The nth Order Polynomial Velocity System Applied to E-Mini Futures 1min Bars

Working Paper September 2005

In the previous working papers we showed how the application of a price curve generated by a third and forth degree polynomial could be used to develop a system to buy and sell daily British Pound futures , daily IBM stock prices, IBM 5min Bars and S&P500 Futures 5min Bars . Here we will let the computer decide the "best" polynomial order to use with the Fixed Memory Polynomial Velocity System (FixmV). The FixmV will be used to trade the E-Mini futures contract on an intraday. To test this system we will use one minute bar prices of the E-Mini futures contract traded on the CME and known by the symbol ES for the 53 weeks from April 7, 2004 to May 13, 2005.

The nth Order Fixed Memory System Defined

The least squares forecast nth order fixed memory polynomial velocity is constructed by solving for the coefficients $\beta_0, \beta_1, \beta_2, \beta_3 \ldots \beta_n$ for the discrete orthogonal Legendre polynomials at each bar using the last **N bars** of closing prices and the equation for β_j in the **"Math"** appendix. The **Velocity(T+1)** is constructed from the equation shown in the "Math" appendix and plotted under the price chart.

The velocity of a 2nd order polynomial should change faster than the straight line (1st order velocity). As seen from the 2nd order velocity equation in the "Math" section there is an acceleration component in the calculation of the velocity. This means that the 2nd order velocity will reflect a change in the price trend much faster than the straight line velocity which does not have an acceleration component. The same is true for 3rd and 4th order velocities. Whether higher order polynomial velocities is an advantage or not we will let the computer decide when we use the walk forward optimization technique described below.

At each bar we calculate the nth order (1st through 4th) fixed memory polynomial velocity from the formulas in the "Math" appendix. As we will show below, optimization will determine what order for fixed memory polynomial velocity will be used. When the velocity is greater than the threshold amount ***vup*** we will go long. When the velocity is less than the threshold amount ***-vdn*** we will go short.

Buy Rule:

IF Velocity is greater than the threshold amount ***vup*** then buy at the market.

Sell Rule:

IF Velocity is less than the threshold amount ***-vdn*** then sell at the market.

Intraday Bars Exit Rule:

Close the position 5 minutes before the E-Mini close (no trades will be carried overnight).

Intraday Bars First Trade of Day Entry Rule:

- Ignore all trade signals before 10:00 EST (30 minutes after the open). For the Buy and Sell rules above we have included a first trade of the day entry rule. We've included this rule because often there are gaps in the open creating immediate system buys and sells. Many times these gaps are closed creating a losing whipsaw trade. In order to avoid the opening gap whipsaw trade problem we've delayed the first trade of the day for 30 minutes until after10:00 EST .

Testing The Polynomial Velocity System Using Walk Forward Optimization

There are four system parameters to determine:

1. **degree**, degree=1 for straight line velocity, degree=2 for 2nd order velocity, etc.
2. ***N***, is the number of bars lookback period to calculate the **velocity.**
3. ***vup***, the threshold amount that velocity has to be greater than to issue a buy signal
4. ***vdn***, the threshold amount that velocity has to be less than to issue a sell signal

To test this system we will use one minute bar prices of the S&P 500 E-Mini futures contract traded on the CME and known by the symbol ES from April 7, 2004 to May 13, 2005.

We will test this strategy with the above E-Mini 1 min bars on a walk forward basis, as will be described below. To create our walk forward files we will use the ***add-in*** software product called the Power Walk Forward Optimizer (PWFO). In TradeStation (TS), we will run the PWFO strategy ***add-in*** along with the n^{th} Order Polynomial Velocity Strategy on the E-Mini 1min data from 4/7/2004 to 5/13/2005. The PWFO will breakup and create fifty-three 30 calendar day test sections along with their corresponding 7 calendar day out-of-sample sections from the one year of E-mini TS optimization (see Walk forward Testing below). The test and out-of-sample section dates are shown in **Table 1** on page 6 below. We will then use another software product called the Walk Forward Performance Explorer (WFPE) on each of the fifty-three test and out-of-sample sections generated by the PWFO to find the best test(in-sample) section performance ***filter*** that determines the system input parameters ***(degree, N, vup,*** and ***vdn)*** that will be used on the out-of-sample data. Detailed information about the PWFO and the WFPE can be found at www.meyersanalytics.com

For the test data we will run the TradeStation optimization engine on the 53 weeks of E-Mini 1 min bars with the following ranges for the nth order polynomial velocity strategy inputs variables.:

1. degree from 1 to 4
2. N from 10 to 75 in steps of 5.
3. vup from 0.05 to 0.6 steps of 0.05
4. vdn from 0.05 to 0.6 in steps of 0.05

This will produce 8064 different cases or combinations of the input parameters for each of the 53 PWFO output files.

Walk Forward Out-of-Sample Testing

What is walk forward analysis?. Optimization is a misnomer on today's trading platforms. Optimization should really be called a combinatorial search. That is, we have the software output certain performance parameters for every combination of a range of input parameters. We then sort that range on some performance parameter. Whenever you run an optimization on noisy data on a fixed time interval, the best performance will almost always be due to curve fitting the noise and signal. This curve fitting is also called "Data Mining" or "Data Snooping". A fixed number of prices on a fixed time interval has many spurious movements which are also called noise. When we run many different combinations of the input parameters, the best performance will be from those cases that are able to capture profits from the spurious movements and the price or signal dynamics. While the signal dynamics, if there, will repeat, the same spurious price movements will not. If the spurious movements that were captured by a certain set of input parameters were a large part of the net profits, then choosing these input parameters would produce losses when traded on future data. These losses occur because the spurious movements will not be repeated in the same way. The input parameters chosen from the test section performance are "cherry picked" to perform well on only those exact same spurious movements. This is why curve fitted systems with no out-of-sample testing cause losses in the out-of-sample section from something that looked great in the test section.

