ELECTRONOTES 208

Newsletter of the Musical Engineering Group

1016 Hanshaw Road, Ithaca, New York 14850

Volume 22, Number 208 January 2012

GROUP ANNOUNCEMENTS

Contents of EN#208

A WHITE NOISE CURIOSITY

-by Bernie Hutchins

INTRODUCTION

Most of us have a good notion of what "White Noise" is. Audibly it is a hiss like the leakage of air from an automobile tire. We used to say it was like the "interstation" noise on an FM radio receiver – but most modern receivers automatically mute in the absence of a usable signal. In electronic music, we use white noise as source material for some sounds, and sometimes as a source for random (control) voltages to be converted into pitch sequences, etc. A sharp enough bandpass filtered white noise can easily "carry a melody" and is perhaps the premier example of colored noise – across the audible spectrum. Here we will review a bit, but want to discuss a few issues not covered in the past as well.

(1) THE PROBLEM WITH A (NOT) FLAT SPECTRUM

Here we will be looking at a discrete sequence of random samples, and the use of the FFT to display the corresponding spectrum. Consecutive samples of a white noise sequence are independent – uncorrelated. The spectrum is flat. This is in analogy with the spectrum

of white light. Everyone knows that this works. Should be easy to demonstrate. Well, there's a problem.

First however, this is not a similar-sounding problem that comes up from time to time on the Internet. Typically someone states that they generated a white noise signal, typically with Matlab, using ***rand*** or perhaps ***randn***, calculated the FFT of the sequence, and it was not flat! What is wrong, they shout. Along will come some suggestions, almost always useless, until someone points out that it is the "Expectation" (in its statistical, mathematical definition) that the spectrum will be flat, but you don't expect a flat FFT magnitude in any one example. Similarly, there are many arguments as to whether the measured mean should be zero (it shouldn't). It is easy to waste time trying to find hints on the web! My problem, as will be reiterated below, is that a well-averaged spectrum is not flat on the end or ends. What do I mean "reiterated"?

A digression here. Many of us have an admirable collection of unfinished projects. From time to time, we may accidentally bump into one which looks near finished, or at least, salvageable for some quick purpose. In the present instance, I wanted to develop some material on "red" noise. Looking at a result, I saw in the corner of a graph something that was unexplained, and which I thought was similar to a result for white noise, which I thought I had explained. But I couldn't remember the explanation. Had I just imagined I had solved the problem? I guess so. One advantage of a publishing enterprise is that you can usually find previous material (on the shelves). It was just a matter of looking it up and reading my own explanation.

The reference here was to a previous App Notes, AN-360 "Flat and Not-So-Flat Spectra" dated August 2004. It is available at:

http://electronotes.netfirms.com/AN360.pdf

and is not very long. Everything there seems to be correct. Evident there is the problem which I remember worrying about. But it is not mentioned there, and certainly an explanation is not there. It is seen there by looking at Fig. 3 (reproduced as Fig. 1 for this paper). The spectrum is fairly flat, but it dips at both ends, slightly, but noticeably.

Now, here is a secret you would of course have figured out for yourself. When an author writes about random stuff (i.e., about random signals and processes, not at random!), and wishes to give an example (a "typical" example), not unlikely some sort of fine tuning enters at this point. You really should just take the first example that comes out. You are not supposed to look at a result, perhaps grimace, and then try again because you feel there is something too special (untypical) about what fate handed you. But, don't count on that.

There is perhaps some merit in applying a small amount of selection. Perhaps you have run a program a dozen times to get everything working. Now you want an example. It is quite possible that some examples might be so untypical as to be misleading. So, a little selection is probably okay. With AN-360 I do remember the curious result of Fig. 3 – the dip on both ends. I also remember trying to get a better example (if only it had just gone away in some instances), and playing with the programs. I also remember thinking I would need to explain this, and that the explanation would be interesting. Reading AN-360, it seems I never even mentioned it. In consequence, there was no explanation! Fig. 1 below shows Fig. 3 from AN-360. The dips are there at k=0 and at k=50.

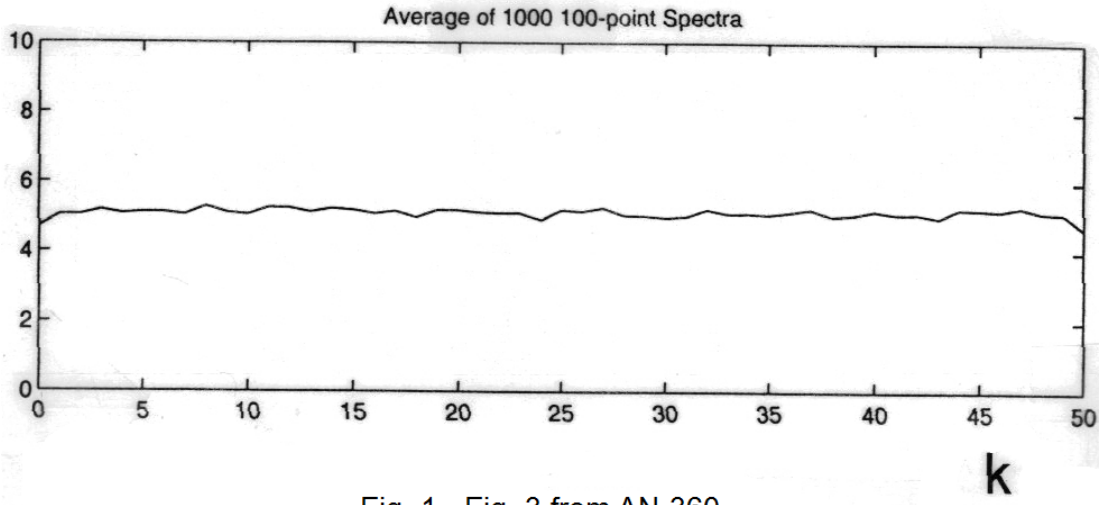
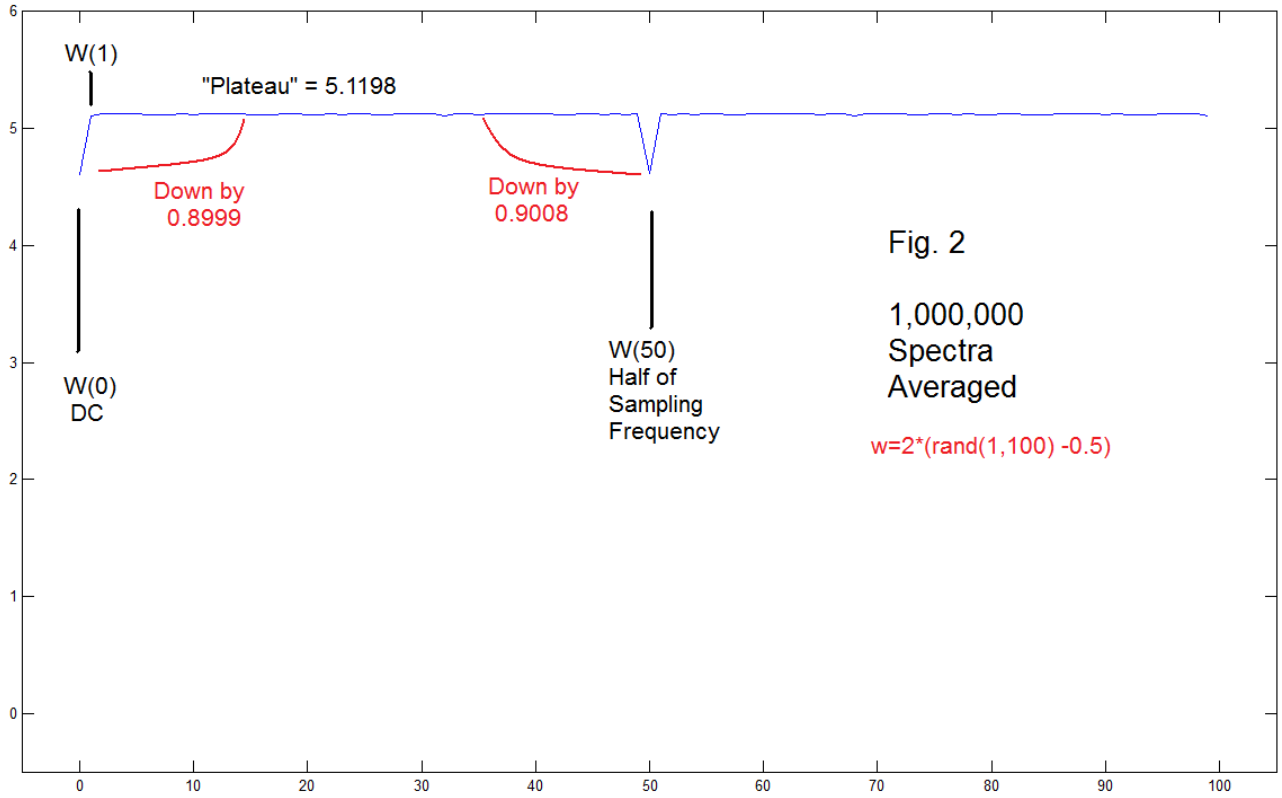


