ELECTRONOTES
1016 Hanshaw Road
Ithaca, NY 14850

APPLICATION NOTE NO. 411

May 20, 2014

DTFT AND FOURIER SERIES: N EQUATIONS IN N UNKNOWNS

We have long noted that the so-called DTFT (Discrete Time Fourier Transform) is essentially the time/frequency dual of the Fourier Series (FS):

FOURIER SERIES

$$c(k) = \left(\frac{1}{P}\right) \int_{-P/2}^{P/2} f(t) e^{-\frac{2\pi jkt}{P}} dt \qquad (1a)$$

$$f(t) = \sum_{k=-\infty}^{\infty} c(k) e^{2\pi jkt/P} \qquad (1b)$$

DISCRETE-TIME FOURIER TRANSFORM

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{-jn\omega T} \qquad (2a)$$

$$x(n) = \left(\frac{1}{\omega_s}\right) \int_{-\omega_s/2}^{\omega_s/2} X(\omega) e^{jn\omega T} \, d\omega \qquad (2b)$$

Here P is the period of the time-domain periodic waveform and $\omega_s = 2\pi f_s = 2\pi/T$ where $T=1/f_s$ is the sampling time, f_s being the sampling frequency. It is somewhat traditional to assume that T=1 (thus f_s=1, and $\omega_s = 2\pi$). In this case, equations (2) would become:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{-jn\omega} \qquad (3a)$$

$$x(n) = \left(\frac{1}{2\pi}\right) \int_{-\pi}^{\pi} X(\omega) e^{jn\omega} \, d\omega \qquad (3b)$$

It is useful as well to consider the case where X = H, a digital filter transfer function:

$$H(\omega) = \sum_{n=-\infty}^{\infty} h(n)e^{-jn\omega} \qquad (4a)$$

$$h(n) = \left(\frac{1}{2\pi}\right) \int_{-\pi}^{\pi} H(\omega)e^{jn\omega}\, d\omega \qquad (4b)$$

in which case equation (4a) is a means of computing the frequency response H(ω) based on the impulse response h(n), while equation (4b) is the most elementary method of designing a filter's impulse response h(n) from a desired H(ω). Indeed, persons familiar with digital filters probably understand the DTFT mainly in terms of filter responses H(ω) and h(n).

One final point is when we have a finite length x(n), from n=0 to N-1, and we evaluate H(ω) at N equally spaced frequencies $\omega_k = 2\pi k/N$. Thus:

$$X(k) = X(\omega_k) = \sum_{n=0}^{N-1} x(n)e^{-j(\frac{2\pi}{N})nk} \qquad (5a)$$

with inverse:

$$x(n) = (1/N) \sum_{k=0}^{N-1} X(k)e^{j(\frac{2\pi}{N})nk} \qquad (5b)$$

which is the Discrete Fourier Transform (DFT) almost always computed by the Fast Fourier Transform algorithm (exact answer – not approximation). Do not confuse the DFT and the DTFT.

Directing our attention back to the DTFT of equations (2) or (3), we note that everything we know about the FS applies if we are able to invert our thinking with regard to time and frequency. This, once done, is extremely useful. So what do we know about the FS?

TOY EXAMPLE OF FOURIER SERIES

Here we will be looking at a toy example of a FS, but not the usual type of toy examples where we might be calculating the FS of a square or a pulse. The first observation is that there are relatively few traditional examples where we can calculate a FS. Most books might list a half dozen, while some reference works might have a hundred. (But try to find a train of semi-circles!). But it is after all just a matter of using equation (1a) – if we know the functional form of f(t) and can do the integration.

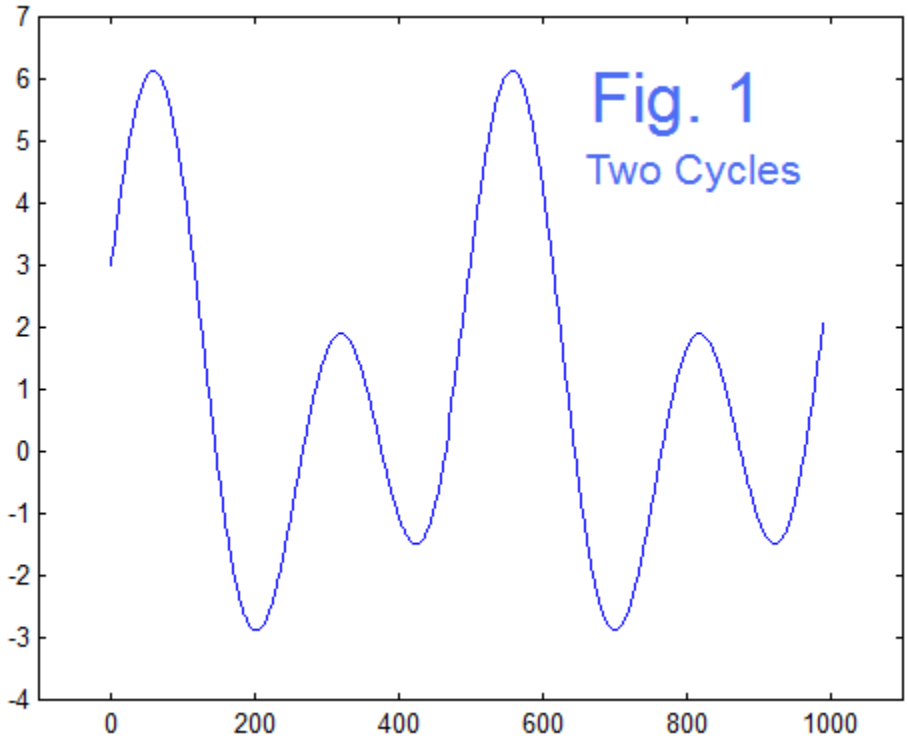
Sooner or later a student will observe that the Fourier Series generally listed are not bandlimited! That is, k runs from -∞ to +∞. If we then ask what sort of periodic signal would have a FS that is bandlimited, we arrive at the circular conclusion that it is a signal composed of a finite sum of sinusoidal waveforms in harmonic relationship. We can easily write down an example – here using a trig form alternate of equation (1b).

$$f(t) = a_0 + a_1 \text{Cos}(\omega t) + b_1 \text{Sin}(\omega t) + a_2 \text{Cos}(2\omega t) + b_2 \text{Sin}(2\omega t) \qquad (6)$$

That is, we have a constant, a fundamental, and a second harmonic with the phase determined by choosing a Sine and a Cosine contribution for each frequency. Finding the FS is just "by inspection" if there are actual numbers for the a's and b's in equation (6). But suppose we just had f(t) as data – at least a very dense set of samples so that numerical integration would be very accurate. This is nothing more than the usual "inner product" calculation and would work (we will illustrate this in a bit).

