The Discrete Fourier Transform Illusion

The Fourier Transform is a mathematical technique named after the famed French mathematician Jean Baptiste Joseph Fourier 1768-1830. Today, the Fourier Transform is widely used in science and engineering in digital signal processing. The application of Fourier mathematical techniques is prevalent in our everyday life from everything from television to wireless telephones to satellite communications.

Stocks & Commodities Magazine is no stranger to Fourier analysis. Back in 1983 Jack Hudson and Anthony Warren published a number of articles on using the Fourier Transform and the Fast Fourier Transform (FFT) in analyzing stock prices.

Here we will take a different approach and will examine how the Discrete Fourier Transform (DFT) and it's modern implementation called the Fast Fourier Transform (FFT) can be applied to the S&P 500 daily index.

What is the DFT?

The overview is that the mathematical technique called the DFT takes a discrete time series of n equally spaced samples and transforms or converts this time series through a mathematical operation into set of n complex numbers defined in what is called the frequency domain. Why in the heck would we what to do that? Well it turns out that we can do all kinds of neat analysis tricks in the frequency domain which are just to hard to do, computationally wise, with the original time series in the time domain. If we make the assumption that the time series we are examining is made up of signals of various frequencies plus noise, than in the frequency domain we can easily filter out the frequencies we have no interest in and minimize the noise in the data.

A simple example will show what I mean.

Let us define the following signal:

sig(i) = 2*sin(i*12*pi/256) + 1.5*cos(i*8*pi/256) + 1 + 0.025*i for i=0 to 255

A graph of this signal would look like:

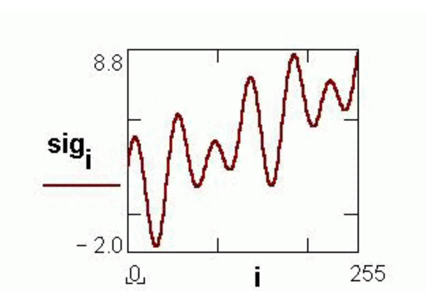

Figure 1, Constructed Signal vs time (i)

This is a signal that has a constant value, a trend and two oscillating frequency components.

Let's add some noise to the signal:

$$\mathbf{sig2(i) = sig(i) + rnd(10) - 5 \quad for\ each\ i,\ i=0\ to\ 255}$$

where **rnd(10)** is a random number between 0 and 10.

A graph of the signal and noise would look like.

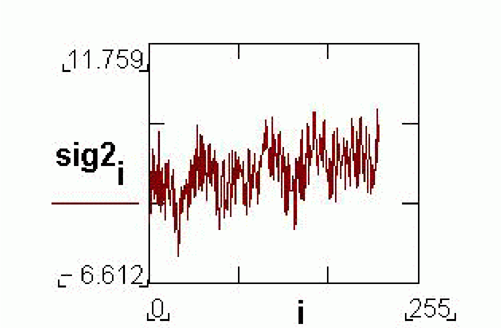

Figure 2, Signal with noise added

With the added noise, the signal has all but disappeared.

Let's take the Fast Fourier Transform of Sig2 and see what it looks like in the frequency domain. Let **f** be the **fft** of the signal **sig2** and let $|f_j|$ be the magnitude the individual frequency components.

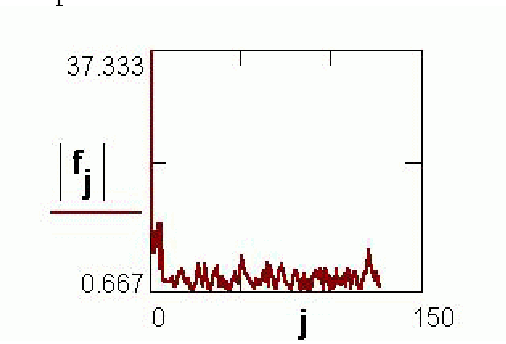

Figure 3, Frequency Magnitude of Sig2

This is not too helpful! What happened? This is an example of what happens when the trend and series average are not taken out of the time series before the **FFT** is done. The trend and the average completely swamp the frequency domain such that none of the characteristics we are looking for can be found.

Taking out the trend and the average,

$$\mathbf{sig3(i) = sig2(i) - 1 - 0.025*i \quad for\ each\ i,\ i=0\ to\ 255.}$$

Taking the **FFT** of **Sig3** . Let **f3** be the **FFT** of the signal **sig3** and let **|f3$_j$|** be the magnitude the individual frequency components.