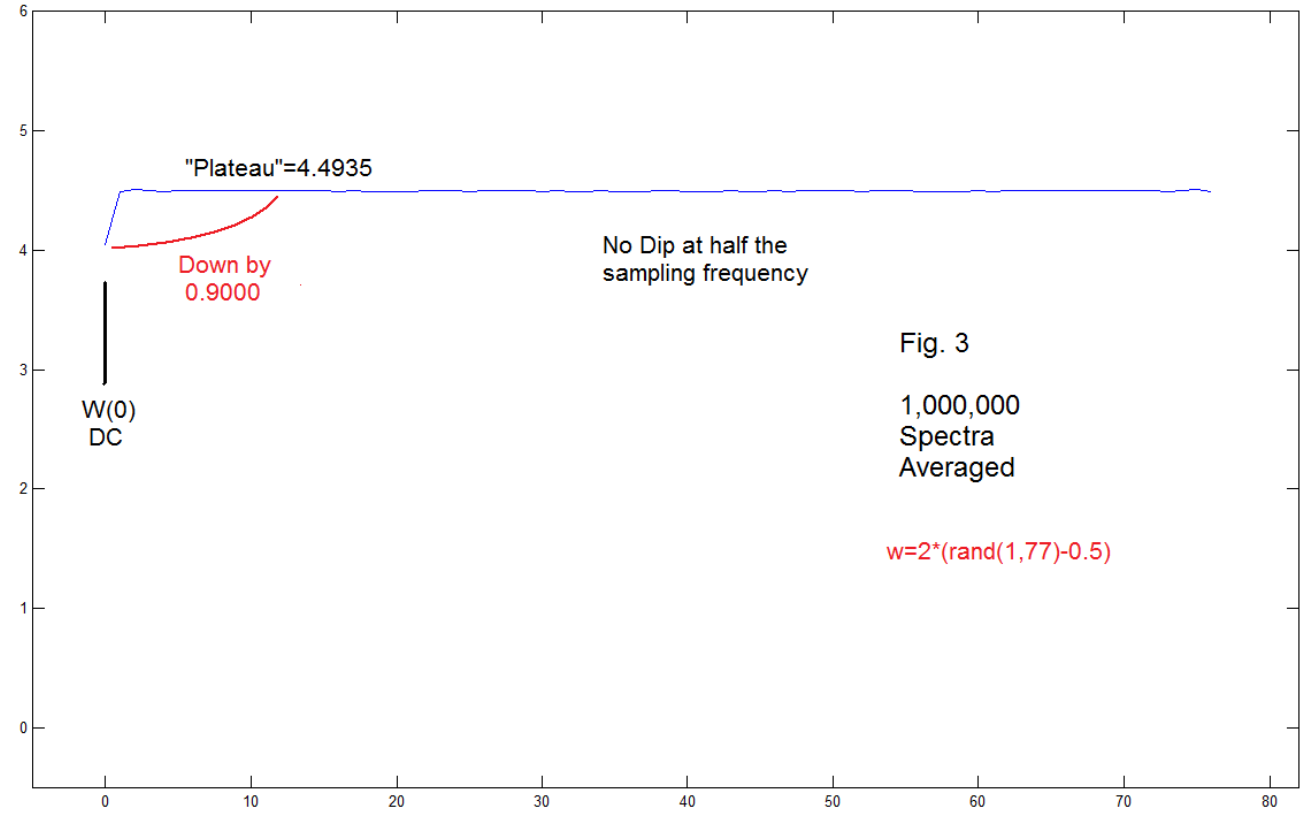
Fig. 1 Fig. 3 from AN-360

(2) THE EVIDENCE

The reaction to a plot such as that in Fig. 1 should be first of all: "That's Funny!" Secondly, we need to determine if it is just a fluke. It's not. So what can we learn about it as a phenomenon? There is something to learn here. What is the evidence?

Fig, 2 shows an updated version of Fig. 1. The difference here is that we are averaging 1,000,000 length 100 spectra. From this we see the same result that was in Fig. 1, better defined because we now have 1000 times more iterations. We also note that the dips are just 1 sample of the FFT wide. The signals were generated using Matlab's ***rand*** function as shown in the figure. The notation W(k) refers to the usual FFT usage, where k runs from 0 to N-1 (0-99 in this case). [Due to Matlab's indexing, in the actual program code the dips are for W(1) and W(51).] The only other thing to note is that the dips are just about exactly to 90% of the plateau. Does this mean anything?


W(1)
"Plateau" = 5.1198
Down by
0.8999
Down by
0.9008
W(0)
DC
W(50)
Half of
Sampling
Frequency
Fig. 2
1,000,000
Spectra
Averaged
w=2*(rand(1,100) -0.5)


"Plateau"=4.4935
Down by
0.9000
No Dip at half the
sampling frequency
W(0)
DC
Fig. 3
1,000,000
Spectra
Averaged
w=2*(rand(1,77)-0.5)

Moving on to another example, Fig. 3 shows the case where we use a length of N=77 rather than N=100. Perhaps the first thing to notice is that this case has no dip at half the sampling frequency. The dip at DC is again 90%, as for N=100. The plateau height is less, 4.4935. This result, and a few more, convinces us that the formula for the plateau height is:

$$P_N = 0.512 \sqrt{N} \qquad (1)$$

So, the examples shown in Fig. 2 and Fig. 3, and additional examples we have tried, convince us that:

(1) For all N, there is a dip at DC, k=0, that is about 0.900 down from the plateau.

(2) For even N, there is also a dip of 0.900 at half the sampling frequency, k=N/2. This point does not exist for odd N.

(3) The plateau height is given by equation (1).

(3) VECTOR SUM

The FFT (i.e., the DFT) is a sequence of numbers each of which is a weighted sum of vectors. Indeed, it is usually written as:

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

If we write this out specifically for k=0 and for k=1, and call our time sequence, the white noise, w(n), we have:

$$W(0) = w(0) + w(1) + w(2) + \ldots.. w(N-1) \qquad (3a)$$

$$W(1) = w(0) + w(1)\, e^{-j2\pi/N} + w(2)e^{-j4\pi/N} + \ldots. w(N-1)e^{-j2\pi(N-1)/N} \qquad (3b)$$

Thus W(0) is the sum of all the random values in the particular instance of the signal w(n) and W(1) is also a sum, but the random values are multiplied by a rotating unit vector in the complex plane. Note we are computing the average value we would get by comparing the absolute values of these sums for a large set of sequences w(n). It is useful here to have an example – a physical notion of how a sum is formed.

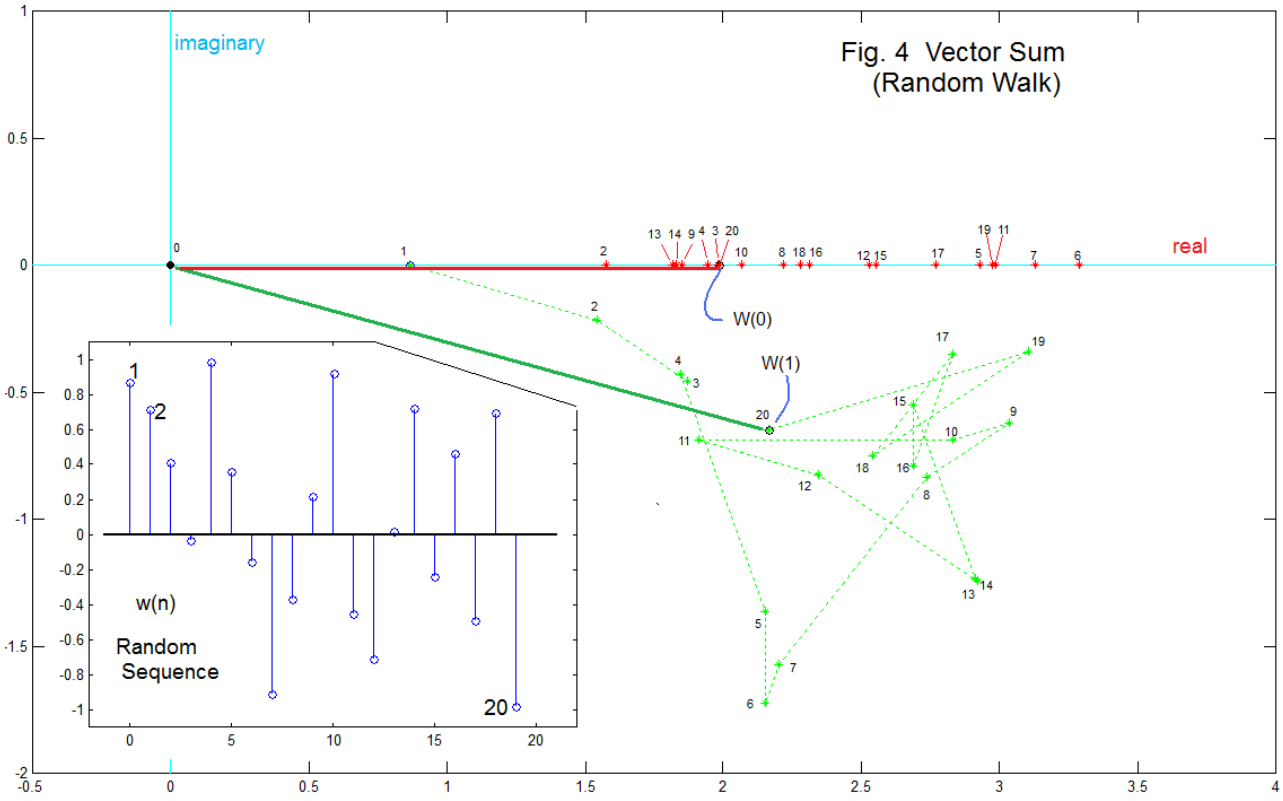


Fig. 4 shows an example for a particular length 20 w(n) shown in the stem plot inset at the lower left. For W(0), the values of w(n) are sequentially summed (red stars) and constitute the usual "drunkard's walk" we expect with integration (accumulation actually). In this case, there are no complex multipliers, so the walk is along the real axis only. We can see the positive and negative contributions opposing each other, so while the individual elements of w(n) have magnitudes between 0 and 1, the sum doesn't get that far away, and ends with the 20th samples at 1.9887. The heavy red line (placed slightly below the axis for visibility) is the overall result. This is all very familiar.