Walk forward analysis attempts to minimize curve fitting by using the law of averages from the Central Limit Theorem on the out-of-sample performance. In walk forward analysis the data is broken up into many test and out-of-sample sections. Usually for any system one has some performance parameter selection procedure or filter used to select the input parameters from the optimization run. For instance, a filter might be all cases that have a profit factor (PF) greater than 1 and less than 3. For the number of cases left we might select the cases that had the best percent profit. This procedure would leave you with one case in the test section output and it's associated input parameters. Now suppose we ran our optimization on each of our many test sections and applied our filter to each test section output. We would than use the input parameters found by the filter in each test section on the out-of-sample section immediately following that test section. The input parameters found in each test section and applied to each out-of-sample section would produce independent net profits and losses for each of the out-of-sample sections. Using this method we now have "x" number of independent out-of-sample section profit and losses from our filter. If we take the average of these out-of-sample section net profits and losses than we will have an estimate of how our system will perform on average. Due to the Central Limit Theorem, as our sample size increases, the

spurious noises results in the out-of-sample section performance should average out to zero in the limit leaving us with what to expect from our system and filter. Mathematical note: This assumption assumes that the out-of-sample returns are from probability distributions that have a finite variance.

Why use the walk forward technique? Why not just perform the TradeStation optimization on the whole price series and choose the input parameters that give the best total net profits or profit factor? Surely the price noise cancels itself out with such a large number of test prices and trades. Unfortunately, nothing could be farther from the truth! Optimization is a misnomer and should really be called combinatorial search. As stated above, whenever we run a combinatorial search over many different combinations of input parameters on noisy data on a fixed number of prices, ***no matter how many***, the best performance parameters found are guaranteed to be due to "curve fitting" the noise and signal. What do we mean by "curve fitting"? Price series that we trade consists of random spurious price movements, which we call noise, and repeatable price patterns (*if they exist*). When we run, say, 8000 different input parameter combinations , the best performance parameters will be from those system input variables that are able to produce profits from the price pattern ***and*** the random spurious movements While the price patterns will repeat, the same spurious price movements will not. If the spurious movements that were captured by a certain set of input parameters were a large part of the total net profits, then choosing these input parameters will produce losses when traded on future data. These losses occur because the spurious movements will not be repeated in the same way. This is why system optimization or combinatorial searches with no out-of-sample testing cause loses when traded in real time from something that looked great in the test section. Unfortunately it is human nature to extrapolate past performance to project future trading results and so results from curve fitting give the illusion, a modern "siren call" so to speak, of future trading profits.

In order to gain confidence that our input parameter selection method using the optimization output of the test data will produce profits, we must test the input parameters we found on out-of-sample data. In addition, we must perform the test/out-of-sample analysis many times. Why not just do the out-of-sample analysis once? Well just as in Poker or any card game, where there is considerable vagaries in hand to hand luck, walk forward out-of-sample analysis give considerable vagaries in week to week out-of-sample profit "luck". That is, by pure chance we may have chosen some input parameter set that did well in the test section data ***and*** the out-of-sample section data. In order to minimize this type of "luck", statistically, we must repeat the walk forward out-of-sample (oos) analysis over many test/oos sections and take an average of our weekly results over all out-of-sample sections. This average gives us an expected weekly return and a standard deviation of weekly returns which allows us to statistically estimate the expected equity and it's range for N weeks in the future.

Finding The Strategy Input Parameters in The Walk Forward Test Sections

To find a performance filter on the test section that the system input parameters in the PWFO files, we will use the Walk Forward Performance Explorer (WFPE) on each of the fifty-three PWFO test/out-of-sample files.

The WFPE revealed that the following filter on the test(in-sample) data will produce the most consistent and reliable out-of-sample results..

Filter: PF<=2.5 and LR<=5 and #Trds>=12 and Smallest lbr

Where:

- **PF** = Profit Factor in Test optimization section
- **LR** = Maximum consecutive loses in a row in test optimization section. Since in real time it is tough to sustain more that five losses in a row and still keep trading, we will eliminate all those cases that have more than five losses in a row
- **#Trds** = The number of trades in the test run. We want our system to trade almost every day. There are 20 to 23 trading days a month, so we only want to look at input parameters that are able to generate at least 12 trades in every one month test sample.
- **lbr** = Total Losing Bars in the test section

The first part of the filter chooses those rows or cases out of the 8064 rows in each PWFO file in-sample(test) section that satisfy the criteria **PF<=2.5 and LR<=5 and #Trds>=12 .** This part of the filter will leave anywhere

from 0 to 200 rows out of the 8064 rows that satisfy this filter criteria. Next we find the one row that has the smallest **lbr**. This **Filter** or selection procedure will leave only one choice for the system input values of ***degree, N, vup,*** and ***vdn***. We then use these input values found in the test section by the **Filter** on the **next week** of one minute bar E-Mini **out-of-sample** price bars **following** the test section.

