ELECTRONOTES
1016 Hanshaw Road
Ithaca, NY 14850

APPLICATION NOTE NO. 416

Nov 7, 2014

MORE CONCERNING NON-FLAT RANDOM FFT

INTRODUCTION

A "curiosity" that has probably long been "peripherally" noted but which apparently went unreported until 2004 [1] involved the non-flatness of the spectrum of what is supposed to be a white-noise signal. An explanation which seems to be correct was offered in ***Electronotes*** No. 208, in 2012 [2]. In essence, we showed that the points of a DFT (FFT) of a random signal constitute a random walk that are sometimes one dimensional (and Gaussian distributed) but more generally are Rayleigh distributed.

In the previous application note AN-415 [3] we offered some related material and remarked that we probably should make the distinction between the spectrum of a signal and its FFT. So here we have a good example of the common case where answering one question leads to a bunch more.

So what does it mean? We have found that the FFT gives a dip at the endpoints [at k=0, and at k=N/2 if this k be an integer (N is even)]. That is, the dip is at DC, and at half the sampling rate when N is even. The dip is to $2^{3/2}/\pi = 0.9003163$ below the flat value. What are the possibilities? (1) The FFT always discounts the ends. Not so. (2) A random white noise has endpoint dips – natural rejection. Contrary to definition, but need investigation for any so-called "random noise generator". (3) We confuse ourselves by considering the FFT of a signal to be its "spectrum".

A SIMPLE VIEWPOINT

Suppose we have a signal that passes through a filter (some system) and this is followed by an analysis device that lets us see the output spectrum. When we see features on the spectrum, we can't automatically tell from the output if they come from the signal or the system (or both of course).

When we measure a frequency response, we attach a sinusoidal oscillator (fixed amplitude) and sweep while watching the output level, and in this case quite reasonably attribute any peaks/dips to the filter. We take this frequency response to represent the output spectrum we would have seen if we had input a flat spectrum.

If we have an all-pass filter, we expect no amplitude features, and if we were to find any, we would suspect an error. This might be a bogus filter, or it could be an error in measurement (for example, AC coupling a scope input and expecting to see a flat response down to DC!).

In the case of measuring the spectrum of white noise using the FFT we have an output, and we suppose the input to be honest (flat) and the measuring tool, the FFT, to be honest as well. So – what is going on?

ROTATING A SPECTRUM OR THE FFT?

We know that the FFT (we use the general term FFT to mean the DFT as well) is discrete in frequency. All energy in a signal is forced to be represented in integer-indexed frequency bins (values of k). The FFT like all its Fourier/Laplace siblings [4] possesses a "modulation property". Here we pick up on the finding that the "dips" are due to the purely real phases at k=0 and k=N/2 (N even) and suppose that we could, perhaps, rotate things so as to avoid DC and half the sampling frequency by this rotation in frequency. Beginning at the beginning, we have the z-Transform:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} = \sum_{n=0}^{N-1} x(n)z^{-n} \qquad (1)$$

where the term on the right is for a finite, length N sequence. The corresponding DTFT is:

$$X\left(e^{j\omega}\right) = \sum_{n=0}^{N-1} x(n)e^{-jn\omega} \equiv X(\omega) \qquad (2)$$

which yields the DFT (FFT) as:

$$X\left(\omega = \frac{2\pi k}{N}\right) \equiv X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}nk} \qquad (3)$$

Equations (1), (2) and (3) are absolutely standard. The DTFT is the z-Transform on the unit circle and the DFT is the DTFT evaluated at frequencies ω=2πk/N. NOW, what if we evaluate (3) between the integers. More generally, let's offset the frequencies by same value ω_0.

$$X\left(\omega = \frac{2\pi k}{N} + \omega_0\right) = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}nk - jn\omega_0} \qquad (4)$$

which we can rewrite as:

$$X'(k) = \sum_{n=0}^{N-1} [x(n)\, e^{-jn\omega_0}]\, e^{-j\frac{2\pi}{N}nk} \qquad (5)$$

which is the FFT of the modulated sequence $[\, x(n)\, e^{-jn\omega_0}\,]$. Accordingly, our attempt to trap the FFT into evaluating at non-integers has failed. It has just told us this is equivalent to shifting the spectrum of the input sequence by ω_0. Fair enough!

Let's be sure we understand this in terms of the integers first. Suppose we have a length 20 sequence consisting of exactly four full cycles of a cosine: x(n) = cos(2π•4•n/20) for n=0 to 19 (Fig. 1, upper left). It's FFT has energy at k=4 and k = 20-4=16 (Fig. 1, upper right). This we can rotate by multiplying the time sequence, point by point, by the exponential:

$$x_s = e^{j2\bullet 2\pi\bullet n/20} \qquad n = 0,1 \ldots\ldots 19 \tag{6}$$

which has a single spike FFT at k=2 (Fig. 1 lower left). Taking the FFT of the product sequence, or convolving the FFT's of the components of the products, results in a shift as seen in Fig. 1, lower right, which shows the magnitude of the now complex FFT. Note that this is not a shifted cosine as the spectral energy is non-symmetric, at k=6 and k=18, shifted up by 2 from the cosine.