But there is an easier way. If we know that f(t) is of the form of equation (6), we just need five data points: $f(t_1)$, $f(t_2)$, $f(t_3)$, $f(t_4)$, and $f(t_5)$. This permits us to solve for a_0, a_1, b_1, a_2, and b_2. This is far less calculation than the inner products, and we don't need very much of the data. **This is pretty much "Part 2" of Prony's method [1].** Recall that Part 1 was "find the poles" and Part 2 was "apply initial conditions". Here the assumption that the correct expression is equation (6) is a declaration of having the poles.

As an example, suppose a_0=1, a_1=2, b_1=1, a_2=0, and b_2=3. We can then plot the function f(t) which is shown with a fundamental period of 500, so the plot of Fig. 1 shows 2 full cycles (to keep the periodicity in mind). We want to recover the a's and b's from 500 samples of data (or fewer!). Some Matlab code and the resulting printout is on page 5.



```
% Generate Test Signal
t = 0:1000;
x = 1 + 2*cos(2*pi*t/500)+ sin(2*pi*t/500) + 3*sin(2*2*pi*t/500);
figure(1)
plot([0:1000],x)
axis([-100 1100 -4 7])
figure(1)

% Here are the possible basis functions
cos0=cos(0*2*pi*[0:499]/500);
cos1=cos(1*2*pi*[0:499]/500);
cos2=cos(2*2*pi*[0:499]/500);
sin1=sin(1*2*pi*[0:499]/500);
sin2=sin(2*2*pi*[0:499]/500);
%
% Do the inner products
a0=sum(x(1:500).*cos0)/500
a1=sum(x(1:500).*cos1)/250
b1=sum(x(1:500).*sin1)/250
a2=sum(x(1:500).*cos2)/250
b2=sum(x(1:500).*sin2)/250

% Or just do 5 equations in 5 unknowns
%  Choose times 0, 1, 2, 7, and 77 arbitrarily
m=[1 1                0              1               0               ;
   1 cos(2*pi/500)    sin(2*pi/500)    cos(2*pi/250)    sin(2*pi/250)   ;
   1 cos(2*2*pi/500)  sin(2*2*pi/500)  cos(2*2*pi/250)  sin(2*2*pi/250) ;
   1 cos(7*2*pi/500)  sin(7*2*pi/500)  cos(7*2*pi/250)  sin(7*2*pi/250) ;
   1 cos(77*2*pi/500) sin(77*2*pi/500) cos(77*2*pi/250) sin(77*2*pi/250)]
%
c=inv(m)*[x(1) x(2) x(3) x(8) x(78)]'
```

Which runs to give:

```
a0 = 1.0000
a1 = 2.0000
b1 = 1.0000
a2 = -6.4126e-016
b2 = 3.0000


m =    1.0000    1.0000         0    1.0000         0
       1.0000    0.9999    0.0126    0.9997    0.0251
       1.0000    0.9997    0.0251    0.9987    0.0502
       1.0000    0.9961    0.0879    0.9846    0.1750
       1.0000    0.5673    0.8235   -0.3564    0.9343


c =

    1.0000
    2.0000
    1.0000
    0.0000
    3.0000
```

The example shows success with both the inner product approach and the linear equations approach, with an immense reduction in calculations for the latter approach. The lines for the generation or the basis sinusoidal waveforms are standard, as are the point-by-point multiplies and sums of the inner (dot) products.

To be completely clear about the alternative N-equations in N-unknowns, we can write down the five equations from equation (6):

$$
\begin{aligned}
f(t_1) &= a_0 + a_1\,\mathrm{Cos}(\omega t_1) + b_1\,\mathrm{Sin}(\omega t_1) + a_2\,\mathrm{Cos}(2\omega t_1) + b_2\,\mathrm{Sin}(2\omega t_1) \\
f(t_2) &= a_0 + a_1\,\mathrm{Cos}(\omega t_2) + b_1\,\mathrm{Sin}(\omega t_2) + a_2\,\mathrm{Cos}(2\omega t_2) + b_2\,\mathrm{Sin}(2\omega t_2) \\
f(t_3) &= a_0 + a_1\,\mathrm{Cos}(\omega t_3) + b_1\,\mathrm{Sin}(\omega t_3) + a_2\,\mathrm{Cos}(2\omega t_3) + b_2\,\mathrm{Sin}(2\omega t_3) \qquad (7) \\
f(t_4) &= a_0 + a_1\,\mathrm{Cos}(\omega t_4) + b_1\,\mathrm{Sin}(\omega t_4) + a_2\,\mathrm{Cos}(2\omega t_4) + b_2\,\mathrm{Sin}(2\omega t_4) \\
f(t_5) &= a_0 + a_1\,\mathrm{Cos}(\omega t_5) + b_1\,\mathrm{Sin}(\omega t_5) + a_2\,\mathrm{Cos}(2\omega t_5) + b_2\,\mathrm{Sin}(2\omega t_5)
\end{aligned}
$$

Note here that ω is known, being the fundamental frequency, and we choose the five values of time fairly arbitrarily. Thus the Cos and Sin terms are all just constants. The five samples $f(t_1)$ to $f(t_5)$ are then numbers which the function f(t) is expected to obtain at the five times. Hence just five linear equations in five unknowns. In the program shown, the times selected (on the possible interval of 0 to 500) were 0, 1, 2, 7, and 77. These do not have to be integers as long as we calculate the test f(t) value at the corresponding times. For example, t_5 can be changed from t=77 to $t=77\sqrt{2}$ = 108.8944 and substitute the value of x at the new t. The five coefficients all come out the same.