Results

Table 1 below on page 6 presents a table of the fifty-three test and out-of-sample windows and the selected optimum parameters and out-of-sample results using the ***filter*** described above.

Figure 1 presents the out-of-sample 1 minute bar chart of ES from 7/9/04 to 7/16/04. with the Velocity Indicator and all the buy and sell signals for those dates.

Discussion of System Performance

At the bottom of **Figure 1** are some statistics that are of interest. In the "Averages" row #56, the first statistic OSNP Average=$206, is the average weekly net profit for the fifty-three out-of-sample weeks and ONT Average= 4.02 is the average number of trades per week. The OSNP Std Dev =$407.9 is the standard deviation of the weekly return. Given a process that generates the above average and standard deviation, it would be interesting to know, statistically, the probabilistic outcome from trading this system for 13 weeks (one quarter). There is a statistical technique called bootstrap Ref [1]. Using the bootstrap technique we would randomly choose 13 of the fifty-three weekly profits in Table 1, with replacement and take the sum of those 13 randomly chosen out-of-sample profits. "With replacement" means we don't eliminate the randomly chosen weekly profit from being chosen again. Let us suppose that we repeat this random choosing of 13 weeks and summing the results 200 times. We would then have 200 different 13 week net profit summations. If we take the average and standard deviation of those 200 different 13 week net profit summations we would have an estimate of what to expect from this system by trading it for 13 weeks.

For this system the 200 bootstrap average of 13 week net profits is $2696 and the standard deviation is $1366. This means that at two standard deviations we can expect our 13 week return from this system to be between -$36 and $5428 95% of the time. This is quite a range and indicates that 13 weeks of trading is not enough time to judge the results of trading this system. Assuming that the bootstrap averages are from a normal distribution an interesting number is called the 99% breakeven time. This is how many weeks do we need to trade this system so that we have a 99% probability that the equity after those number of weeks will be greater than zero. The answer is 15.7 weeks. That is. there is less than a 1% chance that our equity will be negative after 15.7 weeks of trading . Please note that slippage and commissions were not taken into account so the numbers obtained above are higher that could be attained from actual trading..

To see the effect of walk forward analysis take a look at **Table 1**. Notice how the input parameters ***N, vup and vdn*** take sudden jumps from high to low and back . This is the walk forward process quickly adapting to changing volatility conditions in the test sample. In addition, notice how often ***degree*** changes from a straight line velocity with ***degree=1*** to a 2nd and 3rd order velocity with ***degree= 2 and 3***. The 2nd and 3rd order velocity, due of the acceleration component, change much faster than the straight line velocity. When the data gets very noisy with a lot of spurious price movements it's better to have the velocity change slower filtering out the noisy data. During other times when the noise level is not as much it is better to have the velocity break it's ***vup*** and ***vdn*** barriers faster to get onboard a trend faster. This is what the filter is doing. When there is a lot of noise in the test section it switches to the straight line velocity. When the noise level is lower in the test section it switches to the faster changing 2nd and 3rd order velocity. Notice that the best Filter did not use the 4th Order velocity. Probably the 4th Order polynomial velocity changes to fast and is whipsawed by the noise component.

In observing the chart for the two days of 7/9 through 7/16 we can see that the system did very well in catching the major intraday trend of the E-Mini while avoiding many no trend whipsaw losses. When the morning trend changed in midday on 7/14 and 7/15 the Polynomial Velocity system changed direction quickly allowing for the capture of the profits from the trend change to the close of trading. Overall the Polynomial Velocity system did a good job in minimizing the losses due to the inevitable whipsaws that will occur in any trading system and maximizing the profits from the major intraday trend moves of the E-Mini.

References

1. Efron, B., Tibshirani, R.J., (1993), “An Introduction to the Bootstrap”, New York, Chapman & Hall/CRC.
2. Morrison, Norman “Introduction to Sequential Smoothing and Prediction", McGraw-Hill Book Company, New York, 1969.

Table 1 Walk Forward Out-Of-Sample Performance Summary for E-Mini Polynomial Velocity System

ES-1 min bars 4/7/2004 - 5/13/2005. The input values *degree(pw), N, vup, and vdn* are the values found from applying the filter to the test Sample optimization runs.