The case of W(1) is different. Because of the complex multipliers we have a random walk in the two-dimensional complex plane. Quite attractive! Each of the vectors summed here has a length given by the same w(n) but all are at different angles. In fact, the angles increment 18 degrees (360/20) with each step. So, in the figure, relative to W(0), we see the same starting point (at 0,0) and the same first step, since the first angle is zero. Then the second and third step bend away at 18° angles. The fourth step is small, and negative, so appears as a jog, followed by a larger step 5, and so on. We end up at step 20 as shown, and as summarized by the heavy green line.

In this case, the green line is slightly longer than the red one. One example proves nothing here. What we claim is that on average, green lines will be longer than red lines. This is easy enough to program, and we do find that – in 1,000,000 trials (run but not printed on the screen of printed here – for sure) red lines were about 0.9002 relative to the green lines. No surprise – it is the same calculation the FFT does. What we hope to eventually get is some notion, based on this simple physical picture, of why this happens.

(4) WHAT DOES THE MEAN MEAN

In our experiments, we used the Matlab ***rand*** function. The ***rand*** function returns one or many random numbers that have a uniform distribution between 0 and 1. This means that numbers like 0.2, 0.3115, 0.9, etc all have the same probability. But they are all positive. Very often we want a distribution that is between negative and positive limits. Hence, while the instruction:

```
w = rand(1,350)
```

would give you 350 random numbers between 0 and 1, we would often use instead:

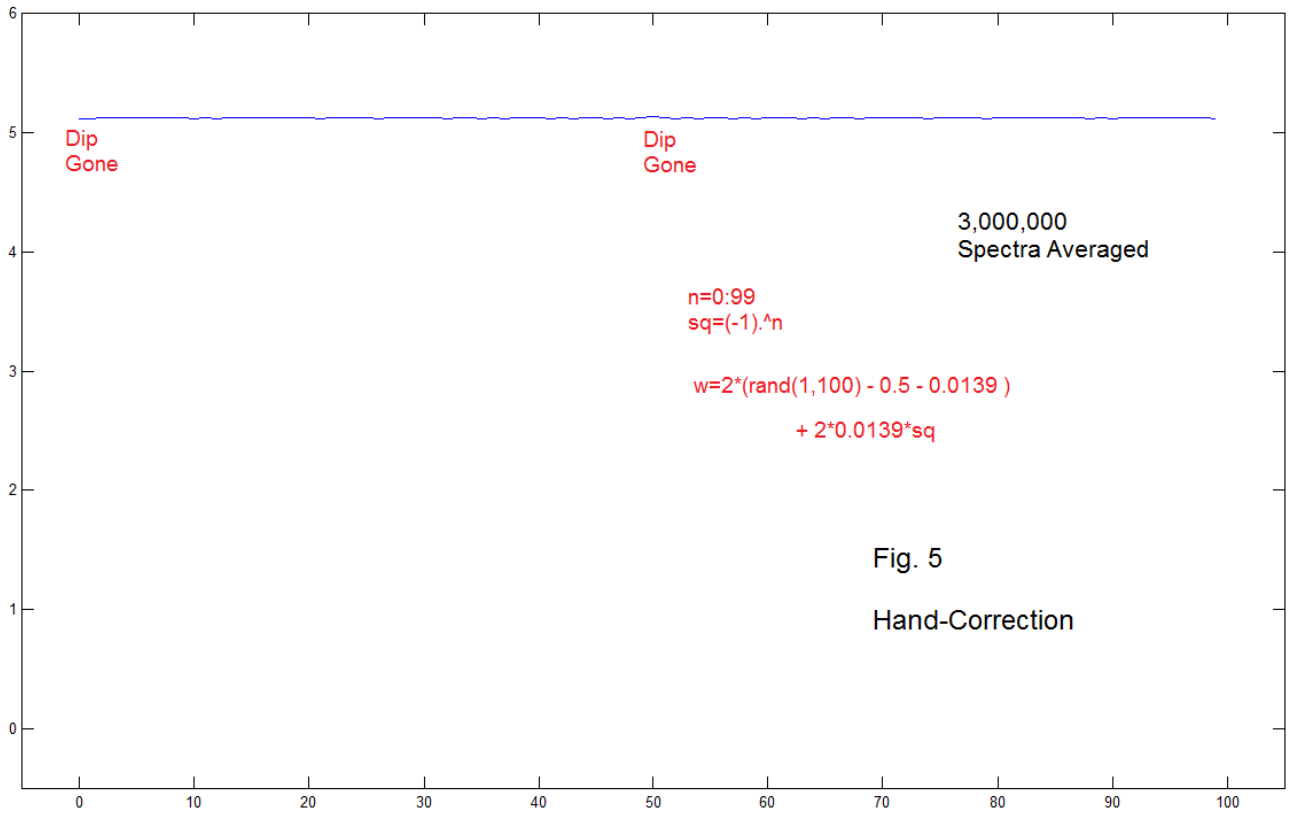
```
w = 2*(rand(1,350) – 0.5)
```

to give you 350 random numbers uniformly distributed between -1 and +1. Thus it may appear that the rand function itself has a mean of 0.5 and we are removing this. So, perhaps we might think that the X(0) of the FFT should be, not only smaller than the other frequencies (which we observe), not only just down to the 90% we observe, but all the way to zero. Not so. Any finite length w will have a DC bias. For example, we could have a perfectly good length 4 sequence w that has values -0.2, 0.7, 0.8, and -0.5. This has a net DC value of 0.8.

But is there perhaps something wrong with the ***rand*** function? These aren't really random numbers but rather pseudo-random numbers – generated by an algorithm. What if that is faulty?

The first test we can make is to use a different random number function, ***randn***. This function produces a normal distribution with mean 0, and variance 1. Making this substitution we observed the same dips in the spectrum that we got with ***rand***. This is more evidence. This is perhaps a good time to mention that we also tried a substitute for the Matlab ***fft*** function – the computation of the DFT with a matrix – to the same result we got with ***fft***.

Another test is to remove the dips by adding to the random signal. This we do by changing the DC bias, and by adding in an alternating sequence (half the sampling frequency) in the case of an even length sequence. By experiment we find the following to work (Fig. 5).



This result is particular to the length 100 sequence. We note several interesting things. First, it works. Secondly, there is a fairly large DC shift needed, nearly 3%. Thirdly, the same constants used for the DC reappears when we want to get rid of the dip at half the sampling frequency. This perhaps makes the point that the situation at ½ the sampling frequency is obviously related to the DC case. Indeed, since we are just multiplying by an alternating sequence of +1 and -1, this is in turn a perfectly good random sequence, just as it might have been generated directly by ***rand***. It would be another red random walk as in Fig. 4.

What we show is that whatever is going on here, its nature is of the same effect as an actual signal with conventional spectral content. We needed to check that this was the case. Before we go on, was it just some silly problem? No – it's something else.

(5) ESTABLISHING THE MEAN

Fig. 2 and Fig 3 show grand (final) results, the average of a million FFTs, while Fig. 4 shows the minute details of just two points of one FFT. Clearly we need to take a look at how the averages evolve as more and more signals are averaged.

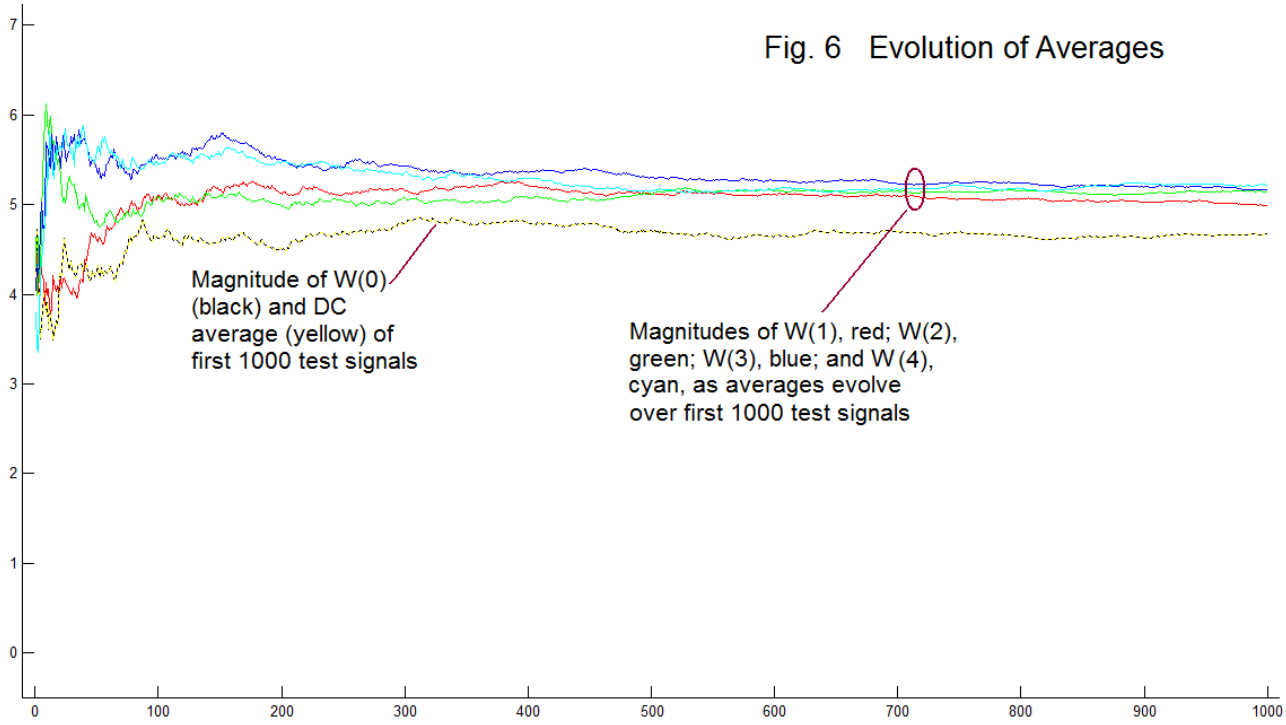


Fig. 6 shows the average magnitudes as they evolve over 1000 random signals. Here we see the evolution for the first FFT point W(0), along with the evolving average of the DC term (average of individual means). This is the curve that is black, W(0), overplotted exactly with the DC average (dashed yellow). They are the same curve. The remaining four curves are for W(1), W(2), W(3), and W(4). The W(0) curve will eventually converge toward something like 4.61 and the others, toward something like 5.11 (see later).

