1986

Kalman filtering approach to market price forecasting

James Martin Rankin
Iowa State University

Rankin, James Martin

KALMAN FILTERING APPROACH TO MARKET PRICE FORECASTING

Iowa State University PH.D. 1986

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark __√__.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages ✓
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Dissertation contains pages with print at a slant, filmed as received _____
16. Other _____

_____

_____

University
Microfilms
International

Kalman filtering approach to market price forecasting

by

James Martin Rankin

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Department: Electrical Engineering and Computer Engineering
Major: Electrical Engineering (Control Systems)

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa

1986

TABLE OF CONTENTS

INTRODUCTION

The Kalman filter was a significant breakthrough in the area of linear filtering and prediction. It has been used in the processing of signals imbedded in noise for over twenty five years. A major application of Kalman filtering is the solution of navigational problems where information is received from multiple noisy sources. The Kalman filter has also been used for applications outside the area of navigation. C. R. Szelag [35] published an article in the Bell System Technical Journal using a Kalman filter to forecast telephone loading. The Kalman filter has even made its way into the economic literature. The Kalman filter has been used to forecast economic quantities such as sales and inventories [23].

This project examines the use of the Kalman filter to forecast intraday stock and commodity prices. The price forecasts are based on a market's price history with no external information included. For the Kalman filter to produce beneficial forecasts, the market must not be a random walk process, but must exhibit a statistically significant autocorrelation pattern which can be modeled. Once an appropriate Kalman filter model is determined, strategies for increasing profits can be studied.

This dissertation presents the analysis techniques used to detect autocorrelation in a market and the models used to

describe the correlation. Several stock indexes and commodity markets are tested for autocorrelation. The Kalman filter algorithm and an adaptive Kalman filter algorithm are also presented and then are used to forecast prices for the Dow Jones Transportation index. Several buy and sell strategies are used to investigate the use of the Kalman filter forecasts to benefit market traders.

HISTORY OF MARKET ANALYSIS

Market analysts are divided into two groups: fundamentalists and technical analysts. Fundamentalists base their analysis on the law of supply and demand and other economic principles. Technical analysts believe that future market behavior is not totally random, but related to past market behavior. This project is based on the theories used in technical analysis.

Random Walk Argument

Since the turn of the century, the question of whether market prices are random walk processes or not has been argued. In 1900, Bachelier [in Cootner [8]] proposed that price differences are independent and that market prices follow a random walk model. If price changes are independent, then price forecasting is not beneficial since the best estimate is just the previous price. Working [39] and Kendall [21] stated that security prices are statistically independent of past history and that changes between successive items tend to be largely random. From these statistical results, the inference was made that mechanical trading rules will not work.

Euguene Fama [13], considered the Father of the efficient market theory, justified the random walk model with the following logic:

1. Stock prices are the accumulation of randomly generated noise which is unrelated to real-world economic and political events.

2. If a noise generating process is dependent, there are enough noise sources that the resulting actions are neutralized and price differences are independent.

3. If a new strategy appears that allows a profit, the number of traders using the new strategy will grow until the strategy is no longer profitable.

Fama tested his random walk hypothesis by calculating frequency distributions, normal probability charts, autocorrelation functions, and run tests. (Run tests study the number of consecutive price changes which have the same sign.)

Technical analysts argue that since mechanical trading rules can produce a profit, markets must not be random walk. Alexander [1] tested one such mechanical trading rule, the filter technique. Alexander's results indicated that filters of all different sizes and all different time periods yield profits significantly larger than a simple buy-and-hold policy.

In an apparent attempt to satisfy the technical analysts, Fama tested Alexander's filter technique. Alexander's original tests were discovered to be flawed since they did not account for slippage (the price change from when the order was placed until the order was filled). Fama's tests resulted in the buy-and-hold policy showing a larger profit than the filter technique.

Studies were also done showing that not all markets are random walk processes. Cargill and Rausser [5] disproved the random walk hypothesis by using autocorrelation functions and spectral analysis. Their results invalidated the random walk as a general explanation of futures behavior, but for a number of commodities the random walk model is consistent with price behavior. A significant number of the futures contracts studied had either a first or second lag that was statistically significant. From this evidence, it was not possible to infer that these coefficients were selected from a population with zero autocorrelation.

Data Sampling

Previous market studies have been based on yearly, quarterly, monthly, weekly, and daily prices. The data considered are usually closing prices or an average of daily closing prices for the period. Studies have also been done on open, high, low, and close prices.

The use of averages (e.g., weekly or monthly) or stock indexes may alter the results of a study. Osborne [26] stated that using averages instead of actual data corrupts the investigation. A positive correlation appears when you might have white noise. Kendall [21] found that indexes appear to behave more systematically than individual stocks. He suggested that this might be due to the reduction of random elements by averaging.

Intraday Analysis

Recently, market analysis has examined intraday, overnight, and weekend effects. Wood, McInish, and Ord [38] formed a stock index by averaging approximately 1000 stocks over a six month period. They calculated the index value minute by minute to form a "typical" day. Autocorrelations of the data showed that only the first thirty minutes of the day had significant correlation and it lasted for only the first twelve lags. Other trading intervals during the day did not exhibit any significiant autocorrelation. It was also found that inclusion of the overnight price difference and infrequent trading induced correlation.

Mechanical Trading Rules

Mechanical trading rules and the growth of personal computers have started to change the way that the markets perform [15]. A few big brokerage houses are using automatic buying and selling programs which sell stocks and buy stock futures when the stock price exceeds the futures price, and buy stocks and sell the futures when the stock price falls below the futures price. Investors then profit in either case. Some Wall Street analysts say the programs are partly responsible for big swings in the Dow Jones Industrial. Quoting Michael Metz, analyst with Oppenheimer & Co., "These programs trade hour by hour, day by day. This is going on all the time. They tend to exaggerate moves once they are

under way" [15]. Robert Colby, analyst from Smith Barney, Harris Upham & Co., said, "For 15 minutes up to half a day I think the programs can be a dominant force, but there is no effect on long term trends" [15].

MARKET ANALYSIS

Introduction

The first market analysis objective is to determine if there is any correlation present in a market. If a correlation pattern does exist, the second objective is to develop a model which generates data with the same statistical parameters. The models used to realize the market process are: Gauss-Markov, damped cosine, and ARIMA. After a model is selected, the model is used to forecast future market prices.

Market analysis is divided into 2 areas: stock indexes and individual commodity markets. Stock indexes were chosen as the initial area of investigation because hourly data were more readily available and because previous research suggested that stock indexes tended to be more correlated. The use of stock indexes allows preliminary models to be identified before analyzing individual markets where the correlation structure may be smaller, if not non-existent.

This chapter examines the analysis techniques used and suggested models for realizing the market processes. Several stock indexes and commodity markets were tested and those exhibiting a correlation pattern were fit to the suggested models.

Analysis Techniques

Two techniques used to determine if a correlation pattern exists in a time-series are the autocorrelation function (ACF) and partial autocorrelation function (PACF). The ACF calculates the correlation between samples which are k periods apart. The ACF is calculated from a sequence of summations as shown in (2-1).

$$r(k) = \frac{N}{N-k} * \frac{\sum_{i=0}^{N-k-1} x(i)*x(i+k)}{\sum_{i=0}^{N-1} x(i)*x(i)} \qquad (2\text{-}1)$$

where N is the number of samples in the time series. The ACF will also show if there are any periodicities in the time series.

The PACF function is based on the ACF and determines the correlation between samples k periods apart after removing any correlation effects from intermediate samples [27]. The PACF formula is shown in (2-2).

$$\phi(k,k) = \frac{r(k) - \sum_{j=0}^{k-1} \phi(k-1,j)*r(k-j)}{1 - \sum_{j=0}^{k-1} \phi(k-1,j)*r(j)} \qquad (2\text{-}2)$$

where $\phi(k,j) = \phi(k-1,j) - \phi(k,k)*\phi(k-1,k-j)$.

The ACF and PACF formulas presented are estimates of the

actual correlation coefficients since they are calculated with sampled data. They are both normalized such that the coefficient at k=0 is equal to 1. Since this value is constant, lag 0 is not shown in any of the following figures.

To determine if any correlation is significant, the coefficient at each lag is checked to see if it is statistically different from the null hypothesis, or r(k)=0. A correlation coefficient is considered statistically significant if it lies outside the 95% confidence interval around r(k)=0 [27]. The 95% confidence level is met if the magnitude of the T-ratio is greater than 1.96. The T-ratio is calculated from

$$T = \frac{r(k)}{s_k} \quad \text{or} \quad = \frac{\phi(k,k)}{s_k} \tag{2-3}$$

where s_k is the standard deviation of the coefficient. The standard deviation, s_k, for the ACF is calculated from

$$s(k) = \left(1 + 2 \sum_{j=1}^{k-1} r(j)^2 \right)^{1/2} * N^{-1/2} \tag{2-4}$$

where the upper summation limit is determined from the moving average length of the model. For example, the upper limit is 3 if the model being used has a MA(3) component. The standard deviation of a PACF coefficient is calculated from

$$s_k = N^{-1/2}. \qquad (2\text{-}5)$$

The market data studied required a large number of samples to reduce the uncertainty of the correlation coefficient due to sample size. The large value of N reduced s_k such that the statistically significant ACF and PACF coefficients were not obscured in noise.

The ACF (2-1) and the PACF (2-2) formulas assume that the time series is a stationary process. Market prices have a non-stationary mean, therefore, a stationary working series is created by taking the first difference of the prices. If the time series also had a non-stationary variance, then the natural logarithm of the prices would have been calculated before doing the differencing. The ACF and PACF are then calculated using the stationary working series.

Models

The next step in market analysis is to choose models which exhibit the same statistical qualities as the market price data. Models were determined by two methods in this study. Both methods involved matching known models to the correlation structure of the sample data. Since the market price realizations are non-stationary, the first difference of the market prices are used for analysis. The first method matches the first difference ACF with common engineering models which provide similar ACFs. The second method uses

Box and Jenkin's Autoregressive Integrated Moving Average (ARIMA) analysis techniques and the associated family of time series models.

When correlation is present in the first difference, the first difference ACF consists of white and colored noise components. The white noise is represented by a spike at lag zero and the colored noise is represented by a significant pattern in the non-zero lags. An engineering model produces a realization of the colored noise by driving white Gaussian noise into a shaping filter [4]. The shaping filter output has the same statistical qualities as the colored noise. The transfer function of the shaping filter is determined from the engineering model's Power Spectral Density (PSD) function which is the Fourier transform of the model's ACF as shown in (2-6).

$$S(s) = F\{ R(t) \} \qquad (2\text{-}6)$$

The models chosen are rational in s^2 so that spectral factorization can be performed on S(s) to obtain the shaping filter transfer function [4]. Spectral factorization separates the PSD into two parts. One part, $S^+(s)$, has all the poles and zeroes in the left half plane, and the second part, $S^-(s)$, contains the poles and zeroes that lie in the right half plane.

$$S(s) = S^+(s)*S^-(s) \qquad (2\text{-}7)$$

$S^+(s)$ is used as the transfer function of the shaping filter in Figure 2-1. The output of the shaping filter, x, has the PSD shown in (2-8).

$$S_x(s) = S^+(s)*S^+(-s)*S_w(s) \qquad (2\text{-}8)$$

If the input, w, is white Gaussian noise which has a PSD of $S_w(s) = 1$, then the PSD of the output is the same as the PSD of the random process being modelled.

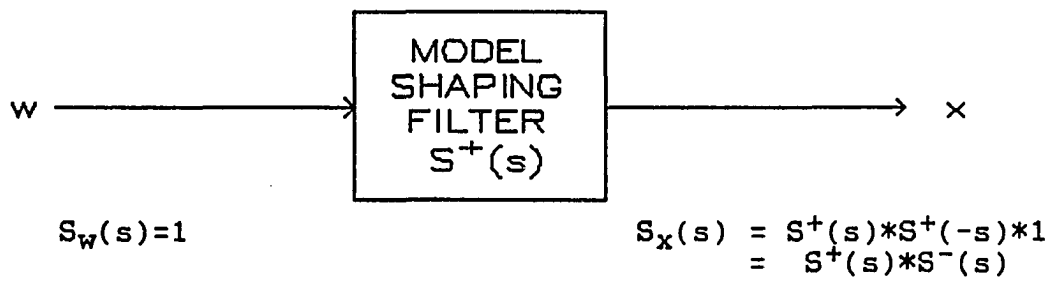

Figure 2-1. Shaping Filter Realization

Using standard linear systems analysis techniques [6], the shaping filter transfer function can be converted to continuous-time, state-space equations of the form shown in (2-9a) and (2-9b).

$$\dot{\underline{x}} = F\underline{x} + Gw \qquad (2\text{-}9a)$$

$$y = B\underline{x} \qquad (2\text{-}9b)$$

where $\underline{x}$ is the state vector, w is the white noise driving function, and y is the output. The continuous-time model

(2-9) is then converted to a discrete-time, state-space model (2-10a) and (2-10b) for use by a Kalman filter.

$$\underline{x}_{k+1} = \Phi * \underline{x}_k + w_k \qquad (2\text{-}10a)$$

$$y_k = B * \underline{x}_k \qquad (2\text{-}10b)$$

When F is time-invariant, the state-transition matrix, Φ, is calculated from

$$\Phi = L^{-1}[(sI - F)^{-1}] \qquad (2\text{-}11)$$

where L^{-1} is the inverse Laplace transform and I is an nxn identity matrix. The white driving sequence, w_k, is calculated using the integral shown in (2-12).

$$w_k = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1} - u) * G * w(u)\, du \qquad (2\text{-}12)$$

The connection matrix, B, is not changed in the conversion to the discrete format.

An additive white noise source is added to the output of the engineering model to realize the complete first difference data process. An extra state is also included to perform the discrete integration needed to convert the first difference data back to the original market price data. The complete block diagram for the market price realization is shown in Figure 2-2.

The engineering models considered are the Gauss-Markov

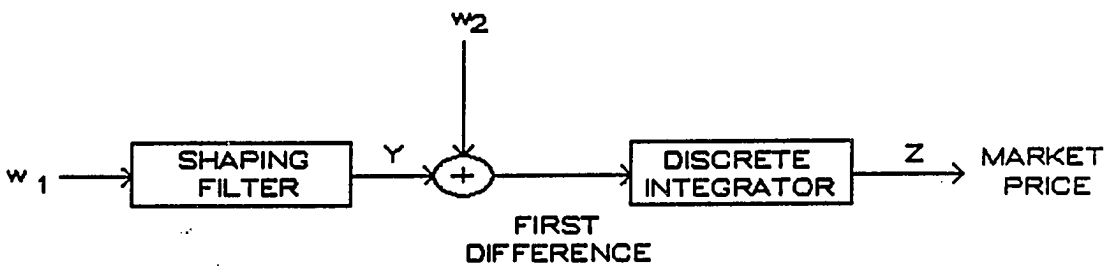

Figure 2-2. Block diagram of Engineering Model Approach to Market Price realization

and damped cosine models. The random walk process is included to show the appropriate model if no correlation is present in the first difference ACF.

Random walk

A random walk process can be described as integrated white Gaussian noise. It is a non-stationary process with a mean of 0 and variance which increases with time. The expected value for the next sample of the process is the same as the present sample.

A state space model for a random walk process can be derived from the differential equation shown in (2-13).

$$\dot{x} = w(t) \tag{2-13}$$

Converting (2-13) to discrete-time form produces the following state-space model for the random walk process.

$$x_{k+1} = x_k + w_k \tag{2-14a}$$

$$y_k = x_k \tag{2-14b}$$

Gauss-Markov

The Gauss-Markov process has an exponential ACF as shown in (2-15) with zero mean and a variance of σ^2.

$$R(nT) = \sigma^2 e^{-\beta|nT|} \tag{2-15}$$

where T is the sampling period. The parameters β and σ^2 are estimated from the ACF being modeled. The farther that two samples are separated, the smaller the correlation between them.

The PSD of the Gauss-Markov model is defined by

$$S(s) = \frac{2\sigma^2\beta}{-s^2 + \beta^2} \tag{2-16}$$

and the shaping filter transfer function is

$$S^+(s) = \frac{\sqrt{2\sigma^2\beta}}{s + \beta} \tag{2-17}$$

Converting the transfer function to a differential equation produces the following continuous-time state equation.

$$\dot{x} = -\beta x + \sqrt{2\sigma^2\beta} * w(t) \tag{2-18a}$$

$$y = 1 * x \tag{2-18b}$$

Converting (2-18) to a discrete-time state equation then gives

$$x_{k+1} = e^{-\beta T} * x_k + w_k \tag{2-19a}$$

$$y_k = 1 * x_k \tag{2-19b}$$

where w_k is defined as

$$w_k = \int_{t_k}^{t_{k+1}} e^{-\beta(t_{k+1}-u)} * \sqrt{2\sigma^2\beta} * w(u)\, du \qquad (2\text{-}20)$$

A graphic example of the additive combination of white noise and the Gauss-Markov model is shown in Figure 2-3. This is similar to a first difference ACF for a market price process which includes an exponential autocorrelation component.

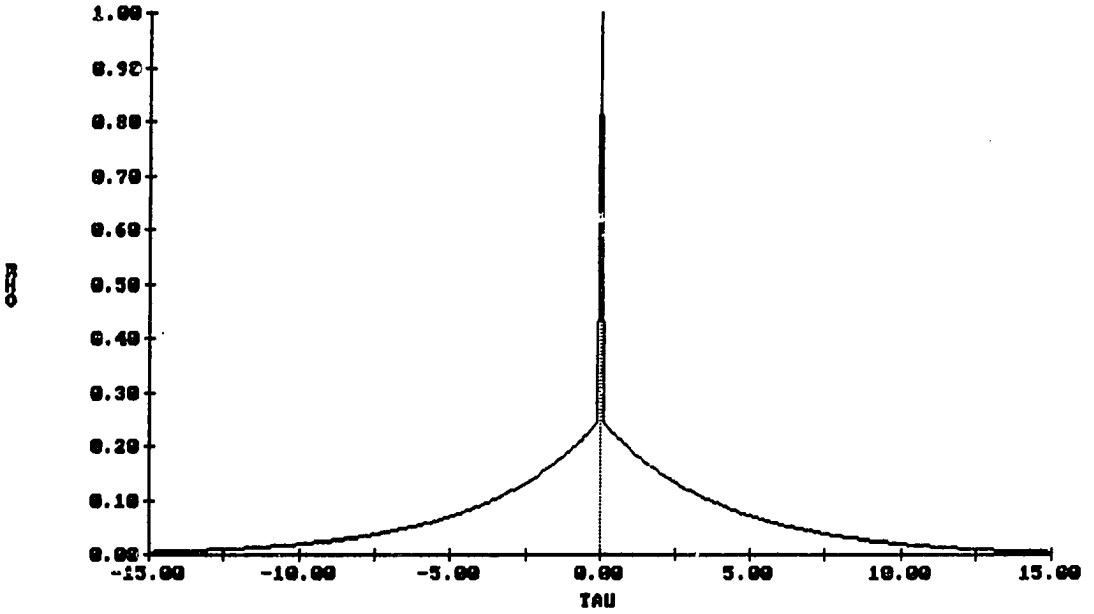

Figure 2-3. Example ACF for a Gauss-Markov model plus additive white noise

Damped cosine

The damped cosine model's ACF is the product of an exponential and a cosine as shown in (2-21).

$$R(nT) = \sigma^2 * e^{-\alpha|nT|} * \cos(\beta nT) \qquad (2\text{-}21)$$

The damped cosine allows negative correlation to occur in the process. The parameters σ^2, α, and β are again to be estimated from the sample ACF.

