SKINNING THE CAT

by John Ehlers

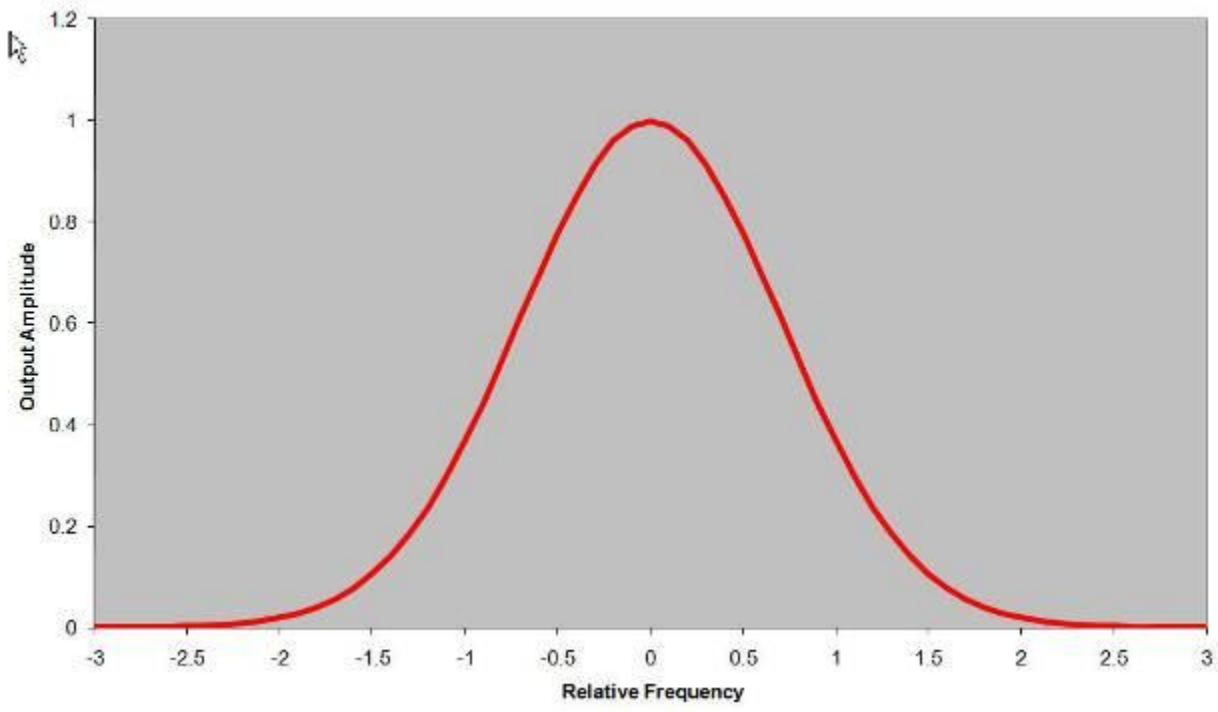
If you want to make your indicators and strategies adaptive to current market conditions it is crucial that you first measure the cycle periods that are present in the data. Given that you know the dominant cycle you can then use that information to dynamically adjust your computations. For example, you can set the observation period of an RSI to be half the dominant cycle. I have previously described a practical way to use Direct Fourier Transforms (DFT) to estimate the market spectrum[1]. But a DFT is not the only way to estimate the market spectrum – there is more than one way to skin a cat.

In this article I describe a way to use bandpass filters to make the spectral estimate. The advantage of using bandpass filters is that the selectivity and the filter transient response can be controlled. This is important because not all filters are good for trading since filters induce lag in the output and therefore cause a lag in you making your trading decisions. In general, the more complicated a filter, the more lag is induced. The simple 2 pole bandpass filter is nice because it provides no lag at the output for a steady state input signal at the frequency to which the bandpass filter is tuned.

First, let's understand some basics about bandpass filters. The response of the filter is shown in Figure 1. This means that when equal amplitude signals at all relative frequencies are applied to the input of the filter the filter rejects frequency components that are both higher and lower than the tuned frequency of the filter. The result that the frequency components at the output of the filter have their amplitudes shaped by the filter as shown in Figure 1. The region within relative frequencies -0.5 to +0.5 is the passband of the bandpass filter because the majority of energy getting through the filter falls in this range.