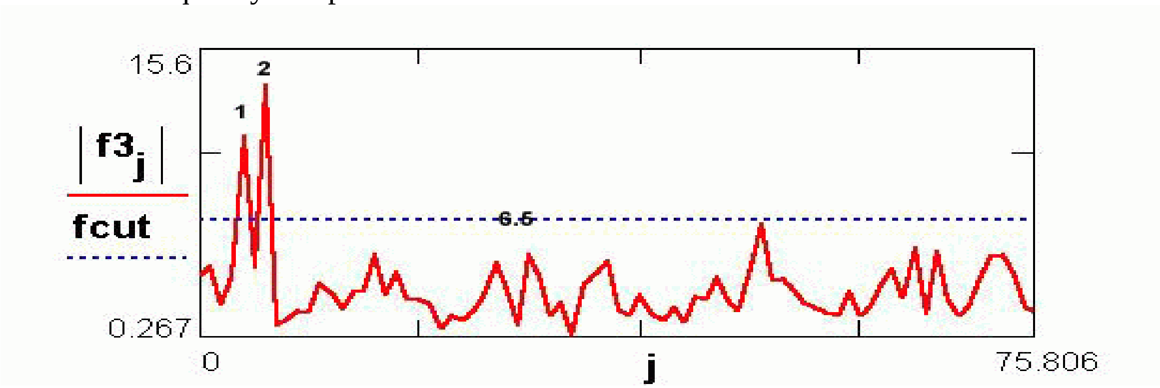

Figure 4, Frequency Magnitude chart of Sig3

Now we can see the two clear frequency peeks 1 and 2 in the FFT frequency magnitude chart.

We can filter the noise out by only accepting those frequencies whose magnitudes are greater than fcut=6.5 . This noise filter is a very simplified form of the Optimal Wiener Noise Filter. Although this noise filter is *not* optimal it will suffice for our purposes.

Thus let us construct a new frequency transform defined by the following equation:

if(|f3(j)|>fcut) then g3(j)=f3(j)
if(|f3(j)|<=fcut) then g3(j)=0

Now we take the inverse **FFT** of **g3**, **ig3=ifft(g3)**. Add back the trend and constant and we have the following graph.

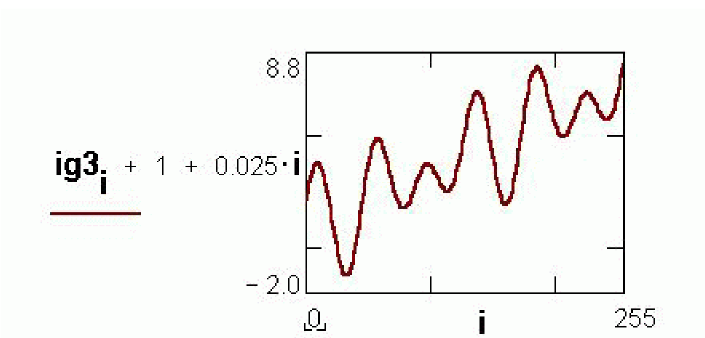

Figure 5, Inverse FFT of the Noise Filtered Signal g3

Comparing the original signal, ***sig***, before the noise was added:

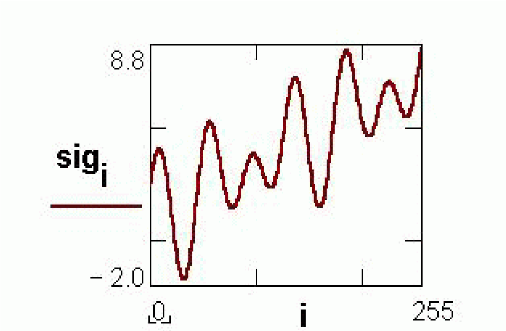

Figure 1, Original Constructed Signal vs time (i)

We see that we have successfully filtered out the added noise and retrieved almost all of the original noiseless signal.

Hey not too bad, but how can we apply these techniques to the S&P500 stock index?

FFT Noise Filtering of The S&P500 Index

To examine the S&P500 index we will start with three years of daily prices from 12/14/95 to 12/18/98. This creates a series of 762 data points. There is no magic attached to the selection of three years of daily data. I just selected this time period for example purposes, and because it contained the October 1997 and August 1998 stock market selloffs.