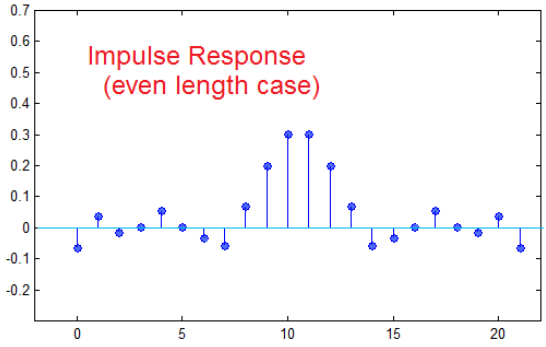
So what we are doing here is not so much FS as it is Prony's Method. In our discussions of Prony's Method we often emphasized that we were NOT doing N-equations in N unknowns until after the transformation to linear difference equations. Here we are eliminating the first part by declaring the frequencies. Surprisingly, when we look at the "dual" problem, the DTFT, we may have a more familiar example with more insight.

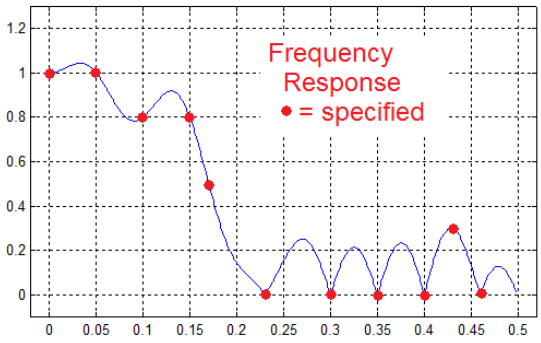
THE DTFT – BACK TO GENEALIZED FREQUENCY SAMPLING

Here is where things should get worse. We need to take something very familiar, the FS, and reverse the roles of time and frequency to arrive at the DTFT. Flipping all the switches is notoriously difficult. The saving thing here will be that we will end up with something we are comfortable with, if not very familiar with – the Generalized Frequency Sampling (GFS) method of digital filter design [2, 4].

The first of these is easy. In the FS we have, underling the whole thing, a continuous periodic function of time, and thus in the DTFT we have a continuous, periodic function of frequency (the frequency response in the case of a filter) [3]. The "output" element in the FS was the coefficients, a discrete equally spaced set of numbers representing spectral strengths. This was a finite (bandlimited) set in our example here. In the DTFT, the output is a discrete set of numbers in time, equally spaced, a finite length set representing the impulse response of the digital filter. Exactly parallel in the dual. For the FS here, the "input" was the known (chosen) samples, x(t), for discrete values of t, not necessarily equally spaced. The idea was that the FS sum is to be forced to go through these input samples. In the DTFT we would have a set of frequency samples, not necessarily equally spaced. The notion then becomes that the frequency response of the filter goes through these "target" samples. We have seen this as the case of "Generalized Frequency Sampling". [In the case of equal spacing, we just had "Ordinary" frequency sampling and used the DFT←→Inverse DFT]. Essentially what we did above with the FS itself we already looked at for digital filter design, IN THE DTFT DUAL, many years ago [4].

The app note AN-337 [4], "A General Review of Frequency Sampling Design" contains many derivations, Matlab programs, and examples. Here we will just show Fig. 2 produced by the program ***amp.m*** (code below, dated 1996 and 2004) which shows a filter design that is not offered as a good filter, but which illustrates a DTFT with unequally spaced samples. Thus it is the counterpart to the FS example above.






```
[ho he]=amp([0 0.05 0.1 0.15 0.17 0.23 0.3 0.35 0.4 0.43 0.46],
[1 1 0.8 0.8 0.5 0 0 0 0 0.3 0] )
```

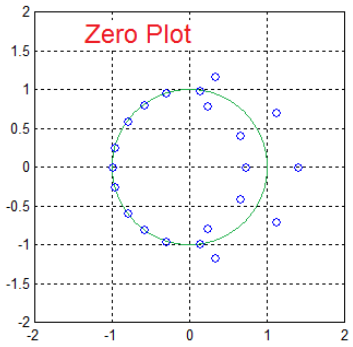


Fig. 2

HERE IS CODE FOR amp.m

```
function [ho,he]=amp(f,a)
%
%   [ho,he] = amp(f,a)
%        f = frequency vector on 0 to fs/2 (fs/2 = 1/2)
%        a = corresponding amplitudes
%
%   Does both even and odd length filters based on amplitude
%
%   B. Hutchins                Jan 1996, Spring 2004

N=length(f);
w=2*pi*f;


% Matrix for even length
  for n=1:N
     for k=1:N
          ME(k,n)=cos((n-1/2)*w(k));
     end
  end


% Matrix for odd length
  for n=1:N
     for k=1:N
          MO(k,n)=cos((n-1)*w(k));
     end
  end

aae=inv(ME)*a.';
aao=inv(MO)*a.';


% even length case
  for n=1:N
     he(n)=aae(n)/2;
  end
  he=[he(N:-1:1),he(1:N)];


% odd length case
  for n=2:N
       ho(n)=aao(n)/2;
  end
  ho=[ho(N:-1:2),aao(1),ho(2:N)];
```

```
% plot odd length result
k=0:(length(ho)-1);
figure(1);
subplot(221)
stem(k,ho);
axis([-2 2*N -.3 .7]);
MHO=abs(freqz(ho,1,500));
MHO=MHO/MHO(1);
subplot(222);plot([0:.001:.499],MHO)
axis([-.02 .52 -.1 1.3]);
grid
subplot(224)
pzplot1(ho,1)

% plot even length result
k=0:(length(he)-1);
figure(2);
subplot(221)
stem(k,he);
axis([-2 2*N -.3 .7]);
MHE=abs(freqz(he,1,500));
MHE=MHE/MHE(1);
subplot(222);plot([0:.001:.499],MHE)
axis([-.02 .52 -.1 1.3]);
grid
subplot(224)
pzplot1(he,1)
```

