TIME WARP – WITHOUT SPACE TRAVEL

John Ehlers

One of the most frustrating aspects of technical analysis is trying to avoid whipsaw trades. When the moving averages are made smoother to avoid these whipsaws, the lag produced by the smoothing often renders the signals to be ineffective. The dilemma is therefore how to strike a balance between the amount of smoothing that can be obtained and the amount of lag that can be tolerated. In this article, I introduce a new tool to address the smoothing versus lag problem more effectively. I originally developed this concept in my recent book.[1] Here, I extend the capability of this tool by creating a highly reactive adaptive smoothing filter.

A moving average is a simple concept involving sampled data. One averages the data over the last "N" samples, moves forward one sample and averages over the new set of "N" samples, and so on. Actually, of each new set of N samples, only the oldest sample is discarded and one new sample is added. In any event, the average is done over a fixed number of samples and moved forward one sample at a time. In this way the average "moves". An engineer views the process differently. He sees the data moving down a fixed delay line that is tapped to get the output of each sample, and the tap outputs are added together to produce the moving average. This process is depicted in the schematic of Figure 1 for a 4 bar moving average. In Figure 1, the symbol Z^{-1} means that there is one unit of delay. In the case of daily data, the delay would be one day. The filter response in terms of the Z Transform is:

1) $$H(z) = 1 + Z^{-1} + Z^{-2} + Z^{-3}$$

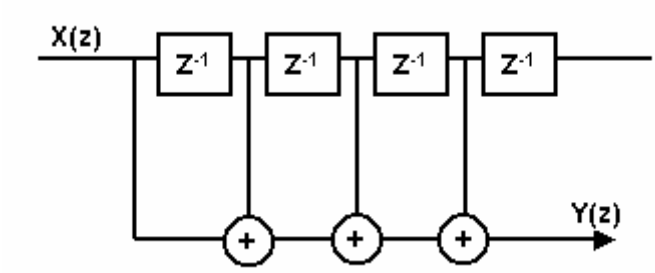


Figure 1. Schematic of a Moving Average

The equation for the moving average, in EasyLanguage format, is:

2) `Filt = (Price + Price[1] + Price[2] + Price[3]) / 4;`

That is, successively older data samples from the newest sample are averaged to attain the filtered output. The tapped delay line concept is favored by engineers because more generalized Finite Impulse Response (FIR) filters can be developed by changing the relative amplitudes of the samples. For example, if we wanted the middle two

[1] John Ehlers, "Cybernetic Analysis for Stocks and Futures", John Wiley & Sons, New York

samples to have twice the weight as the newest sample and oldest sample in our 4 sample example, the schematic diagram would be as shown in Figure 2.

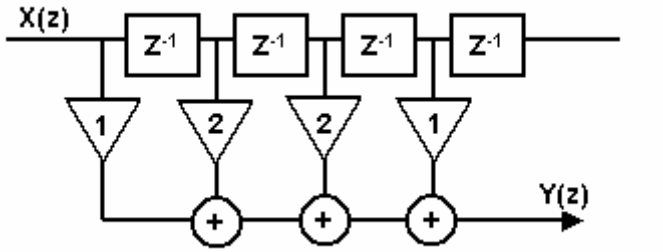


Figure 2. Schematic of a Four Element FIR Filter

The equation for the FIR filter, in EasyLanguage format, is:

3) Filt = (Price + 2*Price[1] + 2*Price[2] + Price[3]) / 6;