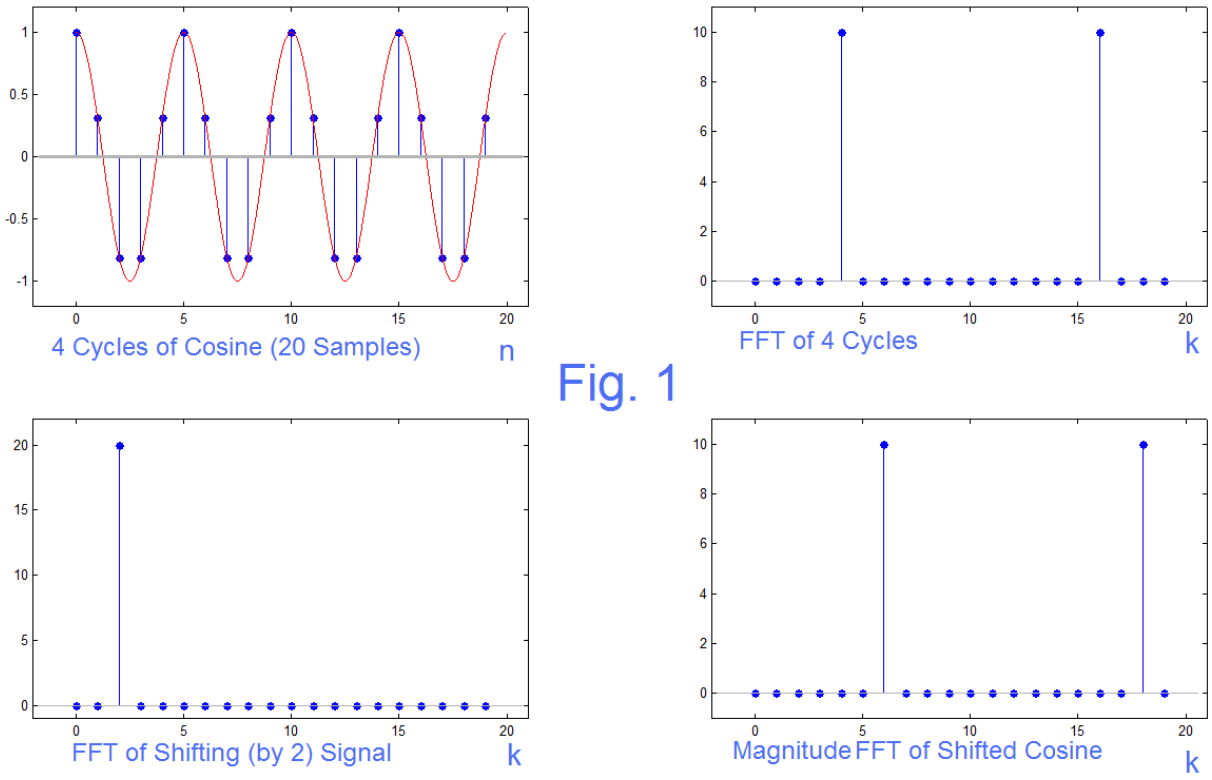


Fig. 1

We need to see what happens if we rotate the spectrum of a random signal in this manner, but first, in a step-by-step controlled manner, what happens if we shift the cosine sample by a non-integer? Accordingly we offer a single change to the setup that led to Fig. 1 – we make the shifting signal have a k index of 2.5 instead of 2.0.

$$x_s = e^{j2.5\bullet 2\pi\bullet n/20} \qquad n = 0,1 \ldots\ldots 19 \tag{7}$$

This means that the resulting shift of the cosine is up by 2.5, and the energy is now redistributed among available frequencies exactly as it is in the familiar "leakage" example common to FFT analysis [5] as shown in Fig. 2.