AND HERE IS CODE FOR PZPLOT1

```
function [z,p]=pzplot(b,a)
%  function [z,p]=pzplot(b,a)
%  Plot the poles and zeros of a transfer function in z-plane
%    z are zeros of numerator polynomial b
%    p are poles of denominator polynomial a
%    z (plotted as o) and p (plotted as x) in the z-plane
%    multiple-order singularities are only indicated as single-order
%  B. Hutchins, EE425, Cornell Univ. Fall 1993

%  find roots
z=roots(b);
p=roots(a);

% find max for plotting
pmax=max([abs(real(z)); abs(real(p)); abs(imag(z)); abs(imag(p)) ]);
sc=ceil(pmax);
```

```
% begin plot
% prepare circle
n=0:500;
r=exp(j*2*pi*n/500);
axis([-sc, sc, -sc, sc]);
plot(r,'g')
grid
hold on
% plot real and imag, not just z itself or else the real part
% may be plotted as verticle if singularities are not complex
plot(real(z),imag(z),'o')
plot(real(p),imag(p),'x')

hold off
axis('equal')
axis([-sc sc -sc sc])
```

****

The discussion in AN-337 [4] goes somewhat further. In the ***amp.m*** program (and in another one) we used unequal spacing. In AN-337 we also went in the direction of equal spacing but an excess of samples. Below we will address the use of excess samples AND unequal spacing, in the context of the original FS. (Because we were interested in input to the design programs that automatically generated the samples, when we went to excess samples, any consideration of the freedom to choose sample frequencies arbitrarily seemed of little practical use.)

Accordingly we considered M equations (M samples) in N unknowns (length-N impulse response) where M was greater than N. This was very useful in practical cases. For example, a cutoff frequency could be specified with considerable precision. Or we could describe a magnitude function in great detail. Because the excess equations could not be solved exactly, but only by least squares (pseudo-inverse) we no longer had the eventual response going through all the samples, but none the less approximating all of them well. The program ***fsamplms.m***, and a weighted version, were offered in AN-337.

EXCESS EQUATIONS AT ARBITRARY TIMES – BACK TO FOURIER SERIES

At this point we will extend the discussion as advertized above: the case of over-determined equations and arbitrary times. There is no reason this should not work. At this point we write a program that sets up and solves a set of equations for specified times, such as in equation (7) and with the same five FS components, except we can have

a lot more equations. We then run this for a variety of conditions. We could have five consecutive times (like 0,1,2,3,4) or five times like in the example (0,1,2,7,77) or we can include irrational times like (0,1,2,7,77$\sqrt{2}$). All these choices give us the perfect result of:

$$[a_0, a_1, b_1, a_2, b_2] = [1 \;\; 2 \;\; 1 \;\; 0 \;\; 3] \tag{8}$$

We can then go to the over-determined cases. Here we will use the Matlab pseudo-inverse ***pinv*** instead of the square matrix inverse. For example, we might have 50 times (0,1,2,…49). Or we would have irrational times. Or we could have random times (not even ordered). All of these give the correct result as in equation (8). The program ***fstest3.m*** is here:

```
************************************************

% fstest3

% Test Signal Plotted for Reference
t1=0:1000;
x1= 1 + 2*cos(2*pi*t1/500)+ sin(2*pi*t1/500) + 3*sin(2*pi*t1/250);
figure(1)
plot([0:1000],x1)
axis([-100 1100 -4 7])
figure(1)

nt=50               % number of equations
t=[];
% choose times
%t=[0 1 2 3 4]
% t=0:nt-1
t=500*rand(1,nt)

% Form Matrix for Equations
M = zeros(nt,5);
for k=1:nt
   M(k,1) = 1;
   M(k,2) = cos(2*pi*t(k)/500);
   M(k,3) = sin(2*pi*t(k)/500);
   M(k,4) = cos(2*2*pi*t(k)/500);
   M(k,5) = sin(2*2*pi*t(k)/500);
end
M
```