The damped cosine model's PSD and shaping filter transfer function are shown in (2-22) and (2-23), respectively.

$$S(s) = \frac{2\sigma^2\alpha(-s^2 + \alpha^2 + \beta^2)}{s^4 + 2s^2(\beta^2 - \alpha^2) + (\alpha^2 + \beta^2)^2} \qquad (2\text{-}22)$$

$$S^+(s) = \frac{\sqrt{2\sigma^2\alpha}(s + \sqrt{\alpha^2 + \beta^2})}{s^2 + 2\alpha s + \alpha^2 + \beta^2} \qquad (2\text{-}23)$$

Converting the shaping filter transfer function (2-23) to continuous state-space format results in (2-24).

$$\dot{x}_1 = x_2 \qquad (2\text{-}24a)$$

$$\dot{x}_2 = -(\alpha^2 + \beta^2)x_1 - 2\alpha x_2 + w \qquad (2\text{-}24b)$$

$$y = \begin{bmatrix} \sqrt{2\sigma^2\alpha}\sqrt{\alpha^2+\beta^2} & \sqrt{2\sigma^2\alpha} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad (2\text{-}24c)$$

The discrete state equations for the damped cosine model are presented in (2-25a) and (2-25b).

$$x_1(k+1) = (D_1+\alpha D_2)x_1(k) + D_2x_2 + W_1(k) \qquad (2\text{-}25a)$$

$$x_2(k+1) = -(\alpha^2+\beta^2)D_2x_1(k) + (D_1-\alpha D_2)x_2(k) + W_2(k) \qquad (2\text{-}25b)$$

where

$$D_1 = e^{-|\alpha T|} * \cos(\beta T) \qquad (2\text{-}26)$$

$$D_2 = [e^{-|\alpha T|} * \sin(\beta T)]/\beta \qquad (2\text{-}27)$$

and

$$\begin{bmatrix} W_1(k) \\ W_2(k) \end{bmatrix} = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}-u)*G*w(u)\, du \qquad (2\text{-}28)$$

where Φ is the state transition matrix.

An example of the additive combination of the Damped Cosine model and white noise is shown in Figure 2-4.

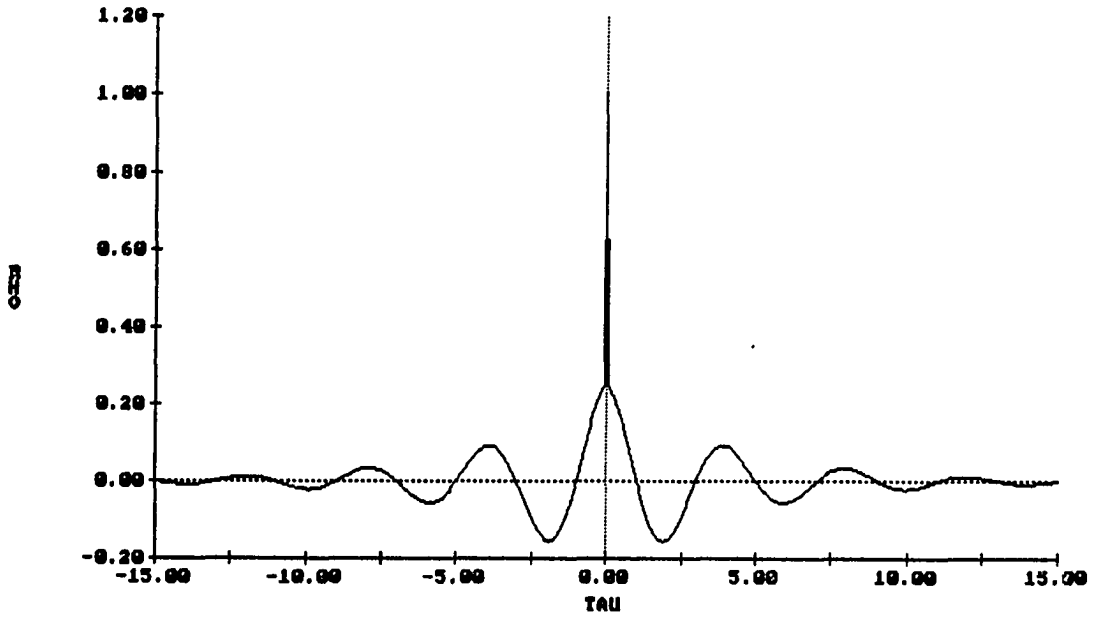

Figure 2-4. Example ACF of the damped cosine model plus additive white noise

ARIMA models

Box and Jenkin's ARIMA models are actually a family of discrete models used to realize time series data [27]. The

ARIMA models are determined through a three stage iterative procedure. The identification, or first, stage uses the ACF and PACF of the sample data to suggest possible models. The estimation, or second, stage optimizes the model parameters such that the mean square forecast error is minimized for the given data sample. The diagnostic stage examines the forecast errors to determine if the model is acceptable. If it is not, the procedure returns to the identification stage.

ARIMA models consist of 3 main parts: the autoregressive (AR) part which specifies how the next value is correlated with previous values, the integrated (I) part which specifies the number of differences required to transform the original time series to a stationary working series, and the moving average (MA) part which specifies how the next value is correlated to previous noise values. The AR terms represent the characteristic equation of the process. They specify the sinusoidal and/or exponential patterns in the time series. The integrated and MA terms account for the non-stationarity and white noise inputs to the process. The ARIMA (p,d,q) family of models are of the form

$$(1-\phi_1 B - \cdots - \phi_p B^P)(1-B)^d Z_k = (1-\theta_1 B - \cdots - \theta_q B^q) w_k \quad (2\text{-}29)$$

where

p is the number of AR terms,

d is the number of differences,

q is the number of MA terms,

ϕ_i are the AR coefficients,

θ_i are the MA coefficients,

B is a unit time delay operator,

Z_k is the measurement at time t_k, and

w_k is the residual error at time t_k.

The ARIMA models are converted to state space through an iterative process. First, the model equation in (2-29) is multiplied through and arranged as shown in (2-30).

$$Z_k = \phi_1 Z_{k-1} + \phi_2 Z_{k-2} + \cdots - \theta_1 w_{k-1} - \cdots + w_k \qquad (2\text{-}30)$$

The state definition begins by replacing Z with x_1 in (2-30) and then substituting (2-31) into (2-30).

$$x_j(k-j+1) = \phi_j x_1(k-j) + \theta_{j-1} w(k-j+1) \qquad (2\text{-}31)$$

where j is the number of states and is equal to the larger of p+d or q+1. Each succeeding state equation is of the form

$$x_i(k+1) = \phi_i * x_1(k) + x_{i+1}(k) + \theta_{i-1} * w(k+1) \qquad (2\text{-}32)$$

where i varies from j-1 down to 1. As each state equation is defined, (2-32) is substituted into (2-30). For example, this process leads to the following matrix format for an ARIMA (3,0,2) model.

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} \phi_1 & 1 & 0 \\ \phi_2 & 0 & 1 \\ \phi_3 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 1 \\ -\theta_1 \\ -\theta_2 \end{bmatrix} W_{k+1} \qquad (2\text{-}33)$$

The ARIMA models can be used to simulate the discrete forms of the engineering models previously discussed. The difference occurs in the way that the parameters are allowed to vary. The optimization of the ARIMA models assumes each parameter varies independently. When the engineering models are simulated, the parameters are dependent on each other to guarantee that the proper relationships in the engineering models are not disturbed. The Gauss-Markov plus white noise model of the first difference is a special case of an ARIMA (1,1,1) and the damped cosine plus white noise model is a special case of an ARIMA (2,1,2).

Stock Indexes

Stock indexes are weighted averages of selected stock prices. A stock index may concentrate on stocks in a particular industry, e.g., the Dow Jones 20 Transportations Index, or an index may be a collection of stocks across the entire market such as the Standard and Poor's Composite 500 Index.

The Dow Jones 20 Transportations, Dow Jones 30 Industrials, and Standard and Poor's 40 Financials indexes chosen for this project are based on the New York Stock

Exchange. These indexes are reported hourly throughout the day and at the market closing. An opening price is also reported for the Dow Jones indexes. Hourly data for the Dow Jones Indexes and Standard and Poor's Financial Index were obtained from The Wall Street Journal [12] and Standard and Poor's Corporation Records - Daily News Section [33], respectively. Daily, weekly, and monthly prices for all stock indexes were found in the Daily Stock Price Record [34].

Dow Jones 20 Transportations Index

The Dow Jones 20 Transportations Index (DJT) was the initial index studied and ended up being the baseline data for the analysis. The first sample realization (DJT #1) consisted of 1036 hourly readings from February 22, 1985, to September 23, 1985. Figure 2-5 depicts the DJT #1 hourly data. The first difference ACF and PACF suggest that a small colored noised component is present. The ACF and PACF for DJT #1's first difference are shown in Figure 2-6. The first difference correlation pattern indicates that ARIMA (2,0,0) and ARIMA (1,0,1) models should be estimated along with the Gauss-Markov and damped cosine models. Estimating the model parameters provide the following models for the DJT #1 first difference:

Gauss-Markov:

$$R(nT) = 0.3922\, e^{-0.3227|nT|} \qquad (2\text{-}34)$$

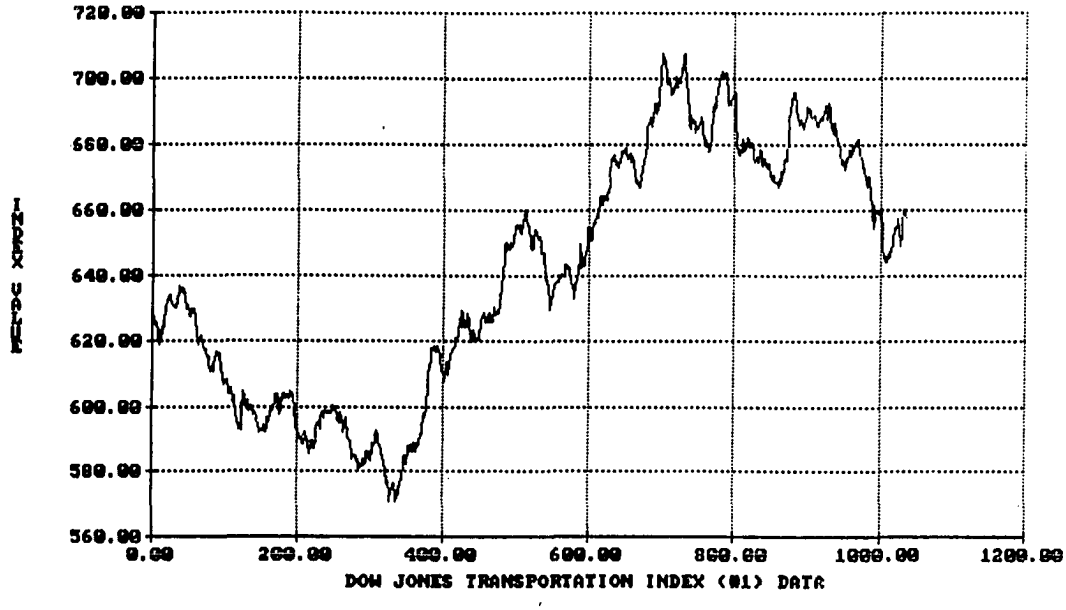

Figure 2-5. DJT #1 realization

ACF

LAG	COEFF	S	T-VAL	-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
1	.174	.031	5.61	| (***)******
2	.106	.032	3.30	| (***)**
3	.069	.032	2.13	| (***)
4	.086	.032	2.65	| (***)*
5	.053	.033	1.61	| (***)
6	.056	.033	1.70	| (***)
7	.036	.033	1.10	| (***)
8	-.030	.033	-.91	| (***)
9	.001	.033	.03	| (*)
10	.034	.033	1.04	| (***)
11	-.014	.033	-.43	| (**)
12	.008	.033	.24	| (*)

PACF

LAG	COEFF	S	T-VAL	-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
1	.174	.031	5.61	| (***)******
2	.078	.031	2.50	| (***)*
3	.040	.031	1.28	| (***)
4	.063	.031	2.03	| (***)
5	.021	.031	.69	| (**)
6	.031	.031	1.01	| (***)
7	.011	.031	.36	| (**)
8	-.054	.031	-1.74	| (***)
9	.002	.031	.07	| (*)
10	.033	.031	1.08	| (***)
11	-.028	.031	-.90	| (**)
12	.013	.031	.40	| (**)

Figure 2-6. ACF and PACF of DJT #1

ARIMA (2,0,0):

$$Z_k = 0.1612\ Z_{k-1} + 0.0773\ Z_{k-2} + w_k \qquad (2\text{-}35)$$

ARIMA (1,0,1):

$$Z_k = 0.7222\ Z_{k-1} - 0.5792\ w_{k-1} + w_k \qquad (2\text{-}36)$$

The damped cosine model reverted to the Gauss-Markov model for DJT #1.

To confirm that the statistically significant autocorrelation found for DJT #1 was not caused by non-typical sample data, second and third realizations of DJT data were collected. The second realization (DJT #2) consisted of 896 hourly readings from January to June, 1984. The 128 days of samples provided an ACF which had positive correlation for the first 5 lags. The ACF and PACF for DJT #2 are shown in Figure 2-7.

The suggested models for the DJT #2 first difference are:

Gauss-Markov:

$$R(nT) = 0.3311\ e^{-0.3689|nT|} \qquad (2\text{-}37)$$

Damped Cosine:

$$R(nT) = 0.1875\ e^{-0.0567|nT|}\ \cos(0.0555nT) \qquad (2\text{-}38)$$

ARIMA (3,0,0)

$$Z_k = 0.0925\ Z_{k-1} + 0.0425\ Z_{k-2} + 0.1108\ Z_{k-3} + w_k \qquad (2\text{-}39)$$

ARIMA (1,0,1):

$$Z_k = 0.7448\ Z_{k-1} - 0.6459\ w_{k-1} + w_k \qquad (2\text{-}40)$$

ACF

LAG	COEFF	S	T-VAL	-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
1	.100	.033	2.99	(***)**
2	.060	.034	1.76	(***)
3	.117	.034	3.46	(***)***
4	.058	.034	1.68	(***)
5	.055	.034	1.59	(***)
6	-.007	.035	-.19	(*)
7	-.029	.035	-.83	(**)
8	-.070	.035	-2.02	(***)
9	-.012	.035	-.35	(**)
10	-.003	.035	-.08	(*)
11	-.064	.035	-1.85	(***)
12	.029	.035	.83	(**)

PACF

LAG	COEFF	S	T-VAL	-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
1	.100	.033	2.99	(***)**
2	.050	.033	1.50	(***)
3	.108	.033	3.23	(***)**
4	.035	.033	1.04	(***)
5	.036	.033	1.09	(***)
6	-.032	.033	-.95	(***)
7	-.040	.033	-1.20	(***)
8	-.077	.033	-2.29	*(***)
9	.003	.033	.09	(*)
10	.012	.033	.34	(**)
11	-.044	.033	-1.33	(***)
12	.050	.033	1.51	(***)

Figure 2-7. ACF and PACF of DJT #2

The DJT #2 first difference required an ARIMA (3,0,0) model instead of an ARIMA (2,0,0). The ARIMA (2,0,0) model could not reduce the correlation in the forecasts errors below an acceptable level.

The third realization (DJT #3) was gathered from July through December, 1983. It consists of 896 samples from the 128 day period. The DJT #3 ACF had a positive correlation for the first seven lags. Figure 2-8 shows the ACF and PACF for DJT #3.

The models suggested for the DJT #3 realization are:

ACF

LAG	COEFF	S	T-VAL	-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
1	.113	.033	3.39	(***)***
2	.102	.034	3.00	(***)**
3	.070	.034	2.04	(***)
4	.061	.034	1.79	(***)
5	.071	.034	2.05	(***)*
6	.059	.035	1.71	(***)
7	.032	.035	.91	(***)
8	-.035	.035	-1.02	(***)
9	-.017	.035	-.50	(**)
10	-.009	.035	-.27	(*)
11	-.033	.035	-.94	(***)
12	.004	.035	.12	(*)

PACF

LAG	COEFF	S	T-VAL	-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
1	.113	.033	3.39	(***)***
2	.090	.033	2.69	(***)**
3	.050	.033	1.50	(***)
4	.041	.033	1.23	(***)
5	.051	.033	1.52	(***)
6	.036	.033	1.08	(***)
7	.007	.033	.21	(*)
8	-.057	.033	-1.71	(***)
9	-.023	.033	-.69	(**)
10	-.008	.033	-.23	(*)
11	-.031	.033	-.93	(***)
12	.014	.033	.42	(**)

Figure 2-8. ACF and PACF of DJT #3

Gauss-Markov:

$$R(nT) = 0.3298\, e^{-0.3090|nT|} \qquad (2\text{-}41)$$

Damped Cosine:

$$R(nT) = 0.2767\, e^{-0.1501|nT|} \cos 0.0307nT \qquad (2\text{-}42)$$

ARIMA (2,0,0)

$$Z_k = 0.1031\, Z_{k-1} + 0.0900\, Z_{k-2} + w_k \qquad (2\text{-}43)$$

ARIMA (1,0,1):

$$Z_k = 0.7710\, Z_{k-1} - 0.6664\, w_{k-1} + w_k \qquad (2\text{-}44)$$

All three DJT hourly realizations did exhibit a positive correlation for the first few lags. One goal of this

analysis is to determine if a single model is acceptable for all realizations of a given market. These DJT realizations all had the Gauss-Markov and ARIMA (1,0,1) models in common. These models will be tested later in a Kalman filter to determine which one produces the optimum forecasts.