Filter= PF<2.5 and LR<=5 and #Trds>=12 and Smallest lbr

Week	Test Dates			Out-0f-Sample Dates			osnp	Equity	ollt	odd	ont	avosnp	pw	N	vup	vdn
1	04/07/04	To	05/07/04	05/10/04	To	05/14/04	950	950	(275)	(513)	17	55.9	3	50	0.55	0.4
2	04/14/04	To	05/14/04	05/17/04	To	05/21/04	363	1313	(88)	(88)	2	181.5	2	60	0.35	0.35
3	04/21/04	To	05/21/04	05/24/04	To	05/28/04	150	1463	0	0	1	150	3	70	0.55	0.55
4	04/28/04	To	05/28/04	05/31/04	To	06/04/04	(213)	1250	(113)	(213)	2	-106.5	3	70	0.55	0.55
5	05/05/04	To	06/04/04	06/07/04	To	06/11/04		1250					2	40	0.35	0.55
6	05/12/04	To	06/11/04	06/14/04	To	06/18/04	(213)	1037	(125)	(213)	2	-106.5	3	50	0.45	0.6
7	05/19/04	To	06/18/04	06/21/04	To	06/25/04	50	1087	(25)	(50)	3	16.7	3	45	0.5	0.4
8	05/26/04	To	06/25/04	06/28/04	To	07/02/04	575	1662	(213)	(363)	9	63.9	3	30	0.6	0.55
9	06/02/04	To	07/02/04	07/05/04	To	07/09/04	50	1712	0	0	2	25	2	25	0.5	0.35
10	06/09/04	To	07/09/04	07/12/04	To	07/16/04	238	1950	(300)	(300)	5	47.6	2	25	0.5	0.35
11	06/16/04	To	07/16/04	07/19/04	To	07/23/04	488	2438	(200)	(225)	7	69.7	1	10	0.5	0.3
12	06/23/04	To	07/23/04	07/26/04	To	07/30/04	0	2438	(188)	(188)	6	0	2	25	0.5	0.45
13	06/30/04	To	07/30/04	08/02/04	To	08/06/04	375	2813	(188)	(300)	4	93.8	2	25	0.5	0.5
14	07/07/04	To	08/06/04	08/09/04	To	08/13/04	313	3126	(113)	(113)	3	104.3	2	25	0.55	0.55
15	07/14/04	To	08/13/04	08/16/04	To	08/20/04	513	3639	(188)	(188)	3	171	2	40	0.3	0.6
16	07/21/04	To	08/20/04	08/23/04	To	08/27/04		3639					2	30	0.4	0.55
17	07/28/04	To	08/27/04	08/30/04	To	09/03/04	(175)	3464	(300)	(525)	5	-35	1	25	0.15	0.25
18	08/04/04	To	09/03/04	09/06/04	To	09/10/04	(175)	3289	(175)	(175)	3	-58.3	1	10	0.4	0.3
19	08/11/04	To	09/10/04	09/13/04	To	09/17/04		3289					1	10	0.3	0.5
20	08/18/04	To	09/17/04	09/20/04	To	09/24/04	(50)	3239	(163)	(375)	5	-10	1	35	0.1	0.15
21	08/25/04	To	09/24/04	09/27/04	To	10/01/04	375	3614	0	0	1	375	2	15	0.6	0.55
22	09/01/04	To	10/01/04	10/04/04	To	10/08/04		3614					3	55	0.45	0.45
23	09/08/04	To	10/08/04	10/11/04	To	10/15/04	275	3889	(138)	(138)	6	45.8	3	55	0.4	0.45
24	09/15/04	To	10/15/04	10/18/04	To	10/22/04	263	4152	(163)	(288)	7	37.6	3	60	0.45	0.4
25	09/22/04	To	10/22/04	10/25/04	To	10/29/04	213	4365	(263)	(413)	9	23.7	3	60	0.4	0.4
26	09/29/04	To	10/29/04	11/01/04	To	11/05/04	1013	5378	(288)	(538)	12	84.4	3	40	0.45	0.55
27	10/06/04	To	11/05/04	11/08/04	To	11/12/04	63	5441	0	0	1	63	3	50	0.55	0.5
28	10/13/04	To	11/12/04	11/15/04	To	11/19/04	0	5441	(175)	(175)	2	0	3	50	0.55	0.6
29	10/20/04	To	11/19/04	11/22/04	To	11/26/04	(88)	5353	0	0	1	-88	2	65	0.35	0.2
30	10/27/04	To	11/26/04	11/29/04	To	12/03/04	(238)	5115	(350)	(463)	4	-59.5	3	40	0.55	0.6
31	11/03/04	To	12/03/04	12/06/04	To	12/10/04	438	5553	(188)	(263)	5	87.6	3	45	0.55	0.45
32	11/10/04	To	12/10/04	12/13/04	To	12/17/04	(563)	4990	(450)	(713)	4	-140.8	3	45	0.6	0.4
33	11/17/04	To	12/17/04	12/20/04	To	12/24/04		4990					3	40	0.6	0.5
34	11/24/04	To	12/24/04	12/27/04	To	12/31/04	513	5503	0	0	3	171	1	10	0.35	0.3
35	12/01/04	To	12/31/04	01/03/05	To	01/07/05	1213	6716	(50)	(50)	5	242.6	3	35	0.55	0.6
36	12/08/04	To	01/07/05	01/10/05	To	01/14/05	(350)	6366	(475)	(475)	4	-87.5	1	10	0.35	0.45
37	12/15/04	To	01/14/05	01/17/05	To	01/21/05	638	7004	0	0	3	212.7	2	75	0.3	0.2
38	12/22/04	To	01/21/05	01/24/05	To	01/28/05	163	7167	(113)	(163)	5	32.6	2	70	0.35	0.15
39	12/29/04	To	01/28/05	01/31/05	To	02/04/05	(100)	7067	0	0	1	-100	3	75	0.45	0.35
40	01/05/05	To	02/04/05	02/07/05	To	02/11/05	338	7405	0	0	1	338	1	25	0.35	0.15
41	01/12/05	To	02/11/05	02/14/05	To	02/18/05	213	7618	0	0	1	213	3	40	0.6	0.6
42	01/19/05	To	02/18/05	02/21/05	To	02/25/05	475	8093	0	0	1	475	3	40	0.6	0.6