Let sp(i), i=1 to 762 represent the S&P500 daily closing price series. As we observed above, before we can FFT this series we have to subtract the trend and the intercept.

Let use define **Ave** as the average of the sp(i), **tbar** as the average of the i's, sumtt as the sum of the i^2's, sumtx as the cross sum of the sp(i) and i's, and **N** equal to 762.
Then,

ave = (Σsp(i))/N for i=1 to N

tbar= (N+1)/2

Sumtt = N*(N*N-1)/12

Sumtx = Σ(sp(i)-ave)*(i-tbar) for i=1 to N

The Slope or Trend of the **sp(i)** can now be calculated as

Slope =sumtx/sumtt

We can now create a detrended series to input into the FFT as,

x(i) = sp(i)-ave-slope*(i-tave) for i=1 to N

There is one more thing we have to do before we can input x(i) into the FFT routine. The FFT requires that the number of data points input into it be a power of 2. The next highest power of 2 larger than N=762 is 1024. What do we do?. Well, all we have to do is pad the end of x(i) for i = 763 to 1024 with zeros. The zero padding of x(i) for i = 763 to 1024 will not effect the frequency transform.

With the zero padded time series x(i), we now take the FFT of x.

Let **fftx** be the **FFT** of **x** and let $|\mathbf{fftx_j}|$ be the magnitude of the individual frequency components.