[1] John Ehlers, "Fourier Transforms for Traders", Stocks & Commodities, January 2007

Figure 1. Bandpass Filter Response to Normalized Frequencies

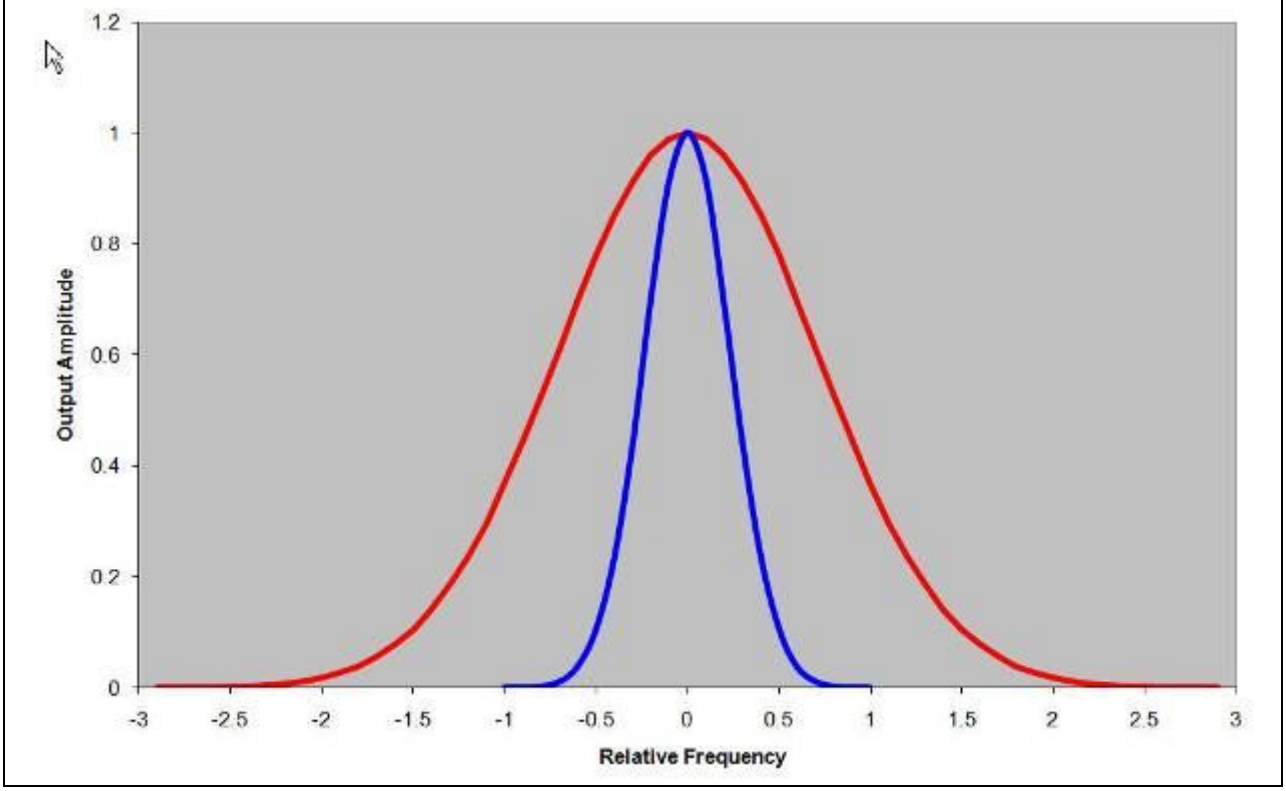


From a trading perspective, the rejection of both lower and higher frequency components has two effects. First, the lower frequencies (longer wavelengths) contain the trend information in the input waveform so that rejecting them detrends the data. Secondly, the higher frequencies contain rapid variations as a function of time so that rejecting them performs a smoothing operation on the data. What could be better than a filter that has no lag, detrends the data and also smoothes it? The big IF in this situation is that all this is true only IF the filter is tuned to the steady state dominant cycle in the data. Because the bandpass filter has frequency selectivity, we can make use of this fact to determine the dominant cycle in the data. I will describe how we do that later.

The bandpass filter can have an adjustable passband. Figure 2 shows the original filter having a passband of relative frequencies from -0.5 to +0.5 in red as well as a filter having a much greater selectivity in blue. That is, the passband is more narrow. In principal we can make the passband of a bandpass filter arbitrarily narrow, but we don't want to do this in trading because narrowing the passband has a negative influence on the transient response of the filter. For example, a bell is designed to be a highly selective bandpass filter. When the clapper strikes the bell, the bell continues to ring out at its tuned frequency. Thus, we hear the response to the disturbing event, but the event itself is lost. Further, the ringing continues long after the initial event has come and gone. Any highly selective filter has this ringing phenomena, and this ringing is not helpful in the interpretation of market action. Yes, we wish to extract the dominant cycle

information from the data, but we also wish that any transient filter response to a sudden change is quick to die down. The only way to do this is to widen the passband of the bandpass filter. Dealing with transients in most trading data is not a great problem because the dominant cycles tend to drift with time rather than jumping from one cycle period to another. On the other hand, if you are using intraday data you must be wary of gap openings because these represent severe transients that can distort a filter-based analysis.