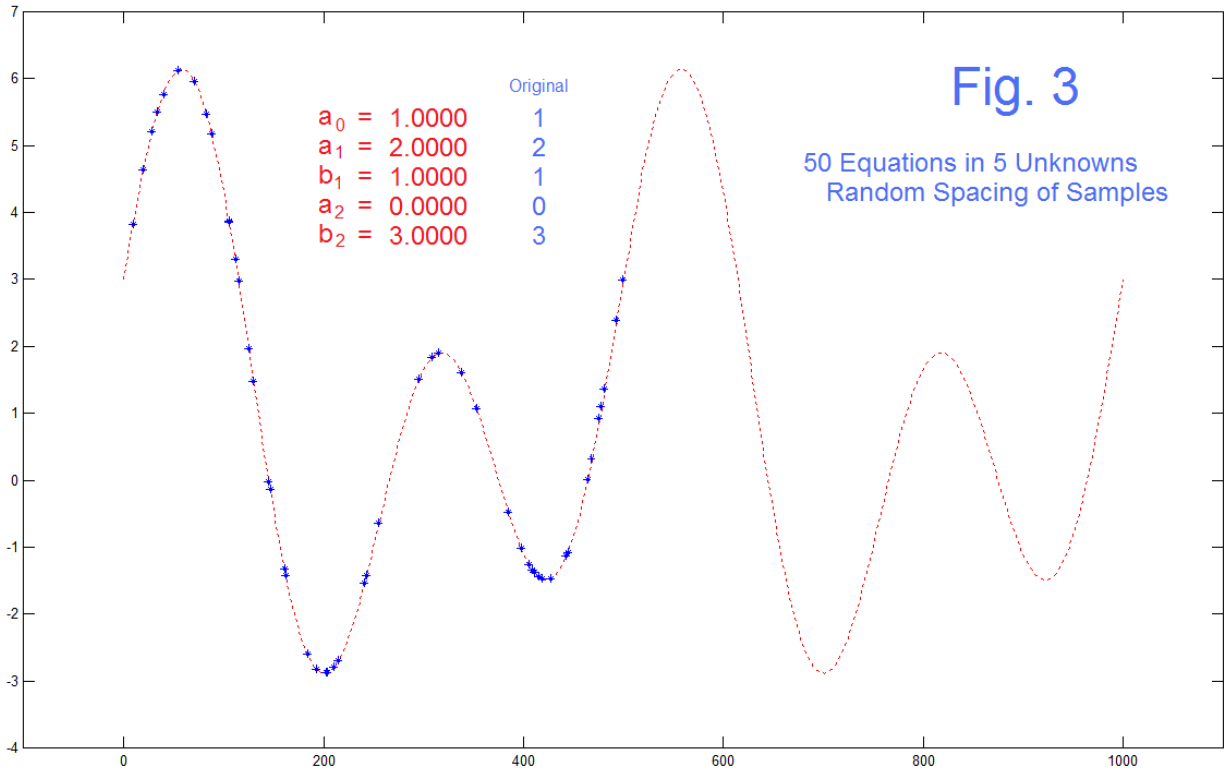
```
xx=[];
for k=1:nt
   xx(k)= 1 + 2*cos(2*pi*t(k)/500)+ sin(2*pi*t(k)/500) + 3*sin(2*pi*t(k)/250);
end

% Add Noise
an=0;
xx=xx+an*randn(1,nt);

% Show Data Used
figure(2)
plot(t1,x1,'r:')
hold on
plot(t,xx,'*') plot(t,xx,'*')
axis([-100 1100 -4 7])
hold off

% Pseudo-Inverse
c=pinv(M)*xx'
```

****

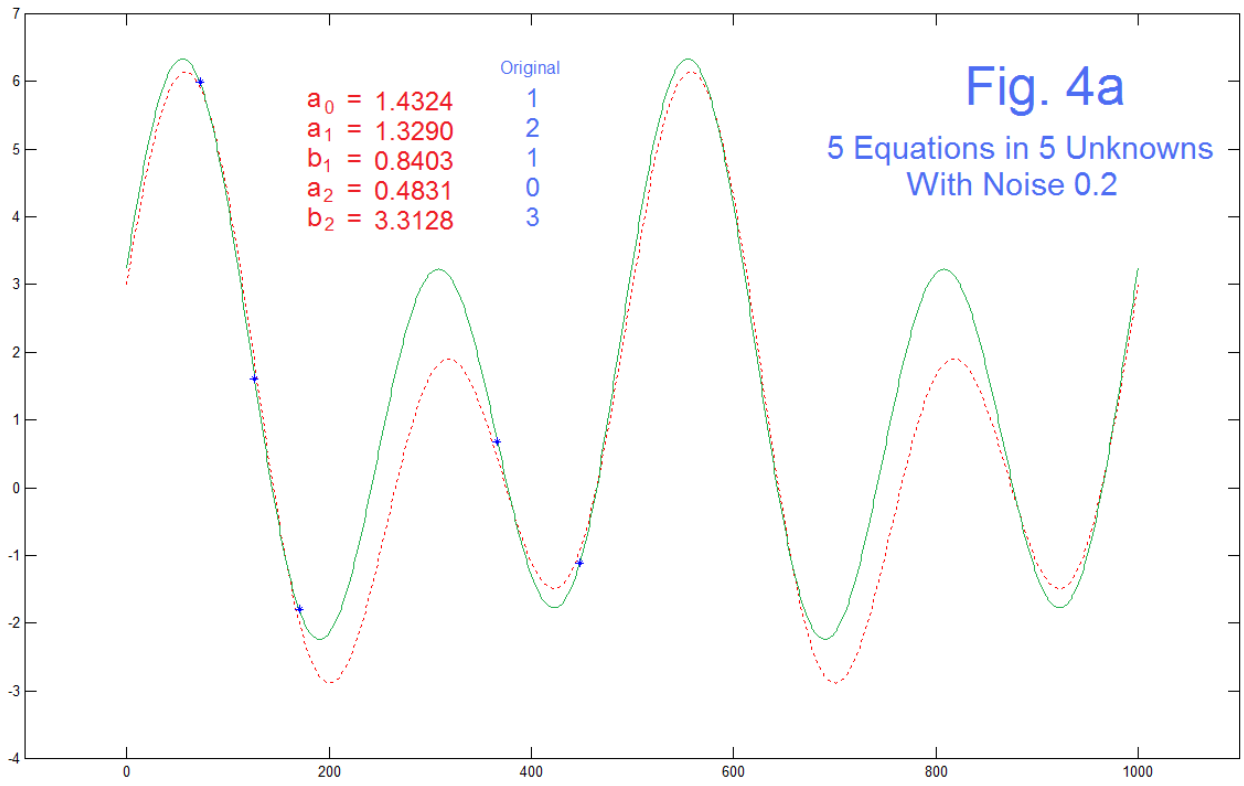


In Fig. 3, we have run the program to give 50 randomly timed samples, so have 50 equations in 5 unknowns. The 50 samples are the blue stars, and the dashed red line is simply the original signal from which these times were taken. So the finding is that we get the coefficients back, as in equation (8), perfectly. The new thing here, beyond the corresponding ***fsamplms.m*** (over-determined), is the uneven spacing of the samples.

AND WHAT HAPPENS WITH NOISE?