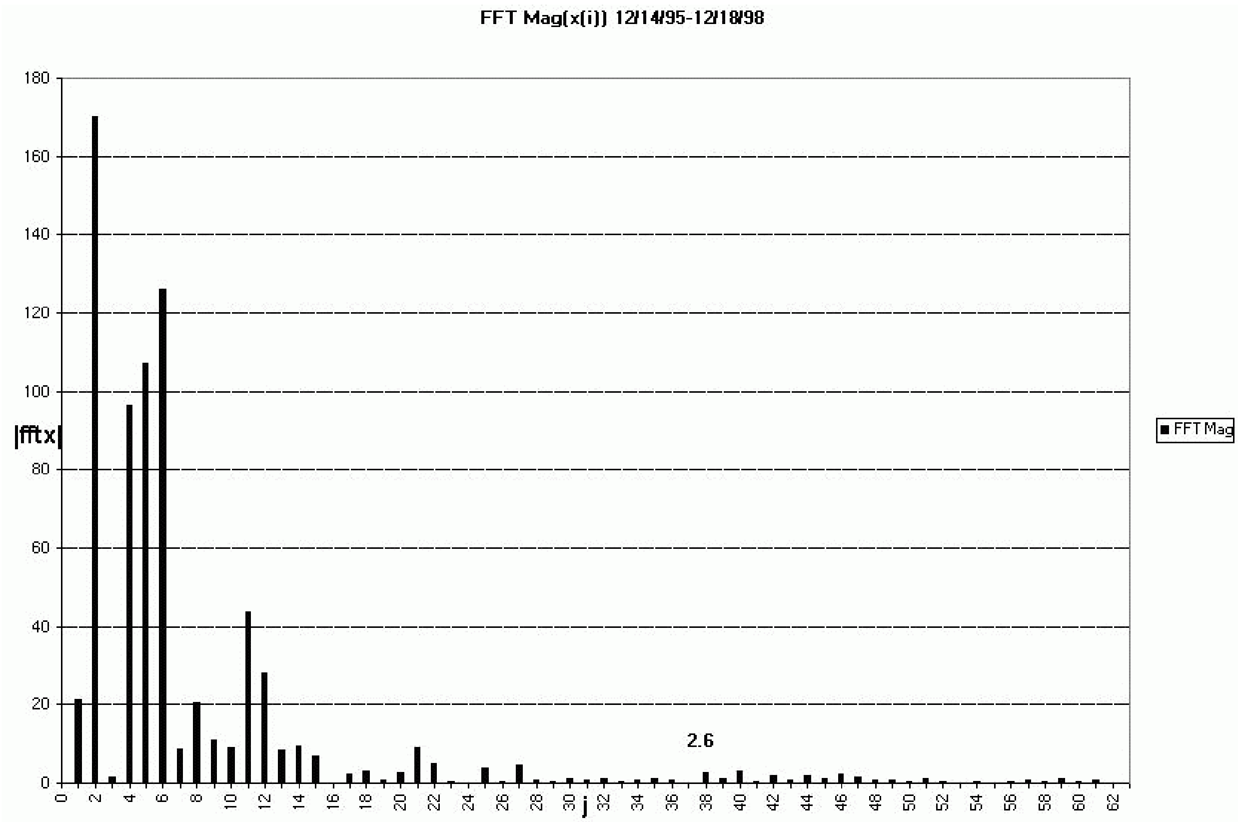

Figure 6, Frequency Magnitude Chart of Detrended S&P500 Index

Examining the frequency magnitude chart in figure 6 we will chose the cutoff frequency magnitude to be 2.5.

Thus we will create a new filtered frequency transform, **fftx2(j) j=0 to 1023** with the following equations:

if(|fftx(j)|>fcut) then fftx2(j)=fftx(j)
if(|fftx(j)|<=fcut) then fftx2(j)=0

We next perform an inverse FFT on the filtered frequencies fftx2, add back the slope and average and the results are shown in Figure 7.

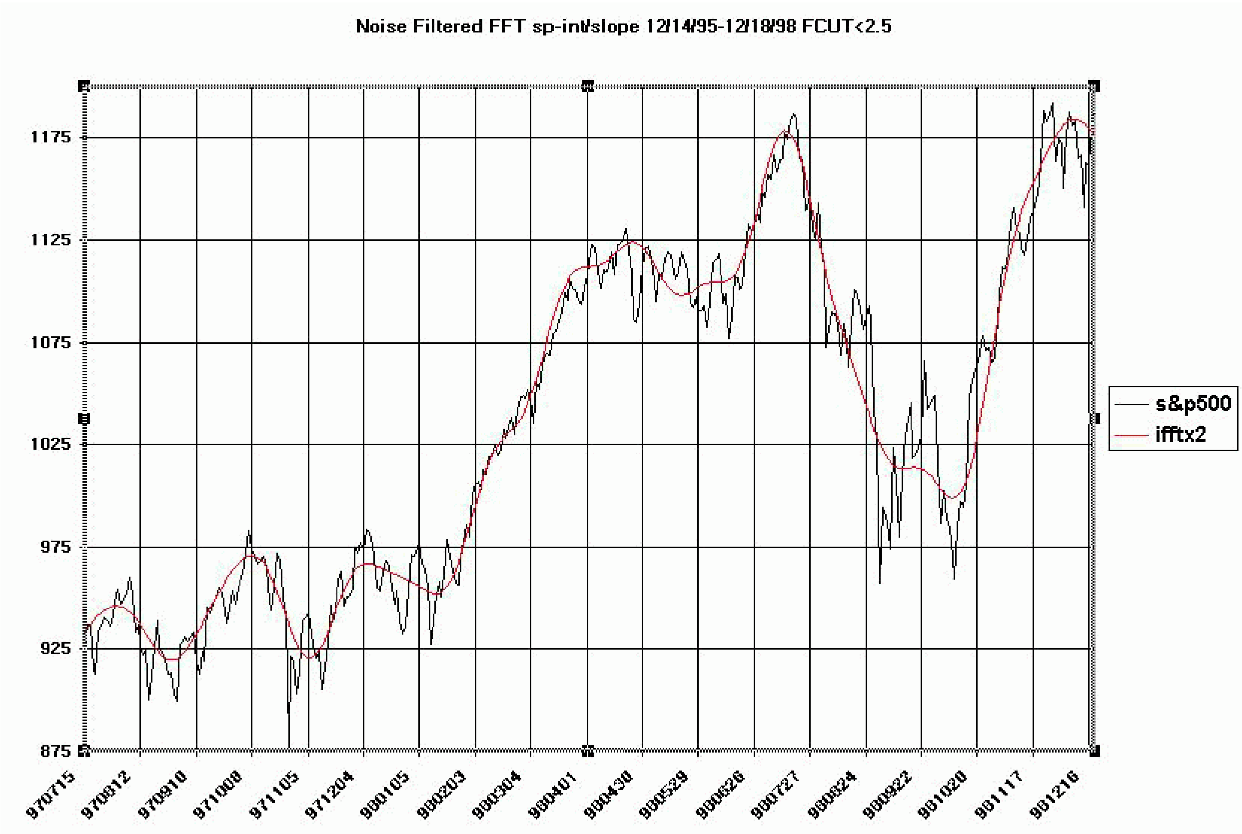

Figure 7, Inverse FFT of the Noise Filtered S&P500 Frequency Spectrum.