Week	Test Dates			Out-0f-Sample Dates			osnp	Equity	ollt	odd	ont	avosnp	pw	N	vup	vdn
43	01/26/05	To	02/25/05	02/28/05	To	03/04/05	150	8243	(13)	(13)	3	50	3	70	0.35	0.5
44	02/02/05	To	03/04/05	03/07/05	To	03/11/05	(325)	7918	(463)	(750)	4	-81.2	1	35	0.15	0.15
45	02/09/05	To	03/11/05	03/14/05	To	03/18/05	263	8181	(25)	(25)	2	131.5	2	75	0.2	0.25
46	02/16/05	To	03/18/05	03/21/05	To	03/25/05	88	8269	(275)	(363)	3	29.3	2	70	0.2	0.3
47	02/23/05	To	03/25/05	03/28/05	To	04/01/05	638	8907	0	0	1	638	2	35	0.35	0.4
48	03/02/05	To	04/01/05	04/04/05	To	04/08/05	(188)	8719	0	0	1	-188	3	45	0.55	0.55
49	03/09/05	To	04/08/05	04/11/05	To	04/15/05	475	9194	(438)	(863)	8	59.4	3	50	0.45	0.5
50	03/16/05	To	04/15/05	04/18/05	To	04/22/05	1225	10419	0	0	5	245	2	30	0.4	0.45
51	03/23/05	To	04/22/05	04/25/05	To	04/29/05	188	10607	(413)	(488)	5	37.6	2	30	0.4	0.5
52	03/30/05	To	04/29/05	05/02/05	To	05/06/05	(50)	10557	(275)	(275)	4	-12.5	3	70	0.55	0.5
53	04/06/05	To	05/06/05	05/09/05	To	05/13/05	(688)	9869	(700)	(700)	2	-344	2	35	0.55	0.5
						Averages	**206**				**4.02**	**51.1**				
				Standard Deviation			**407.9**									
				200Boot 13wk Sum Avg			**2696**									
				200Boot 13wk Sum Std			**1366**									

PF = Profit Factor in Test optimization section
LR=Maximum consecutive loses in a row in test optimization section
#Trds = Minimum number of trades in the test section.
lbr=Total Losing Bars in the test section
osnp = Weekly Out-of-sample net profit in $
ollt = The largest losing trade in the out-of-sample section in $.
odd = The close drawdown in the out-of-sample section in $.
ont = The number of trades in the out-of-sample week.
Equity = running sum of the weekly out-of-sample net profits in $

Figure 2 Walk Forward Out-Of-Sample Performance Summary for E-Mini Fixed Memory Polynomial Velocity System 1 minute bar chart of ES from 7/9/04 to 7/16/04