Figure 2. Original and Narrow Passband Bandpass Filters



Basically, we want to take advantage of the selectivity of the bandpass filter to discern the dominant cycle in the data. After finding that dominant cycle we can dynamically tune any of a host of indicators. Figure 3 shows the basic concept of how we do this. If the input signal falls within the passband of Filter A, then the output of Filter A will have a larger amplitude than the output of Filter B. Conversely, if the input signal falls within the passband of Filter B, then the output of Filter B will have a larger amplitude than the output of Filter A. So, all we have to do to measure the spectrum of the input signal is to establish a contiguous bank of overlapping filters as shown in Figure 4 and measure the amplitude of the signal at the output of each filter.

Figure 3. Bandpass Filters Enable Signal Selection

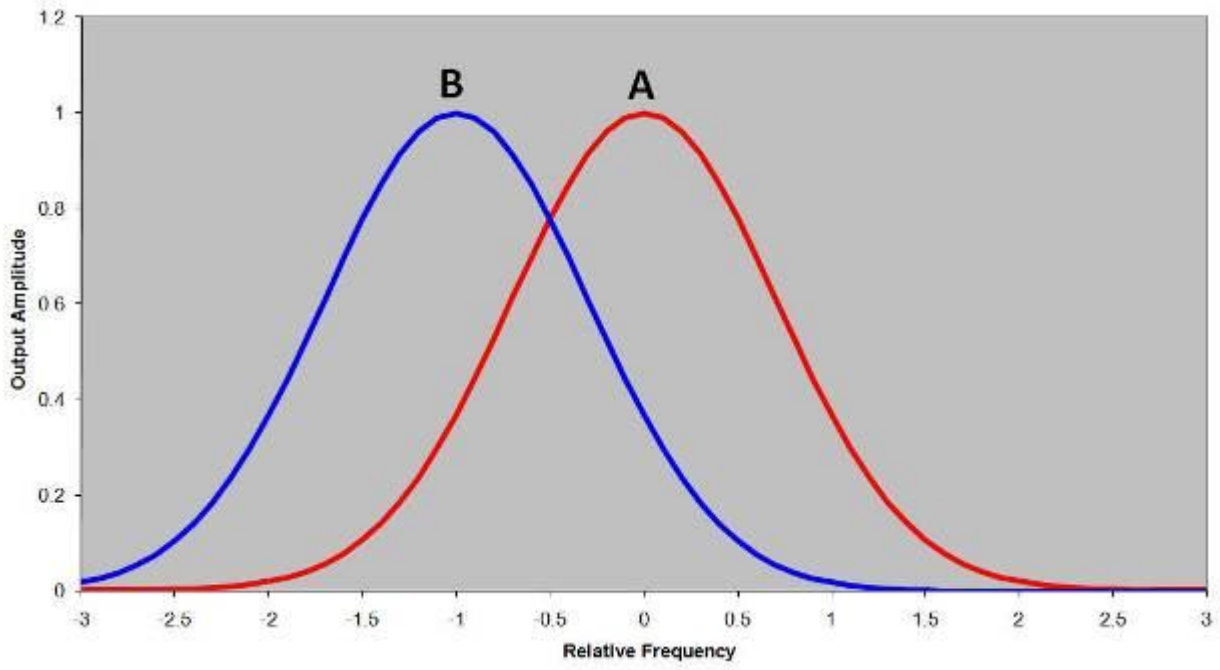
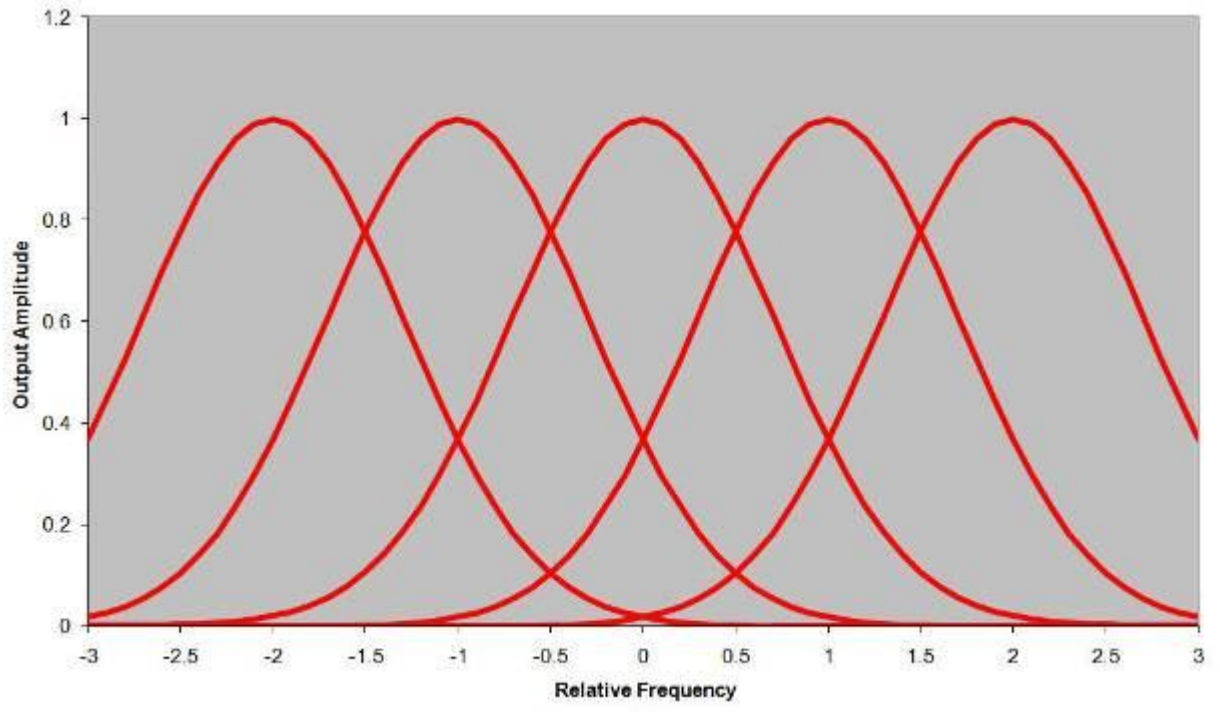