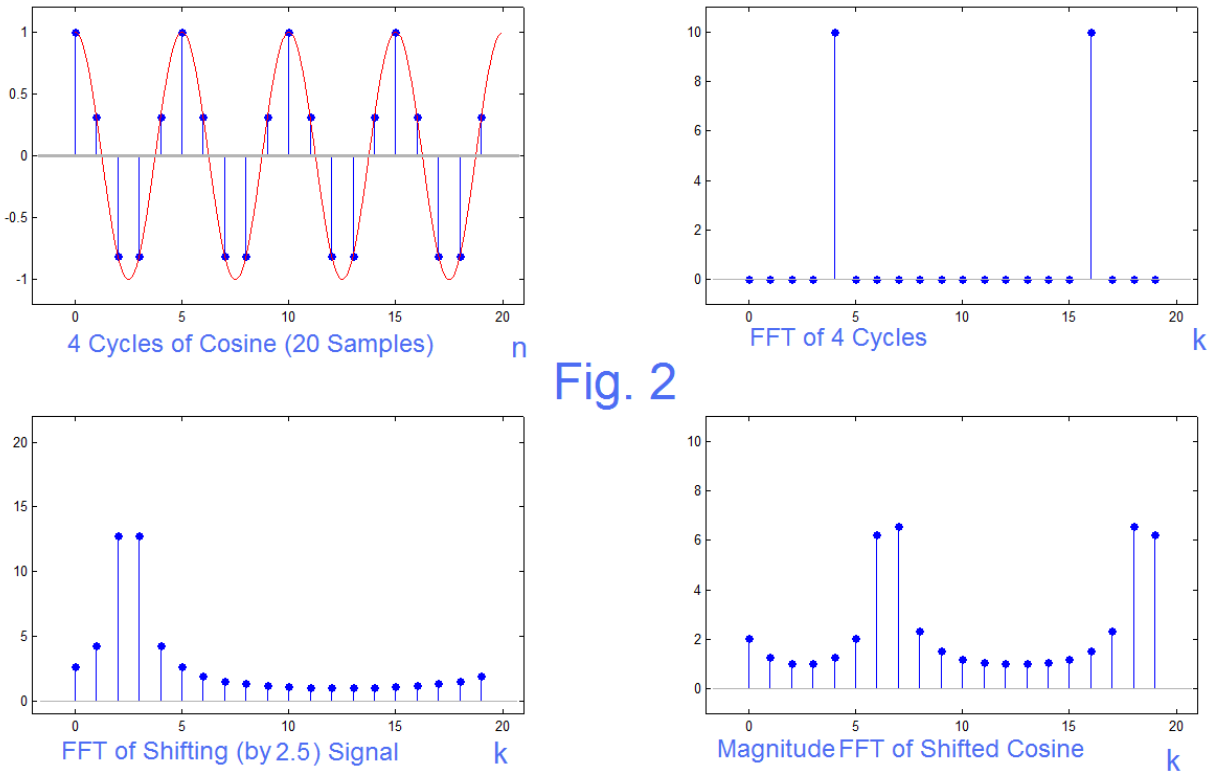


Fig. 2

ROTATIING THE RANDOM SPECTRUM

We are now in the correct position to see what really happens when we try to rotate the spectrum of a random signal so as to avoid the real "random walk" summations. This we do first by looking at the previous result [2] where we detected and offered an explanation for endpoint dips. Fig. 3, Fig. 4, and Fig. 5 are all the result of averaging 10,000,000 FFT magnitudes. The signal lengths are all 100 samples, so as usual, k=0 is DC, while k=100/2=50 is half the sampling rate, so we have a dip there too (in Fig. 3). So this is the base case.

Fig. 4 shows the case where after generating the input data, it is multiplied point-by-point by the rotating signal:

$$x_s = e^{j5 \bullet 2\pi \bullet n/100} \qquad n = 0,1 \ldots\ldots 99 \qquad (8)$$

When we applied such a rotation to a periodic signal, we saw a rather exacting shift to the right, as expected. Without fully stating at this point the full significance of the result, note that the FFT spectrum of Fig. 3 is indeed shifted up by 5 index values of k. So it seems like Fig. 3 was the spectrum and it has been shifted. The key sticking point here is seen in equations (4) and (5) where we saw that there was a cross-over of interpretation with regard to whether the unity magnitude exponential being employed has the function of changing the frequencies where a DTFT is evaluated, or is this sequence a modulation of the input. Presumably we can understand this in both interpretations.