Examining Figure 7, we can see that the noise filtered inverse FFT of the S&P500 spectrum has smoothed out the noisy jiggles in the S&P500 chart and even ***leads*** the major tops in October 1997 and July 1998 of the S&P500! Is this the equivalent of stock market prediction nirvana at last? Nope! Read on to discover the flaws in this advanced version of the "Siren Call of Optimized Systems".

The Illusion Revealed

First it is observed that in the data series above used to create the FFT and subsequent noise filter, that every past data point has full knowledge of all future data points in the fixed 3 year time frame. When the FFT went to fit this data it already knew where all the tops and bottoms were. It's the job, so to speak, of the FFT to minimize the error between the points on the curve it generates and the points on the real data curve. Thus it's almost impossible not to get a good fit.

Let's examine what would happen in real time. We will look at the July 1998 top and subsequent bottom. The FFT curve in Figure 7 clearly shows a rising trend on 6/26/98. Figure 8 represents the noise filtered FFT on the S&P500 with the ending date of 6/26/98. That is, this is what you would have seen if you ran this analysis on 6/26/98.

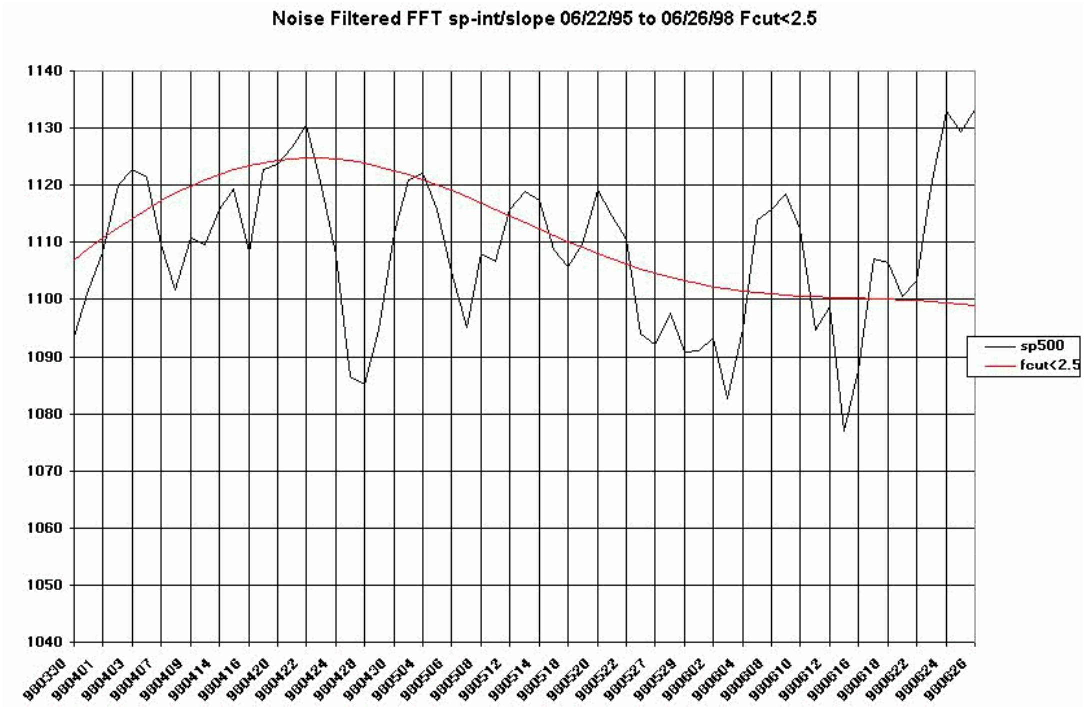

Figure 8, Inverse FFT of the Noise Filtered S&P500 Frequency Spectrum, End Date 6/26/98.

As one can see on 6/26/98 the FFT curve is moving down *not up* as shown in Figure 7. What about a week later on 7/2/98? Figure 9 shows the FFT curve with an end date of 7/02/98.