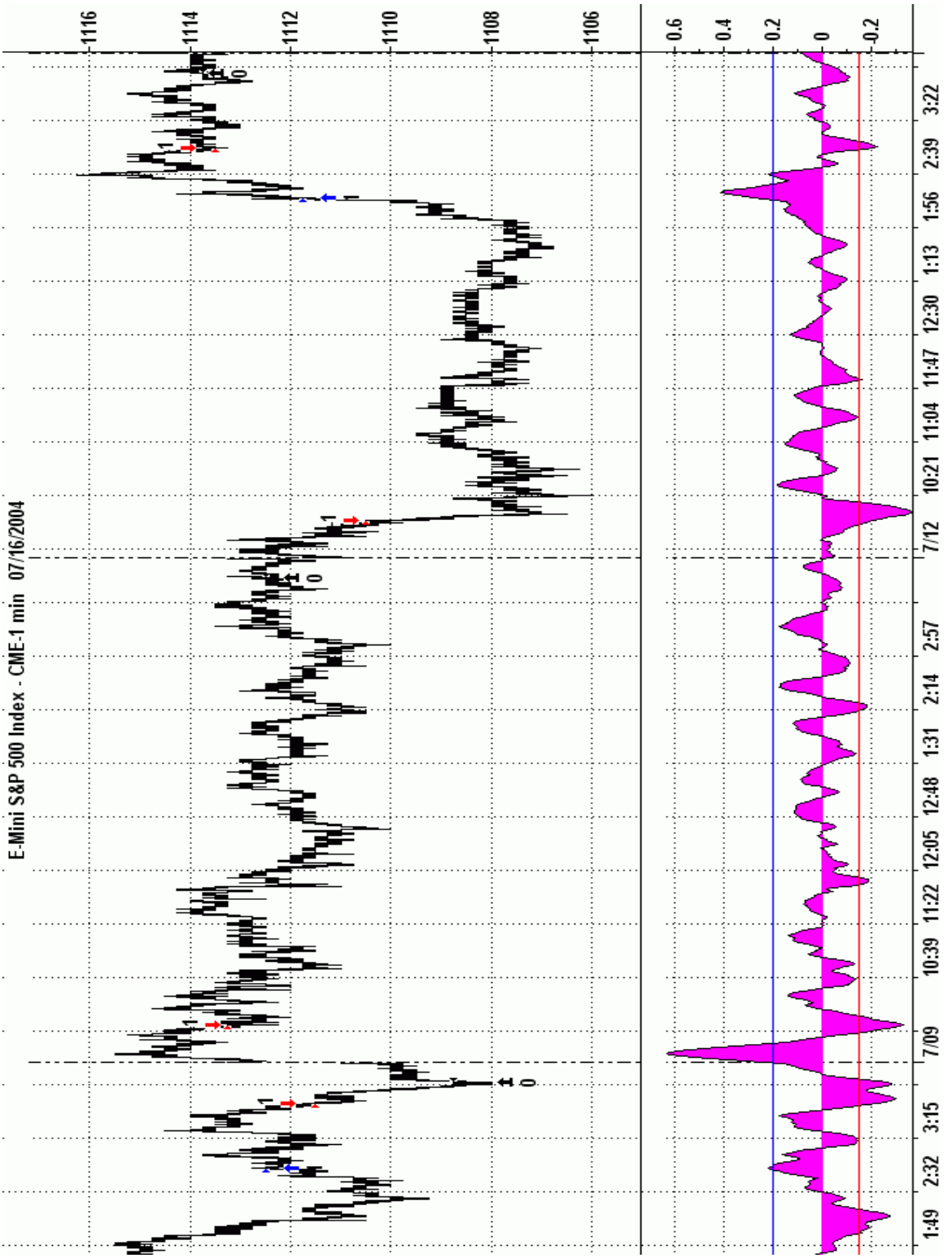


Figure 2 Walk Forward Out-Of-Sample Performance Summary for E-Mini Fixed Memory Polynomial Velocity