To determine if the positive correlation could be found for larger sampling periods, the Dow Jones Transportation index was also sampled daily, weekly, and monthly. The daily closing data did not provide any statistically significant correlation. Monthly and weekly data from January, 1977, through September, 1985, did not exhibit any significant correlation, either.

The Dow Jones Transportations Index had significant autocorrelation when sampled hourly, but appears to be a random walk process when sampled weekly and monthly. The Dow Jones 30 Industrials and the Standard and Poor's 40 Financials indexes were also tested to see if the hourly correlation is present in other indexes.

Dow Jones 30 Industrials Index

The Dow Jones 30 Industrials Index (DJI) was sampled hourly from January 2 through March 28, 1985. The data did not provide any significant correlation. Daily closing prices from January 2 through June 28, 1985, and monthly readings from January 1981 through September 1985 did not produce any statistically significant correlation, either.

Standard and Poor's 40 Financials Index

Examining hourly prices from May through September, 1985, provided a significant correlation pattern. The correlation is apparent for six lags which is the number of measurements available each day. The ACF and PACF are shown in Figure 2-9.

ACF

LAG	COEFF	S	T-VAL
1	.160	.049	4.04
2	.115	.041	2.83
3	.103	.041	2.51
4	.092	.042	2.21
5	.038	.042	.90
6	.103	.042	2.44
7	.014	.042	.34
8	.017	.042	.41
9	.015	.042	.35
10	.029	.042	.68
11	-.036	.042	-.84
12	.000	.042	-.01

PACF

LAG	COEFF	S	T-VAL
1	.160	.040	4.04
2	.092	.040	2.32
3	.074	.040	1.87
4	.059	.040	1.48
5	.000	.040	.01
6	.080	.040	2.02
7	-.026	.040	-.66
8	-.004	.040	-.11
9	-.001	.040	-.03
10	.016	.040	.39
11	-.046	.040	-1.16
12	-.002	.040	-.05

Figure 2-9. ACF and PACF of the S&P Financial Index

The following engineering models were suggested by estimating the model parameters from the first difference ACF and PACF:

Gauss-Markov:

$$R(nT) = 0.0119\ e^{-0.3172\ |nT|} \qquad (2\text{-}45)$$

Damped cosine model:

$$R(nT) = 0.0099\ e^{-0.1767\ |nT|}\ \cos(0.0273nT) \qquad (2\text{-}46)$$

The first difference ACF and PACF also suggested ARIMA (2,0,0) and (1,0,1) models, but their parameters were not estimated.

The S&P Financial index was also sampled daily from January through June, 1985, but no correlation pattern was present.

Commodity Markets

Several commodity markets were examined to see if any correlation structure existed in a single market. Since a customer can buy and sell in a commodity market, the results could be used more directly than those from stock indexes. The commodity markets examined were corn, soybeans, United States Treasury bonds, gold, and Standard and Poor's Composite 500 Index futures.

Transaction data for each commodity was obtained directly from the respective commodity exchange. These data were then sampled each half hour, each fifteen minutes, and each minute. The transaction data were also used without regard to the time between transactions.

Corn

The July 1986 corn contract on the Chicago Board of Trade (CBOT) was sampled every half hour from January 2 to January 15, 1986. The coefficient for lag 3 was the only statistically significant lag. This value was probably a result of sample variation.

Corn prices were then sampled every minute from January 6 to January 10, 1986. The 1604 samples produced a negative exponential pattern in the first difference. This pattern could be modeled with a Gauss-Markov model or an ARIMA (1,0,1) model. The correlation only exists for approximately 6 minutes. The first difference ACF and PACF are shown in Figure 2-10. Corn was not traded very actively during the period sampled, therefore the first difference of the samples is quite often zero. The effect this had on the ACF and PACF was not studied.

Analysis of the corn prices at each transaction was based on 659 transactions from January 2 through January 15, 1986. The first two ACF lags were significantly correlated with the first lag having a T-ratio of -11.03. The first difference of the transaction data appears to fit an ARIMA (1,0,0) process. The ACF and PACF are shown in Figure 2-11.

United States Treasury Bonds

The U.S. Treasury Bond market at the CBOT was sampled every half hour from January 2 through January 22, 1986.

ACF

LAG	COEFF	S	T-VAL	-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
1	-.160	.025	-6.41	******(**)
2	-.066	.026	-2.57	(***)
3	-.037	.026	-1.45	(***)
4	.000	.026	.00	(*)
5	.008	.026	.33	(*)
6	-.057	.026	-2.23	(***)
7	.011	.026	.44	(**)
8	-.043	.026	-1.66	(***)
9	.000	.026	.00	(*)
10	-.003	.026	-.11	(*)
11	.009	.026	.33	(*)
12	-.035	.026	-1.34	(***)

PACF

LAG	COEFF	S	T-VAL	-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
1	-.160	.025	-6.41	******(**)
2	-.094	.025	-3.76	***(**)
3	-.066	.025	-2.64	*(**)
4	-.025	.025	-1.01	(**)
5	-.004	.025	-.17	(*)
6	-.063	.025	-2.54	*(**)
7	-.011	.025	-.44	(**)
8	-.056	.025	-2.22	*(**)
9	-.025	.025	-1.00	(**)
10	-.019	.025	-.74	(**)
11	-.003	.025	-.11	(*)
12	-.044	.025	-1.75	(**)

Figure 2-10. ACF and PACF of Corn Sampled by Minute

There was no significant correlation present in this data realization.

The transaction data for January 2 and 3, 1986, produced significant correlation in the first and third lags of the ACF, but no significant pattern could be identified.

Soybeans

The July 1986 contract for soybeans on the CBOT was sampled each half-hour, each minute, and by transaction in January 1986. There was no significant correlation in the the ACF or PACF for any sample period.

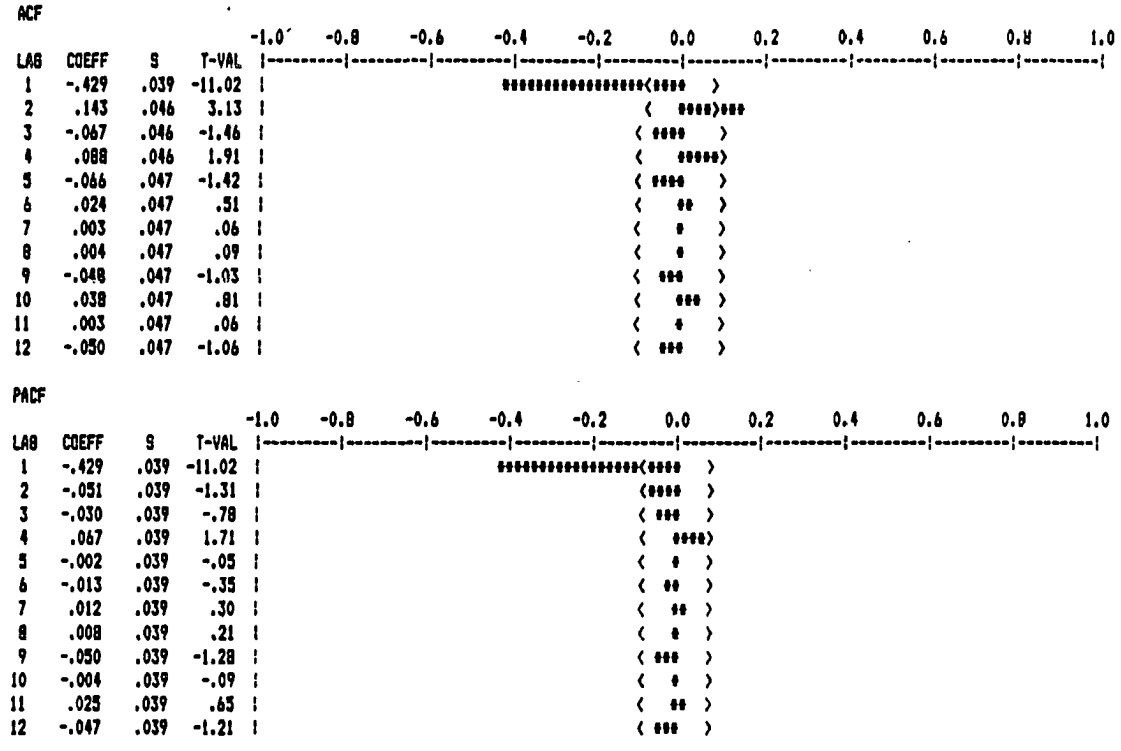

Figure 2-11. ACF and PACF of Corn Transactions

Gold

Gold prices were obtained from the Commodity Exchange in New York City for the June 1986 contract. Sampling every half-hour from January 6 through January 17, 1986, supplied 120 data points, but the analysis did not show any correlation in the ACF and suggests that gold is a random walk process when sampled every half hour.

Examining the transaction data for January 6 and 7, 1986, the first ACF lag was significant, but no correlation pattern could be discerned.

Standard and Poor's 500 Composite Index Futures

The S&P 500 Futures contract is based on the S&P 500 index for stocks. It is a new type of futures contract which does not have an underlying commodity to deliver. Transaction data for the March 1986 and June 1986 contract months were obtained from the Chicago Mercantile Exchange.

Half hour sampling of the March 1986 contract from January 2 to February 28, 1986, only provided significant correlation at lags 6 and 13.

A realization was also created by sampling the March 1986 data every 15 minutes from January 2 through February 5, 1986. The first lag of the ACF showed a negative correlation, but it was the only lag to be significant.

Portfolio Analysis

With favorable results provided by the stock indexes and insignificant correlation provided by the individual commodity markets, a small portfolio was constructed and analyzed to see if the averaging of individual markets would provide a more interesting correlation structure. The portfolio consisted of two different contracts from the same commodity market: the March 1986 and June 1986 S&P 500 Futures contracts. The portfolio's hourly value was calculated as the average of the two contracts. The portfolio was analyzed from January 2 through February 5, 1986.

The portfolio's first difference ACF exhibited positive correlation for the first two lags. This was an interesting result since there was no correlation present in either of the two contracts when analyzed individually. Further analysis of the portfolio revealed that the ACF coefficients were very similar to the cross correlation coefficients between the two contracts. A portfolio's autocorrelation would probably not be significant if the individual components did not have significant autocorrelation and were not significantly cross-correlated.

Market Open/Close Effect

Initial analysis on market prices considered the period from market closing until the next market opening to be one sample period. There was some concern about what effect this open/close period had on the analysis. To study the open/close effect, the price change between close and open was removed from the first difference data. The positive correlation that was exhibited in the DJT data still existed, but its magnitude was decreased. The ACF of the DJT #1 series with the open/close price difference removed is displayed in Figure 2-12. The first lag was reduced by approximately 20% for the three DJT realizations. The positive correlation only lasts 6 lags now without the open/close interval instead of the 7 lags that were positive

ACF

LAG	COEFF	S	T-VAL	-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
1	.140	.034	4.17	(***)****
2	.076	.034	2.23	(***)*
3	.043	.034	1.24	(***)
4	.071	.034	2.05	(***)*
5	.000	.035	.01	(*)
6	.032	.035	.92	(***)
7	-.003	.035	-.08	(*)
8	.052	.035	1.50	(***)
9	.011	.035	.32	(**)
10	-.043	.035	-1.23	(***)
11	.021	.035	.61	(**)
12	.000	.035	.00	(*)

PACF

LAG	COEFF	S	T-VAL	-1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
1	.140	.034	4.17	(***)****
2	.058	.034	1.72	(***)
3	.025	.034	.75	(**)
4	.059	.034	1.76	(***)
5	-.021	.034	-.63	(**)
6	.026	.034	.78	(**)
7	-.013	.034	-.39	(**)
8	.048	.034	1.44	(***)
9	-.001	.034	-.03	(*)
10	-.055	.034	-1.63	(***)
11	.034	.034	1.00	(***)
12	-.009	.034	-.27	(*)

Figure 2-12. ACF of DJT #1 with Open/ Close Price Difference Removed

before. This is probably related to the fact that there are now only 6 price differences each day instead of seven.

Random Number Generator

There was also a concern that the positive correlation patterns found in the DJT realizations could have been produced by some unknown idiosyncrasy in the computer program used to analyze the data. Thus, a random number generator was used to produce 1035 samples from a normal distribution with zero mean and a variance identical to DJT #1. These data were then analyzed, and there were no significantly

correlated lags in either the ACF or PACF.

Next, a randomly generated price index was created by averaging four individual random sequences similar to the one used above. Lag 4 of the ACF was the only lag outside the 95% confidence interval for statistical significance. There was no exponential or cosine pattern present in the ACF. This check then supports the theory that non-trivial correlation structure does, in fact, exist in certain market first differences.

Summary

The market analysis section has described the analysis procedures and algorithms, suggested models for realizing the process, tested data samples from various stock indexes and commodities, and examined results from variations in the analysis. The analysis procedures included the autocorrelation function (ACF) and partial autocorrelation function (PACF) which are used to calculate correlation in the data. Statistical significance of the correlation can be measured with the T-ratio test.

Suggested models include the continuous Gauss-Markov and damped cosine models and the discrete ARIMA models. The conversion of the Gauss-Markov and damped cosine models to differential equations and then to discrete state-space format is explained. The conversion of the ARIMA models to state space form was also explained.

Various stock indexes were examined for correlation. The Dow Jones 20 Transportations Index and Standard and Poor's 40 Financials Index exhibited a statistically significant correlation pattern when sampled hourly. The indexes were also sampled with larger time periods, but no significant autocorrelation was detected. The Dow Jones 30 Industrials Index was tested, but no significant correlation was found even when the sampling was done hourly.

Corn, soybeans, U.S. Treasury bonds, gold, and Standard and Poor's Composite 500 Index Futures were the individual commodity markets tested. Corn has a correlation pattern present when sampled each minute and by transaction. Treasury bonds, S&P 500 Index Futures, and gold produced significant lags, but no correlation pattern amenable to modelling was exhibited. The soybean market did not show a correlation pattern for any of the sample periods tested.

Additional results were also presented for the portfolio, open/close, and random number generator analyses. The portfolio analysis investigated the effect of using indexes instead of individual markets. It suggests that any cross correlation present between the portfolio components is a major contributor to the correlation shown by the portfolio. The open/close analysis studied the effects of including the open/close difference in the first difference calculations. The open/close difference does enhance the

correlation present in the market price data. The correlation pattern for the Dow Jones Transportation Index was still significant even with the open/close interval removed. The random number generator study examined the possibility that the correlation structure shown by the Dow Jones Transportation and Standard and Poor's Financial Indexes were induced by the analysis procedures. A similar correlation structure was not present when the random data was analyzed. Therefore, the correlation shown is actually present in the respective index realizations.

KALMAN FILTER

Introduction

In 1960, R. E. Kalman introduced a recursive algorithm to solve the linear filtering and prediction problem using a state-space approach [4]. The Kalman filter is a linear, discrete-time system which provides a recursive solution to a set of difference equations. The recursive nature of the Kalman filter requires only the previous values of the state vector to be retained to produce future estimates. This recursive algorithm makes the Kalman filter useful for real-time applications. The state space format makes it easy to implement the Kalman filter on a digital computer.

The Kalman filter provides the optimum estimate in a least squares sense of a random process which is being sampled with noisy measurements. The Kalman filter can be used to "filter" the best estimate or it can be used to forecast future values of the random process.

The Kalman filter models a process as the output of white noise passing through a linear system. The states are selected such that the filter output is formed from the linear combination of the states.

A Kalman filter can also be used to model non-stationary processes if a linear differential equation relating the process to white noise can be determined. If the model parameters are time-varying, an adaptive Kalman filter can

often be used to estimate the non-stationary process.

This chapter on the Kalman filter will present: 1) the formulation of the Kalman filter algorithm, 2) the conversion of models discussed in the previous chapter into Kalman filter format, 3) an adaptive Kalman filter algorithm for estimating non-stationary processes, 4) conversion of the ARIMA (1,1,1) model into the adaptive Kalman filter, and 5) the results from using the Kalman filter to forecast market prices for the Dow Jones Transportation Index.

The Kalman filter and adaptive Kalman filter algorithms are presented to provide the reader with a basic understanding of these Kalman filters. Readers interested in a more in depth discussion of the Kalman filter should refer to the text by Brown [4]. The text by Haykin [18] provides a reference for adaptive filtering in general, while the article by Sastri [30] provided the specific adaptive Kalman filter algorithm used in this project.

Kalman Filter Algorithm

The Kalman filter is based on a discrete state space approach where the random process is modeled by a state equation (3-1a) and a measurement equation (3-1b).

$$\underline{x}_{k+1} = \Phi_k \underline{x}_k + \underline{w}_k \qquad (3\text{-}1a)$$

$$z_k = \underline{H}_k \underline{x}_k + v_k \qquad (3\text{-}1b)$$

For a process having a single noisy output and modeled using

n internal states and m white noise inputs, $\underline{x}$ is the n-dimensional state vector, $\underline{w}$ is the m-dimensional white noise input vector, z is the noisy output measurement, and v is the additive measurement noise. For the single output system, z and v are both scalars. The other parameters in the state description are the state transition matrix, Φ, and the connection vector, $\underline{H}$. The (nxn) state transition matrix describes the change in the states from t_k to t_{k+1} when there are no driving functions, i.e., $\underline{w} = \underline{0}$. The n-dimensional connection vector describes the linear combination of states which comprise the output.

The process and measurement noise parameters, $\underline{w}$ and v, respectively, are uncorrelated white Gaussian sequences with zero mean and variances (covariances) defined by:

$$E\,[w_i * \underline{w}_k^T] = \begin{cases} Q_k & i=k \\ 0 & i \neq k \end{cases} \qquad (3\text{-}2)$$

$$E\,[v_i * v_k] = \begin{cases} R_k & i=k \\ 0 & i \neq k \end{cases} \qquad (3\text{-}3)$$

$$E\,[w_i * v_k] = \underline{0} \quad \text{for all i and k} \qquad (3\text{-}4)$$

The values of Q and R are calculated prior to execution of the Kalman filter.