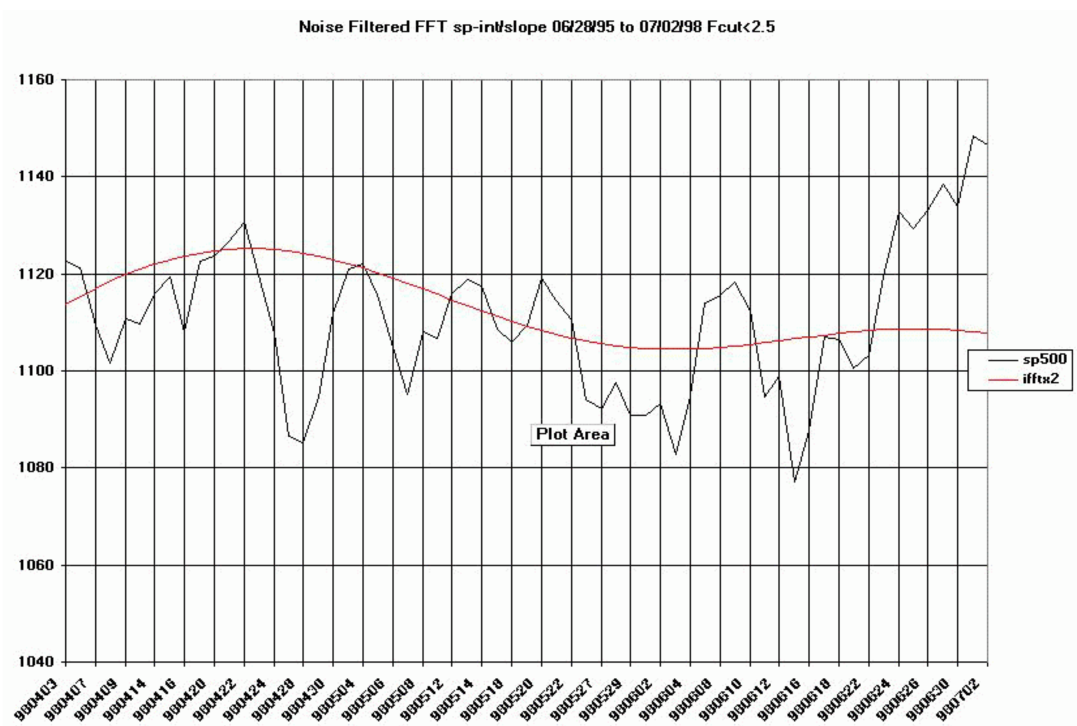

Figure 9, Inverse FFT of the Noise Filtered S&P500 Frequency Spectrum, End Date 07/02/98.

In Figure 7 the FFT curve is moving strongly up on 7/02/98. However the FFT curve in Figure 9 with the end date of 7/02/98 is only slightly up from it's low. The difference between these two curves illustrates the difference between generating a curve with no knowledge of the future as in figures 8 and 9 and generating a curve with full knowledge of the future as in Figure 7.

Figure 10 presents the FFT curve at the S&P500 peak on 7/17/98 with end date 7/17/98.