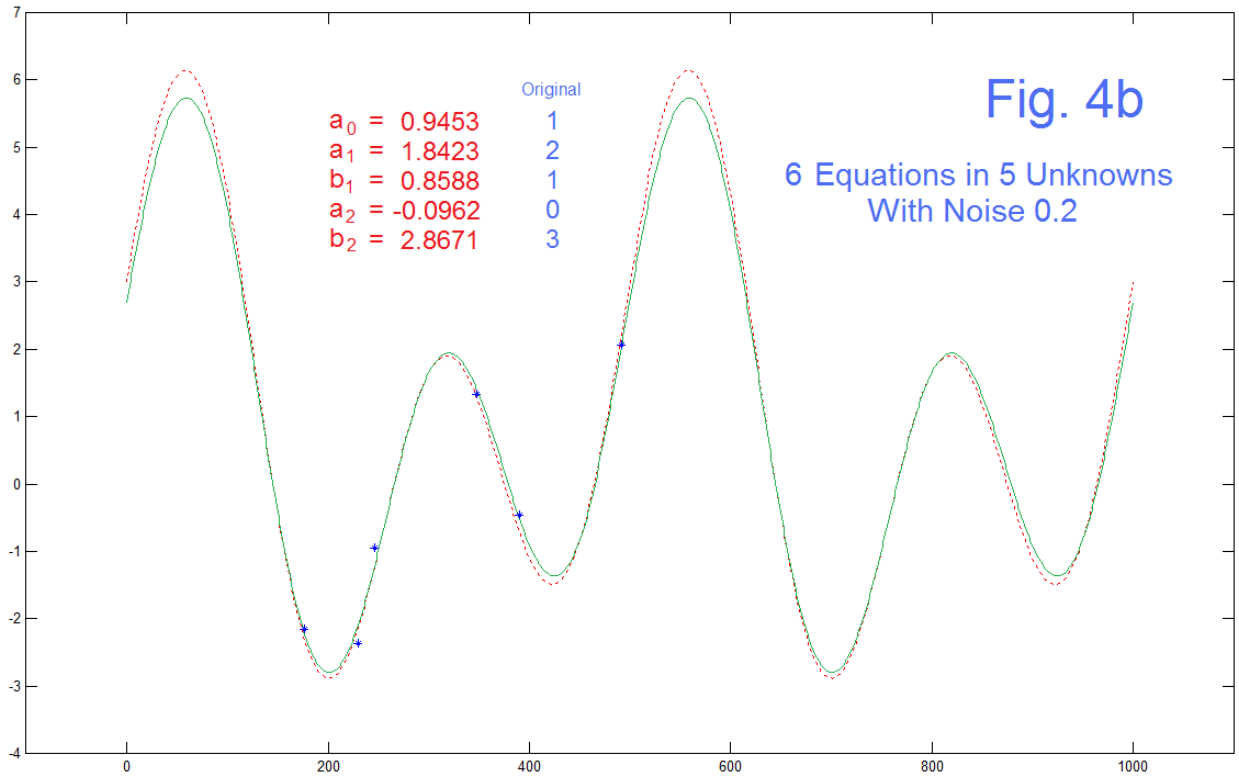
The final wrinkle here is to add noise to the target sample values. Here we are looking at a case similar to what we would expect if we had a large number of noisy data point but had some reason to suppose that they should fit a few sinusoidal components, for which we had reason to suspect a particular fundamental frequency. Here we will look at four cases: (1) the case of just 5 noisy equations, (2) the case of 6 noisy equations, and (3,4) cases (two) of many noisy equations. We shall be interested in how well we can calculate the underlying FS coefficients, and how well the resulting curve fits the actual data.

Here we use the program ***fstest3*** above where we now make the noise amplitude equal to 0.2 instead of 0. This means that the answer of equation (8) is not obtained. We can use the new (calculated from noisy samples) coefficients and plot that series, and this we added to the program and plot in green.