System 1 minute bar chart of ES from 7/9/04 to 7/16/04

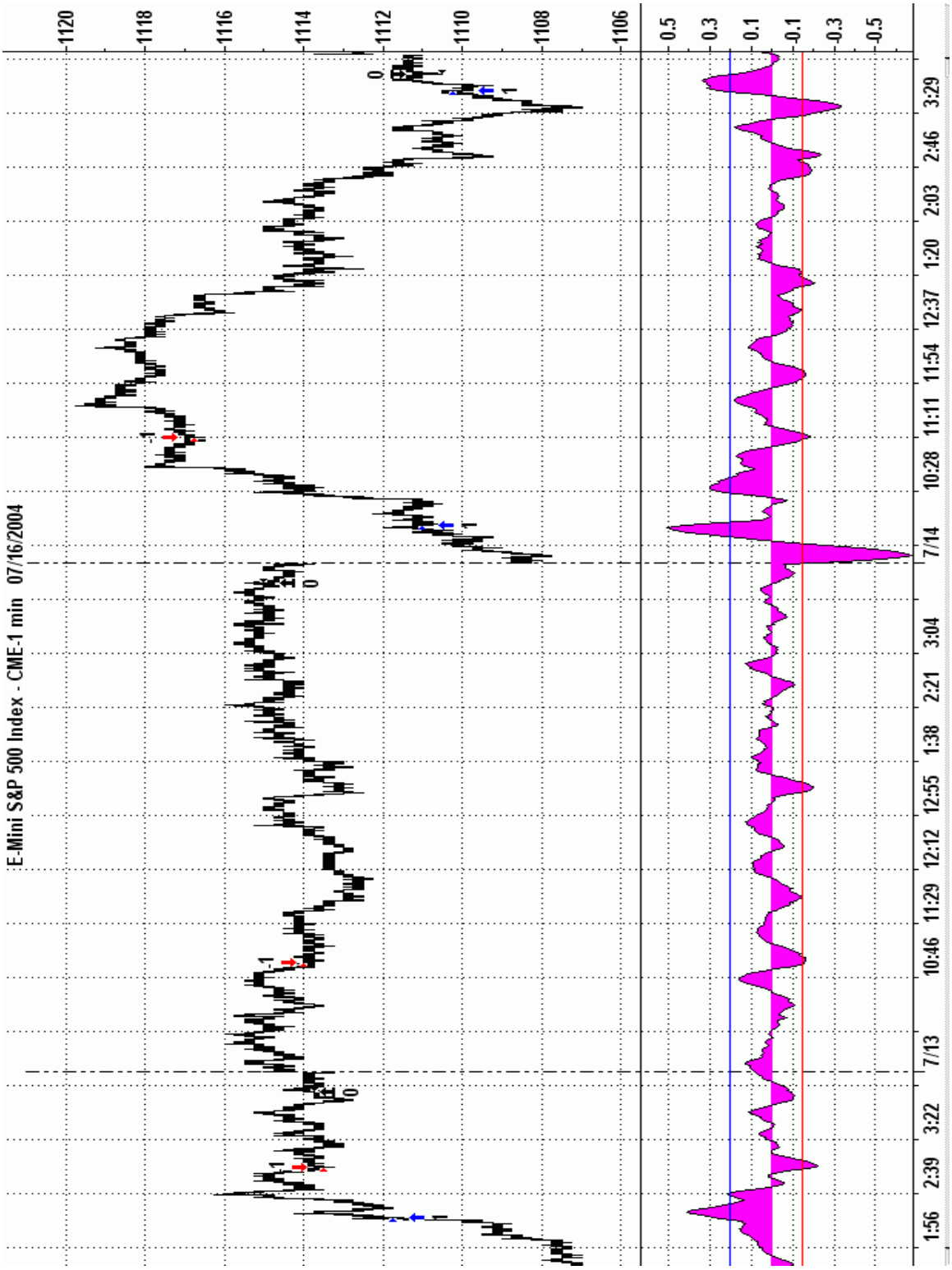
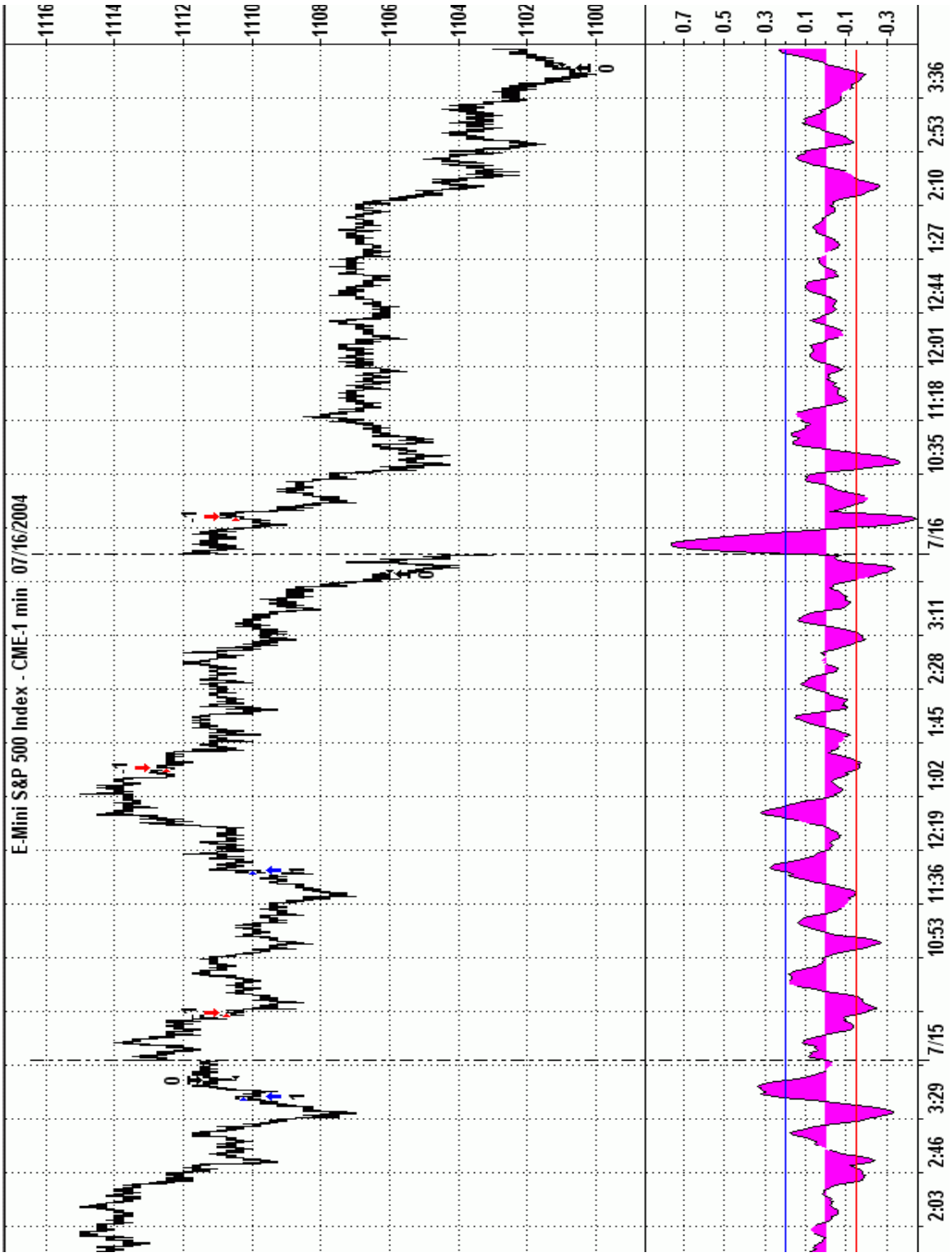