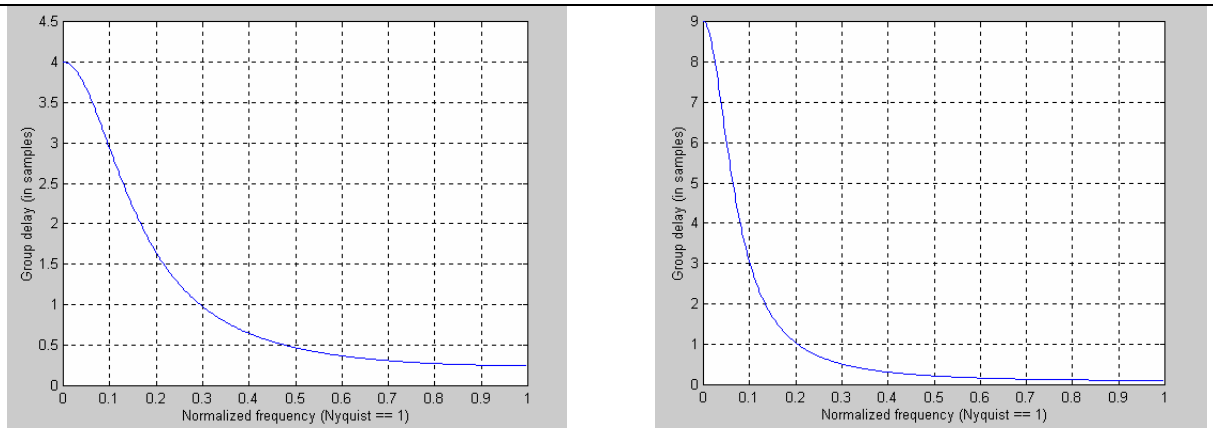
This is exactly the same filter used to eliminate the 2 bar and 3 bar cycle components in Figure 1. The multipliers on price are called the coefficients of the filter. Note that the filter is always normalized to the sum of the coefficients. This normalization is done so that the output will be the same as the input if all the samples have the same value. In engineering terms, the DC gain is equal to unity. The FIR filter can be made to have additional smoothing by making the filter longer. However, the lag of a FIR filter is approximately half the filter length. The result is that if we want greater smoothing we must accept the additional lag in conventional filters.

Conventional filters use the Z transform to describe the filter transfer characteristic, where Z^{-1} denoted a unit delay. There are a semi-infinite number of orthonormal functions for transform arithmetic. One such function is formed from Laguerre Polynomials. The mathematical expression for a k^{th} order Laguerre transfer response is:

4) $$H(z) = \frac{1-\gamma}{1-\gamma Z^{-1}}\left[\frac{Z^{-1}-\gamma}{1-\gamma Z^{-1}}\right]^{k-1}$$

The Laguerre Transform can be represented as a EMA low pass filter (the first term), followed by a succession of all-pass elements instead of unit delays (the k-1 terms). All terms have exactly the same damping factor γ. We see these are all pass networks by examining the frequency response. When frequency is zero, the Z^{-1} term has a value of 1, and therefore the element evaluates to $(1-\gamma)/(1-\gamma) = 1$. Similarly, when frequency is infinite, Z^{-1} has a value of –1, and therefore the element evaluates to $(-1-\gamma)/(1+\gamma) = -1$. The element has a unity gain at all frequencies between zero and infinity, and therefore is an all pass network. However, the phase from input to output shifts over the frequency range, causing the lag to be variable as a function of frequency. The degree

to which the lag is variable depends upon the value of the damping factor γ. For example, the lag, or group delay, for $\gamma = 0.6$ and $\gamma = 0.8$ are shown in Figure 3.



$\gamma = 0.6$ $\gamma = 0.8$

Figure 3. All Pass Network Lag is a Function of Frequency and Damping Factor

Therefore, we can make a filter using the Laguerre elements instead of the unit delay whose coefficients are also [1 2 2 1]/6 as with the FIR filter. The difference is that we have warped the time between the delay line times. The schematic of the Laguerre filter is shown in Figure 4.