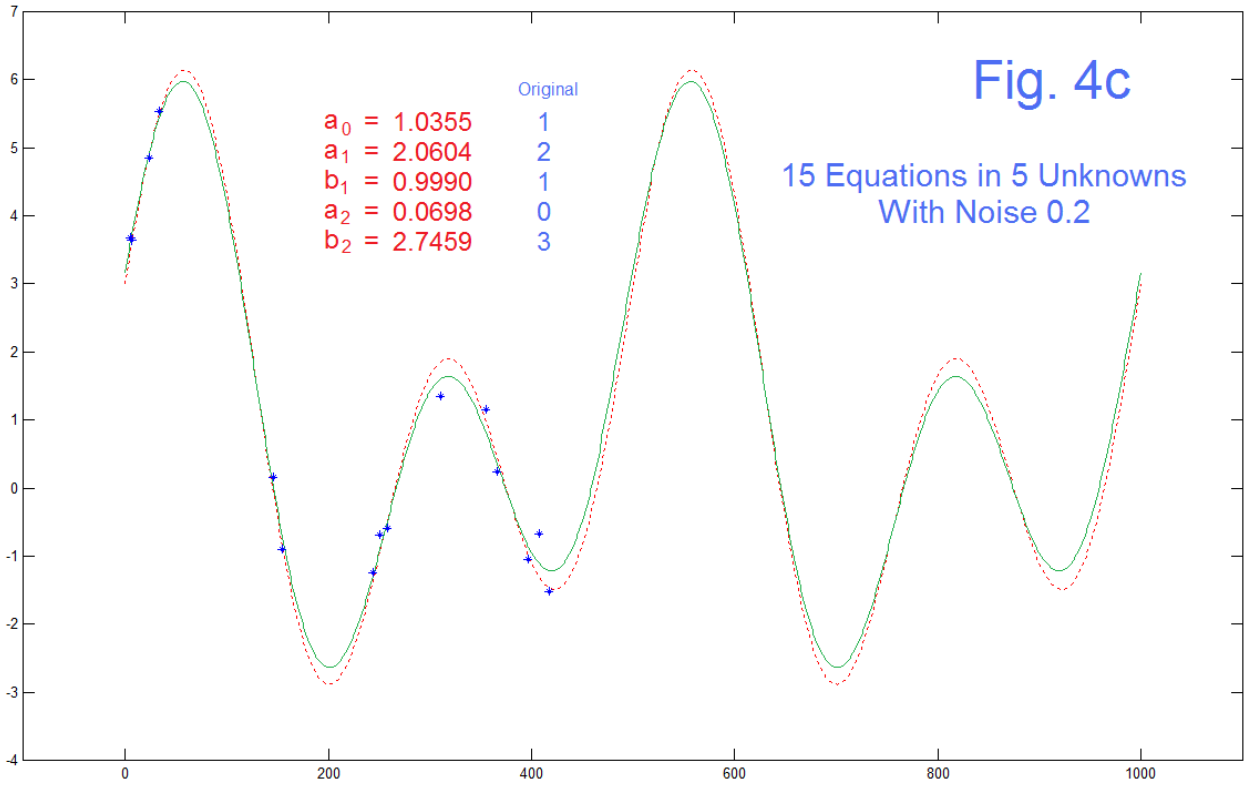


In the case of 5 samples (Fig. 4a), the green curve goes exactly through the noisy point. Note that the calculated coefficients are of course no longer (1 2 1 0 3), as they should not be. Further we have seen in our runs examples where the green curve was all over the place (and running off scale), so results are not always (not usually) even this good. But they always go through the original five noisy points. This example is 5 equations in 5 unknowns, so we expect the green curve to go through the five blue stars.

Fig. 5b shows the change from 5 to 6 points, which seems a minor change, but which presents a fundamental change. This example happens to be what seems to be a better fit to the original red curve. It is in general true that the more noisy points we add, the better the fit to the red curve. This varies according to the particular noise for a particular run (again – some curves run all over the plot). And the fit to the red is not the point here (we will see this later in Fig. 4c and Fig. 4d). Here, the point is that the green curve no longer goes through the 6 blue stars. We have 6 equations and only 5 unknowns. Note as well that the coefficients are changed and not well approximating the original red case.



While the change from 5 equations in 5 unknowns to 6 equations in 5 unknowns is a fundamental departure into over-determined land, often times the most impressive examples of the least squares and "pseudo inverse" involve many many excess equations, not just one extra (from 5 to 6, as in Fig. 4b) as we can see an improvement occurring with many extra equations. That is, not only does the least squares approach permit us to solve the over-determined set, but shows useful results.


Original
a_0 = 1.0355 1
a_1 = 2.0604 2
b_1 = 0.9990 1
a_2 = 0.0698 0
b_2 = 2.7459 3
Fig. 4c
15 Equations in 5 Unknowns
With Noise 0.2

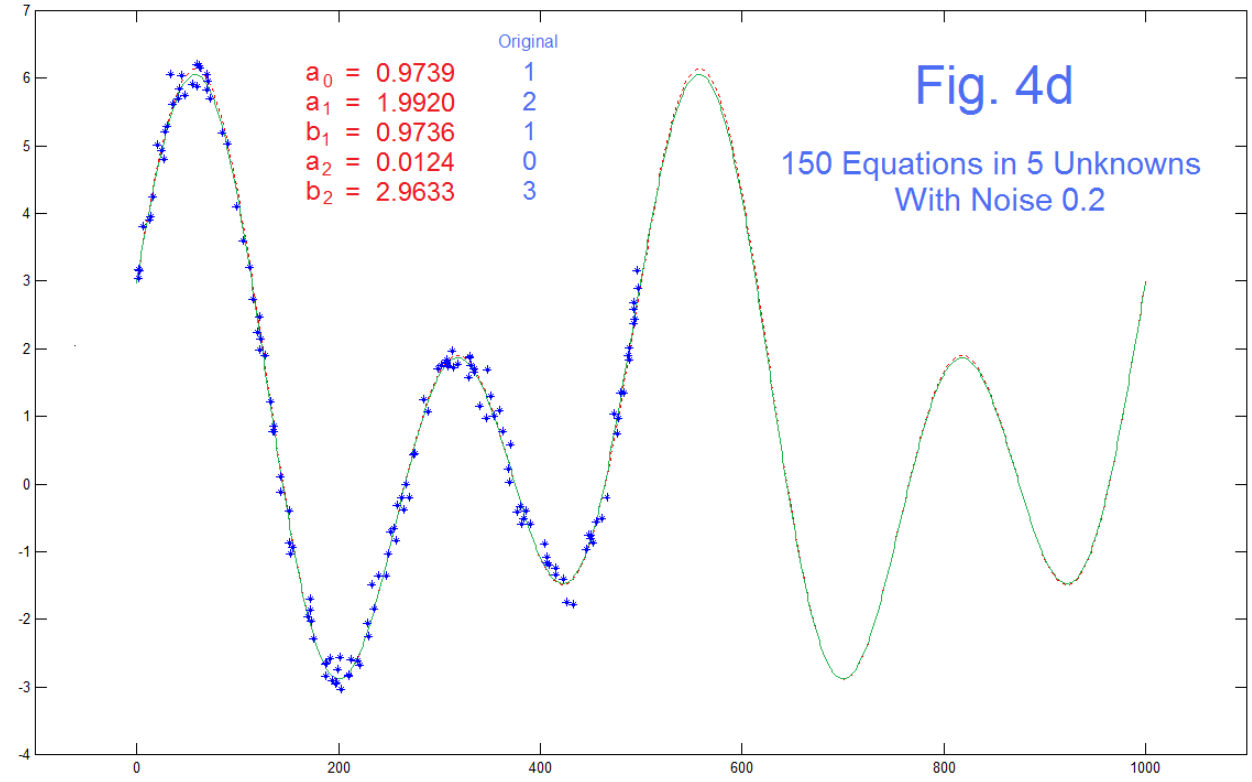
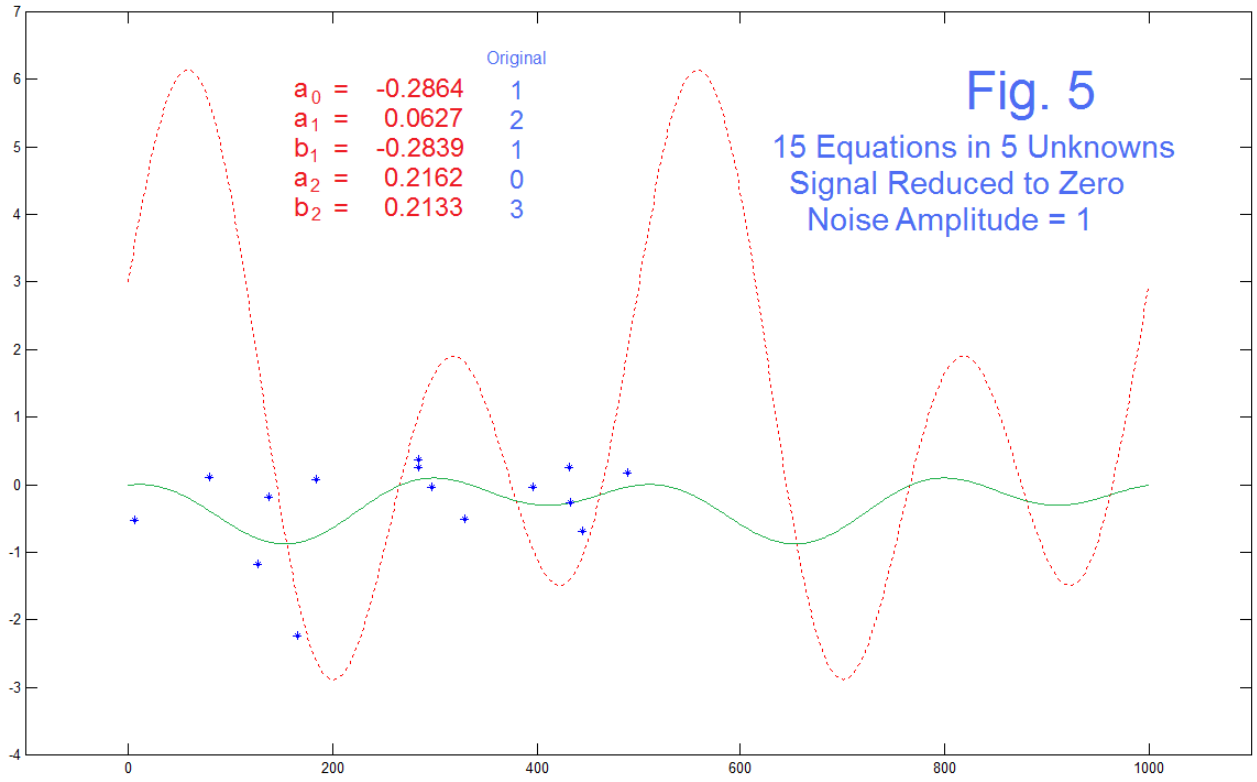