Above we mentioned that authors may be selective when choosing an example to post when the actual appearance is subject to random features, which may be “misleading”. This is a good example of a case where it was necessary to run the program perhaps a dozen times before we got a “suitable” demonstration. Not that all of the examples would not have eventually converged correctly, but rather we wanted one that converged nicely within the 1000 signals so as not to clutter the page. The point is correctly made.

(6) THE DISTRIBUTION

We have found the following. We can interpret the DFT of a random signal as a summation of random steps, and the magnitude of the DFT is the distance (always positive) from the end of the last step back to the origin. In the case of W(0), and of W(N/2) in the case of even N, the steps are a familiar random walk (more colorfully, a "Drunkard's Walk") along a real line. For all other W(k), the random walk is in the complex plane. We are interested in the distribution of the lengths of the endpoints. Fig. 7 shows our example.

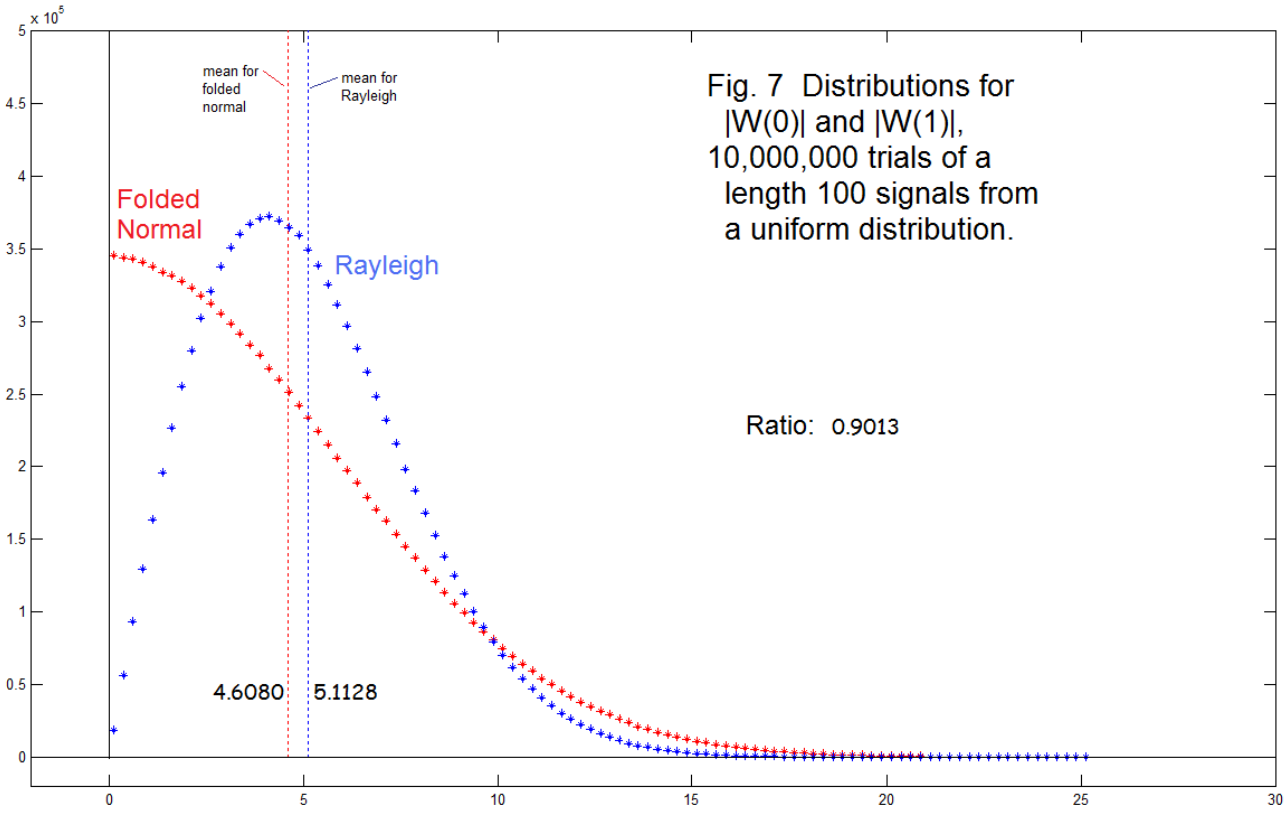


Fig. 7 Distributions for |W(0)| and |W(1)|, 10,000,000 trials of a length 100 signals from a uniform distribution.

This lovely picture is the answer. The difference we observe is because the two cases have different statistical distributions. This is extremely interesting.

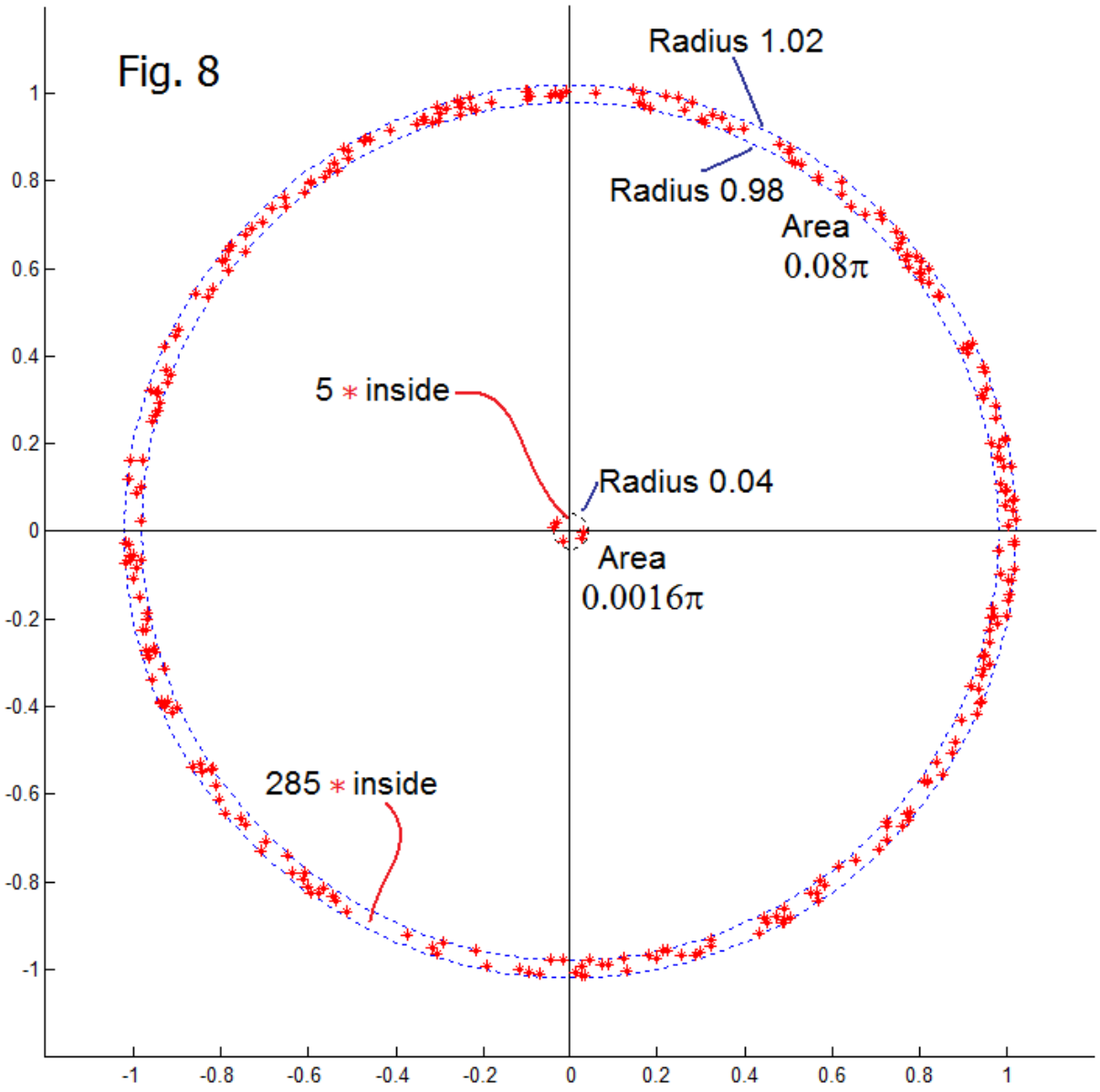
First of all we can discuss the red curve. This is the result of the real random walk. Although the individual steps are from a uniform distribution (more distributions follow), we would expect the endpoints of the random walk to have a Gaussian ("normal" or "bell-

shaped") distribution, with zero mean, as a result of the "central limits theorem". This is the case – except we are NOT plotting final positions but rather the absolute values of these final positions. All the results are positive, so effectively, the Gaussian is folded over. It is the mean of the absolute values – not the absolute value of the mean. No wait! It is the mean of the absolute values of the means of the individual sequences. Hard to describe, but what is going on is likely quite transparent. The "folded normal" is well-studied. So much for the red curve.