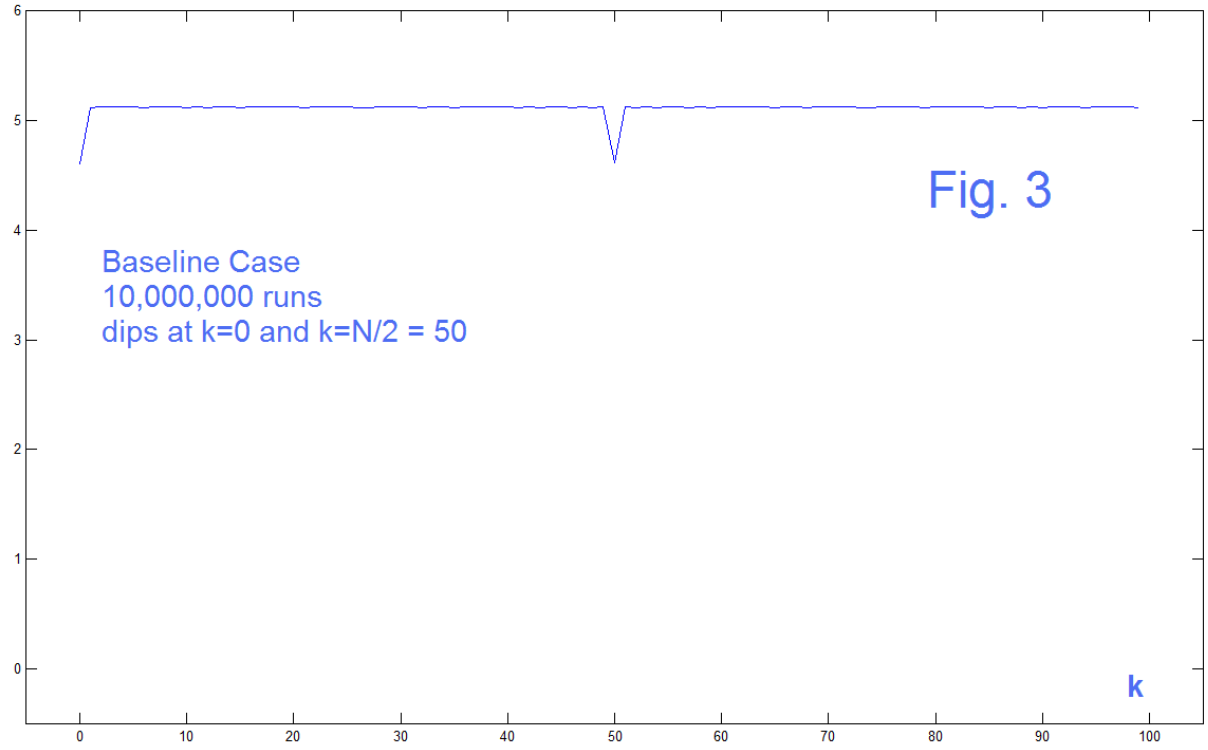


Fig. 3
Baseline Case
10,000,000 runs
dips at k=0 and k=N/2 = 50
k
0
1
2
3
4
5
6
0
10
20
30
40
50
60
70
80
90
100