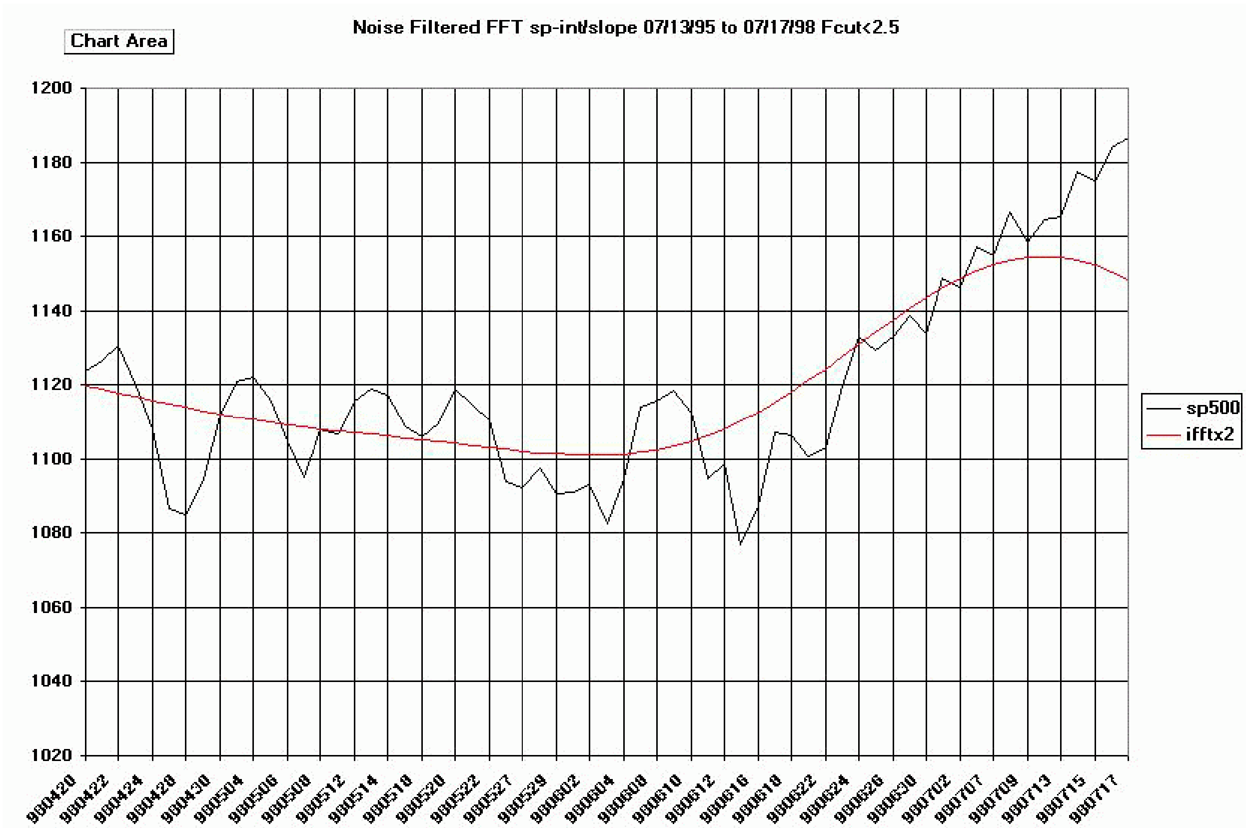

Figure 10, Inverse FFT of the Noise Filtered S&P500 Frequency Spectrum, End Date 07/17/98.

In Figure 10 the FFT curve with end date 7/17/98 clearly shows a down trend before the S&P500 starts it's downtrend from the 7/17/98 high. In this case the end point FFT correctly predicted the turning point of the S&P500. However, the 6/26/98 FFT curve also predicted a downturn in the S&P500 when the S&P500 did just the opposite by rallying +5%.

One last example. In Figure 7 the S&P500 rallied from a low of 960 on 8/31/98 to a high of 1066 on 9/23/98. On 9/29/98 the S&P500 backed off to 1049. In Figure 7 the FFT curve was flat during this first leg up from the 8/31/98 bottom and was moving down on 980929. Thus if we would have be able to use the FFT curve in Figure 7, we would have avoided the subsequent drawdown to the low on 10/8/98 from buying into that first rally. On Figure 7 it's real clear what we *should have done*. However, let's look at the FFT curve derived on 9/29/98 with only data on and before that date. Figure 11 represents this FFT curve.