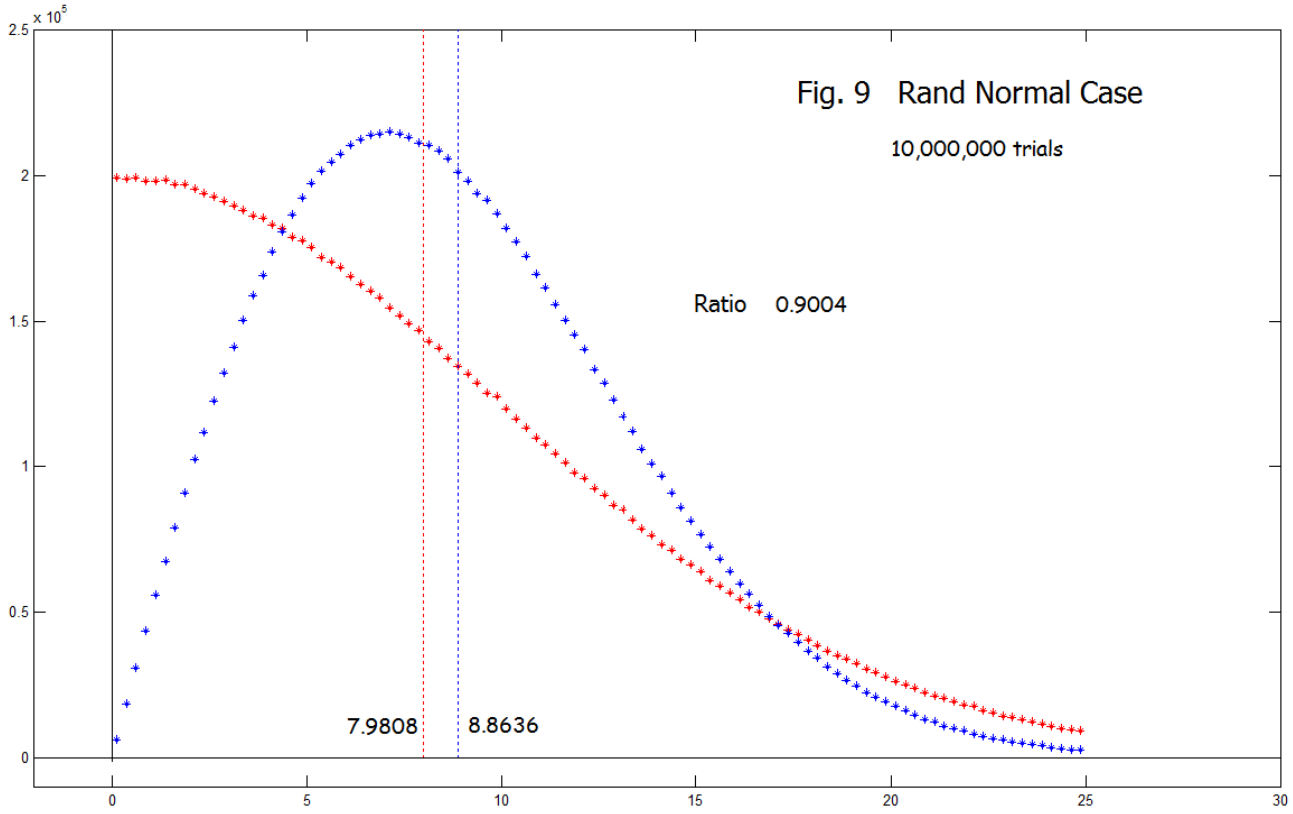
The blue curve is a surprise. We have provisionally labeled this as being a "Raleigh distribution" based on appearance alone. This is also a well-studied distribution. It is quite different from the folded Gaussian. For one thing, it is zero at zero. Recall that this is a random walk in the complex plane (the green walk in Fig. 4). It is only a complex plane here because we started with the DFT – we can just consider it a two-dimensional plane if we prefer. The corresponding "Drunkard's Walk" is generally presented as an interesting story of a drunkard leaving a bar and trying to find his way home, walking along city blocks and selecting new directions (including a reversal of direction) at each intersection. It is a proven result that the drunkard will always (probability 1) get home eventually (or to any other point, including a return to the bar, the point 0,0). The probability of getting there in a given finite number of steps may be quite low, of course.

The DFT case is different in some ways. We are not restricting the walk to city blocks (a discrete grid). And the lengths of the steps are chosen at random from a uniform distribution, projected along a predetermined set of DFT angles ($2\pi nk/N$). This is not the same as choosing, independently, two random draws, one for real, and one for imaginary, which would lead to a Rayleigh distribution. This we can investigate by looking at a non-DFT case where the angles, as well as the lengths of the steps, are also chosen at random, and we will do this (see later). For the moment, let's continue with the issue of low probability in the region of the origin.

When we measure distances radially, going no further than a very small distance from the origin is a very small circular area. On the other hand, this same small distance out further defines an annular region of significant area. Fig. 8 shows this idea. Here we have taken trial points from uniform distributions (uniform from -1.5 to +1.5) in the two perpendicular directions, and kept only points that fell into the center circular region (0.04 in radius) or in the annular region from 0.98 to 1.02. The difference in areas is 50:1. In this case, there were 57 times more hits inside the ring as compared to the circle. Accordingly we understand how there is (relatively) vanishingly small probability of hitting examples that fall very close to the center. (Eventually, the ring would expand beyond the underlying cluster, and would not be populated.)



We have found that the difference between the W(0) and W(1) averaged magnitudes (which we continually find to be very close to 90%) needs to be ascribed to the different distributions. This we will address shortly. We also need to justify treating the real and imaginary parts in the DFT case as independent. For the moment, we want to show that this ratio is independent of the distribution of the original random signals.


x 10^5
2.5
2
1.5
1
0.5
0
0
5
10
15
20
25
30
Fig. 9 Rand Normal Case
10,000,000 trials
Ratio 0.9004
7.9808
8.8636

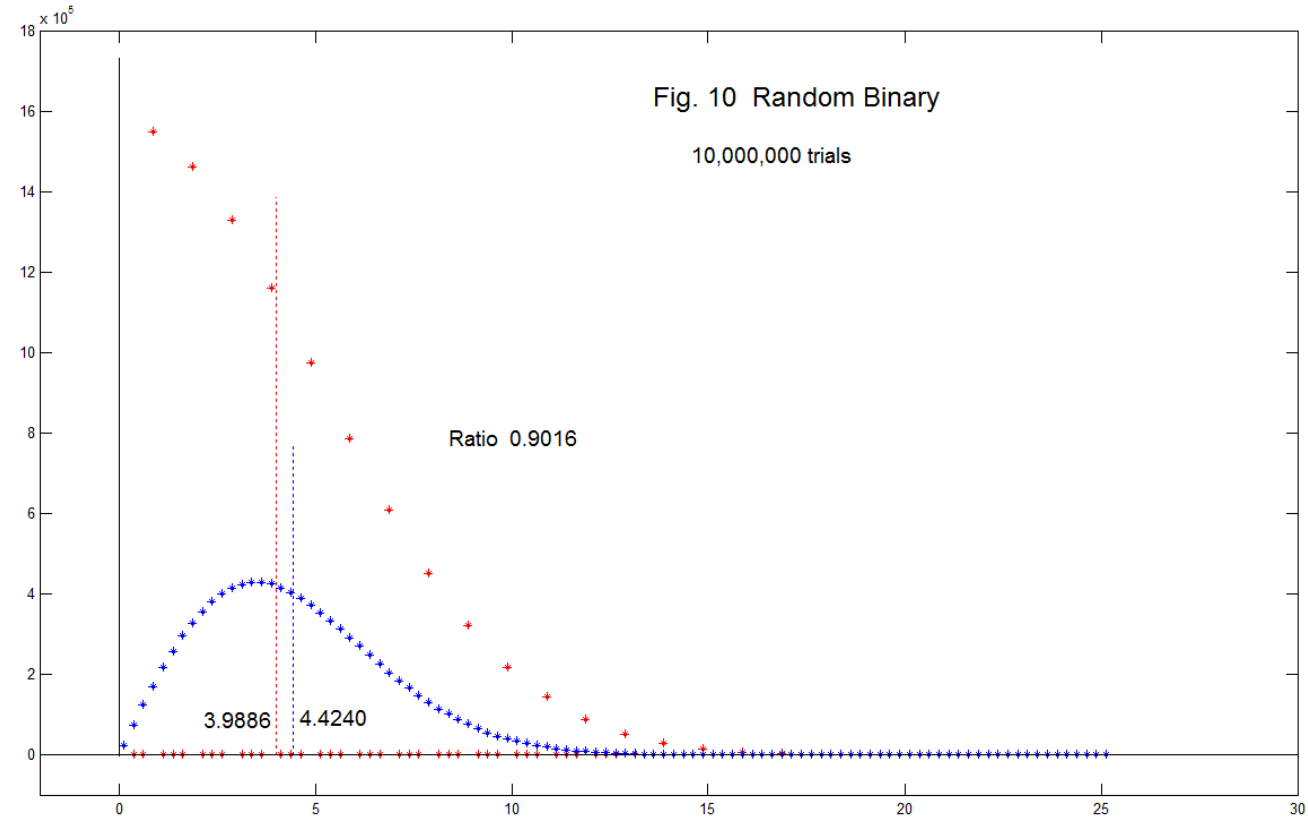

x 10^5
18
16
14
12
10
8
6
4
2
0
0
5
10
15
20
25
30
Fig. 10 Random Binary
10,000,000 trials
Ratio 0.9016
3.9886
4.4240

Fig. 9 shows the case where we use a normal distribution rather than a uniform one. We get essentially the same result. Because of the normal distributions, we truncated the plot on the right side a bit. The ratio of means is the same 90%. In Fig. 10, we have used a binary sequence, the sign of the uniform binary case times 1/2 . Because of this effective quantization of results to integers, 3 of 4 of the bins of the histogram are empty (the remaining ones 4 times larger. But the ratio of means is again 90%. Accordingly we find the result robust against changes of random distribution.