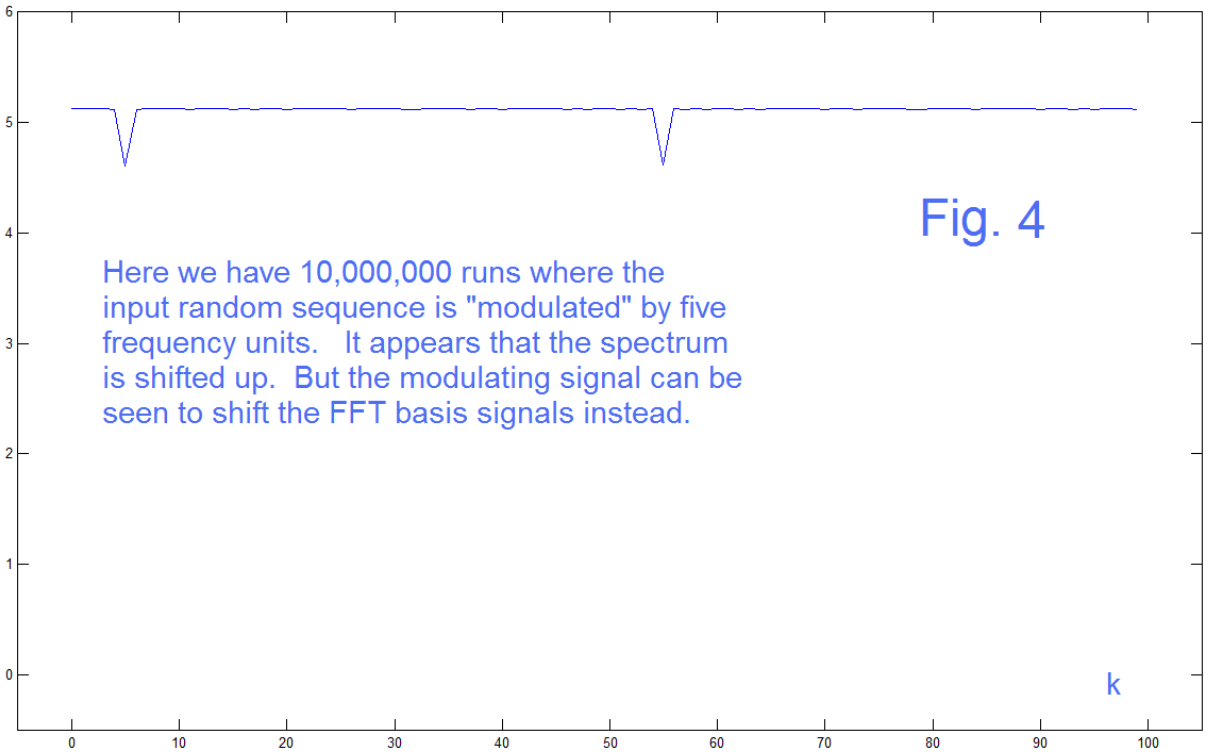
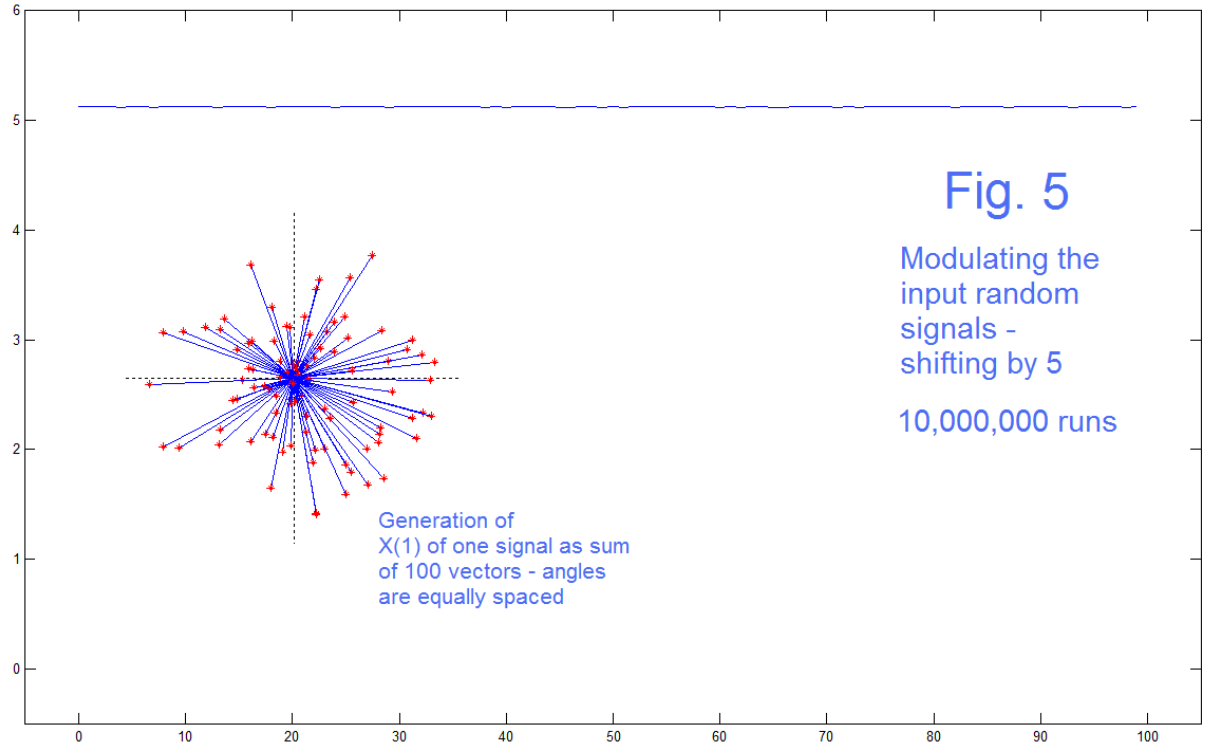


Fig. 4
Here we have 10,000,000 runs where the
input random sequence is "modulated" by five
frequency units. It appears that the spectrum
is shifted up. But the modulating signal can be
seen to shift the FFT basis signals instead.
k
0
1
2
3
4
5
6
0
10
20
30
40
50
60
70
80
90
100



The next step is to try to shift the spectrum by 5.5 instead of by 5, using:

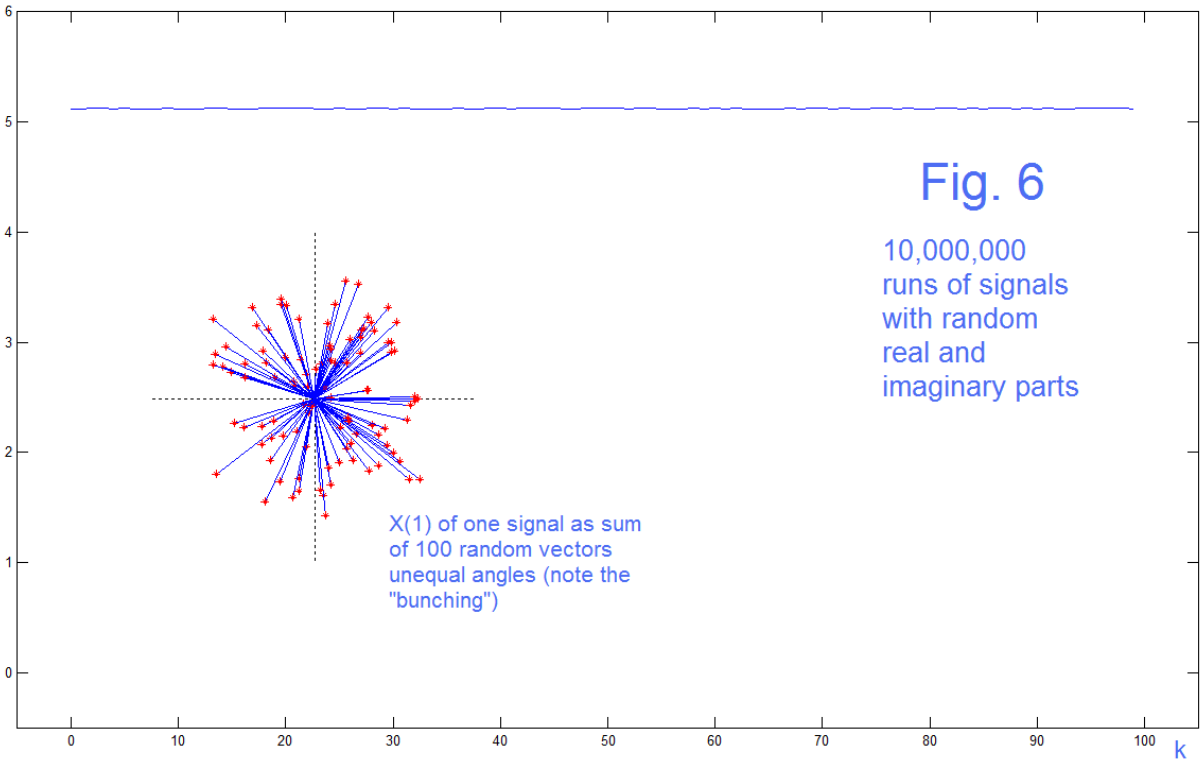
$$x_s = e^{j5.5 \bullet 2\pi \bullet n/100} \qquad n = 0,1 \ldots \ldots 99 \qquad (9)$$

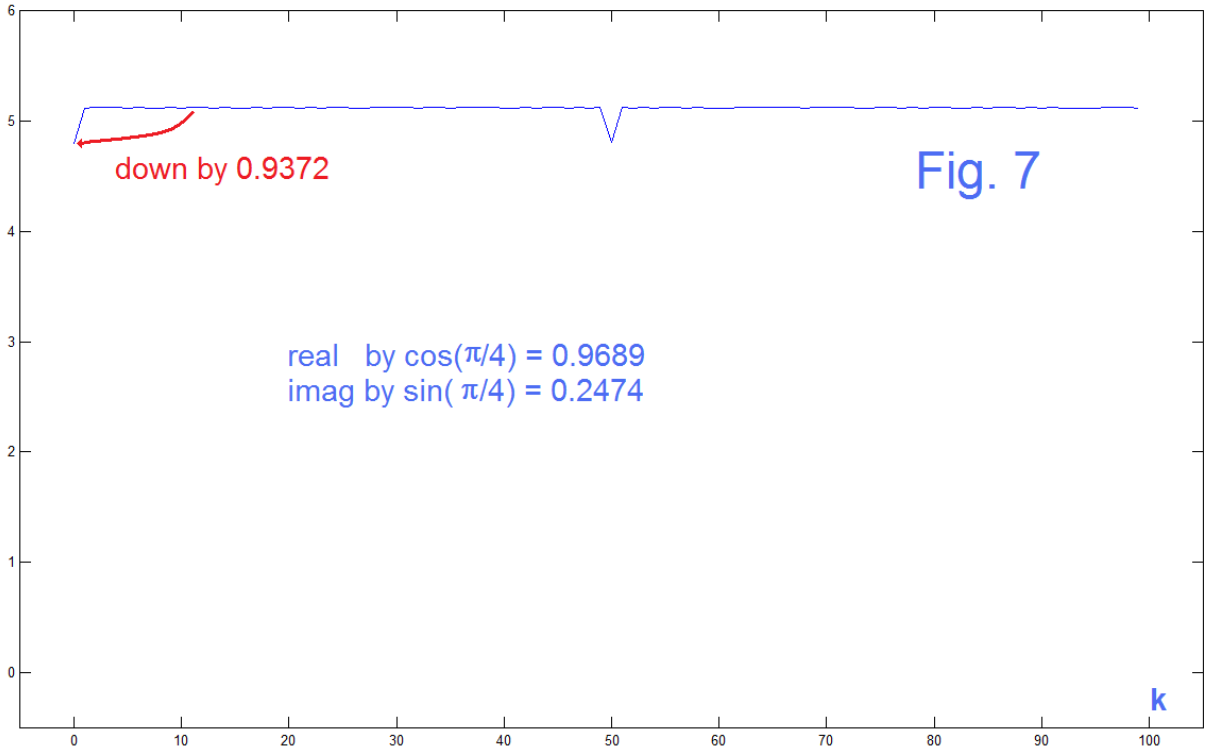
and the result is shown in Fig. 5. **IT'S FLAT**!