Original
a_0 = 0.9739 1
a_1 = 1.9920 2
b_1 = 0.9736 1
a_2 = 0.0124 0
b_2 = 2.9633 3
Fig. 4d
150 Equations in 5 Unknowns
With Noise 0.2

Fig. 4c shows the case of 15 noisy samples (three times the original number) while Fig. 4d shows 150 noisy samples. At this point, we see that as expected, the green curve does not in general pass through the 15 blue stars, although the coefficients are starting to move closer to the coefficients of the original signal. This proceeds as in Fig. 4d where we have 150 noisy samples. Here the green curve pretty much overplots the dashed red curve. That is, the noise has sufficiently averaged out that the correct original coefficients have been obtained, and this can also be seen in the calculated coefficients as printed in the figures. With 150 points (or more) we have obtained almost the correct result, even with considerable noise.

The final test here is to look at the case where we have pure noise and no signal. Thus we make the noise amplitude 1 and multiply the original signal by 0, and this is shown in Fig. 5. As expected, by chance there is some relatively small frequency content at dc, the first harmonic, and the second harmonic, such that we see a small amplitude green curve that changes completely with every run. While we plot the red dashed curve in Fig. 5, for reference back, we expect none of it, and see none of it, except by chance.



SUMMARY

We tend to think of the FS equations as an essential pairing, while in fact each is individually valid. Equation (1b), the series sum, is rather simple in suggesting that a periodic function is a sum of sinusoidal waveforms (or complex exponentials). Once we decide on the fundamental and the bandwidth (how many terms) we then may have several paths to determining the coefficients. Here the lesser known path of solving linear equations has been explored. Along with the standard N equations in N unknowns, often thought of as equally spaced samples, we have looked at unequally spaced samples, an excess (perhaps a large excess of equations) to be solved in the least squares sense, and arbitrary spaced samples. We have found it very useful to relate this to the "frequency sampling" digital filter design methods which are really FS with the time domain an frequency domains reversed, as these are somewhat familiar and known to be useful. This is not to say that the ramifications are immediately obvious. These connections are well worth pursuing.

We have also brought in the often present complication of noisy data, and found that over-determined solutions are extremely useful. Finally, looking at this procedure as essentially the same thing as the second part of Prony's method seems quite revealing.

REFERENCES

[1] B. Hutchins, ‘‘Resonators and Friends - Prony with Noise" Electronotes Volume 23, Number 221 March 2014
http://electronotes.netfirms.com/EN221.pdf

[2] Electronotes Digital Filter Design Series, Electronotes EN#198, pg. 12…..
http://electronotes.netfirms.com/EN198.pdf

[3] B. Hutchins, "Fourier Map", Electronotes Application Note No. 410, May 6, 2014
http://electronotes.netfirms.com/AN410.pdf

[4] B. Hutchins, "A General Review of Frequency Sampling Design," Electronotes Application Note No. 337, April 1996
http://electronotes.netfirms.com/AN337.pdf