(7) RANDOM VS. DFT ANGLES

One more graph is needed here, Fig. 11, which will look almost exactly like Fig. 7. The difference here is that in Fig. 11, the angles were chosen at random. Specifically, for Fig. 7, the angles were set in Matlab to the DFT angles as:

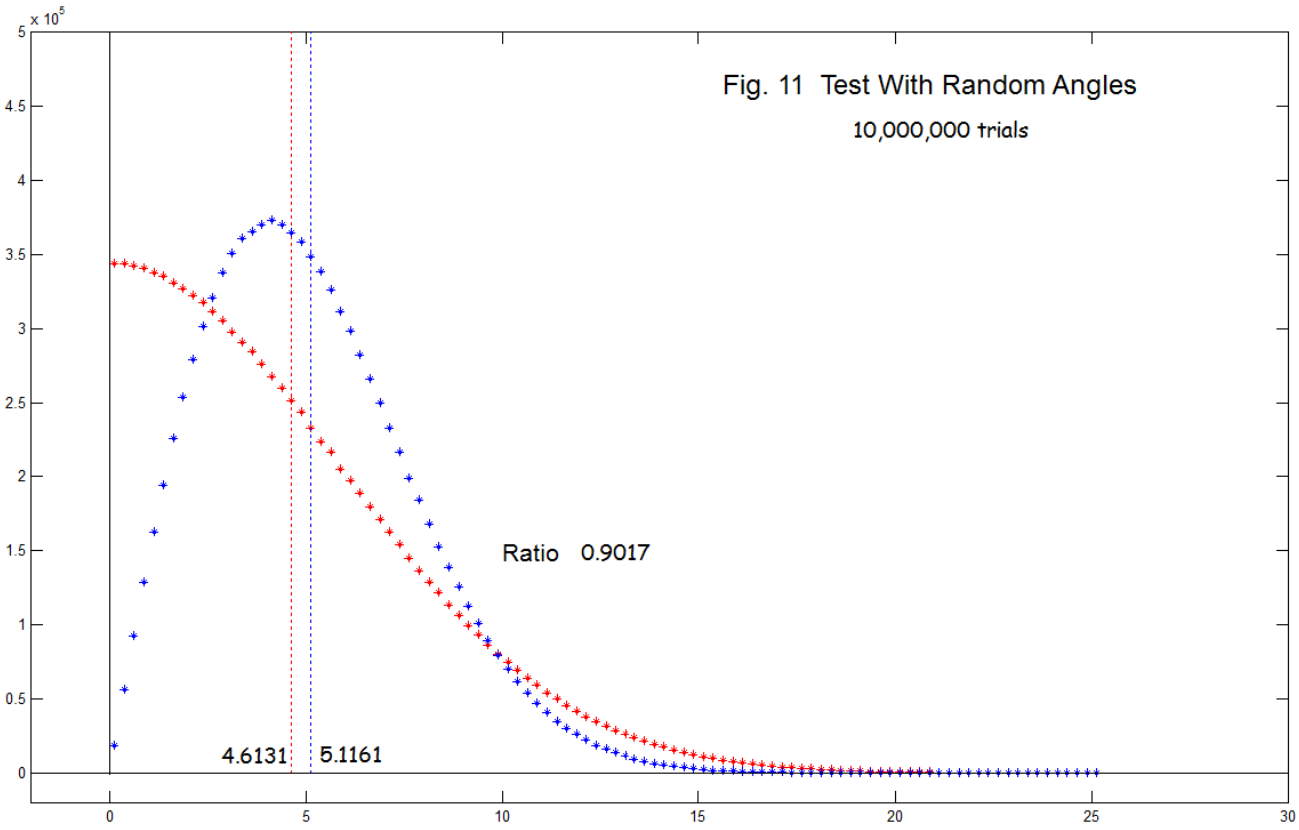
```
n=0:99

exp(-j*n*2*pi/N)
```

which is the same for each of the random signals. In contrast, for Fig. 11, they are set as:

```
exp(-j*rand(1,N)*2*pi)
```

separately for each random trial. Fig. 12 compares the set of 100 equally spaced angles corresponding to the DFT with one instance of 100 randomly chosen angles. The purpose of this is to justify Fig. 7 as having a Rayleigh distribution since the distribution of the angles is much the same. In Fig. 7, we have all the 100 possible angles, 3.6° apart, in sequence. In Fig. 11, there are 100 angles, randomly chosen.

Note well that Fig. 11 is not a plot of DFT magnitudes. Rather it is the magnitude of random walks in the complex plane. According to theory, we have a Rayleigh distribution if we have random samples that have real and imaginary parts that are what is called “IID” (Independent and Identically Distributed). If angles are predetermined (as with the DFT), it would seem that the real and imaginary parts are not independent. With a particular known given angle φ, the complex sample’s components depend only on the magnitude. Change the magnitude, change both real and imaginary. This is true regardless of where the angle came from (Fig. 13).


x 10^5
Fig. 11 Test With Random Angles
10,000,000 trials
Ratio 0.9017
4.6131
5.1161

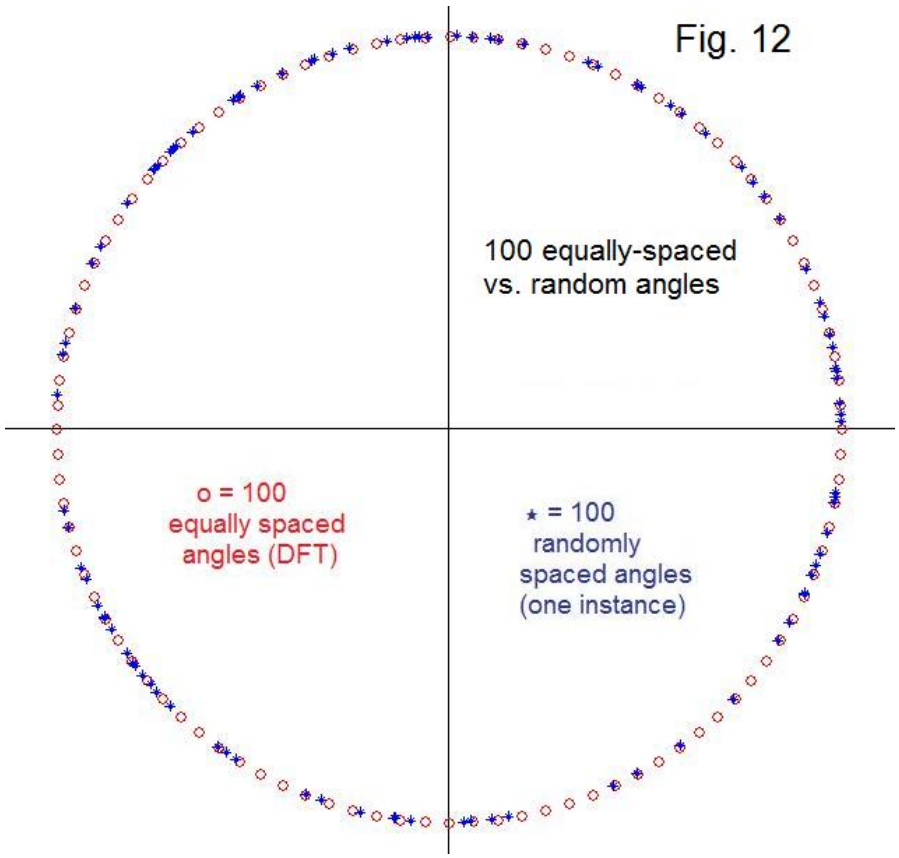


Fig. 12
100 equally-spaced
vs. random angles
o = 100
equally spaced
angles (DFT)
⋆ = 100
randomly
spaced angles
(one instance)

As we mentioned above, one guideline as to whether a complex random number has a Rayleigh distribution for the magnitude seems to be that the real and imaginary parts are IID, independent and identically distributed. We could do this by drawing a random number for the real part, and another draw for the imaginary part. Two random numbers. We also see no problem in drawing a random number for the magnitude, and a second random number for the angle. Again, two random numbers. This is what we did in obtaining Fig. 11. Since it seems very nearly identical to Fig. 7, we might want to claim that even though the angles in Fig. 7 are known (not random), the distributions are the same (Fig. 12), or at least, the information about the angle does not matter to the distribution of magnitudes (Fig. 13). Any strong opinions here?