Recall that the original intent of rotating the spectrum was to avoid the cases of summing up real numbers in a standard random walk. In fact, there were two cases where we were summing real values of x(n), k=0 or DC, and k=N/2 or half the sampling rate when N is even. (Note that for the case of odd N, there was no FFT value taken for k=N/2, and we reported that no dip was seen there. Accordingly one interpretation of Fig. 5 is that in a similar manner we have now also removed the DC case.)

At some point, the thought jumps to mind that we possibly should have pursued the notion that it is not the modulation of the original x(n) as much as it is the fact that multiplying by the modulating sequence makes all (virtually all) the input sequence into the FFT complex. That is, and obvious "control" for our study. This is an easy test which we should have tried sooner. Thus we form each of our random input values by two random numbers, one for a real part, and the second for an imaginary part (multiplying both by $\sqrt{2}$ so as not to change the input power. Fig. 6 (10,000,000 runs) shows the result. **IT TOO IS ABSOLUTELY FLAT!** This control result is HIGHLY significant.

The result of Fig. 6 is also probably not a surprise had be thought about it and remembered that the difference between dip or no-dip was whether the random walk was one dimensional or two dimensional. The easiest way to guarantee a two-





dimensional random walk is to take steps in two dimensions (complex random input signal). This we try - instead of relying on a complex multiplying (modulating or shifting) sequence. So this makes sense. We do have some loose ends to tie up.

POINT 1: FFT Itself Has Modulating Sequences

Here looking back to equations (4) and (5) we saw that using a modulating sequence to shift a spectrum involved the modulating exponential sequence multiplied by the exponential sequence that was inherent to (basis) the FFT itself. In fact it is the "inner products" of these "basis" sequences with the signal that constitute the coefficients that are the output of the FFT. Thus nothing new was expected or found as long as the sequence indices were shifted by integers (Fig. 4).

POINT 2: Shifting Sequence Merges With FFT Basis

Closely related to Point 1 is the fact that if you do merge the modulating signal (such as $e^{j(2\pi/N)m\bullet n}$) with the FFT basis ($e^{-j(2\pi/N)k\bullet n}$) we will note the emergence of a sequences of all ones (k=m), and hence, the sum of real values of x(n) if x(n) be purely real to begin with (Fig. 4).

POINT 3: Random Versus Ordered Angles

This is a sticky point. On the one hand, we argue that a perfectly random complex x(n) scatters random walk steps in two dimensions. See the inset plots in Fig. 5 and Fig. 6. Since the real and imaginary parts are random and independent in Fig. 6, we have random length steps at random angles. Note the "bunching" of vectors in Fig. 6 as compared to Fig. 5 where the angles are all equally spaced. Our feeling that there might be something "wrong" with the bunching is exactly the same sort of effect we saw in bunching of head or tails in a random coin flip [3]. Fig. 5 on the other hand relied on random length steps [the real values of x(n)] but at an ordered set of equally spaced angles. This is a bit slippery. It is the same thing we discussed in [2], at page 11. Probably just fine. Both provide a scattering of angles.

POINT 4: What If Imaginary Part Is Small?

We saw that when the input signals x(n) were purely real, we had the (base case) endpoint dips to about 90% (Fig. 3). We saw that there were two ways in which we could make the input signals (at least mostly) complex: Fig. 5 by modulating a real x(n) with a complex sequence; Fig. 6 by forming x(n) to be already complex. In both cases, the dips went away. Do the dips suddenly go away just by virtue of the data becoming complex? Obviously not. We could just add a very small imaginary part to a real x(n) (perhaps 0.000001•j) and we would have a complex input, but would suspect no significant change of result.

So once again we have the issue of a control test for the study. Instead of setting equal amplitudes to the random generators for real and imaginary, what if we back this off and have a small (but not too small) random imaginary part. For our study (Fig. 7) we have chosen to keep the real part 0.9689 and make the imaginary part smaller at 0.2474 (keeping same power). The purely real case (Fig. 3) has a dip to 0.9003 while the complex case (Fig. 6) has no dip. True enough, the case of a smaller imaginary part has a dip of 0.9373. less dip than 0.9003.