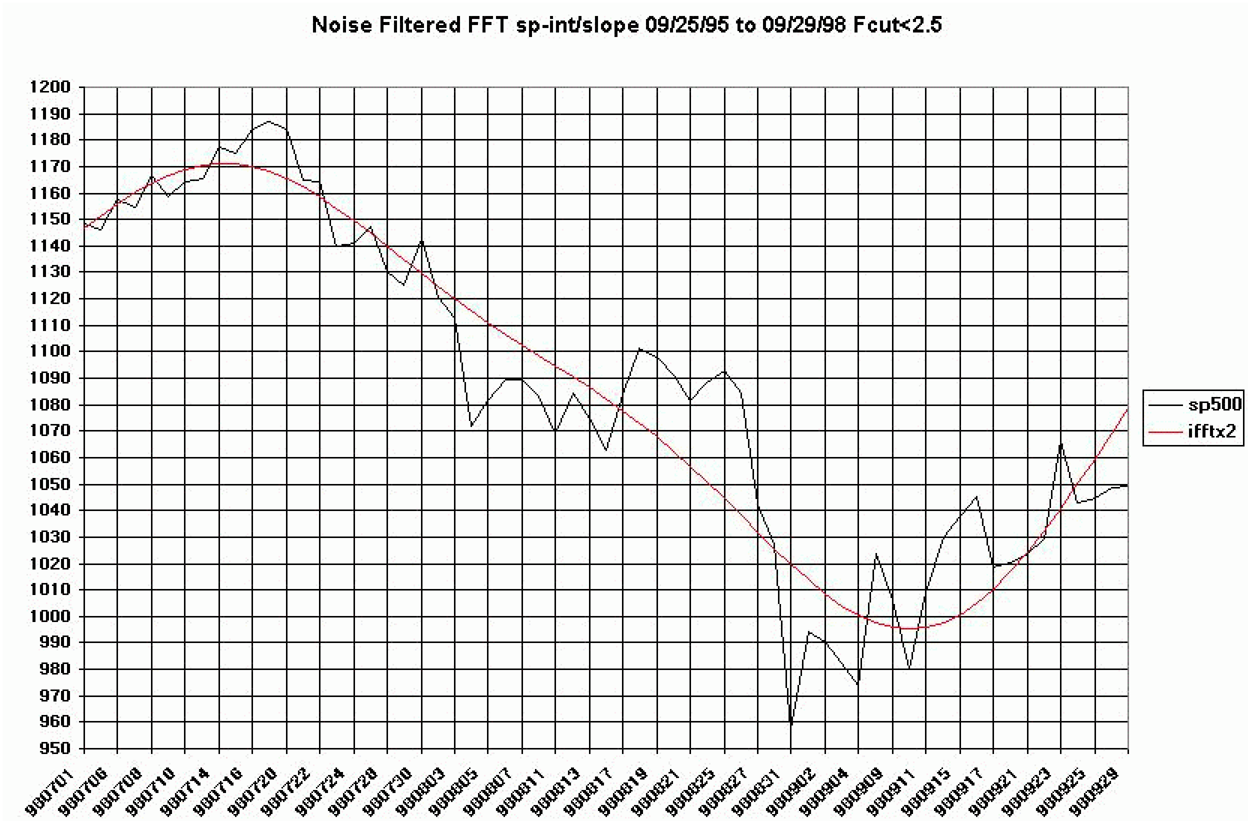

Figure 11, Inverse FFT of the Noise Filtered S&P500 Frequency Spectrum, End Date 09/29/98.

As we can see from Figure 11, if we performed the Noised Filtered FFT routine on 9/29/98 the FFT curve was moving strong up *not down* as in Figure 7. Two days later on 10/1/98 the S&P500 was down 6% to 986 from the 9/29/98 price of 1049. The real FFT curve on 9/29/98 gave no hint of the coming loss. Even the FFT curve derived on 10/6/98 data (not shown) was still moving strongly up and this was only 2 days before the final bottom!

Conclusion

It should now be obvious from the examples above that indicators, no matter the mathematical sophistication, should only be looked at on a walk forward basis. Examining how an indicator performed on the past data that it was optimized on or curve fitted on will only create an *illusion* of good performance that will not be recreated in real time.

After all this should we abandon this FFT technique? Not at all! There are many variables to test to see if we can make the FFT into a viable walk forward system. For instance we could create an "end point" noise filter FFT. That is, we can create a data window that slides forward one day at a time. For each data window we calculate and record the FFT curve end point. Since all these FFT curve end points were only derived from data before the end point, we can connect them, that is connect the dots, and generate a new end point FFT curve. This new end point FFT curve would be similar in construction to the "Cubic" curve that was generated in my "British Pound Cubed" article.
This new end point FFT curve will be the topic of my next article.

References:

Lyons, Richard G., "Understanding Digital Signal Processing", Addison-Wesley Publishing, 1997.

Meyers, Dennis [1998], "The British Pound, Cubed", *Stocks & Commodities*, Volume 16: November.

Meyers, Dennis [1998], "Surfing The Linear Regression Curve", *Stocks & Commodities*, Volume 16: May.

MathCad 8, Mathsoft, Inc., Cambridge MA, 1998, http://www.mathsoft.com

Info on Dennis Meyers

Dennis Meyers has a doctorate in applied mathematics in engineering. He is a member of the Chicago Board Options Exchange(CBOE), a private trader, and president of Meyers Analytics. His firm specializes in consulting for financial institutions and developing publicly available analytical software for traders. He can be reached (312) 280-1687, via his Web site at http://www.MeyersAnalytics.com or via E-mail at info@MeyersAnalytics.com.