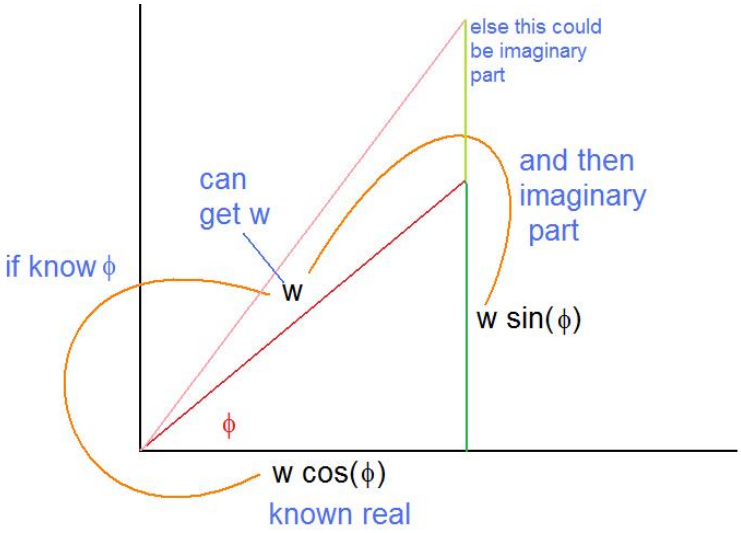
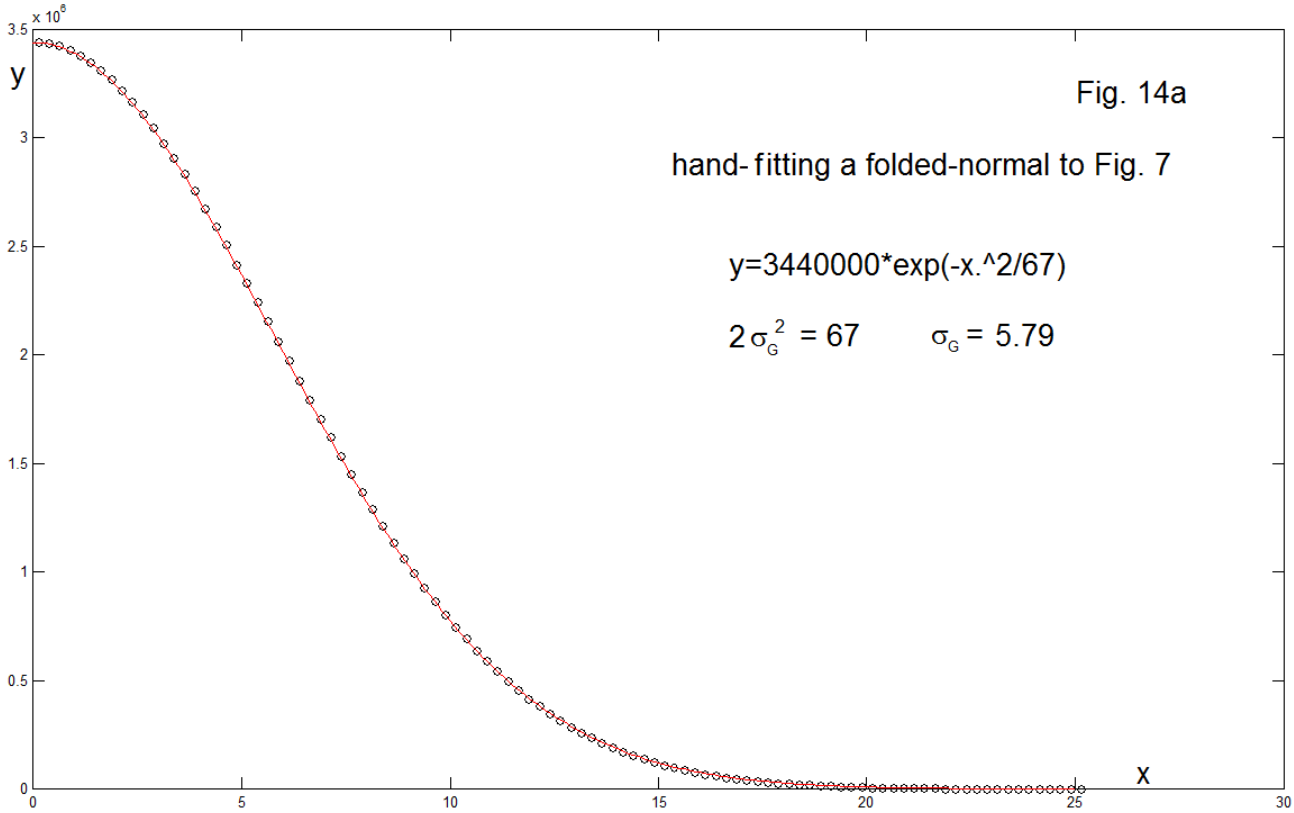


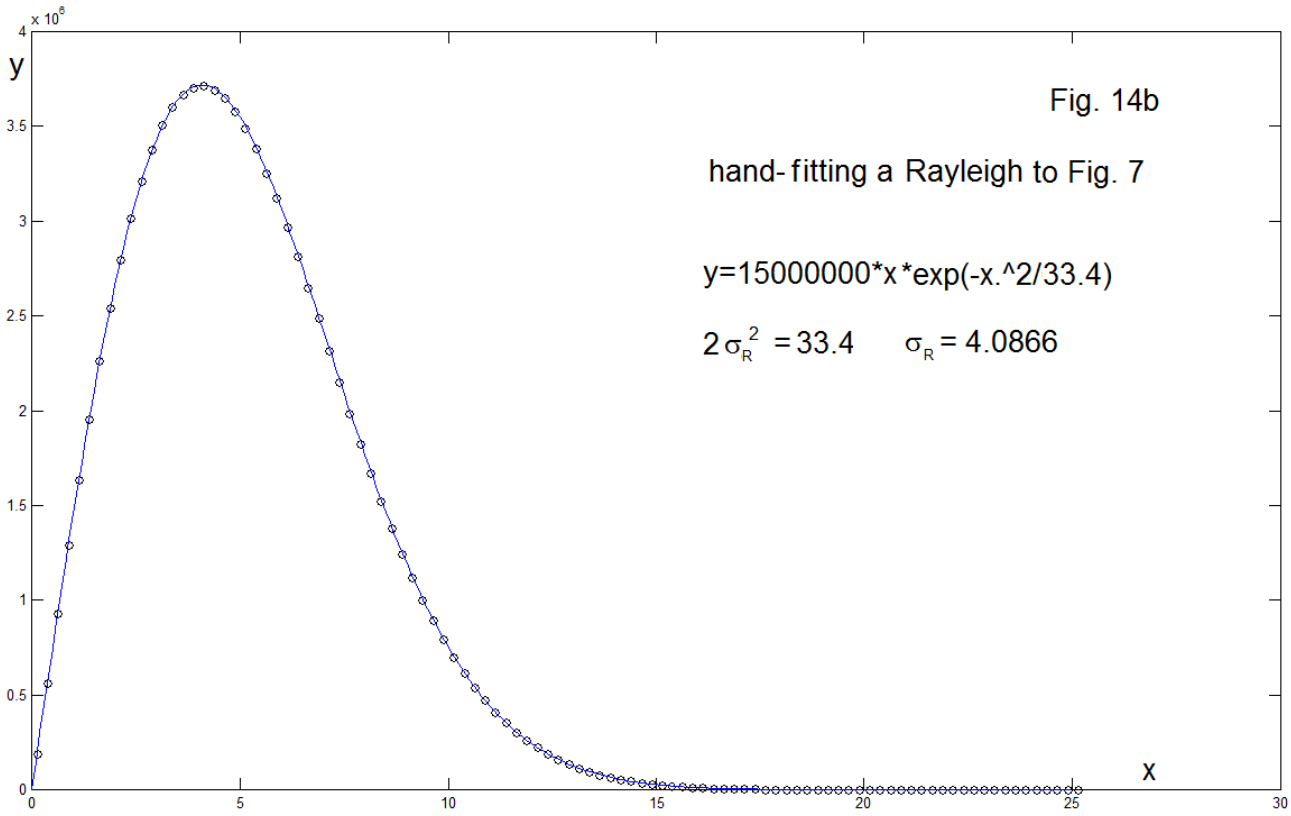
Fig. 13

Does knowing the angle matter? Yes, if we know ϕ we can compute the magnitude w from the real part, and then the imaginary part. If we are not given ϕ, the imaginary part could be anything. Likewise if we know the real and imaginary parts, we can compute the angle ϕ, but this will not in general match any preselected angle.

(8) COMPLETING THE CALCULATIONS

Proceeding with the assumption that we are dealing with a phenomenon which has an origin in a difference of probability distribution, we can calculate this ratio of 90% that keeps appearing. This is not as easy as one might suppose, because various sources have different definitions of the parameters involved. In consequence, we want to use the most basic mathematical forms of the distributions, and fit the curves to the experimental data (i.e., to Fig. 7). Once the curve is fit, we can check some numbers and hopefully know how to plug these into formulas for the means. Fig. 14a shows the fit for the folded normal distribution, and Fig. 14b the corresponding case for the Rayleigh distribution (fit to Fig. 7).


x 10^6
y
x
Fig. 14a
hand-fitting a folded-normal to Fig. 7
y=3440000*exp(-x.^2/67)
$2\sigma_G^2 = 67$
$\sigma_G = 5.79$


x 10^6
y
x
Fig. 14b
hand-fitting a Rayleigh to Fig. 7
y=15000000*x*exp(-x.^2/33.4)
$2\sigma_R^2 = 33.4$
$\sigma_R = 4.0866$

Here the Gaussian (folded normal) should have the general form (Wickipedia):

$$y_G = A_1 \exp(-x^2/2\sigma_G^2) \tag{4a}$$

and this we fit (by hand – by eye) to $A_1 = 3440000$ and $\sigma_G = 5.79$. The parameter A_1 is only important to the curve-fitting height. It is very large only because of the very large number of signals in the histogram. All that matters is σ_G. Notice the we have been careful here NOT to assume the σ's are the same, hence the subscript G here.