Figure 4. A Filter Bank Comprises a Channelized Receiver



Since the passband of each filter is relatively narrow, the signal at each output can be characterized as a sinewave with a slowly varying phase. You may recall from calculus

that $d(Sin(\omega t))/dt = \omega Cos(\omega t)$. Therefore, multiplying the rate of change of the output of each filter by Period/2π results in a Cosine wave being created at the output of each filter also. Both the Sine wave And Cosine wave have the generalized amplitude "A". Further, from the familiar trigonometric identity

$$A^2 = A^2 Sin^2(x) + A^2 Cos^2(x)$$

we have a measure of the amplitude of the signal at the output of each filter by squaring and adding the Sine wave and Cosine wave together. That is basically all there is to generating the spectrum of the market data. Some of the details are easier to explain with reference to the EasyLanguage code in Figure 5. When translating the code from EasyLanguage to other languages, several general features of the code should be noted. First, the entire code is completed for each bar and the value of the variables are retained from bar to bar. On the other hand, the values of arrays are not automatically indexed to each price bar. Secondly, EasyLanguage uses degrees rather than radians as arguments of its trigonometric function. Finally, the notation Price[1], i.e. with the square brackets, means the value of Price one bar ago with respect to variables. The square bracket notation also gives the position within an array.

Figure 5. EasyLanguage Code to Display the Spectrum Derived from a Filter Bank

```
Inputs:
        Price((H+L)/2),
        ShowDC(False);

Vars:
        delta(0.1),
        gamma(0),
        alpha(0),
        beta(0),
        N(0),
        Period(0),
        MaxAmpl(0),
        Num(0),
        Denom(0),
        DC(0),
        DomCyc(0),
        Color1(0),
        Color2(0),
        alpha1(0),
        HP(0),
        SmoothHP(0);

Arrays:
        I[50](0),
        OldI[50](0),
        OlderI[50](0),
        Q[50](0),
```

```
        OldQ[50](0),
        OlderQ[50](0),
        Real[50](0),
        OldReal[50](0),
        OlderReal[50](0),
        Imag[50](0),
        OldImag[50](0),
        OlderImag[50](0),
        Ampl[50](0),
        OldAmpl[50](0),
        DB[50](0);

alpha1 = (1 - Sine (360 /  40)) / Cosine(360 / 40);
HP = .5*(1 + alpha1)*(Price - Price[1]) + alpha1*HP[1];
SmoothHP = (HP + 2*HP[1] + 3*HP[2] + 3*HP[3] + 2*HP[4] + HP[5]) / 12;
IF CurrentBar < 7 Then SmoothHP = Price - Price[1];
IF CurrentBar = 1 THEN SmoothHP = 0;

delta = -.015*CurrentBar + .5;
If delta < .15 then delta = .15;

If CurrentBar > 6 Then Begin
        For N = 8 to 50 Begin
                beta = Cosine(360 / N);
                gamma = 1 / Cosine(720*delta / N);
                alpha = gamma - SquareRoot(gamma*gamma - 1);
                Q[N] = (N / 6.283185)*(SmoothHP - SmoothHP[1]);
                I[N] = SmoothHP;
                Real[N] = .5*(1 - alpha)*(I[N] - OlderI[N]) + beta*(1 + alpha)*OldReal[N] -
alpha*OlderReal[N];
                Imag[N] = .5*(1 - alpha)*(Q[N] - OlderQ[N]) + beta*(1 + alpha)*OldImag[N]
- alpha*OlderImag[N];
                Ampl[N] = (Real[N]*Real[N] + Imag[N]*Imag[N]);
        End;
End;
For N = 8 to 50 Begin
        OlderI[N] = OldI[N];
        OldI[N] = I[N];
        OlderQ[N] = OldQ[N];
        OldQ[N] = Q[N];
        OlderReal[N] = OldReal[N];
        OldReal[N] = Real[N];
        OlderImag[N] = OldImag[N];
        OldImag[N] = Imag[N];
        OldAmpl[N] = Ampl[N];
End;
```

```
MaxAmpl = Ampl[10];
For N = 8 to 50 Begin
        If Ampl[N] > MaxAmpl then MaxAmpl = Ampl[N];
End;