Each iteration of the Kalman filter is started with an a priori estimate, $\hat{\underline{x}}^-_k$, which is the expected value of the state just before assimilating the measurement. The estimation error, $\underline{e}^-_k$, between the actual state, $\underline{x}_k$,

and the a priori state estimate, $\hat{\underline{x}}^-_k$ is defined by (3-5).

$$\underline{e}^-_k = \underline{x}_k - \hat{\underline{x}}^-_k \tag{3-5}$$

The estimation error is assumed to have zero mean and a covariance matrix, P^-_k, defined as

$$P^-_k = E[\underline{e}^-_k \, \underline{e}^{-T}_k] = E[(\underline{x}_k - \hat{\underline{x}}^-_k)(\underline{x}_k - \hat{\underline{x}}^-_k)^T] \tag{3-6}$$

The P^-_k matrix describes the confidence level of the a priori state estimate accuracy.

After the current measurement, z_k, the a priori state estimate is updated to incorporate the measurement data. The a posteriori estimate, $\hat{\underline{x}}_k$, is defined by the following update equation (3-7).

$$\hat{\underline{x}}_k = \hat{\underline{x}}^-_k + \underline{K}_k(z_k - \underline{H}_k\hat{\underline{x}}^-_k) \tag{3-7}$$

where $\underline{K}_k$ is the Kalman gain vector at time, t_k. The n-dimensional Kalman gain vector contains the weighting factors used to combine the new measurement with the a priori estimate to achieve an optimal a posteriori estimate. An optimum estimate minimizes the mean-square error of the updated estimate. The Kalman gain vector which produces an optimal estimate takes into account the confidence in the a priori estimate, P^-_k, and the reliability of the measurement, R_k. The Kalman gain is given by (3-8).

$$\underline{K}_k = P^-_k\underline{H}_k^T(\underline{H}_kP^-_k\underline{H}_k^T + R_k)^{-1} \qquad (3\text{-}8)$$

With a scalar measurement, the inversion in the Kalman gain is just a scalar inversion.

The error covariance matrix for the a posteriori state estimate is calculated from

$$P_k = (\underline{I} - \underline{K}_k\underline{H}_k)P^-_k \qquad (3\text{-}9)$$

where $\underline{I}$ is an (n x n) identity matrix.

At this point, an updated state estimate and its error covariance matrix have been calculated for the measurement at step k. To prepare for the next iteration of the Kalman filter, an a priori state estimate, $\hat{\underline{x}}^-_{k+1}$, and an a priori error covariance matrix, P^-_{k+1}, must be projected ahead from their a posteriori estimates. $\hat{\underline{x}}^-_{k+1}$ for the next measurement can be estimated by taking the expected value of the state equation (3-1a). Since the expected value of $\underline{w}_k$ is zero, the a priori estimate becomes

$$\hat{\underline{x}}^-_{k+1} = \Phi_k\hat{\underline{x}}_k \ . \qquad (3\text{-}10)$$

The a priori error covariance matrix is projected ahead by

$$P^-_{k+1} = \Phi_kP_k\Phi_k^T + Q_k. \qquad (3\text{-}11)$$

The recursive Kalman filter algorithm consists of the Kalman gain equation (3-8), state estimate (3-7) and error

covariance (3-9) update equations, and state estimate (3-10) and error covariance (3-11) projection equations. Initially, the Kalman filter must be provided with an estimate of the state vector, $\hat{\underline{x}}_0^-$, and its error covariance matrix, P_0^-. A block diagram of the Kalman filter algorithm is shown in Figure 3-1.

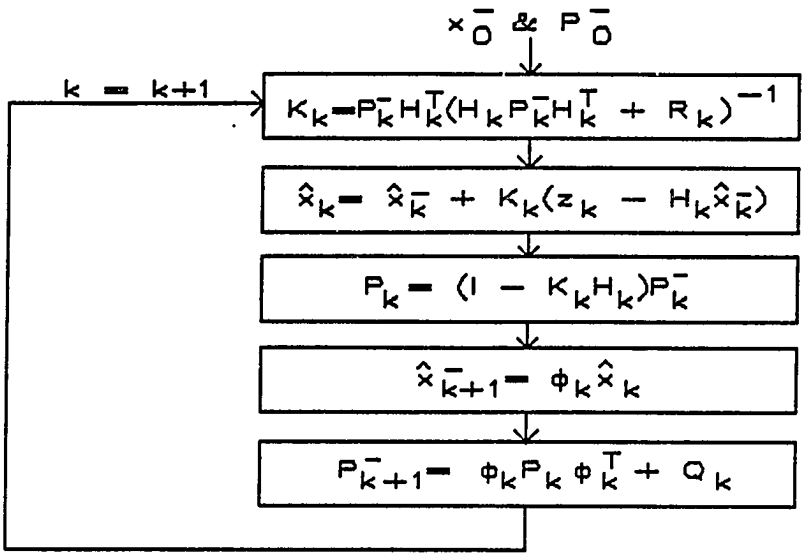

Figure 3-1. Block diagram of the Kalman filter algorithm

The Kalman filter can also provide multiple step ahead forecasts. The N-step ahead forecast equation is

$$\hat{\underline{x}}^-_{k+N} = \Phi_{k+N,k} \hat{\underline{x}}_k \qquad (3\text{-}12)$$

where $\Phi_{k+N,k}$ is the N-step ahead transition matrix. This forecast equation is kept separate from the recursive Kalman filter algorithm.

Kalman Filter Models

To use the Gauss-Markov, damped cosine, or ARIMA models for forecasting market prices, the Kalman filter parameters must be determined for each model. The state transition matrix, Φ, and the connection matrix, $\underline{H}$, can be determined directly from the discrete state equations (3-1). The measurement noise variance, R, is determined from the measurement data. The process noise covariance matrix, Q, is calculated from the state equations. Initial estimates of the state vector, $\hat{\underline{x}}^-$, and error covariance matrix, P^-, are derived from any prior knowledge of the process being modeled. If this knowledge is not available, the states are normally initialized to zero and the error covariance matrix is started with relatively large values to signify the uncertainty of the state estimate. Numerical calculations presented in this section are derived from the DJT #1 realization.

Measurement noise for a given market will be assumed to be independent of which model is being used. For market prices, the only measurement error will be the round-off error occurring from the quantitization of the price. Stock prices report their value in eighth of a dollar increments. If a stock price is assumed to be uniformly distributed over its quantitization interval, then its noise variance is defined as

$$R = \int_{-A/2}^{A/2} (u^2/A)\, du = A^2/12 \qquad (3\text{-}13)$$

where A is the quantitization interval. For an individual stock price, A = \$0.125 and R = 0.0013.

The process noise matrix, Q, is calculated from (3-14).

$$Q = E[\ \underline{W}_k * \underline{W}_k^T\ ] \qquad (3\text{-}14)$$

For the Gauss-Markov and damped cosine models, the white noise vector, $\underline{W}_k$, is defined as

$$\underline{W}_k = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}-u) * G * \underline{w}(u)\, du \qquad (3\text{-}15)$$

where $G*\underline{w}(t)$ are the driving functions for the continuous state equations. Depending on G, $\underline{W}_k$ may be comprised of multiple white noise sources, w_i, in which case the off-diagonal terms of Q will be non-zero. For the ARIMA models, the elements of $\underline{W}_k$ are all scaled versions of the single white noise source. $\underline{W}_k$ for the ARIMA (1,1,1) model is shown in (3-16).

$$\underline{W}k = \begin{bmatrix} w_k \\ -\theta_1 w_k \end{bmatrix} \qquad (3\text{-}16)$$

Q is then dependent on the moving average (MA) terms, θ_i, in the ARIMA model.

The starting value of the state vector, $\hat{\underline{x}}_0^-$, can

either be the last known value for each state or can be the expected value of the respective state equation. For the market price analysis, the state estimates were initialized at their nominal average values as determined by DJT #1 data.

The initial value for the error covariance matrix, P_0^-, is based on how the initial state estimates were determined. If a state is initialized with its last known value, then its error (co)variance represents the error that could occur after one time period. If the state is initialized with its nominal average value, then the error (co)variance is calculated from the corresponding variances in the data used to obtain the averages.

The state equations listed below are from the Models section of the Market Analysis chapter. The Gauss-Markov model also includes a white noise source, W_2, to account for the large white noise term in the first difference of the DJT data. The damped cosine model is not included in this section since the DJT #1 data did not fit this model.

Random Walk

The state equations for the random walk model are

$$x_{k+1} = 1*x_k + W_k \qquad (3\text{-}17a)$$

$$y_k = 1*x_k + v_k \qquad (3\text{-}17b)$$

Note that $\Phi = 1$ and $H = 1$. Solving (3-17a) for W_k shows that the white sequence is equal to the first difference of

the price. Therefore, Q for the random walk model is equal to the variance of the first difference, or Q = 2.9353 for DJT #1. The state estimate is initialized to the mean of the DJT #1 realization, $\hat{x}_0^- = 640.75$, and the error covariance is initialized to the variance of the DJT #1 data, $P_0^- = 1810$.

Gauss-Markov

The state equations for the Gauss-Markov plus white noise model are shown in (3-18). Although, the Gauss-Markov model is a single state model, a second state is required for the discrete integration to convert the first difference model to an actual market price model.

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1.0 & e^{-\beta T} \\ 0.0 & e^{-\beta T} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} W_1(k) + W_2(k+1) \\ W_1(k) \end{bmatrix} \quad (3\text{-}18a)$$

$$y(k) = [1 \quad 0] \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + v(k) \quad (3\text{-}18b)$$

The state transition matrix for the Gauss-Markov model is

$$\Phi = \begin{bmatrix} 1 & e^{-\beta T} \\ 0 & e^{-\beta T} \end{bmatrix} = \begin{bmatrix} 1.0 & 0.7239 \\ 0.0 & 0.7239 \end{bmatrix} \quad (3\text{-}19)$$

and the connection matrix is

$$H = [\, 1 \quad 0 \,] . \quad (3\text{-}20)$$

The covariance matrix for the process noise is

$$Q = \begin{bmatrix} E[(W_1 + W_2)^2] & E[W_1^2] \\ E[W_1^2] & E[W_1^2] \end{bmatrix} = \begin{bmatrix} 2.9383 & 0.3201 \\ 0.3201 & 0.3201 \end{bmatrix}. \quad (3\text{-}21)$$

The initial state vector for DJT #1 is

$$\underline{\hat{x}}_0^- = \begin{bmatrix} 640.75 \\ 0.00 \end{bmatrix} \quad (3\text{-}22)$$

and the initial state covariance matrix is

$$P_0^- = \begin{bmatrix} 1810.0 & 2.4359 \\ 2.4359 & 0.6726 \end{bmatrix} . \quad (3\text{-}23)$$

ARIMA model

The first differences of the three DJT data sets were realized with an ARIMA (1,0,1) model. When the actual price is to be forecast instead of the first difference, a discrete integration must be included in the model. Therefore, the ARIMA (1,0,1) model is converted to an ARIMA (1,1,1) model. The ARIMA (1,1,1) model has the forecast equation shown in (3-24).

$$Z_k = (1+\phi_1)Z_{k-1} - \phi_1 Z_{k-2} - \theta_1 w_{k-1} + w_k \quad (3\text{-}24)$$

Using the conversion procedure in the Market Analysis chapter, the ARIMA (1,1,1) state equations are:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} (1+\phi_1) & 1 \\ -\phi_1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} W(k) \\ -\theta_1 W(k) \end{bmatrix} \qquad (3\text{-}25a)$$

$$Z(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + v(k) \quad . \qquad (3\text{-}25b)$$

An additional state is also required for the ARIMA (1,1,1) model to provide the discrete integration. From the state equations, the state transition matrix is

$$\Phi = \begin{bmatrix} 1+\phi_1 & 1 \\ -\phi_1 & 0 \end{bmatrix} = \begin{bmatrix} 1.7222 & 1 \\ -0.7222 & 0 \end{bmatrix} \qquad (3\text{-}26)$$

and the connection matrix is

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix}. \qquad (3\text{-}27)$$

The process noise covariance matrix, initial state estimate, and initial error covariance matrix for the ARIMA (1,1,1) are shown in (3-28), (3-29), and (3-30), respectively.

$$Q = \begin{bmatrix} E[W_k^2] & -\theta_1 E[W_k^2] \\ -\theta_1 E[W_k^2] & \theta_1^2 E[W_k^2] \end{bmatrix} = \begin{bmatrix} 2.82 & -1.63 \\ -1.63 & 0.95 \end{bmatrix} \qquad (3\text{-}28)$$

$$\hat{\underline{x}}_0^- = \begin{bmatrix} 640.75 \\ -462.75 \end{bmatrix} \qquad (3\text{-}29)$$

$$P_0^- = \begin{bmatrix} 1810.0 & -1307.2 \\ -1307.2 & 945.0 \end{bmatrix} \qquad (3\text{-}30)$$

Adaptive Kalman Filter Algorithm

An adaptive Kalman filter provides a method of realizing a process for which the system model is not well-defined. The filter allows the model parameters to vary such that the model adapts to the incoming data. This parameter variation allows the adaptive filter to model a system in which the process parameters are either not known exactly or may be time-varying [18]. The adaptive filter modifies the model parameters after each iteration to incorporate any information provided by the new measurement. If an unknown parameter is a random constant, the adaptive filter uses the measurement data to drive the model parameter towards the process parameter value. The steady state value of the model parameter will reflect the actual process parameter. When a process parameter is assumed to be time-varying, the filter allows the model parameter to track the process parameter. The model parameter varies slowly, never reaching a steady state value. This variability allows the adaptive filter to model non-stationary processes. The adaptive Kalman filter used in this study was based on ARIMA modelling of the process.

The rate at which the adaptive Kalman filter tracks the process is regulated by an adaptive control. The adaptive control is also responsible for guaranteeing that the filter remains adaptive. In the basic Kalman filter, the state

estimates become more accurate as the filter operates and, therefore, the error covariance is continually reduced to reflect this accuracy. This, in turn, causes the Kalman gain to place less weight on newer measurements. In the adaptive Kalman filter, the parameters may be time-varying, in which case the accuracy of the state estimates is limited. To allow the filter to remain adaptive, the adaptive control constantly increases the a priori error covariance matrix which reduces the confidence in the parameter estimate. This prevents the Kalman gain from becoming too small and not providing sufficient weight to new data.

For the adaptive Kalman filter, the unknown parameters become elements of the state vector. The state equation for a parameter is then dependent on whether it is assumed to be a random constant, or time-varying. If a parameter is assumed to be a random constant, then its state equation is

$$x_{k+1} = x_k. \tag{3-31}$$

If the parameter is assumed to be time-varying, then the state is modelled as a random walk process with the following state equation:

$$x_{k+1} = x_k + w_k. \tag{3-32}$$

The state and measurement equations for the adaptive Kalman Filter are:

$$\underline{x}_{k+1} = \Phi\underline{x}_k + G\underline{w}_k \qquad (3\text{-}33a)$$

$$Z_k = \underline{H}_k * \underline{x}_k + v_k \qquad (3\text{-}33b)$$

where the state transition matrix, Φ, is an nxn identity matrix and G is a known (nxm) matrix connecting the white sequences to the state vector.

The measurement equation (3-33b) now defines the relationship between the measurement, Z_k, and the model parameters, $\underline{x}_k$. If any of the parameters are autoregressive (AR) terms, then the measurement connection matrix, $\underline{H}$, contains data from previous measurements. For example, if the x_1 term defined the first AR parameter, ϕ_1, then the $\underline{H}_1$ term would be Z_{k-1}. Elements of $\underline{H}$ which correspond to AR parameters must be updated after each measurement.

For the adaptive Kalman filter suggested by Sastri [30], the measurement noise, v, and process noise, $\underline{W}$, may be correlated as shown in (3-34).

$$E[v * \underline{W}] = \underline{C} \qquad (3\text{-}34)$$

where $\underline{C}$ is an m-dimensional correlation vector. This correlation vector is necessary for the conversion of the ARIMA model to adaptive Kalman filter format.

The adaptive Kalman filter algorithm utilized for this project is very similar to the Kalman filter described

previously. The recursive loop contains a Kalman gain equation, update equations for the state estimate and its error covariance matrix, and project ahead equations for the state estimate and error covariance matrix. The individual equations are changed slightly to account for any correlation between the process and measurement noise.

The adaptive Kalman filter is initialized with an a priori state estimate, $\hat{\underline{x}}^-$, and an a priori error covariance matrix, P^-, before every iteration. If the state vector contains AR terms, then an updated measurement connection matrix, $\underline{H}$, is also provided at the start of each iteration.

The first step in the adaptive Kalman filter is to calculate the Kalman gain vector. The Kalman gain vector determines how the estimate error will be combined with the a priori state estimate, $\hat{\underline{x}}^-$, to arrive at an updated state estimate, $\hat{\underline{x}}$. The Kalman gain vector for the adaptive algorithm is shown in (3-35).

$$\underline{K} = (P^-\underline{H}^T + G\underline{C})[\underline{H}P^-\underline{H}^T + \underline{H}G\underline{C} + (\underline{H}G\underline{C})^T + R]^{-1} \qquad (3\text{-}35)$$

Since the Kalman gain is a function of $\underline{H}$, the Kalman gain vector will be also be dependent on previous data if the state vector contains AR parameters.

After the measurement for step k is collected, the a posteriori state estimate and its error covariance matrix are calculated. The state estimate is updated as shown in

(3-36).

$$\hat{\underline{x}}_k = \hat{\underline{x}}^-_k + \underline{K}_k(z_k - \underline{H}_k\hat{\underline{x}}^-_k) \qquad (3\text{-}36)$$

The update equation for the a posteriori error covariance matrix is shown in (3-37).

$$P_k = P^-_k - \underline{K}_k(\underline{H}_k P^-_k + \underline{C}^T G^T) \qquad (3\text{-}37)$$

When the $\underline{H}$ matrix is data dependent, the P matrix also becomes related to past data since it is a function of $\underline{H}$. Therefore, P is a conditional error covariance matrix, conditioned on the input data [18].

To prepare the adaptive Kalman filter for the next iteration, a priori estimates must be determined for the state vector and its error covariance matrix. If the state vector contains AR terms, then the $\underline{H}$ matrix is also updated. The equations for projecting the state estimate and its error covariance matrix ahead to the k+1 step are shown in (3-38) and (3-39), respectively.

$$\hat{\underline{x}}^-_{k+1} = \Phi\hat{\underline{x}}_k \qquad (3\text{-}38)$$

$$P^-_{k+1} = B(\Phi P_k \Phi^T + GQG^T), \qquad B \geq 1 \qquad (3\text{-}39)$$

The error covariance matrix project ahead equation (3-39) contains the adaptive control, B, which controls the speed at which the filter adapts to the process. Caution must be used not to set B too large as this will cause the filter to

become unstable. The $\underline{H}$ matrix is updated by replacing the elements corresponding to AR parameters with the latest measurement data.