For the Rayleigh, the form is:

$$y_R = A_2 \, x \exp(-x^2/2\sigma_R^2) \tag{4b}$$

and this we fit with $A_2 = 1500000$ and $\sigma_R = 4.0866$. We noted that the curve in Fig. 14a had a number 67 where Fig. 14b has a number 33.4. In fact, we needed a factor of 2 to appear.

The formulas for the means are:

$$\mu_G = (2/\pi)^{1/2} \sigma_G \tag{5a}$$

and

$$\mu_R = (\pi/2)^{1/2} \sigma_R \tag{5b}$$

Yes, the π/2 factors are inverted between the two. Thus the ratio between the means is:

$$R = \mu_G/\mu_R = (2/\pi)(\sigma_G/\sigma_R)^{1/2} \tag{6}$$

and we note (from our curve fitting), that

$$\sigma_G^2/\sigma_R^2 = 2 \tag{7}$$

so

$$R = (2)^{3/2}/\pi = 0.9003163 \tag{8}$$

which is in excellent agreement with what we have found from our plots

The number R, 2 square root of 2 divided by π, actually appeared early in our study. This was because in thinking about averages here, and the fact that any magnitude is shared by sine and cosine components, the average value of a sine (2/π) appeared, as did √2 as the result of combining at an average angle of 45° (also relating the RMS and peak values of a sinewave). It's elementary stuff in AC waveforms. Can we make it fit? So the appearance of what seemed to be the correct value of R, in terms fundamental constants

was exciting. The apparent 90% was not just a number, that might have been exactly 0.9, based on our experiments, but something within experimental accuracy, that was much more fundamental. Next all that would be required was the right handwaving!

Great, but the problem was that the handwaving was backward. X(1) should have been smaller than X(0). So onward - and eventually the means of distribution. The warning was there, and perhaps still is: the numbers coming out right is essential, but not a proof. Hence an uneasiness about the findings. The current author appreciates math, but prefers that it be accompanied by an intuitive understanding as well. To paraphrase someone I once read, but don't remember exactly (Tukey, von Neumann, Hamming ?), a mathematical proof is best when presented such that all see that the result as inevitable. Comments are invited.

(9) SOME CONCLUSIONS

It seems we have gained some insight into why the spectrum drops to 90% at one or both ends. The points on the end or ends obey a different distribution. This is progress, but in a sense, it does not do a great deal more than move the problem to a new question – why do they obey different distributions? Since the end points are the "odd men out" perhaps we could understand them as special (limiting) cases of the Rayleigh distribution? And, we are talking about the DFT here, which we are forcing to be our spectrum analysis tool.

What would happen with continuous white noise? There is still a limiting case at DC, but it is now only a single point on a continuous line, and there is no notion of sampling frequency in this case. Is DC perhaps special in the discrete case as just another multiple of the sampling rate (zero times the sampling rate)?

Too many questions still remain here. The value of what we have presented here is perhaps just to let people who encounter the dips know that others have seen them and made follow-up studies. That is, people experimenting with Matlab's ***rand*** function must first get past the mistaken notion that any individual instance of a random signal should be flat (let alone each and every one), and instead to do an appropriate measure of averaging. Thus realizing the need for averaging, it would likely then be a disappointment if yet another unexpected phantom shows up at the door. Hence we need the discussion here to be "out there".

(10) SOME PROGRAMS

Here are a few Matlab programs that were used in this study and for making figures here.

PROGRAM 1: bigtest.m

This program (or similar) made the distribution figures:

```
% bigtest
% Detailed Histograms
NS=100                  % length 100 random sequences
%
n=0:(NS-1);             % set up DFT exponentials for k=1
e=exp(-j*n*2*pi/NS);    % exponential for k=1 DFT

s0=0;     % sum corresponding to DFT 0
s1=0;     % sum corresponding to DFT 1
S0=0;     % sum corresponding to abs DFT 0
S1=0;     % sum corresponding to abs DFT 1

H0=zeros(1,101);   % zero histogram sum for multiple histograms
H1=zeros(1,101);   %   same for H1

% run with TWO nested "for loops" to avoid an oversized array
%   for the histogram.  Multiple (many) Matlab "hist" outputs are
%   summed and stored in H0 and H2, and just plotted at the end
Nk=1000;
Nm=10000;         % total trials is Nk*Nm
for k=1:Nk
    for m=1:Nm

      w=2*(rand(1,NS)-.5);                % uniform distribution
        % w=(1/2)*sign((rand(1,NS)-0.5)); % random +1/2 or -1/2
        % w=randn(1,NS);                  % rand normal
        % w=sign(randn(1,NS));            % +1 or -1

        % e=exp(-j*rand(1,NS)*2*pi);  % override DFT if want
                                      %random angles
```

```
      X0=sum(w);           % X(k=0) of individual w
      X1=sum(w.*e);        % X(k=1) of individual w

      s0=s0+X0;            % sum for mean of endpoint
      s1=s1+X1;            % sum for mean of endpoint
      S0=S0+abs(X0);       % sum for mean of abs values
      S1=S1+abs(X1);       % sum for mean of abs values

      SS0(m)=abs(X0);      % store this X(0)for histogram
                             %of this 10000 (loop m)
      SS1(m)=abs(X1);      % store this X(1)for histogram
                             %of this 10000 (loop m)
    end      % end m loop

% compute histograms of this 10000 and add to previous k loops
%
H0=H0+hist(SS0,0.125+[0:.25:25]);
H1=H1+hist(SS1,0.125+[0:.25:25]);
%
end    % end k loop

%  DISPLAY……
NT=Nk*Nm;
MeanX0=s0/NT
MeanX1=s1/NT
MeanAbsX0=S0/NT
MeanAbsX1=S1/NT
RatX0X1=MeanAbsX0/MeanAbsX1

figure(1)
plot(0.125+[0:.25:25],H0,'r*')
hold on
plot([MeanAbsX0 MeanAbsX0],[-10 1000000],'r:')
plot(0.125+[0:.25:25],H1,'b*')
plot([MeanAbsX1 MeanAbsX1],[-10 1000000],'b:')
plot([0 0],[-1000,1000000],'k')
plot([-2 30],[0 0],'k')
hold off
axis([-2 30 -NT/400 NT/20])
figure(1)
```

PROGRAM 2: randomvector.m

This program made Fig. 4, the random walk display – not with all the details.
Because we don't know where the walk will end, some hand-setting of the axes is probably necessary.

```
% randomvector
k0=zeros(1,20);
k1=zeros(1,20);

w=2*(rand(1,20)-0.5);
k0(1)=w(1);
k1(1)=w(1);
for n=1:19
   k0(n+1)=k0(n)+w(n+1);
   k1(n+1)=k1(n)+w(n+1)*exp(-j*2*pi*n/20);
end
figure(1)
plot([-100 100],[0 0],'c')
hold on
plot([0 0],[-100 100],'c')

plot(k0+0.000001*ones(1,20)*j,'r*')
plot(k0(20),0,'ok')
plot(k0(1),0,'ob')

plot(k1,':g')
plot(k1,'*g')
plot(k1(20),'ok')
plot(k1(1),0,'ob')

hold off
axis([-4 4 -4 4])
figure(1)
```

PROGRAM 3: whiteevolve.m

This program made Fig. 6

```
% whiteevolve
clf
NS=100     % length of signals
NT=1000    % Number of iterations

SS0=zeros(1,NT+1);
SS1=zeros(1,NT+1);
SS2=zeros(1,NT+1);
SS3=zeros(1,NT+1);
SS4=zeros(1,NT+1);
MM=zeros(1,NT+1);

for kk=2:NT+2
   w=2*(rand(1,NS)-.5);
   W=abs(fft(w));
   M=abs(sum(w));
   SS0(kk)=(SS0(kk-1)*(kk-1)+W(1))/kk;
   MM(kk)=(MM(kk-1)*(kk-1)+M)/kk;        % mean
   SS1(kk)=(SS1(kk-1)*(kk-1)+W(2))/kk;
   SS2(kk)=(SS2(kk-1)*(kk-1)+W(3))/kk;
   SS3(kk)=(SS3(kk-1)*(kk-1)+W(4))/kk;
   SS4(kk)=(SS4(kk-1)*(kk-1)+W(5))/kk;
end

figure(1)
hold on
plot([1:NT],SS0(3:NT+2),'k')
plot([1:NT],MM(3:NT+2),'y:')
plot([1:NT],SS1(3:NT+2),'r')
plot([1:NT],SS2(3:NT+2),'g')
plot([1:NT],SS3(3:NT+2),'b')
plot([1:NT],SS4(3:NT+2),'c')
hold off
axis([-10 NT+10 -.5 8])
figure(1)
```

PROGRAM 4: angledisplay.m

This program made the Fig. 12, a display of regularly spaced (DFT) angles and random angles.

```
% angledisplay
clf
sa=(360/100)*2*pi/360;
a=0;
for k=1:100
   a=a+sa;
   vdft(k)=exp(-j*a);
   vran(k)=exp(-j*rand*2*pi);
end
figure(1)
hold on
plot([0 0],[-5 5],'k')
plot([-5 5],[0 0],'k')
plot(real(vdft),imag(vdft),'or')

plot(real(vran),imag(vran),'*b')

hold off
axis([-1.2 1.2 -1.2 1.2])
axis('square')
figure(1)
```

Additional programs for other figures are likely easy to recreate. The code however is available upon request.

* * * * * * * * * * *