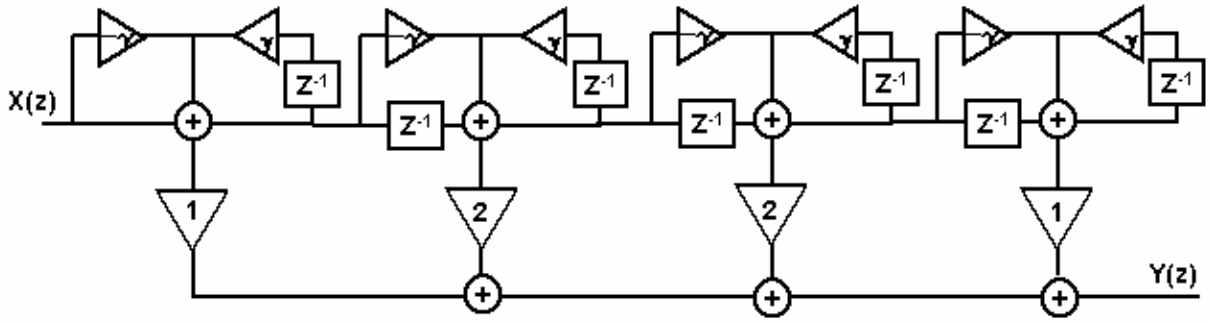


Figure 4. Schematic of a Laguerre Filter

The EasyLanguage code for a four element Laguerre Filter are given in Figure 5. Since gamma is a TradeStation function, I have made the substitution $\alpha = 1 - \gamma$. L0 is the output of the first section, and is just an EMA. The following three sections are identical in their form. The four sections of the Laguerre delay line are summed exactly the same way as one would sum a linear delay line for a FIR filter. The Laguerre output is the "Filt" variable. An identical length FIR filter is also computed for comparison.

Figure 5. EasyLanguage Code for the Laguerre Filter

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

Vars:    L0(0),
         L1(0),
         L2(0),
         L3(0),
         Filt(0)
         FIR(0);

L0 = alpha*Price + (1 - alpha)*L0[1];
L1 = -(1 - alpha)*L0 + L0[1] + (1 - alpha)*L1[1];
L2 = -(1 - alpha)*L1 + L1[1] + (1 - alpha)*L2[1];
L3 = -(1 - alpha)*L2 + L2[1] + (1 - alpha)*L3[1];

Filt = (L0 + 2*L1 + 2*L2 + L3) / 6;
FIR = (Price + 2*Price[1] + 2*Price[2] + Price[3]) / 6;

Plot1(Filt, "Filt");
Plot2(FIR, "FIR");
```

The results of the Laguerre and FIR filter are shown in Figure 6. Remember both filters have identical lengths. The FIR filter has a lag of only 1.5 bars and only moderately smoothes the price data. On the other hand, the Laguerre filter is dramatically smoother and also has significant lag. You can decrease the smoothing and the lag by decreasing the damping factor. When the damping factor is reduced to zero, the Laguerre filter is identical to the FIR filter. This is a simple way to control the action of a moving average and still use only a few data samples in the calculation.

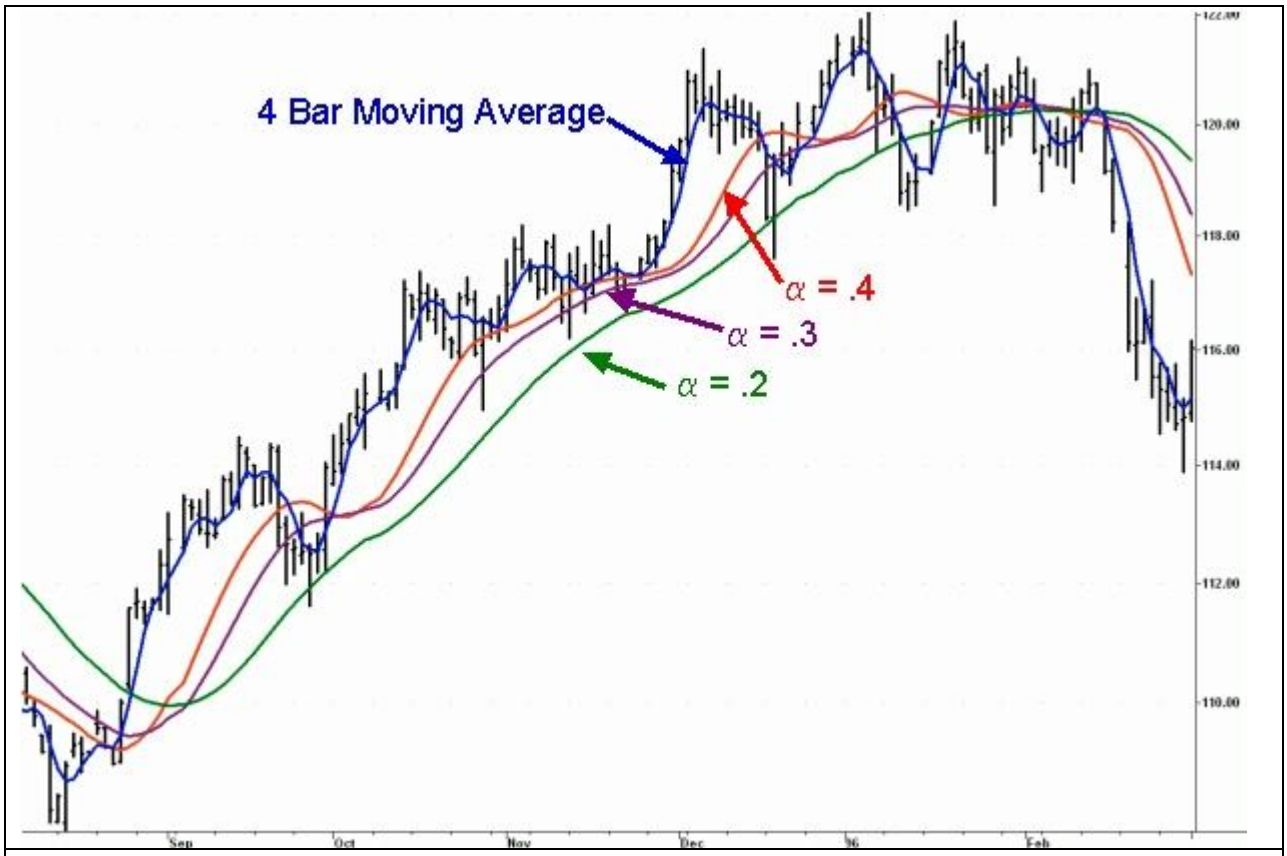


Figure 6. Four Element Laguerre Filter is Dramatically Smoother than a Conventional Four Element FIR Filter

The story does not end with conventional filters. As I am fond of saying, “Truth and science always triumphs over ignorance and superstition”. If we can generate superior smoothing with very short filters, it follows that we should be able to take advantage of the wide tuning range of this short filter and make it the ultimate adaptive filter.

The Adaptive Laguerre Filter is described with reference to Figure 7. The process starts by taking the absolute value of the difference between the current price and the previously computed filter value. This way the adaptive parameter is largest when then the difference is greatest, causing the filter to be more responsive to the price change. We establish tuning parameter alpha as the normalized difference relative to the largest range of differences over the observation length. This way, if the difference is small relative to the recent maximum and minimum differences, alpha will be small and the filter will therefore be very smooth. So, if the price changes rapidly, the filter will follow the price closely. However, if the price moves in a sideways fashion, the filter will not change much. This characteristic is very beneficial in eliminating whipsaw trades in sideways markets while enabling early entry into and exits from trending trades. I use a five bar median filter to smooth the alpha tuning factor because, without it, alpha is very noisy and undesired adaptation to noise spikes can result.

Figure 7. EasyLanguage Code for the Adaptive Laguerre Filter