For N = 8 to 50 Begin
   IF MaxAmpl <> 0 AND (Ampl[N] / MaxAmpl) > 0 THEN DB[N] = -10*Log(.01 / (1 -
.99*Ampl[N] / MaxAmpl)) / Log(10);
        If DB[N] > 20 then DB[N] = 20;
End;
Num = 0;
Denom = 0;
For N = 8 to 50 Begin
        If DB[N] <= 3 Then Begin
                Num = Num + N*(20 - DB[N]);
                Denom = Denom + (20 - DB[N]);
        End;
        If Denom <> 0 Then DC = Num / Denom;
End;
DomCyc = Median(DC, 10);
If ShowDC = True Then Plot1(DomCyc, "DC", RGB(0, 0, 255), 0, 2);


For N = 8 to 50 Begin
   IF DB[N] <= 10 THEN Begin
      Color1 = 255;
      Color2 = 255*(1 - DB[N] / 10);
   END;
   IF DB[N] > 10 THEN Begin
      Color1 = 255*(2 - DB[N] / 10);
      Color2 = 0;
   END;
        If N = 8 Then Plot8(N, "S8", RGB(Color1, Color2, 0),0,5);
        If N = 9 Then Plot9(N, "S9", RGB(Color1, Color2, 0),0,5);
        If N = 10 Then Plot10(N, "S10", RGB(Color1, Color2, 0),0,5);
        If N = 11 Then Plot11(N, "S11", RGB(Color1, Color2, 0),0,5);
        If N = 12 Then Plot12(N, "S12", RGB(Color1, Color2, 0),0,5);
        If N = 13 Then Plot13(N, "S13", RGB(Color1, Color2, 0),0,5);
        If N = 14 Then Plot14(N, "S14", RGB(Color1, Color2, 0),0,5);
        If N = 15 Then Plot15(N, "S15", RGB(Color1, Color2, 0),0,5);
        If N = 16 Then Plot16(N, "S16", RGB(Color1, Color2, 0),0,5);
        If N = 17 Then Plot17(N, "S17", RGB(Color1, Color2, 0),0,5);
        If N = 18 Then Plot18(N, "S18", RGB(Color1, Color2, 0),0,5);
        If N = 19 Then Plot19(N, "S19", RGB(Color1, Color2, 0),0,5);
        If N = 20 Then Plot20(N, "S20", RGB(Color1, Color2, 0),0,5);
```

```
        If N = 21 Then Plot21(N, "S21", RGB(Color1, Color2, 0),0,5);
        If N = 22 Then Plot22(N, "S22", RGB(Color1, Color2, 0),0,5);
        If N = 23 Then Plot23(N, "S23", RGB(Color1, Color2, 0),0,5);
        If N = 24 Then Plot24(N, "S24", RGB(Color1, Color2, 0),0,5);
        If N = 25 Then Plot25(N, "S25", RGB(Color1, Color2, 0),0,5);
        If N = 26 Then Plot26(N, "S26", RGB(Color1, Color2, 0),0,5);
        If N = 27 Then Plot27(N, "S27", RGB(Color1, Color2, 0),0,5);
        If N = 28 Then Plot28(N, "S28", RGB(Color1, Color2, 0),0,5);
        If N = 29 Then Plot29(N, "S29", RGB(Color1, Color2, 0),0,5);
        If N = 30 Then Plot30(N, "S30", RGB(Color1, Color2, 0),0,5);
        If N = 31 Then Plot31(N, "S31", RGB(Color1, Color2, 0),0,5);
        If N = 32 Then Plot32(N, "S32", RGB(Color1, Color2, 0),0,5);
        If N = 33 Then Plot33(N, "S33", RGB(Color1, Color2, 0),0,5);
        If N = 34 Then Plot34(N, "S34", RGB(Color1, Color2, 0),0,5);
        If N = 35 Then Plot35(N, "S35", RGB(Color1, Color2, 0),0,5);
        If N = 36 Then Plot36(N, "S36", RGB(Color1, Color2, 0),0,5);
        If N = 37 Then Plot37(N, "S37", RGB(Color1, Color2, 0),0,5);
        If N = 38 Then Plot38(N, "S38", RGB(Color1, Color2, 0),0,5);
        If N = 39 Then Plot39(N, "S39", RGB(Color1, Color2, 0),0,5);
        If N = 40 Then Plot40(N, "S40", RGB(Color1, Color2, 0),0,5);
        If N = 41 Then Plot41(N, "S41", RGB(Color1, Color2, 0),0,5);
        If N = 42 Then Plot42(N, "S42", RGB(Color1, Color2, 0),0,5);
        If N = 43 Then Plot43(N, "S43", RGB(Color1, Color2, 0),0,5);
        If N = 44 Then Plot44(N, "S44", RGB(Color1, Color2, 0),0,5);
        If N = 45 Then Plot45(N, "S45", RGB(Color1, Color2, 0),0,5);
        If N = 46 Then Plot46(N, "S46", RGB(Color1, Color2, 0),0,5);
        If N = 47 Then Plot47(N, "S47", RGB(Color1, Color2, 0),0,5);
        If N = 48 Then Plot48(N, "S48", RGB(Color1, Color2, 0),0,5);
        If N = 49 Then Plot49(N, "S49", RGB(Color1, Color2, 0),0,5);
        If N = 50 Then Plot50(N, "S50", RGB(Color1, Color2, 0),0,5);
End;
```