Figure 2 Walk Forward Out-Of-Sample Performance Summary for E-Mini Fixed Memory Polynomial Velocity System 1 minute bar chart of ES from 7/9/04 to 7/16/04



Appendix: n^{th} Order Polynomial Next Bar's Forecast Math

What is the n^{th} Order Polynomial ?

The n^{th} Order Polynomial, also called the n^{th} Order Fixed Memory Polynomial, is simply the least square fit of a polynomial of the form $\mathbf{b_0+b_1*t+b_2*t^2+b_3*t^3+\ldots b_n*t^n}$ to a ***fixed*** number of past data points. Where **t** is discrete time bars. Time could be daily bars or one minute bars. We use the term "Fixed Memory" to designate that only a fixed number of data points are used to calculate the polynomial coefficients. It is assumed that the time bars occur at fixed intervals of time so tic bars would not be appropriate for this analysis. Least squares is a mathematical technique where the squared vertical distance between the data and the curve that is being fit to the data is minimized. When the net squared distance (also called the sum of the squared errors) is minimized, a unique set of coefficients $\mathbf{b_0,b_1,b_2,\ldots,b_n}$ for the polynomial is determined. This type of error minimization is mathematically solvable and is widely used in science and mathematics.

For a 4^{th} order polynomial equation, the least squares coefficients are obtained from the solution of the following matrix equation.

$$\begin{vmatrix} T & \sum t & \sum t^2 & \sum t^3 & \sum t^4 \\ \sum t & \sum t^2 & \sum t^3 & \sum t^4 & \sum t^5 \\ \sum t^2 & \sum t^3 & \sum t^4 & \sum t^5 & \sum t^6 \\ \sum t^3 & \sum t^4 & \sum t^5 & \sum t^6 & \sum t^7 \\ \sum t^4 & \sum t^5 & \sum t^6 & \sum t^7 & \sum t^8 \end{vmatrix} \begin{vmatrix} a_0 \\ b_0 \\ c_0 \\ d_0 \\ e_0 \end{vmatrix} = \begin{vmatrix} \sum p(t) \\ \sum (p(t)*t) \\ \sum (p(t)*t^2) \\ \sum (p(t)*t^3) \\ \sum (p(t)*t^4) \end{vmatrix}$$

where

p(T) is the current bar's price, **p(T-1)** is the previous bar's price and **p(1)** is the price **T** bars ago.
T is the number of Bars in the Least Squares estimation
$\mathbf{\sum p(t)}$ is the summation of prices from **t=1 to T** bars
$\mathbf{\sum p(t)*t}$ is the summation of prices times t from **t=1 to T** bars
$\mathbf{\sum t}$ is the summation of the integer **t** from **t=1 to T** bars
$\mathbf{\sum t^2}$ is the summation of the integer **t** squared from **t=1 to T** bars
etc.

Once the coefficients to the polynomial have been solved for we generate the forecast for the next bar's price which is given for the equation by:

$$\mathbf{P_f = a_0 + b_0*(T+1) + c_0*(T+1)^2 + d_0*(T+1)^3 + e_0*(T+1)^4}$$

Where $\mathbf{P_f}$ stands for price forecast.

With these coefficients, we can also generate the forecast for the next bar's ***velocity*** and ***acceleration*** by the equations:

$$\mathbf{Velocity(T+1) = dP_f/dt = b_0 + 2c_0*(T+1) + 3d_0*(T+1)^2 + 4e_0*(T+1)^3}$$

$$\mathbf{Acceleration(t+1) = d^2P_f/d^2t = 2\,c_0 + 6d_0*(T+1) + 12e_0*(T+1)^2}$$

We use the next bar forecast because changes in the trend are more quickly reflected in the forecast price, velocity and acceleration than in the end point price, velocity and acceleration.

Programs that solve for the solution to matrix equations can be found in the book "Numerical Recipes" by W. Press, et. al. However these type of matrix equation solvers are very slow and for these type of problems are unstable. They cause numerical errors and floating point overflows due to matrix inversion ill conditioning which produces false results.

Fortunately these type of problems can be solved by a fast, efficient and accurate algorithm using Discrete Orthogona1 Legendre Polynomials. This method is explained in detail in Norman Morrison' book entitled "Introduction to Sequential Smoothing and Prediction", Chapter 7 page 223., referenced at the end of this section.

Without going into detail here (see Morrison reference), the polynomial filter can now be represented by:

$$P_e(t) = \sum_{j=0}^{n} \beta_j * \phi_j(t) \quad j=0 \text{ to } n$$

Where **n** is the polynomial order, **T** is the total number of Bars of data used in the Least Squares fit and

$$\beta_j = \sum_{k=0}^{T-1} p(t-T+k) * \phi_j(k)$$

$\phi_j(t)$ = the *normalized* discrete Legendre polynomial. t = an integer from 0 to T

The coefficients, $\beta_0, \beta_1, \beta_2, \beta_3, , , , \beta_n$ for a n^{th} order polynomial can now be solved for by the equation above and we can generate the forecast for the next bar's close, velocity and acceleration which are given by the equations

$$P_F(T+1) = \beta_0*\phi_0(T+1) + \beta_1*\phi_1(T+1) + \beta_2*\phi_2(T+1) + \beta_3*\phi_3(T+1) + \ldots + \beta_n*\phi_n(T+1)$$

$$\text{Velocity} = (dP_F/dt)_{(T+1)} = \beta_1*(d\phi_1/dt)_{(T+1)} + \beta_2*(d\phi_2/dt)_{(T+1)} + \beta_3*(d\phi_3/dt)_{(T+1)} ++ \beta_n*(d\phi_n/dt)_{(T+1)}$$

$$\text{Acceleration} = (d^2P_F/d^2t)_{(T+1)} = \beta_2*(d^2\phi_2/d^2t)_{(T+1)} + \beta_3*(d^2\phi_3/d^2t)_{(T+1)} + \ldots + \beta_n*(d^2\phi_n/d^2t)_{(T+1)}$$

The n^{th} Order Fixed Memory Forecast Next Bar's Velocity System Defined

The least squares forecast is constructed by solving for the least squares coefficients $\beta_1, \beta_2, , , \beta_n$ at each bar using the last **T** bars of closing prices and the Discrete Orthogonal Legendre Polynomial equations for β_j above. Then **Velocity** = $\mathbf{dP_F(T+1)/dt}$ is constructed from the velocity equation above and plotted under the price chart. In general what we will be doing is following the plotted curve of **Velocity** which is calculated at each bar from the previous T bars. When the velocity is greater than a threshold amount ***vup*** we will go long. When the velocity is less than a threshold amount ***-vdn*** we will go short.

Buy Rule:
IF Velocity is greater than the threshold amount ***vup*** then buy at the market.

Sell Rule:
IF Velocity is less than the threshold amount ***-vdn*** then sell at the market.

References

1. Morrison, Norman "Introduction to Sequential Smoothing and Prediction", McGraw-Hill Book Company, New York, 1969.