[1] B. Hutchins, “Flat and Not-So-Flat Spectra,” Electronotes Application Note No. 360, August 2004
http://electronotes.netfirms.com/AN360.pdf

[2] B. Hutchins, “A White Noise Curiosity,” ***Electronotes***, Volume 22, Number 208 January 2012.
http://electronotes.netfirms.com/EN208.pdf

[3] B. Hutchins, “Do Random Sequences Need “Improving”?,” Electronotes Application Note No. 415, Nov 1, 2014
http://electronotes.netfirms.com/AN415.pdf

[4] B. Hutchins, “Fourier map,” Electronotes Application Note No. 410, May 6, 2014
http://electronotes.netfirms.com/AN410.pdf

[5] B. Hutchins, “Two Views of “Leakage” in DFT-Based Spectral Estimation,” Electronotes Application Note No. 340, Nov. 1996. (Not currently posted – email to request copy and/or posting)

MATLAB CODE

The Matlab code here is offered as the best form of documentation of what was done. The code is not offered as optimal.

```
% AN416.m
% Makes Fig. 1 - Fig. 7 of AN-416 with same numbering
% Notation is not necesarily the same

% Fig. 1   Shift frequency 4 by 2
x=cos(2*pi*4*[0:19]/20)
xc=cos(2*pi*4*[0:1999]/2000);
X=fft(x)
xs=exp(j*2*pi*2*[0:19]/20)
XS=fft(xs)
xxs=x.*xs
XXS=fft(xxs)
figure(1)
subplot(221)
stem([0:19],x)
hold on
plot([0:1999]/100.,xc,'r')
hold off
axis([-2 21 -1.2 1.2])
subplot(222)
stem([0:19],abs(X))
axis([-2 21 -1 11])
```

```
subplot(223)
stem([0:19],abs(XS))
axis([-2 21 -1 22])
subplot(224)
stem([0:19],abs(XXS))
axis([-2 21 -1 11])

% Fig. 2   Shift frequency 4 by 2.5
x=cos(2*pi*4*[0:19]/20)
X=fft(x)
xs=exp(j*2*pi*2.5*[0:19]/20)
XS=fft(xs)
xxs=x.*xs
XXS=fft(xxs)
figure(2)
subplot(221)
stem([0:19],x)
hold on
plot([0:1999]/100.,xc,'r')
hold off
axis([-2 21 -1.2 1.2])
subplot(222)
stem([0:19],abs(X))
axis([-2 21 -1 11])
subplot(223)
stem([0:19],abs(XS))
axis([-2 21 -1 22])
subplot(224)
stem([0:19],abs(XXS))
axis([-2 21 -1 11])


% Fig. 3, 4, 5
XT=zeros(1,100);
XT0=zeros(1,100);
XT1=zeros(1,100);
N=1000000
% rotate by 5
w0=2*pi*5/100
% rotate by 5.5
w1=2*pi*5.5/100
offset0=exp(j*[0:99]*w0);
offset1=exp(j*[0:99]*w1);
for m=1:N
   x=2*(rand(1,100)-.5);
   x0=x.*offset0;
   x1=x.*offset1;
   X= fft(x) ;
   X0=fft(x0);
   X1=fft(x1);
   XT=XT+abs(X);
   XT0=XT0+abs(X0);
   XT1=XT1+abs(X1);
end
figure(3)
plot([0:99],XT/N)
axis([-5 105 -.5 6])
figure(4)
plot([0:99],XT0/N)
axis([-5 105 -.5 6])
```

```
figure(5)
plot([0:99],XT1/N)
axis([-5 105 -.5 6])
%
XTav=sum(XT(10:40))/(31*N)
XT0av=sum(XT0(10:40))/(31*N)
XT1av=sum(XT1(10:40))/(31*N)


% Fig. 6  Test for complex random input
XT=zeros(1,100);
XT0=zeros(1,100);
XT1=zeros(1,100);
N=1000000
for m=1:N
   x=sqrt(2)*(rand(1,100)-.5)+ sqrt(2)*j*(rand(1,100)-0.5);
   X= fft(x) ;
   XT=XT+abs(X);
end
XTav=sum(XT(10:40))/31
XT(1)
XT(1)/XTav
figure(6)
plot([0:99],XT/N)
axis([-5 105 -.5 6])

% Fig. 7 Make imaginary part smaller
XT=zeros(1,100);
XT0=zeros(1,100);
XT1=zeros(1,100);
N=1000000
for m=1:N
   x=2*cos(.25)*(rand(1,100)-.5)+ 2*sin(.25)*j*(rand(1,100)-0.5);
   X= fft(x) ;
   XT=XT+abs(X);
end
XTav=sum(XT(10:40))/31
XT(1)
XT(1)/XTav
figure(7)
plot([0:99],XT/N)
axis([-5 105 -.5 6])
```