an interesting result if we carry calculations all the way through – something almost "obvious" and often insightful jumps out. This did not seem to be the case, but we can here look at the result. It is:

$$a = \frac{y^2 - xy - yz + xz}{-y^2 + 2xy - x^2} \qquad (19)$$

Nothing apparently insightful here. What it is is a means of calculating **a** from the data. Note from equation (17) that when **e**=0, **a** is just the ratio of two consecutive values. It is amusing to note that if the sequence is decaying to 0, then the value of z is just y times (y/x) or $z = y^2/x$. This when substituted into equation (19) indeed gives **a** = y/x right back.

The point here was to suppose that in allowing **e** to vary the response might be better disposed to noise averaging. This appears to be a poor hunch (left for someone else to verify for sure).

Fig. 16 shows the case corresponding to Fig. 15 except for just the first 10 estimates of a noisy signal (noise uniform of amplitude 0.01). This is the red circles with the average being the red line, which is not that far from the correct answer. The average is in fact better than any of the red circle individual calculations (pretty much as expected). Also shown are the calculations of **a** using equation (19), the blue stars, along with the blue-line average of the stars. This is not as good as the simple ratio. [Note that z is not simply y^2/x or the two would be identical. Instead z is noisy.] Different runs of the program will of course vary quite a bit.

SUMMARY

Fond as we are of Fourier analysis, there are objects that do not yield easily to standard methods, even though their nature is not complicated. In the case of linear ramps, the simple component (a linear function of t) is simply not part of the usual Fourier basis. Likewise an exponential ramp is not conveniently included in a Fourier model. Both should be understood as they are often included as information-bearing components of "signals" of interest. We have shown how these two types of ramps can be manipulated with and without a Fourier methodology, and how noise can be accommodated to some degree in the analysis.

REFERENCES

[1] “Fourier Map”, Electronotes Application Note No. 410, May 6, 2014
http://electronotes.netfirms.com/AN410.pdf

[2] “A Review of Fourier Methods in Signal Processing and Musical Engineering”, ***Electronotes***, Vol. 15, Numbers 155-160 (Special Issue D), Nov. 1983-April 1984, pp 3-52

[3] “Calculating the DFT and the Fourier Series: Each with the Other”, ***Electronotes***, Vol. 19, No. 188, February 1997

[4] There are many excellent books on wavelets. A few give an introductory view that is useful but do not really provide enough of the mathematics. Others resemble math books. This one is not a bad middle ground: C. Sidney Burrus and Ramesh A. Gopinath, ***Introduction to Wavelets and Wavelet Transforms: A Primer,*** Prentice-Hall 1997

[5a] “Polynomial Fitting for Sample-Rate Changing at Rational and Irrational Frequency Ratios”, Electronotes Application Note AN-317, Jan 1992.
http://electronotes.netfirms.com/AN317.PDF

[5b] “An Intuitive Approach to Polyphase Rate Changing” , ***Electronotes***, Vol. 21, No. 203, Jan. 2004 http://electronotes.netfirms.com/EN203.pdf

[5c] “Sample-Rate Changing by Polyphase On Demand”, ***Electronotes,*** Vol. 22, No. 205 March 2008 (short easy presentation pages 14-16).
http://electronotes.netfirms.com/EN205.pdf

[6] “Prediction, Deconvolution, and System Idetification; Part 1: Undriven Systems”, ***Electronotes***, Vol. 17, No. 179, June 1992; (Short Presentation) “Reader’s Question (Recovering a Sineave from Samples”, ***Electronotes***, Vol. 21, No. 204, August 2004
http://electronotes.netfirms.com/EN204.pdf

[7] “Resonators and Friends - Prony with Noise”, ***Electronotes***, Volume 23, Number 221 March 2014 http://electronotes.netfirms.com/EN221.pdf

[8] “Tools for Investigating Kautz Functions”, ***Electronotes***, Volume 22, Number 207 December 2011 http://electronotes.netfirms.com/EN207.pdf

[9] “Practical Aspects of Obtaining Spectra from FFT’s”, ***Electronotes***, Vol. 18, No. 181, April 1993

[10] "A White Noise Curiosity," ***Electronotes***, Volume 22, Number 208 January 2012. http://electronotes.netfirms.com/EN208.pdf ; More Concerning Non-Flat Random FFT", Electronotes Application Note No. 416, Nov 7, 2014 http://electronotes.netfirms.com/AN416.pdf

[11] "Time Domain Least Squared Low-Pass" Electronotes Application Note No. 318 February 1992 http://electronotes.netfirms.com/AN318.PDF

[12] "Basic Elements of Digital Signal Processing", ***Electronotes***, Vol. 20, No. 198, June 2001, pp12-21 http://electronotes.netfirms.com/EN198.pdf

[13a] "Resonators and Friends – Prony with Noise", ***Electronotes***, Vol. 23, No 221, March 2014 http://electronotes.netfirms.com/EN221.pdf

[13b] "Savitzky-Golay Smoothing", Electronotes Application Note AN-404, Feb. 13, 2014 http://electronotes.netfirms.com/AN404.pdf

[13c] "Examples of Inverting Smoothers", Electronotes Application Note AN-408, April 5, 2014 http://electronotes.netfirms.com/AN408.pdf

[13d] "DTFT and Fourier Series: N Equations in N Unknowns", Electronotes Application Note AN-411, May 20, 2014 http://electronotes.netfirms.com/AN411.pdf

[14] "Time Domain Weighted Least Squares", Electronotes Application Note No. 417 Nov 22, 2014 http://electronotes.netfirms.com/AN417.pdf

[15] Jackson, L.B., "Filter Design by Modeling," Chapter 10 of ***Digital Filters and Signal Processing*** (2^{nd} Ed), Kluwer (1989). See pp 249-252 for concise and readable presentation (couple of obvious typos).

[16] "Looking Again at the RC Low-Pass," ***Electronotes***, Volume 22, Number 210 May 2012 http://electronotes.netfirms.com/EN210.pdf ; "Exponential Decay – Continuous and Discrete Impulse Response", Electronotes Application Note No. 423, April 27, 2015 http://electronotes.netfirms.com/AN423.pdf

Matlab files online at Electronotes site as http://electronotes.netfirms.com/EN226MFiles.txt