There are only two inputs to the indicator. One is price, and it is my habit to use the average of the high and low of each bar. Using closing prices is perfectly fine. The other input is the decision whether or not to show the display of the Dominant Cycle (ShowDC). As usual, variables and arrays are defined and initialized. The first calculation is a 40 bar high pass filter used to detrend the data. The detrended data is then smoothed in a low-lag 6 tap FIR filter.

Delta is the half-bandwidth of each bandpass filter relative to its center frequency. For example, if delta=.15, then a filter centered at a 20 bar cycle period would have its passband extend from a 17 bar cycle to a 23 bar cycle. Delta starts at a value of .5 to minimize a startup transient. Delta gradually reduces to reach a value of .15 after about 50 bars at the beginning.

The next section of code actually computes the bandpass filters in channels centers from 8 bars to 50 bars in one bar increments. I have taken the approach of using the smoothed and detrended data as the I[N] variable (notation for the Inphase component) and the rate of change data as the Q[N] variable (notation for the Quadrature component) before applying the bandpass filter. Both components are filtered in identical channelized bandpass filters so they have identical lag and phase shifts. The two components are squared and summed to compute the instantaneous amplitude of the signal at the output of each filter channel.

Next, the code finds the largest amplitude signal for normalization purposes. Then, the normalized amplitude is converted to decibels simultaneously with application of the nonlinear transform to sharpen the display. The nonlinear transformation was described in my article on the practical application of DFTs for trading.[2] The dominant cycle is then computed as the center of gravity from those channels whose amplitude is larger than -3 dB relative to the largest amplitude channel. Using the center of gravity results in a smoother result than just selecting the highest amplitude channel.

The channel amplitudes are then converted to a color value. If the amplitude falls between 0 and 10 dB, the color transitions from yellow to red and if the amplitude falls between 10 and 20 dB, the color transitions from red to black. Finally, each channel output is plotted as a colorized line, where vertical position in its subgraph corresponds to the amplitude of the signal at the output of that channel. This way, the measured spectrum can be plotted in synchronism with the barchart.