The algorithm for the adaptive Kalman filter is listed in Figure 3-2.

0. Initialize with $\underline{H}_0$, $\hat{\underline{x}}^-_0$ and P^-_0.
1. $\underline{K}_k = (P^-_k\underline{H}^T_k + G\underline{C})[\underline{H}_kP^-_k\underline{H}_k^T + \underline{H}_kG\underline{C} + (\underline{H}_kG\underline{C})^T + R]^{-1}$
2. $\hat{\underline{x}}_k = \hat{\underline{x}}^-_k + \underline{K}_k(z_k - \underline{H}_k\hat{\underline{x}}^-_k)$
3. $P_k = P^-_k - \underline{K}_k(\underline{H}_kP^-_k + \underline{C}^TG^T)$
4. $\hat{\underline{x}}^-_{k+1} = \Phi\hat{\underline{x}}_k$
5. $P^-_{k+1} = B(\Phi P_k\Phi^T + GQG^T), \quad B \geq 1$
6. Update $\underline{H}_{k+1}$ with lastest measurement data, if required
7. k = k+1
8. Go to 1.

Figure 3-2. Adaptive Kalman filter algorithm [30]

Adaptive Kalman Filter Models

The adaptive Kalman filter used in this project was created from an ARIMA (1,1,1) model. This model requires 2 states: one state, x_1, is used for the AR term, ϕ_1, and the second state, x_2, tracks the accumulated white noise inputs. Both states use the random walk model (3-32) to allow parameter variation.

The state description for the ARIMA (1,1,1) adaptive Kalman filter is shown in (3-40).

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} W_1(k) \\ W_2(k) \end{bmatrix} \qquad (3\text{-}40a)$$

$$Z_k = [Z_{k-1} \quad 1] \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + v_k \qquad (3\text{-}40b)$$

The state transition matrix, Φ, and the process noise connection matrix, G, are (2x2) identity matrices. The first element of the measurement connection matrix, $\underline{H}$, is always the previous measurement as shown in (3-41).

$$\underline{H} = [\, Z_{k-1} \quad 1 \,] \qquad (3\text{-}41)$$

This corresponds to the first element of the state vector, x_1, which represents the AR term, ϕ_1. The $\underline{H}$ matrix must be updated before each iteration of the Kalman filter.

The expressions for W_2 and v are obtained from the second state equation and the measurement equation. Solving the measurement equation (3-40b) for x_2 and substituting it into the state equation for x_2 (3-40a) provides an equation similar to the ARIMA (1,1,1) forecast equation (3-24). This new equation is shown in (3-42).

$$Z_k = (1+x_1)Z_{k-1} - x_1 Z_{k-2} - v_{k-1} + v_k + w_{2k} \qquad (3\text{-}42)$$

Comparing (3-42) and the ARIMA (1,1,1) forecast equation, the following relationships are noted.

$$v_k = \theta_1 w'_k \tag{3-43}$$

$$w_{2k} = (1-\theta_1) w'_k \tag{3-44}$$

where w' is substituted for w in (3-24). Using these equations, the measurement noise variance, R, the process noise variance, Q_{22}, and the covariance between W_2 and v can be calculated as shown in (3-45), (3-46), and (3-47), respectively.

$$R = \theta_1^2 \, E[w'^2] = 0.9460 \tag{3-45}$$

$$Q_{22} = (1-\theta_1)^2 \, E[w'^2] = 0.4493 \tag{3-46}$$

$$C_2 = \theta_1(1-\theta_1) \, E[w'^2] = 0.6873 \tag{3-47}$$

The state x_1 is also assumed to vary, but very slowly. The process noise variance for x_1 reflects this by using a small value for Q_{11}. The change in x_1 is also assumed to be independent of the change in x_2 and, thus, also independent of v. For this reason, the terms Q_{12}, Q_{21}, and C_1 are all set equal to zero.

The initial estimate for the state vector is shown in (3-48). x_1 is initialized with the optimum ϕ_1 value determined for the DJT #1 data. The x_2 estimate was

$$\hat{\underline{x}}^-_0 = \begin{bmatrix} 0.7222 \\ 178.00 \end{bmatrix} \tag{3-48}$$

calculated so the expected value of the measurement equation (3-40b) would yield the DJT #1 mean, or 640.75.

The error covariance matrix was initialized to

$$P^{-}_{0} = \begin{bmatrix} 0.0078 & 0.0000 \\ 0.0000 & 0.9460 \end{bmatrix} \qquad (3\text{-}49)$$

The P_{11} term is the variance of the ϕ_1 estimate as determined by the ARIMA estimation procedure. The P_{22} term is calculated from the measurement equation (3-40b).

The adaptive control value, B = 1.0001, was selected as to produce the minimum MSE for the DJT #1 data.

Kalman Filter Results

Results for the Kalman filter and adaptive Kalman Filter are based on the three DJT realizations. The Kalman and adaptive filters are used to forecast the next hourly price (1-step ahead) using the various models. The MSE of the forecasts are calculated to determine model usefulness.

Tests on the DJT #1 data used the last 536 out of 1036 samples to calculate the MSE. For the DJT #2 and #3 realizations, the MSE was calculated on the last 796 out of 896 data points. The first 100 data samples in DJT #2 and #3 were used to initialize the filter. Since model parameters were optimized for the DJT #1, the DJT #2 and #3 data sets are used to determine if a model also applies to other time frames in the same process.

The trivial random walk model is used as a baseline measurement for the tests. The random walk model forecasts that the next price will be same as the last price. If there is correlation present in the process, then the non-trivial models should be able to outperform the random walk model in terms of MSE. The MSEs for one-step ahead forecasts are shown in Table 3-1. The percentage change from the random walk model is shown in parentheses.

Table 3-1. MSE for 1-step ahead foreasts

Model	DJT #1	DJT #2	DJT #3
Random Walk	3.4016	4.1730	3.5781
Gauss-Markov	3.2600 (-4.2)	4.1389 (-0.8)	3.5337 (-0.7)
ARIMA (1,1,1)	3.2582 (-4.2)	4.1173 (-1.3)	3.5138 (-1.8)
Adaptive filter ARIMA(1,1,1)	3.2579 (-4.2)	4.1097 (-1.5)	3.5069 (-2.0)

Each suggested model had a lower MSE than the random walk model. This was true for all three data sets. The maximum MSE improvement ranged from -4.2% for DJT #1 to -1.5% for DJT #2. These small percentages highlight the fact that the DJT's first difference is very nearly pure white noise.

The best results were provided by the adaptive Kalman filter, although the improvement was only 0.2% better than

the basic Kalman filter for the DJT #2 and #3 realizations. Between the two basic Kalman filter models, the ARIMA (1,1,1) model provided a lower MSE, especially for the DJT #2 and #3 realizations.

Summary

The Kalman filter is a discrete state-space solution to linear filtering and prediction problems. Its recursive algorithm produces an optimal estimate of a stochastic process corrupted by additive white noise. The Kalman filter parameters are formulated from a state description of the random process to be modeled.

This project used the Kalman filter to forecast future values of the Dow Jones 20 Transportations Index (DJT). The random walk, Gauss-Markov, and ARIMA (1,1,1) models discussed in the Market Analysis chapter are used by the Kalman filter to produce these forecasts.

An adaptive Kalman filter was introduced as a tool for forecasting prices when the random process model has unknown variable parameters. The adaptive filter continually estimates the model parameters from the most recent data. Development of the adaptive Kalman filter for the ARIMA (1,1,1) and DJT data were examined.

The three DJT realizations were used to test the forecast accuracy of the Kalman filter models. Model comparisons were based on minimum MSE using the random walk

model as a baseline. The Gauss-Markov, ARIMA (1,1,1), and adaptive ARIMA (1,1,1) Kalman filters provided a smaller MSE that the random walk model for all three DJT realizations. The adaptive filter produced the best results in all three cases.

BUY AND SELL STRATEGIES

Introduction

The ability to produce optimum forecasts of future market prices is not the final solution to the problem of profiting in the markets. There must be a method of utilizing the forecasts to achieve maximum profits. Since the Kalman Filter forecasts are based on minimizing MSE, a buy/sell strategy which increases profits as the MSE decreases would be desirable.

Forecast errors generated by the Kalman filter are assumed to be normally distributed. The probability that the actual price change will be positive (4-1) can be approximated using the forecast and the Normal distribution.

$$P(\Delta\$\geq 0) = 1 - P(\Delta\$<0) \approx 1 - \int_{-\infty}^{0} N(\hat{\Delta}\$, MSE)\, d(\Delta\$) \qquad (4\text{-}1)$$

where $N(\hat{\Delta}\$, MSE)$ is a normal distribution with mean of $\hat{\Delta}\$$ and a variance equal to the MSE of the forecasts. A diagram showing the probabilistic relationship between the actual price change and the forecast price range is shown in Figure 4-1. This relationship is exploited by the buy and sell strategies.

Buy and sell strategies tested were based on using the market either as 1) a speculator or 2) a consumer. The speculator scenario tries to maximize profits by buying or

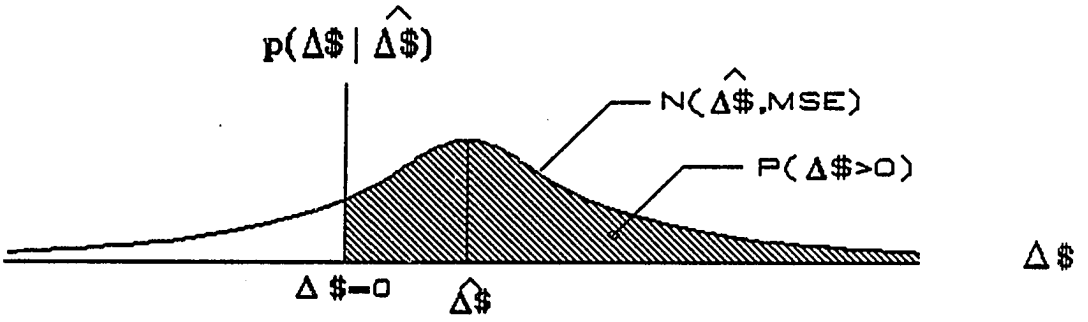

Figure 4-1. Relationship between actual and forecast price change

selling stock each hour depending on the Kalman filter forecast for the next hour. The consumer scenario assumes that a contract, e.g., corn, must be bought once a day, or once every two days, or once a week. The consumer waits to make a purchase until the Kalman filter forecasts the price to increase.

The buy/sell strategies are tested on the data from the three Dow Jones Transportation (DJT) realizations. The adaptive Kalman filter presented in the last chapter provides the price forecasts on which the buy/sell decisions are based.

Speculator Strategy

The speculator strategy used the Kalman filter forecast to determine whether to buy or sell stock. Testing for the speculator strategy included step, hysteresis, linear, and quadratic strategies. Each strategy was tested with and

without commissions being charged.

Trading for this scenario is performed according to the following trading rules. No transactions are allowed between the market closing and the following opening, i.e., overnight. The maximum amount of stock purchased each transaction is limited by the current assets (cash plus stock value) which are re-evaluated after each measurement, leveraging is not allowed. Stock transactions can be made in odd lots and may include fractional shares. There is no slippage, i.e., the transaction price is the same as the last measurement. The analysis begins with $10,000 in assets and half of that amount is invested in stock. Profit comparisions are made over the last 536 data points of DJT #1 and the last 796 data points of DJT #2 and #3. (These are the same data used for the MSE comparison in the last chapter.)

A buy-and-hold strategy was used as a baseline for profit comparison. The buy-and-hold strategy invests all the assets at the beginning of the comparison period and lets the stock accumulate over the period. No other buying or selling takes place during the test. The buy-and-hold strategy reflects any price change in the stock. Comparing the buy and sell strategies to the buy-and-hold strategy shows whether the buying and selling increased profits over what the stock would have done on its own.

A second baseline used for profit comparison was slope-projection. The premise behind this strategy is that the price difference between the last price and the next will continue along the same slope as the last price difference, i.e., the price differences will be equal. If the last price difference was positive, then the assets are converted to shares. If the last difference was negative, then all the shares are converted to cash. No transactions occur if the price difference is in a dead-zone around zero. The width of the dead-zone was adjusted to maximize profits for the DJT #1 data.

Each speculator strategy is controlled by a maximum investment factor, a minimum investment factor, and a bias value. The linear and quadratic strategies also have a clip value. These values are optimized for each strategy to provide the maximum profit for the DJT #1 data set. The maximum and minimum investment factors limit the percentage of assets that can be invested at each transaction. The maximum investment factor ranges between 50 and 100% of the assets and the minimum investment factor ranges between 0 and 50% of the assets. For step, linear, and quadratic strategies, the bias value provides an interval where the investment factor is set to an average investment, or 50% of the assets. For the hysteresis strategy, the bias value specifies the hysteresis window size. The bias value is

effective when the price change forecast is small, since the probabilities of profit and loss are almost equal. The clip value determines where the buy and sell strategy reaches the investment limits. A forecast value exceeding the clip value will invest shares at the minimum or maximum percentage.

The step function strategy invests at either the minimum or maximum investment factor when the forecast price is outside the bias values. If the forecast is within the bias values, then the investment is set at 50% of the current assets. The step function is shown in Figure 4-2.

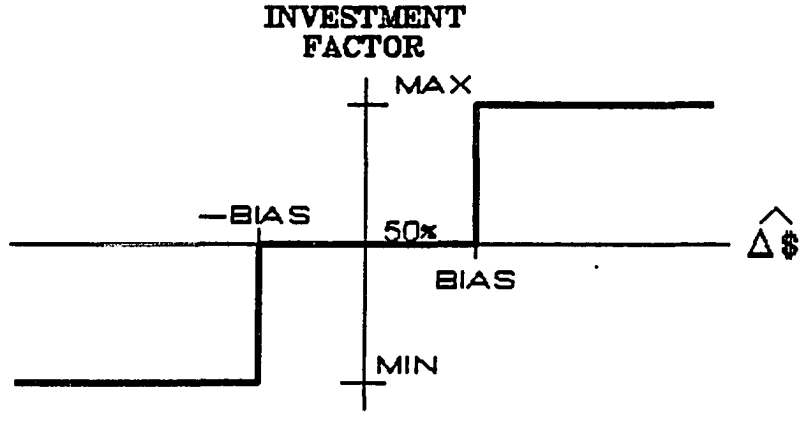

Figure 4-2. Step function speculator strategy

The hysteresis strategy, shown in Figure 4-3, invests either the maximum or minimum limits of its assets on each transaction. When the forecast price change is inside the bias interval, no transaction occurs and the investment remains at the previous value.

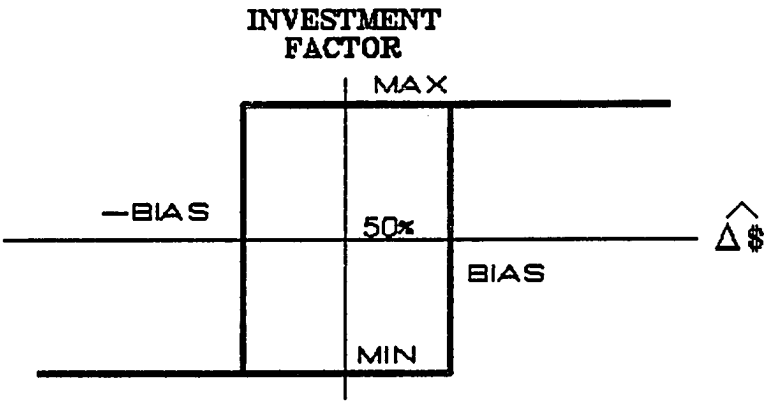

Figure 4-3. Hysteresis speculation strategy

The linear strategy sets the number of shares invested proportional to the price change forecast. The linear function reaches the investment limits when the forecast reaches the clip value. The bias value sets the investment factor to 50%. Forecast values between the bias and clip values cause the investment factor to follow a linear slope in that region. The linear strategy is shown in Figure 4-4.

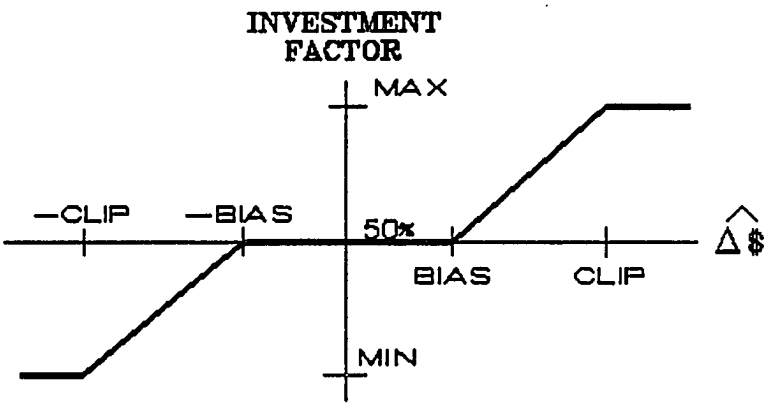

Figure 4-4. Linear speculator strategy

The quadratic speculator strategy is similar to the linear strategy. Bias and clip values are used to determine the quadratic function. Figure 4-5 displays the quadratic speculator strategy. If the forecast lies between the bias and clip values, then the percentage of assets invested in stock is determined by the quadratic curve. The closer the forecast is to the clip value, the closer the percentage is to the maximum or minimum limit.

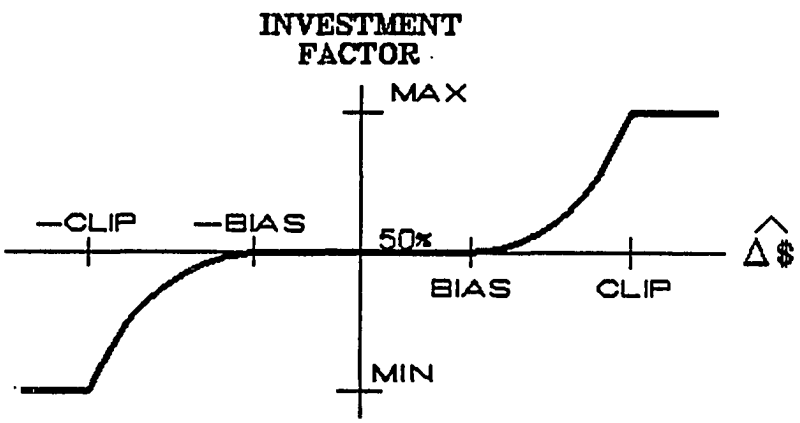

Figure 4-5. Quadratic speculator strategy

The speculator strategies were tested on the three DJT realizations with no broker commissions on any transaction. The optimum value for the bias and clip values for the DJT #1 realization are shown in Table 4-1. The minimum and maximum investment factors were 0% and 100%, respectively, for each strategy.