```
Inputs:  Price((H+L)/2),
         Length(20);

Vars:    Diff(0),
         HH(0),
         LL(0),
         count(0),
         alpha(0),
         L0(0),
         L1(0),
         L2(0),
         L3(0),
         Filt(0),
         FIR(0);

Diff = AbsValue(Price - Filt[1]);
HH = Diff;
LL = Diff;
For count = 0 to Length - 1 begin
         If Diff[count] > HH then HH = Diff[count];
         If Diff[count] < LL then LL = Diff[count];
End;
If CurrentBar > Length and HH - LL <> 0 then alpha = Median(((Diff - LL) / (HH - LL)), 5);

L0 = alpha*Price + (1 - alpha)*L0[1];
L1 = -(1 - alpha)*L0 + L0[1] + (1 - alpha)*L1[1];
L2 = -(1 - alpha)*L1 + L1[1] + (1 - alpha)*L2[1];
L3 = -(1 - alpha)*L2 + L2[1] + (1 - alpha)*L3[1];

Filt = (L0 + 2*L1 + 2*L2 + L3) / 6;

Plot1(Filt, "Laguerre");
```

The performance of the Adaptive Laguerre Filter is shown in Figure 8. The filter response speaks for itself.

Figure 8. Adaptive Laguerre Filter Response

CONCLUSIONS

The Laguerre Transform provides a time warp such that the low frequency components are delayed much more than the high frequency components. This time distortion enables very smooth filters to be built using a short amount of data. Time-warped indicators react faster because a shorter amount of data is used. Therefore, a highly reactive smoothing filter using the Laguerre tuning parameter may very well result in the ultimate adaptive filter.