Figure 6. Channelized Receiver Spectrum Display

[2] John Ehlers, "Fourier Transforms for Traders", Stocks & Commodities, January 2007

I chose an interesting example display as a 1000 tick chart for the day session of the S&P e-mini futures contract on 2 August 2007. This chart is of passing interest because a equitick chart has a nonlinear horizontal time axis because each bar contains exactly 1000 ticks. This way, the chart is kind of a money flow, or at least activity flow, indicator itself. The chart shows a relatively consistent cycle period between 30 and 35 bars, with a little jitter in the late morning.

Perhaps the bigger question is “So what?”. How is this information useful? In Figure 7 I show the code where the measured dominant cycle is used to tune a bandpass filter. The code is exactly the same as that to compute the spectrum except the measured dominant cycle is then used to tune a bandpass filter. The sine and cosine outputs of this filter are plotted so that the crossovers can be used as buy and sell signals. Since both the inphase and quadrature components are available, additional tweaking is easily done to modify the phase of the plotted indicator. Figure 8 shows the response of the dominant cycle-tuned bandpass filter for the same equitick chart of Figure 6. The sine component is plotted in red and the cosine component is plotted in cyan. The sine component is clearly a smoothed replica of the price bars and the cosine component is a leading function – sometimes leading a little too much – of the sine component.