A running comparison of the assets using the hysteresis

Table 4-1. Optimized bias and clip values for DJT #1 with no commission

Strategy	Bias	Clip
Step	0.06	N/A
Hysteresis	0.00	N/A
Linear	0.06	0.07
Quadratic	0.05	0.06

strategy and the buy-and-hold strategy is shown in Figure 4-6 for DJT #1. The hysteresis strategy does not exhibit any dramatic decrease in assets similar to the buy-and-hold strategy.

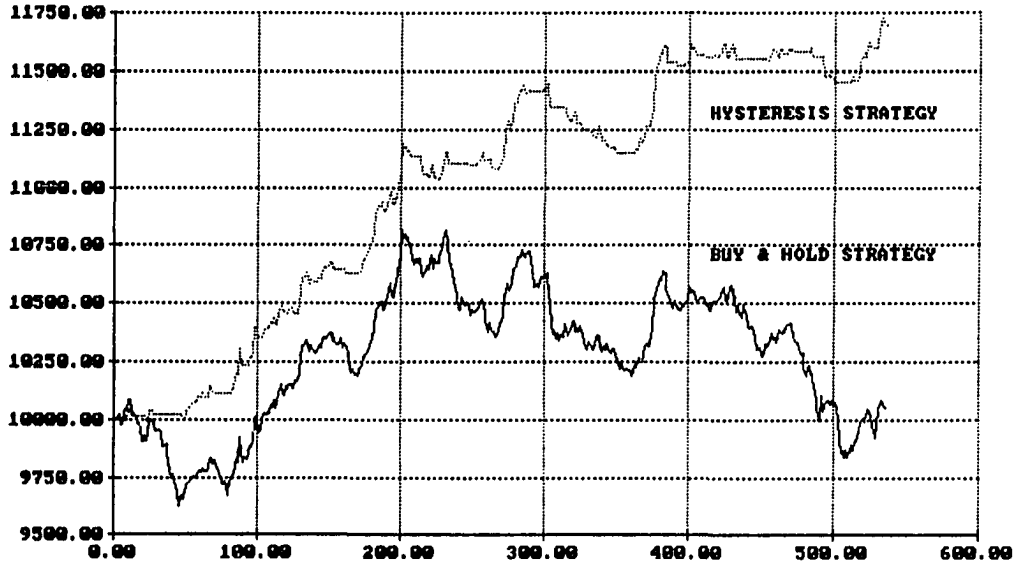

Figure 4-6. Running asset comparison between hysteresis strategy and buy-and-hold strategy for DJT #1

Table 4-2 shows the results of the speculator strategies with no commission charged. The numbers in parentheses are the percentage change from the buy-and-hold strategy.

All the speculator strategies out-performed the buy-and-hold strategy in all three tests. They also showed a profit from the starting value of $10,000. The step function strategy produced the best results for DJT #1, the hysteresis strategy the best for DJT#2, and the linear strategy provided the greatest profit for DJT #3.

Table 4-2. Final asset comparison for speculator strategies with no commission

Strategy	DJT #1	DJT #2	DJT #3
Buy & Hold	$10,054	$ 7,814	$10,613
Slope-projection	$11,847 (17.8)	$11,084 (41.8)	$10,323 (-2.7)
Step	$11,744 (16.8)	$10,818 (38.4)	$11,765 (10.9)
Hysteresis	$11,701 (16.4)	$11,291 (44.5)	$11,827 (11.4)
Linear	$11,711 (16.5)	$10,863 (39.0)	$11,891 (12.0)
Quadratic	$11,711 (16.5)	$10,902 (39.5)	$11,784 (11.0)

Slope-projection had greater profits than the Kalman filter strategies for DJT #1, but the slope-projection profits were less for DJT #2 and #3. slope-projection may

provide greater profits on a given data sample, but the Kalman filter strategies are more profitable on the average.

The speculator strategies were tested again with a commission rate of 0.25%. The commission rate was applied to the amount of cash exchanged during the transaction. The strategies were again optimized to provide the maximum profit for DJT #1. The step, linear and quadratic strategies produced the greatest profit, $10025.92, when the investment was 50% of the assets for all transactions, i.e., they reverted to a simple buy-and-hold strategy using half of the assets. The hysteresis strategy was the only one which proved profitable. The clip value for the hysteresis strategy was $0.21. Table 4-3 shows the results for the buy-and-hold, slope-projection, and hysteresis strategies with a commission rate of 0.25%. The numbers in parentheses again represent the percentage change from the respective buy-and-hold value.

Table 4-3. Asset comparison for speculator strategies with commission

Strategy	DJT #1	DJT #2	DJT #3
Buy-and-hold	$10,054	$ 7,814	$10,613
Slope-projection	$10,259 (2.0)	$ 9,560 (22.3)	$ 9,835 (-7.3)
Hysteresis	$10,484 (4.3)	$ 8,613 (10.2)	$10,046 (-5.3)

The hysteresis strategy did not provide a consistent inprovement over the results shown by the buy-and-hold strategy. In DJT #2 tests, the hysteresis strategy was more profitable than the buy-and-hold strategy, but the final assets were 14% less than the starting value. In the DJT #3 test, the hysteresis strategy did increase assets, but the final assets were less than those achieved by the buy-and-hold strategy.

Consumer Strategy

This strategy is based upon a consumer requiring continual purchasing of goods to keep his operation running. The consumer waits to make the purchase until the Kalman filter forecasts an increase in price. After the purchase is made, the consumer still uses the Kalman filter forecasts. If the forecast price drops below the purchase price before the end of the buying period, then the consumer sells the previous contract purchased and waits for the Kalman filter to predict the next price increase. The consumer buys 1 contract or, for the DJT data, one share at each purchase.

The consumer strategy, shown in Figure 4-7, includes a decision point that determines what forecast price change to use as a buy/wait threshold. If the price change forecast is below this decision point, the consumer waits to make his purchase. The decision point may be at a positive or negative value. The decision point was chosen to minimize

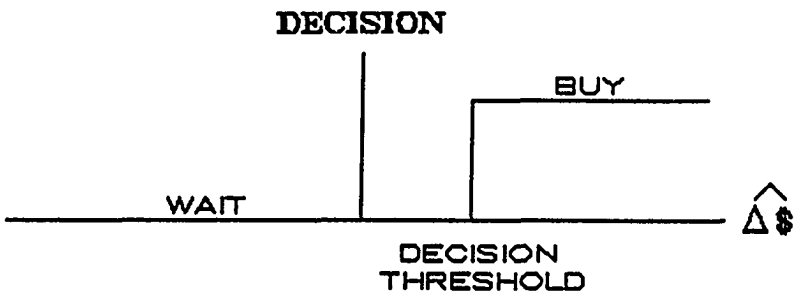

Figure 4-7. Diagram of decision process for consumer strategy

expenses for DJT #1.

The consumer strategy is compared to two baselines. The first baseline has the consumer make the purchase at the opening price on the first day of the buying period. The second baseline uses the closing price on the last day of the buying period. The comparison period was enlarged slightly in each of the DJT realizations so that the comparison would run over an integral number of days. The DJT #1 test used the last 539 data points, or 77 days. DJT #2 and #3 test used the last 798 data samples, or 114 days. There was no commission charged on transactions. The tests were conducted with one, two, three, four, and five days in the buying period. The results for the consumer strategy are shown in Table 4-4.

Kalman filter forecasting did reduce the expenses for all three DJT realizations, but the amount of reduction was never more than 0.7%.

Table 4-4. Expense comparison for consumer strategy

Strategy	DJT #1	DJT #2	DJT #3
1 Day Period			
Open	$51,772	$57,576	$65,559
Close	$51,785	$57,459	$65,585
Kalman filter	$51,697	$57,429	$65,506
2 Day Period			
Open	$26,220	$28,832	$32,747
Close	$26,230	$28,718	$32,782
Kalman filter	$26,128	$27,670	$32,699
3 Day Period			
Open	$18,141	$19,788	$22,401
Close	$18,145	$19,665	$22,433
Kalman filter	$18,047	$19,642	$22,342
4 Day Period			
Open	$13,439	$14,754	$16,642
Close	$13,443	$14,622	$16,679
Kalman filter	$13,352	$14,607	$16,587
5 Day Period			
Open	$10,748	$12,231	$13,765
Close	$10,741	$12,088	$13,806
Kalman filter	$10,678	$12,061	$13,705

Summary

This section examined the conversion of Kalman filter price forecasts to increased profits in the market place. This conversion was studied from the viewpoints of a speculator and a consumer. The speculator strategy used step, hysteresis, linear, and quadratic functions to improve profits over slope-projection and buy-and-hold strategies. The speculator strategies were more profitable than the buy-and-hold in each case and more successful than slope-projection on the average. If the speculator must pay a commission on each transaction, then then speculator strategies will not increase the profits. The speculator strategy search was not exhaustive, but it did show that the Kalman filter could be used to increase profits.

The consumer strategy tried to reduce operational expenses by forecasting when the market price was rising. The Kalman filter forecasts did not provide significant improvement over buying on the open or the close. Any savings realized were very small in comparison to the large expenses accrued.

SUMMARY

This project examined the use of a Kalman filter to forecast market prices. The first step was to determine if there was any correlation present in the market data. If no correlation was present, then the market process followed the trivial random walk model and a Kalman filter would not provide any increased forecasting accuracy. If correlation was found to be present, the market process was modelled using the continuous Gauss-Markov and damped cosine models or the discrete Box and Jenkin's ARIMA models. These models are convertible to a discrete state space format for use in a Kalman filter.

Market analysis was performed on stock indexes and individual commodity markets. Significant correlation was found in the hourly data for the Dow Jones Transportation and Standard and Poor's Financial stock indexes. These correlation patterns were not present when the indexes were sampled daily, weekly, or monthly. Corn, soybeans, U.S. Treasury bonds, gold, and the S&P 500 Futures were the commodity markets examined. These commodities did not provide any significant correlation when they were sampled every half-hour. Corn did exhibit a small amount of correlation for minute by minute sampling and for transaction sampling. However, the correlation patterns exhibited were of very short duration. U.S. Treasury bonds, S&P 500

Futures, and gold had some significant lags when transaction data was studied, but no correlation pattern was apparent. Correlation was very small in the commodity markets studied, but enough statistically significant ACF spikes were present to suggest that the commodities might not be random walk processes.

With stock indexes showing more correlation patterns than individual markets, a portfolio was constructed from two commodities to determine the effects of averaging. A significant cross correlation between the two commodities provided an autocorrelation pattern for the portfolio when there was no autocorrelation pattern for the two commodities individually.

The Gauss-Markov and ARIMA (1,1,1) models were converted to Kalman filter format to forecast the three DJT realizations. The ARIMA (1,1,1) model was also placed in an adaptive Kalman filter format. The Kalman filter forecasts were more accurate than forecasts provided by the random walk model. The best results were produced by the ARIMA (1,1,1) model in the adaptive Kalman filter.

Two buy and sell strategies were tested to see if the more accurate Kalman filter forecasts could be used to increase profits. The speculator strategy found that the Kalman filter could be used to forecast hourly prices and provide a greater profit than a buy-and-hold or

slope-projection strategy. The speculator strategy could not consistently produce a profit if commissions were charged on each transaction. A consumer strategy failed to significantly reduce expenses when the Kalman filter was used to forecast price increases.

In conclusion, it appears that sufficient correlation can be found in the stock indexes to use a Kalman filter to produce improved forecasts. A method of using the stock index forecasts to benefit in traded market(s) must be determined. If a market with significant correlation is found, the buy and sell testing has shown that the Kalman filter forecasts can be used to increase profits.

An area which should be examined further is the creation of a portfolio. An investor may be able to profit by trading a small portfolio instead of single markets. The idea of combining markets which have a significant cross-correlation may prove beneficial.

REFERENCES

1. Alexander, Sidney S. "Price Movements in Speculative Markets: Trends or Random Walk." Industrial Management Review, 2 (May 1961), 7-26.

2. Athans, Michael. "The Importance of Kalman Filtering Methods for Economic Systems." Annals of Economic and Social Measurement, 3 (1974), 49-64.

3. Bear, Robert M. Commentary. Review of Research in Futures Markets, 3 (1984), 278-279.

4. Brown, Robert Grover. Introduction to Random Signal Analysis and Kalman Filtering. New York, NY: John Wiley & Sons, 1983.

5. Cargill, Thomas F., and Rausser, Gordon C. "Time and Frequency Domain Representation of Futures Prices." Journal of the American Statistical Association, 67 (March 1972), 23-30.

6. Chan, Shu-Park; Chan, Shu-Yun; and Chan, Shu-Gar. Analysis of Linear Networks and Systems. Reading, MA: Addison-Wesley Publishing Company, 1972.

7. Cooper, J. Phillip. "Time Varying Regression Coefficients: A Mixed Estimation Approach and Operational Limitations of the General Markov Structure." Annals of Economic and Social Measurement, 2 (October 1973), 525-530.

8. Cootner, Paul H., ed. Random Character of Stock Market Prices. Cambridge, MA: MIT Press, 1964.

9. Cootner, Paul H. "Stock Prices: Random vs. Systematic Changes." Industrial Management Review, 3 (Spring 1962), 24-45.

10. Cornell, Bradford. "The Weekly Pattern in Stock Returns: Cash versus Futures: A Note." Journal of Finance, 40 (June 1985), 583-588.

11. Dimson, Elroy. "Risk Measurement When Shares Are Subject to Infrequent Trading." Journal of Financial Economics, 7 (June 1979), 197-226.

12. "The Dow Jones Averages." The Wall Street Journal, daily table for years 1983-1985.

13. Fama, Eugene F. "The Behavior of Stock-Market Prices." Journal of Business, 38 (January 1965), 34-105.

14. Fisher, Lawrence, and Kamin, Jules H. "Forecasting Systematic Risk: Estimates of "Raw" Beta that Take Account of the Tendency of Beta to Change and the Heteroskedasticity of Residual Returns." Journal of Financial and Quantitative Analysis, 20 (June 1985), 127-149.

15. Gladstone, Rick. "Computerized Trading Programs Inject Wild Swings Into The Market." The Des Moines Register, February 9, 1986, p. 8F.

16. Granger, Clive W. J., and Morgenstern, Oskar. "Spectral Analysis of New York Stock Market Prices." Kyklos, 16 (1963), 1-27.

17. Harvey, A. C. "A Unified View of Statistical Forecasting Procedures." Journal of Forecasting, 3 (1984), 245-275.

18. Haykin, Simon. Adaptive Filter Theory. Englewood Cliffs, NJ: Prentice-Hall, 1986.

19. Irwin, Scott H., and Uhrig, William J. "Do Technical Analysts Have Holes in Their Shoes?" Review of Research in Futures Markets, 3 (1984), 265-277.

20. Kahl, Douglas R., and Ledolter, Johannes. "A Recursive Kalman Filter Forecasting Approach." Management Science, 29 (November 1983), 1325-1333.

21. Kendall, M. G. "The Analysis of Economics Time Series -- Part 1: Prices." Journal of the Royal Statistical Society, 96 Part I (1953), 11-25.

22. Martell, Terrence F., and Helms, Billy P. "A Reexamination of Price Changes in the Commodity Futures Market." Proceedings of the International Futures Trading Seminar, 5 (May 1978), 136-152.

23. McWhorter, Archer, Jr. "Time Series Forecasting Using the Kalman Filter: An Empirical Study." Proceedings of the American Statistical Association, (August 1975), 436-441.

24. Morrison, G. W., and Pike, D. H. "Kalman Filtering Applied to Statistical Forecasting." Management Science, 23 (March 1977), 768-774.

25. Oppenheim, Alan V., and Schafer, Ronald W. Digital Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.

26. Osborne, M. F. M. "Brownian Motion in the Stock Market." Operations Research, 7 (March-April, 1959), 145-173.

27. Pankratz, Alan. Forecasting with Univariate Box-Jenkins Models. New York, NY: John Wiley & Sons, 1983.

28. Roberts, Harry V. "Stock-Market "Patterns" and Financial Analysis: Methodological Suggestions." Journal of Finance, 14 (March 1959), 1-10.

29. Sarris, Alexander H. "Kalman Filter Models: A Bayesian Approach to Estimation of Time-varying Regression Coefficients." Annals of Economic and Social Measurement, 2 (1973), 501-523.

30. Sastri, T. "An Adaptive Autoregressive Model." Computers and Industrial Engineering, 9 (1985), 9-27.

31. Schartz, Robert A., and Whitcomb, David K. "Evidence on the Presence and Causes of Serial Correlation in Market Model Residuals." Journal of Financial and Quantitative Analysis, 12 (1977), 291-313.

32. Scholes, Myron, and Williams, Joseph. "Estimating Betas from Nonsynchronous Data." Journal of Financial Economics, 5 (December 1977), 309-327.

33. Standard and Poor's Corporation. Daily Stock Price Record: New York Stock Exchange. New York, NY: Standard and Poor's Corporation.

34. Standard and Poor's Corporation. Standard and Poor's Corporation Records - Daily News Section. New York, NY: Standard and Poor's Corporation.

35. Szelag, C. R. "A Short-Term Forecasting Algorithm for Trunk Demand." The Bell System Technical Journal, 61 (January 1982), 67-96.

36. Theobald, Michael. "The Analytic Relationship between Intervaling and Nontrading Effects in Continuous Time." Journal of Financial and Quantitative Analysis, 18 (June 1983), 199-208.

37. Walpole, Ronald E., and Myers, Raymond H. Probability and Statistics for Engineers and Scientists. New York, NY: Macmillan Publishing Co., Inc., 1972.

38. Wood, Robert A.; McInish, Thomas H.; and Ord, J. Keith. "An Investigatation of Transactions Data for NYSE Stocks." Journal of Finance, 40 (July 1985), 723-739.

39. Working, Holbrook. "A Random-Difference Series for Use in the Analysis of Time Series." Journal of the American Statistical Association, 29 (March 1934), 11-24.

40. Working, Holbrook. "Note on the Correlation of First Differences of Averages in a Random Chain." Econometrica, 28 (October 1960), 916-918.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my wife, Wendy, and my daughters, Kara and Anna, for their support and understanding during the many hours spent researching this project.

I would also like to thank my major professor, Dr. R. Grover Brown, for his help and the guidance he provided me while undertaking this research.

My appreciation also goes to George Chadima and the Norand Corporation for the opportunity to pursue this topic.

The time and commitment of my committee, Dr. John Basart, Dr. Herbert T. David, Dr. Bion Pierson, and Dr. Terry Smay, is also appreciated.

I would also like to thank the Electrical Engineering Endowment Fund Advisory Committee for selecting me as the recipient of the Palmer Electrical Engineering Fellowship.

APPENDIX: SOFTWARE LISTINGS

Autocorrelation and Partial Autocorrelation

The ACF and PACF program reads in the sample realization from an external file. The first difference of the data is calculated to provide a stationary working series. The first difference mean is then subtracted from the data to simplify the calculation of the autocovariance terms. Twelve lags are determined for both the ACF and PACF. The lag coefficients are normalized such that the coefficient for lag zero is 1. To determine statistical significance, a 95% confidence level is calculated for each lag.

```
      PROGRAM ACORR
C
      REAL RHO(50),PACF(50),PHI(2,50),S,T,PHINUM,PHIDEN
      REAL SUM,INPUT(2000),MEANSQ,NREAL,RHOSUM
      INTEGER N,TAU,NACF,NPACF
C
      NACF = 12
      NPACF = 12
C
C*******************************************
C  OPEN DATA FILE AND READ IN DATA
      OPEN(1,FILE='DOWJONES.TRNSPRT1.HOUR')
      N=0
  100 N = N+1
      READ(1,110,END=500)INPUT(N)
  110 FORMAT(5X,F12.4)
      GOTO 100
  500 CONTINUE
      N = N-1
      WRITE(6,600)N
  600 FORMAT(1H0,'N = ',I5)
C
C*************************************************
C  CONVERT DATA TO FIRST DIFFERENCE
      DO 700 I = 1,N-1
```

```
          INPUT(I) = INPUT(I+1) - INPUT(I)
  700 CONTINUE
      N = N-1
      NREAL = REAL(N)
C
C*****************************************************
C CALCULATE THE FIRST DIFFERENCE MEAN AND SUBTRACT IT
      SUM = 0.0
      MEANSQ = 0.0
      DO 800 I = 1,N
        SUM = INPUT(I) + SUM
        MEANSQ = INPUT(I)**2 + MEANSQ
  800 CONTINUE
      SUM = SUM/NREAL
      MEANSQ = MEANSQ/NREAL
      DO 900 I=1,N
        INPUT(I) = INPUT(I) - SUM
  900 CONTINUE
C
C*************************************************
C CALCULATE THE ACF
      RHOSUM = 0.0
      WRITE(6,1000)
 1000 FORMAT(1H0,'ACF')
      WRITE(6,1100)
 1100 FORMAT(1X,28X,'-1.0',5X,'-0.8',6X,'-0.6',6X,'-0.4',6X,
     1  '-0.2',7X,'0.0',7X,'0.2',7X,'0.4',7X,'0.6',7X,'0.8',
     2  7X,'1.0'/1X,'LAG',3X,'COEFF',5X,'S',5X,'T-VAL',
     3  2X,'¦',10(9('-'),'¦'))
      DO 1200 TAU = 1,NACF
        SUM = 0.0
        DO 1150 I = 1,N-TAU
          SUM = SUM + INPUT(I)*INPUT(I+TAU)
 1150   CONTINUE
        RHO(TAU) = SUM/(MEANSQ*(N-TAU))
        S=SQRT((1.0+2.0*RHOSUM)/NREAL)
        T = RHO(TAU)/S
        CALL PLOT(TAU,RHO(TAU),S,T)
        RHOSUM = RHOSUM + RHO(TAU)**2
 1200 CONTINUE
C
C*************************************************************
C CALCULATE THE PACF
      S = SQRT(1/NREAL)
      WRITE(6,1300)
 1300 FORMAT(1H0,'PACF')
      WRITE(6,1100)
      DO 1500 TAU = 1,NPACF
        PHINUM = 0.0
        PHIDEN = 0.0
```

```
          DO 1400 J=1,TAU-1
            PHINUM = PHINUM + PHI(1,J)*RHO(TAU-J)
            PHIDEN = PHIDEN + PHI(1,J)*RHO(J)
 1400     CONTINUE
          PHI(2,TAU) = (RHO(TAU)-PHINUM)/(1.0-PHIDEN)
          PACF(TAU) = PHI(2,TAU)
          DO 1600 J = 1,TAU-1
            PHI(2,J) = PHI(1,J) - PHI(2,TAU)*PHI(1,TAU-J)
 1600     CONTINUE
          DO 1700 J = 1,TAU
            PHI(1,J) = PHI(2,J)
 1700     CONTINUE
          T=PACF(TAU)/S
          CALL PLOT(TAU,PACF(TAU),S,T)
 1500   CONTINUE
C
       END
C
C*************************************
C      PLOT SUBROUTINE
       SUBROUTINE PLOT(TAU,COEFF,S,T)
       INTEGER TAU,NSTARS,NTWOS
       REAL COEFF,S,T
       EQUIVALENCE (GRAFLN,LINE(1))
       CHARACTER*1 LINE(100)
       CHARACTER*100 GRAFLN
C
       DO 2000 I = 1,100
         LINE(I) = ' '
 2000  CONTINUE
       NTWOS = INT(50.0*1.96*S + 0.5)
       IF (COEFF.LT.0.0) THEN
         NSTARS = INT(50.0*COEFF - 0.5)
         DO 2100 I = 50+NSTARS,50
           LINE(I) = '*'
 2100    CONTINUE
       ELSE
         NSTARS = INT(50.0*COEFF + 0.5)
         DO 2200  I=50,NSTARS+50
           LINE(I) = '*'
 2200    CONTINUE
       ENDIF
       LINE(50-NTWOS) = '<'
       LINE(50+NTWOS) = '>'
       WRITE(6,2300)TAU,COEFF,S,T,GRAFLN
 2300  FORMAT(1X,I2,3X,F6.3,2X,F6.3,2X,F6.2,2X,'¦',A)
       RETURN
       END
```

Kalman Filter

The Kalman filter program produces N-step ahead forecasts for a time series. The mean square error of the forecast errors are calculated to measure model accuracy. An autocorrelation function and Box-Ljung statistic are also calculated on the forecast errors to determine if the errors are correlated. Additional information on the Kalman filter algorithm is presented in the Kalman Filter chapter.

```
      PROGRAM KALMAN
C
      REAL X(5),XEST(5),XAHEAD(5),XPRDCT(10)
      REAL P(5,5),PEST(5,5)
      REAL H(5,5),HTRNSP(5,5),K(5)
      REAL PHI(5,5),PHIT(5,5),PHIAHD(5,5)
      REAL Q(5,5),IDENT(5,5),TEMP(5,5)
      REAL ERRVAR,AVEERR,MSE,TSTAT,CHI,STDERR,RCNT
      REAL XAXIS(0:1500),RHO(0:50),ERROR(0:1500)
      REAL Z,Y,R,YEST,SCALR
C
      INTEGER N,PRDCTN,NACF
      INTEGER CNT,CMPCNT,RHOCNT
C
      LOGICAL OPENPR
C
      CHARACTER*4 OPEN
C
C*************************************************
C  OPEN FILE UNITS   #6 IS PRINTER & #5 IS CARD READER
C  FILE 1 = INPUT RAW DATA, #2 = OUTPUT OF X EST
C
      OPEN(1,FILE='DOWJONES.TRNSPRT1.HOUR')
      OPEN(2,FILE='KALMAN.DATA',STATUS='NEW')
C
C*************************************************
C READ IN N,PHI,H,Q,R, AND INITIAL ESTIMATES OF XEST AND PEST
      READ(5,100)N
      WRITE(6,105)'N = ',N
  100 FORMAT(I2)
  105 FORMAT(1H0,A,I5)
C
```

```
      DO 110 I=1,N
         READ(5,115)(PHI(I,J),J=1,N)
         WRITE(6,120)(PHI(I,J),J=1,N)
         DO 110 J=1,N
          PHIT(J,I)=PHI(I,J)
  110 CONTINUE
  115 FORMAT(6F12.4)
  120 FORMAT(1H0,'PHI : ',10F12.4)
C
      READ(5,130)(H(1,I),I=1,N)
  130 FORMAT(6F8.4)
      WRITE(6,135)(H(1,I),I=1,N)
  135 FORMAT(1H0,'H MATRIX: ',10F8.4)
      DO 136 I=1,N
        HTRNSP(I,1)=H(1,I)
  136 CONTINUE
C
      READ(5,140)R
  140 FORMAT(F8.6)
      WRITE(6,145)R
  145 FORMAT(1H0,'R = ',F8.6)
C
      DO 150 I=1,N
        READ(5,160)(Q(I,J),J=1,N)
      WRITE(6,165)(Q(I,J),J=1,N)
  150 CONTINUE
  160 FORMAT(5F12.6)
  165 FORMAT(1H0,'Q: ',10F12.6)
C
      READ(5,170)(XEST(I),I=1,N)
      WRITE(6,175)(XEST(I),I=1,N)
  170 FORMAT(6F12.4)
  175 FORMAT(1H0,'XEST (TRANSPOSED) = ',6(F8.2,3X))
C
      DO 180 I=1,N
      READ(5,200)(PEST(I,J),J=1,N)
      WRITE(6,205)(PEST(I,J),J=1,N)
  180 CONTINUE
  200 FORMAT(6F12.4)
  205 FORMAT(1H0,'PEST : ',6F12.4)
C
C*****************************************
C READ IN START POINT FOR COMPARISON
      READ(5,220)CMPCNT
  220 FORMAT(I5)
C
C READ IN # OF STEPS AHEAD TO PREDICT
      READ(5,230)PRDCTN
  230 FORMAT(I2)
C
```

```
      WRITE(6,290)
  290 FORMAT(1H ,100(1H-)/1H0,'RESULTS')
C
C**********************************
C CREATE IDENTITY MATRIX
      DO 300 I=1,N
        DO 300 J=1,N
           IF(I.EQ.J) THEN
              IDENT(I,I)=1.0
           ELSE
              IDENT(I,J)=0.0
           ENDIF
  300 CONTINUE
C
C GENERATE PHI MATRIX TO PREDICT AHEAD N STEPS
      DO 310 I=1,N
        DO 310 J=1,N
          PHIAHD(I,J)=PHI(I,J)
  310 CONTINUE
      DO 320 I=1,PRDCTN-1
        CALL MATMPY(N,N,N,PHI,PHIAHD,PHIAHD)
  320 CONTINUE
C
C*************************************************************
C READ IN MEASUREMENTS, PERFORM FILTER OPS, AND STORE DATA  *
C*************************************************************
C
      AVEERR = 0.0
      MSE = 0.0
      CNT=-1
  400 READ(1,410,END=500)Z,OPEN
  410 FORMAT(5X,F12.4,2X,A4)
C
      OPENPR = .FALSE.
      IF (OPEN.EQ.'OPEN') THEN
        OPENPR = .TRUE.
      ENDIF
C
C*********************************************
C  CALCULATE ERROR STATISTICS
      IF (OPENPR) GOTO 1000
        CNT=CNT+1
        IF(CNT.LT.CMPCNT) GOTO 1000
          XAXIS(CNT-CMPCNT) = REAL(CNT-CMPCNT)
          ERROR(CNT-CMPCNT) = Z-XPRDCT(1)
          AVEERR= AVEERR + ERROR(CNT-CMPCNT)
          MSE = MSE + ERROR(CNT-CMPCNT)**2
 1000 CONTINUE
C
C*************************************************
```

```
C  CALCULATE THE KALMAN GAIN
       CALL MATMPY(1,N,N,H,PEST,TEMP)
       CALL MATMPY(1,N,1,TEMP,HTRNSP,K)
       SCALR=1/(K(1)+R)
       DO 440 I=1,N
         K(I)=SCALR* HTRNSP(I,1)
  440 CONTINUE
       CALL MATMPY(N,N,1,PEST,K,K)
C
C*****************************************
C   UPDATE THE STATE MATRIX X
       CALL MATMPY(1,N,1,H,XEST,X)
       SCALR = Z -X(1)
       DO 430 I=1,N
         X(I)= SCALR* K(I)
  430 CONTINUE
       CALL MATADD(N,1,XEST,X,X)
C
C*****************************************
C    UPDATE THE ERROR COVARIANCE MATRIX, P
       CALL MATMPY(N,1,N,K,H,P)
       DO 470 I=1,N
         DO 470 J=1,N
           P(I,J)= -1.0*P(I,J)
  470 CONTINUE
       CALL MATADD(N,N,IDENT,P,P)
       CALL MATMPY(N,N,N,P,PEST,P)
C
       CALL MATMPY(1,N,1,H,X,Y)
C      WRITE(2,420)CNT,Z,Y,XEST(1)
C 420 FORMAT(I5,F12.4,F12.4,F12.4)
C*************************************************
C PROJECT AHEAD XEST AND PEST FOR NEXT STEP
       CALL MATMPY(N,N,N,PHI,P,PEST)
       CALL MATMPY(N,N,N,PEST,PHIT,PEST)
       CALL MATADD(N,N,PEST,Q,PEST)
       CALL MATMPY(N,N,1,PHI,X,XEST)
C
C*****************************************
C FORECAST X MATRIX AHEAD N STEPS
       CALL MATMPY(N,N,1,PHIAHD,X,XAHEAD)
       DO 2000 I=1,PRDCTN-1
         XPRDCT(I)=XPRDCT(I+1)
 2000 CONTINUE
       XPRDCT(PRDCTN)=XAHEAD(1)
C
C ***********************************************
C RETURN TO BEGINNING OF CYCLE
       GOTO 400
  500 CLOSE(1)
```

```
      CLOSE(2)
C
      RHOCNT = CNT-CMPCNT
      RCNT = REAL(RHOCNT)
      AVEERR = AVEERR/(RHOCNT+1)
      MSE = MSE / RCNT
      ERRVAR = MSE - (AVEERR)**2
      STDERR = SQRT(ERRVAR)
      TSTAT = AVEERR /(STDERR/SQRT(RCNT))
C
C*************************************************************

C  PRINT FINAL KALMAN GAIN, STATE ESTIMATE,
C  & ERROR COVARIANCE MATRIX
       WRITE(6,600)(K(I),I=1,N)
  600 FORMAT(1H0,'KALMAN GAIN VECTOR (TRANSPOSED) IS:',
     1  10F8.4)
       WRITE(6,650)(XEST(I),I=1,N)
  650 FORMAT(1H0,'THE EST. STATE VECTOR (TRANSPOSED) IS: ',
     $  8(F10.3,3X))
       WRITE(6,700)'THE FINAL EST. ERROR COVARIANCE MATRIX
     1  IS:'
  700 FORMAT(1H0,A)
       DO 800 I=1,N
         WRITE(6,900)(PEST(I,J),J=1,N)
  800 CONTINUE
  900 FORMAT(1H0,10F12.4)
C*****************************************
C
      WRITE(6,1050)PRDCTN
 1050 FORMAT(1H1,'ERROR STATISTICS ARE FOR ',I5,
     $  '-STEP AHEAD PREDICTION')
      WRITE(6,1070)RHOCNT,AVEERR,TSTAT,MSE,ERRVAR
 1070 FORMAT(1H0,'RESIDUAL STATISTICS:  ','# =',I5,3X,
     1  'MEAN = ',F8.4,3X,'T-STATISTIC = ',F6.2,3X,
     2  'MSE = ',F8.4,3X,'VARIANCE = ',F8.4)
C*****************************************
C  CALCULATE RESIDUAL ACF
C
      RHO(0) = 0.0
      SUM = 0.0
      NACF = 20
      WRITE(6,6000)
 6000 FORMAT(1H0,'AUTOCORRELATION PLOT ')
      WRITE(6,6010)
 6010 FORMAT(1H0,28X,'-1.0',5X,'-0.8',6X,'-0.6',6X,'-0.4',6X,
     1  '-0.2',7X,'0.0',7X,'0.2',7X,'0.4',7X,'0.6',7X,
     2  '0.8',7X,'1.0'/1X,'LAG',3X,'COEFF',5X,'S',5X,
     3  'T-VAL',2X,'|',10(9('-'),'|'))
      DO 4000 I=1,NACF
```

```
          RHO(I)=0.0
          DO 4100 J=0,RHOCNT-I
            RHO(I)= RHO(I) + ERROR(J)*ERROR(I+J)
 4100     CONTINUE
          RHO(I) = RHO(I)/(RHOCNT*MSE)
          SUM = SUM + RHO(I-1)**2
          S = SQRT((1.0 + 2.0*SUM)/RCNT)
          T = RHO(I)/S
          CALL PLOT(I,RHO(I),S,T)
 4000 CONTINUE
C
C*********************************************
C    CALCULATE CHI-SQUARE DISTRIBUTION FOR RESIDUALS
C
         CHISQR = 0.0
         DO 5000 I=1,NACF
           CHISQR = CHISQR + ((RHO(I)**2)/(RHOCNT+1-I))
 5000 CONTINUE
      CHISQR = CHISQR*(RHOCNT+1)*(RHOCNT+2+1)
      WRITE(6,1100)CHISQR
 1100 FORMAT(1H0,'BOX-LJUNG STATISTIC FOR THE RESIDUAL ACF=',
     1  F12.4,3X,'WITH 16 DEGREES OF FREEDOM. 95% C.I.=26.3')
      WRITE(6,1150)
C*****************************************
C  PLOT RESIDUALS
      CALL GRAPH(RHOCNT+1,XAXIS,ERROR,3,7,12.0,9.0,0.0,0.0,
     $  0.0,0.0,'INDEX;','ERROR;','RESIDUALS;',';')
C
      END
C
C*****************************************
C  SUBROUTINES
C   MATRIX ADD SUBROUTINE
      SUBROUTINE MATADD(I,J,A,B,C)
      REAL A(5,5),B(5,5),C(5,5)
      DO 2100 L=1,I
        DO 2100 M=1,J
          C(L,M)=A(L,M)+B(L,M)
 2100 CONTINUE
      RETURN
      END
C
C*****************************************
C   MATRIX MULTIPLY ROUTINE
      SUBROUTINE MATMPY(I,J,L,A,B,C)
      INTEGER I,J,L,M,O,U
      REAL A(5,5),B(5,5),C(5,5),TMP(5,5)
      REAL SUM
      DO 2200 M=1,I
        DO 2200 O=1,L
```

```
            SUM=0.0
            DO 2210 U=1,J
              SUM=SUM + A(M,U)*B(U,O)
 2210 CONTINUE
              TMP(M,O)=SUM
 2200 CONTINUE
      DO 2220 M=1,I
        DO 2220 O=1,L
          C(M,O)=TMP(M,O)
 2220 CONTINUE
      RETURN
      END
C
C*****************************************
      SUBROUTINE PLOT(TAU,COEFF,S,T)
C
      CHARACTER*1 LINE(100)
      CHARACTER*100 GRAFLN
      INTEGER TAU,NSTARS,NTWOS
      REAL COEFF,S,T
      DATA LINE/100*' '/
      EQUIVALENCE (GRAFLN,LINE(1))
C
      DO 50 I=1,100
        LINE(I) = ' '
   50 CONTINUE
      NTWOS = INT(50.0*1.96*S + 0.5)
      IF (COEFF.LT.0.0) THEN
        NSTARS = INT(50.0*COEFF - 0.5)
        DO 100 I=50+NSTARS,50
          LINE(I) = '*'
  100   CONTINUE
      ELSE
        NSTARS = INT(50.0*COEFF + 0.5)
        DO 200 I=50,50+NSTARS
          LINE(I) = '*'
  200   CONTINUE
      ENDIF
      LINE(50-NTWOS) = '<'
      LINE(50+NTWOS) = '>'
      WRITE(6,300)TAU,COEFF,S,T,GRAFLN
  300 FORMAT(1X,I2,3X,F6.3,2X,F6.3,2X,F6.2,2X,'|',A)
      RETURN
      END
```