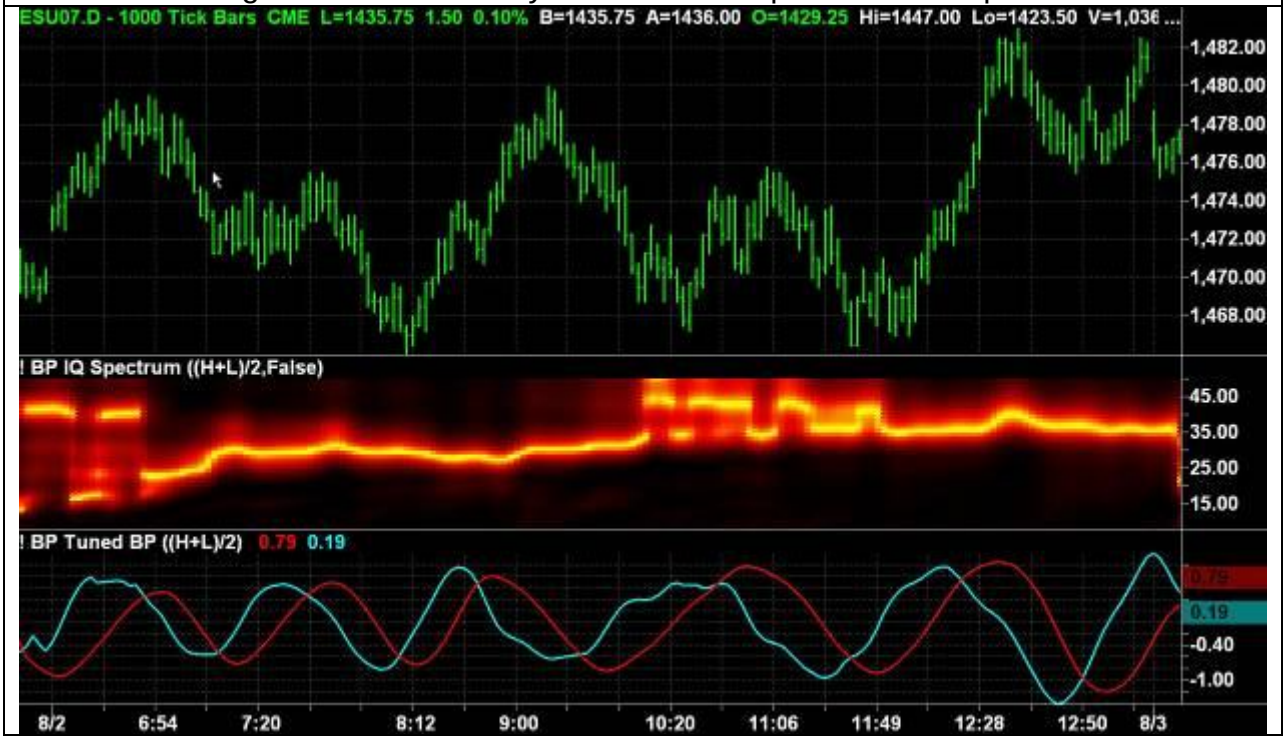
Figure 7. EasyLanguage Code for a Dominant Cycle Tuned Bandpass Filter

```
Inputs:
        Price((H+L)/2);

Vars:
        delta(0.1),
        gamma(0),
        alpha(0),
        beta(0),
        N(0),
        Period(0),
        MaxAmpl(0),
        Num(0),
        Denom(0),
        DC(0),
        DomCyc(0),
        Color1(0),
        Color2(0),
        alpha1(0),
        HP(0),
        SmoothHP(0);

Arrays:
        I[50](0),
        OldI[50](0),
        OlderI[50](0),
```

```
        Q[50](0),
        OldQ[50](0),
        OlderQ[50](0),
        Real[50](0),
        OldReal[50](0),
        OlderReal[50](0),
        Imag[50](0),
        OldImag[50](0),
        OlderImag[50](0),
        Ampl[50](0),
        OldAmpl[50](0),
        DB[50](0);

alpha1 = (1 - Sine (360 /  40)) / Cosine(360 / 40);
HP = .5*(1 + alpha1)*(Price - Price[1]) + alpha1*HP[1];
SmoothHP = (HP + 2*HP[1] + 3*HP[2] + 3*HP[3] + 2*HP[4] + HP[5]) / 12;
IF CurrentBar < 7 Then SmoothHP = Price - Price[1];
IF CurrentBar = 1 THEN SmoothHP = 0;

delta = -.015*CurrentBar + .5;
If delta < .15 then delta = .15;

If CurrentBar > 6 Then Begin
        For N = 8 to 50 Begin
                beta = Cosine(360 / N);
                gamma = 1 / Cosine(720*delta / N);
                alpha = gamma - SquareRoot(gamma*gamma - 1);
                Q[N] = (N / 6.283185)*(SmoothHP - SmoothHP[1]);
                I[N] = SmoothHP;
                Real[N] = .5*(1 - alpha)*(I[N] - OlderI[N]) + beta*(1 + alpha)*OldReal[N] -
alpha*OlderReal[N];
                Imag[N] = .5*(1 - alpha)*(Q[N] - OlderQ[N]) + beta*(1 + alpha)*OldImag[N]
- alpha*OlderImag[N];
                Ampl[N] = (Real[N]*Real[N] + Imag[N]*Imag[N]);
        End;
End;
For N = 8 to 50 Begin
        OlderI[N] = OldI[N];
        OldI[N] = I[N];
        OlderQ[N] = OldQ[N];
        OldQ[N] = Q[N];
        OlderReal[N] = OldReal[N];
        OldReal[N] = Real[N];
        OlderImag[N] = OldImag[N];
        OldImag[N] = Imag[N];
        OldAmpl[N] = Ampl[N];
```

```
End;

MaxAmpl = Ampl[10];
For N = 8 to 50 Begin
        If Ampl[N] > MaxAmpl then MaxAmpl = Ampl[N];
End;

For N = 8 to 50 Begin
   IF MaxAmpl <> 0 AND (Ampl[N] / MaxAmpl) > 0 THEN DB[N] = -10*Log(.01 / (1 -
.99*Ampl[N] / MaxAmpl)) / Log(10);
        If DB[N] > 20 then DB[N] = 20;
End;
Num = 0;
Denom = 0;
For N = 10 to 50 Begin
        If DB[N] <= 3 Then Begin
                Num = Num + N*(20 - DB[N]);
                Denom = Denom + (20 - DB[N]);
        End;
        If Denom <> 0 Then DC = Num / Denom;
End;
DomCyc = Median(DC, 10);
If DomCyc < 8 Then DomCyc = 20;

beta = Cosine(360 / DomCyc);
gamma = 1 / Cosine(720*delta / DomCyc);
alpha = gamma - SquareRoot(gamma*gamma - 1);
Value1 = .5*(1 - alpha)*(SmoothHP - SmoothHP[1]) + beta*(1 + alpha)*Value1[1] -
alpha*Value1[2];
Value2 = (DomCyc / 6.28)*(Value1 - Value1[1]);

Plot1(Value1, "Sine", Red, default, 2);
Plot2(Value2, "Cosine", Cyan, default, 2);
```

Figure 8. Dominant Cycle-Tuned Bandpass Filter Response



CONCLUSIONS

I have shown there is more than one way to measure the spectrum and establish the dominant cycle in market data. Knowing the dominant cycle is crucial to the creation of adaptive indicators, and adaptive indicators can have a significant contribution to your bottom line trading performance because you can now anticipate and implement trades at the cyclic turning points.