Adaptive Kalman Filter

The adaptive Kalman filter program produces forecasts of the process realization. The forecast errors are used to modify the model parameters such that the model more closely resembles the process. The mean square error of the forecast errors is used to as a performance index for model comparison. An autocorrelation function and Box-Ljung statistic are calculated to test for correlation in the forecast errors.

```
      PROGRAM AKF
C
      REAL X(5),XEST(5),P(5,5),PEST(5,5),A(5),ZOLD(5)
      REAL H(5,5),HTRNSP(5),K(5),C(5),Q(5,5)
      REAL TEMP2(5,5),TEMP(5,5),TMPVCT(5)
      REAL BETA,ZBAR,SCALR2
      REAL ERRVAR,AVEERR,MSE,TSTAT,CHI,STDERR,RCNT
      REAL XAXIS(0:1000),RHO(0:25),ERROR(0:1000)
      REAL Z,Y,R,YEST,SCALR
C
      INTEGER N,PRDCTN,NACF
      INTEGER CNT,CMPCNT,RHOCNT,RUNCNT
C
      LOGICAL OPENPR
C
      CHARACTER*4 OPEN
C*****************************************************
C  OPEN FILE UNITS   #6 IS PRINTER & #5 IS CARD READER
C  FILE 1 = INPUT RAW DATA, #2 = OUTPUT OF X EST
C
      OPEN(1,FILE='DOWJONES.TRNSPRT1.HOUR')
      OPEN(2,FILE='KALMAN.DATA',STATUS='NEW')
C
C*****************************************************
C READ IN PARAMETERS
      READ(5,100)N
      WRITE(6,110)'N = ',N
  100 FORMAT(I2)
  110 FORMAT(1H0,A,I2)
C
```

```
      READ(5,200)ZBAR
  200 FORMAT(F8.3)
C
      WRITE(6,205)ZBAR
  205 FORMAT(1H0,'PREVIOUS Z VALUES ARE INITIALIZED TO: ',
     1  F8.3)
C
      DO 210 I=1,5
        ZOLD(I) = ZBAR
        A(I) = 0.0
  210 CONTINUE
C
  260 FORMAT(5(F8.4,4X))
C
      READ(5,300)R
  300 FORMAT(F8.4)
      WRITE(6,310)R
  310 FORMAT(1H0,'R = ',F8.4)
C
      DO 426 J=1,N
      READ(5,427)(Q(J,I),I=1,N)
      WRITE(6,425)(Q(J,I),I=1,N)
  426 CONTINUE
  427 FORMAT(5F12.7)
  425 FORMAT(1H0,'Q = ',5(F12.7))
C
      READ(5,350)(C(I),I=1,N)
      WRITE(6,400)(C(I),I=1,N)
  350 FORMAT(5F12.7)
  400 FORMAT(1H0,'C (TRANSPOSED) = ',5F12.7)
C
      READ(5,410)BETA
      WRITE(6,415)BETA
  410 FORMAT(F12.6)
  415 FORMAT(1H0,'BETA = ',F8.6)
C
      READ(5,260)(XEST(I),I=1,N)
      WRITE(6,460)(XEST(I),I=1,N)
  460 FORMAT(1H0,'XEST (TRANSPOSED) = ',5(F8.3,3X))
C
      DO 500 I=1,N
      READ(5,260)(PEST(I,J),J=1,N)
      WRITE(6,520)(PEST(I,J),J=1,N)
  500 CONTINUE
  520 FORMAT(1H0,'PEST : ',5(F8.4,3X))
C
C*****************************************
C READ IN START POINT FOR COMPARISON
      READ(5,550)CMPCNT
  550 FORMAT(I5)
```

```
C READ IN # OF STEPS AHEAD TO PREDICT
      READ(5,100)PRDCTN
C
      WRITE(6,750)
  750 FORMAT(1H ,80(1H-)/1H0,'RESULTS')
C
C*****************************************
C  CALCULATE THE H MATRIX
      HTRNSP(1) = ZOLD(1)
      HTRNSP(2) = 1.0
      H(1,1) = HTRNSP(1)
      H(1,2) = HTRNSP(2)
      CALL VCTMPY(N,HTRNSP,XEST,XFORE)
C
C*************************************************************
C READ IN MEASUREMENTS, PERFORM FILTER OPS, AND STORE DATA  *
c*************************************************************
C
      AVEERR = 0.0
      MSE = 0.0
      CNT=-1
 1000 READ(1,1100,END=1900)Z,OPEN
 1100 FORMAT(5X,F12.4,2X,A4)
C
      CNT = CNT+1
      OPENPR = .FALSE.
      IF (OPEN.EQ.'OPEN') THEN
        OPENPR = .TRUE.
      ENDIF
C
C*************************************************
C  CALCULATE THE KALMAN GAIN
      CALL MATMPY(N,N,1,PEST,HTRNSP,TMPVCT)
      CALL VCTMPY(N,HTRNSP,TMPVCT,SCALR2)
      CALL VCTMPY(N,HTRNSP,C,SCALR)
      SCALR = 2.0*SCALR + SCALR2 + R
      CALL MATADD(N,1,C,TMPVCT,TEMP2)
      DO 1150 I=1,N
        K(I)= TEMP2(I,1)/SCALR
 1150 CONTINUE
C
C***************************************
C  DETERMINE FORECAST FOR THIS PERIOD AND ADJUST A AND ZOLD
      SCALR = Z -XFORE
      DO 1200 I=1,4
        A(6-I) = A(5-I)
        ZOLD(6-I) = ZOLD(5-I)
 1200 CONTINUE
      A(1) = SCALR
      ZOLD(1) = Z
```

```
C*************************************************
C       ACCUMULATE ERROR STATISTICS
      IF (CNT.GE.CMPCNT) THEN
        IF (OPENPR) THEN
          CNT = CNT - 1
        ELSE
          XAXIS(CNT-CMPCNT) = REAL(CNT-CMPCNT)
          ERROR(CNT-CMPCNT) = Z-XFORE
          AVEERR= AVEERR + ERROR(CNT-CMPCNT)
          MSE = MSE + ERROR(CNT-CMPCNT)**2
        ENDIF
      ENDIF
C
C*************************************
C   UPDATE THE STATE MATRIX X
      DO 1300 I=1,N
        TEMP(I,1)= SCALR* K(I)
 1300 CONTINUE
      CALL MATADD(N,1,XEST,TEMP,X)
C
C*****************************************
C    UPDATE THE ERROR COVARIANCE MATRIX, P
      CALL MATMPY(1,N,N,H,PEST,TEMP)
      DO 1350 I=1,N
        TEMP2(1,I) = C(I)
 1350 CONTINUE
      CALL MATADD(1,N,TEMP2,TEMP,P)
      CALL MATMPY(N,1,N,K,P,TEMP)
      DO 1360 I=1,N
        DO 1360 J=1,N
          TEMP(I,J) = -1.0*TEMP(I,J)
 1360 CONTINUE
      CALL MATADD(N,N,PEST,TEMP,P)
      CALL SYMTRC(N,P)
C
      WRITE(2,1400)CNT,Z,XFORE
 1400 FORMAT(I4,2X,F6.2,2X,F7.3)
C
C*************************************************
C PROJECT AHEAD XEST AND PEST FOR NEXT STEP
      CALL MATADD(N,N,P,Q,TEMP)
      DO 1460 I=1,N
        DO 1460 J=1,N
          PEST(I,J) = BETA*TEMP(I,J)
 1460 CONTINUE
      CALL SYMTRC(N,PEST)
C
      DO 1500 I=1,N
        XEST(I) = X(I)
 1500 CONTINUE
```

```
C*********************************************
C  CALCULATE THE H MATRIX
      HTRNSP(1) = ZOLD(1)
      HTRNSP(2) = 1.0
      H(1,1) = HTRNSP(1)
      H(1,2) = HTRNSP(2)
      CALL VCTMPY(N,HTRNSP,XEST,XFORE)
C
C ******************************************
C RETURN TO BEGINNING OF CYCLE
      GOTO 1000
 1900 CLOSE(1)
      CLOSE(2)
C
      RHOCNT = CNT-CMPCNT
      RCNT = REAL(RHOCNT)
      AVEERR = AVEERR/(RHOCNT+1)
      MSE = MSE / RCNT
      ERRVAR = MSE - (AVEERR)**2
      STDERR = SQRT(ERRVAR)
      TSTAT = AVEERR /(STDERR/SQRT(RCNT))
C
C*************************************************************
C  PRINT KALMAN GAIN, STATE ESTIMATE AND
C   ERROR COVARIANCE MATRIX
      WRITE(6,2000)(K(I),I=1,N)
 2000 FORMAT(1H0,'KALMAN GAIN VECTOR (TRANSPOSED) IS: ',
     1   10F8.4)
      WRITE(6,2100)(XEST(I),I=1,N)
 2100 FORMAT(1H0,'THE EST. STATE VECTOR (TRANSPOSED) IS: ',
     $  8(F10.3,3X))
      WRITE(6,2200)'THE FINAL EST. ERROR COVARIANCE MATRIX:'
 2200 FORMAT(1H0,A)
      DO 2300 I=1,N
        WRITE(6,2400)(PEST(I,J),J=1,N)
 2300 CONTINUE
 2400 FORMAT(1H0,10F12.4)
C
C*****************************************
C  WRITE RESIDUAL STATISTICS
      WRITE(6,2500)PRDCTN
 2500 FORMAT(1H1,'ERROR STATISTICS ARE FOR ',I2,
     $  '-STEP AHEAD PREDICTION')
      WRITE(6,2600)RHOCNT,AVEERR,TSTAT,MSE,ERRVAR
 2600 FORMAT(1H0,'RESIDUAL STATISTICS:  ','# =',I5,3X,
     1  'MEAN = ',F8.4,3X,'T-STATISTIC = ',
     2  F6.2,3X,'MSE = ',F8.4,3X,'VARIANCE = ',F8.4)
C*****************************************
C  CALCULATE RESIDUAL ACF
C
```

```
      RHO(0) = 0.0
      SUM = 0.0
      NACF = 20
      WRITE(6,3000)
 3000 FORMAT(1H0,'AUTOCORRELATION PLOT ')
      WRITE(6,3100)
 3100 FORMAT(1H0,28X,'-1.0',5X,'-0.8',6X,'-0.6',6X,'-0.4',6X,
     1  '-0.2',7X,'0.0',7X,'0.2',7X,'0.4',7X,'0.6',7X,'0.8',
     2  7X,'1.0'/1X,'LAG',3X,'COEFF',5X,'S',5X,'T-VAL',
     3  2X,'|',10(9('-'),'|'))
       DO 3300 I=1,NACF
         RHO(I)=0.0
         DO 3200 J=0,RHOCNT-I
           RHO(I)= RHO(I) + ERROR(J)*ERROR(I+J)
 3200    CONTINUE
         RHO(I) = RHO(I)/(RHOCNT*MSE)
         SUM = SUM + RHO(I-1)**2
         S = SQRT((1.0 + 2.0*SUM)/RCNT)
         T = RHO(I)/S
         CALL PLOT(I,RHO(I),S,T)
 3300 CONTINUE
C
C*********************************************
C   CALCULATE CHI-SQUARE DISTRIBUTION FOR RESIDUALS
C
       CHISQR = 0.0
       DO 3400 I=1,NACF
         CHISQR = CHISQR + ((RHO(I)**2)/(RHOCNT+1-I))
 3400 CONTINUE
      CHISQR = CHISQR*(RHOCNT+1)*(RHOCNT+2+1)
C
      WRITE(6,3500)CHISQR
 3500 FORMAT(1H0,'BOX-LJUNG STATISTIC FOR THE RESIDUAL ACF=',
     1  F10.1,3X,'WITH 16 DEGREES OF FREEDOM. 95% C.I.=26.3')
C*********************************************
C  PLOT RESIDUALS
      CALL GRAPH(110,XAXIS,ERROR,3,7,12.0,9.0,0.0,0.0,
     $  0.0,0.0,'INDEX;','ERROR;','RESIDUALS;',';')
C
      END
C
C*****************************************
C  SUBROUTINES
C   MATRIX ADD SUBROUTINE
      SUBROUTINE MATADD(I,J,A,B,C)
      REAL A(5,5),B(5,5),C(5,5)
      DO 2100 L=1,I
        DO 2100 M=1,J
          C(L,M)=A(L,M)+B(L,M)
 2100 CONTINUE
```

```
      RETURN
      END
C
C*****************************************
C   MATRIX MULTIPLY ROUTINE
      SUBROUTINE MATMPY(I,J,L,A,B,C)
      INTEGER I,J,L,M,O,U
      REAL A(5,5),B(5,5),C(5,5),TMP(5,5)
      REAL SUM
      DO 2200 M=1,I
        DO 2200 O=1,L
          SUM=0.0
          DO 2210 U=1,J
            SUM=SUM + A(M,U)*B(U,O)
 2210 CONTINUE
            TMP(M,O)=SUM
 2200 CONTINUE
      DO 2220 M=1,I
        DO 2220 O=1,L
          C(M,O)=TMP(M,O)
 2220 CONTINUE
      RETURN
      END
C
C*****************************************
C  SUBROUTINES
C   VECTOR MULTIPLY ROUTINE
      SUBROUTINE VCTMPY(I,A,B,SUM)
      INTEGER I
      REAL A(5),B(5)
      REAL SUM
      SUM=0.0
      DO 2200 M=1,I
            SUM=SUM + A(M)*B(M)
 2200 CONTINUE
      RETURN
      END
C
C*********************************************
C  MAKE MATRIX SYMMETRIC
      SUBROUTINE SYMTRC(N,ARRAY)
      REAL ARRAY(5,5)
      DO 1000 I=1,N-1
        DO 1000 J=I+1,N
          ARRAY(I,J) = (ARRAY(I,J) + ARRAY(J,I))/2.0
          ARRAY(J,I) = ARRAY(I,J)
 1000 CONTINUE
      DO 2000 I=1,N
       IF (ARRAY(I,I).LT.0.0) THEN
         ARRAY(I,I) = 0.0
```

```
        ENDIF
 2000 CONTINUE
      RETURN
      END
C
C*************************************
      SUBROUTINE PLOT(TAU,COEFF,S,T)
C
      CHARACTER*1 LINE(100)
      CHARACTER*100 GRAFLN
      INTEGER TAU,NSTARS,NTWOS
      REAL COEFF,S,T
      DATA LINE/100*' '/
      EQUIVALENCE (GRAFLN,LINE(1))
C
      DO 50 I=1,100
        LINE(I) = ' '
   50 CONTINUE
      NTWOS = INT(50.0*1.96*S + 0.5)
      IF (COEFF.LT.0.0) THEN
        NSTARS = INT(50.0*COEFF - 0.5)
        DO 100 I=50+NSTARS,50
          LINE(I) = '*'
  100   CONTINUE
      ELSE
        NSTARS = INT(50.0*COEFF + 0.5)
        DO 200 I=50,50+NSTARS
          LINE(I) = '*'
  200   CONTINUE
      ENDIF
      LINE(50-NTWOS) = '<'
      LINE(50+NTWOS) = '>'
      WRITE(6,300)TAU,COEFF,S,T,GRAFLN
  300 FORMAT(1X,I2,3X,F6.3,2X,F6.3,2X,F6.2,2X,'|',A)
      RETURN
      END
```

Speculator Buy/Sell Strategy

The speculator buy and sell program uses the Kalman filter forecasts to guide market transactions. The actual and forecast prices are stored in an external file created by the Kalman or adaptive Kalman filter programs. The user selects which buy and sell strategy to try. The available

strategies are step, hysteresis, linear and quadratic. The program starts with $10,000 in assets and tries to increase the assets by using the selected strategy. A broker commission, or load factor, can be charged on each transaction.

```
      PROGRAM SPECULATOR
C
      REAL Z(1050),XFORE(1050),DELTAX,ABSDIF
      REAL BFACTR,SFACTR,DLTSHR,CASH,SHARES,ASSETS,BIAS,
      REAL LOAD,OLDSHR,OLDCSH,BROKER,CLIP,PERCNT
C
      INTEGER CNT,CMPCNT,COUNT
C
      LOGICAL OPENPR(1050)
      CHARACTER*4 OPEN
      CHARACTER*1 OPTION
C
C*****************************************************
C  PRINT TITLE
      WRITE(6,10)
   10 FORMAT(1H1)
      WRITE(6,20)
   20 FORMAT(1X,'BUY / SELL STRATEGY:'/1H0,
     1  'S - STEP FUNCTION'/1X,'L - LINEAR'/1X,
     2  'Q - QUADRATIC'/1H0,'ENTER OPTION:')
      READ(5,25)OPTION
   25 FORMAT(A1)
      IF (OPTION.EQ.'S') THEN
        WRITE(6,26)'STEP FUNCTION'
      ELSEIF (OPTION.EQ.'L') THEN
        WRITE(6,26)'LINEAR FUNCTION'
      ELSEIF (OPTION.EQ.'Q') THEN
        WRITE(6,26)'QUADRATIC FUNCTION'
      ENDIF
   26 FORMAT(1X,A)
      WRITE(6,30)
   30 FORMAT(1H0)
C*****************************************************
C  OPEN FILE UNITS: FILE 1 = INPUT RAW DATA, #3 = BATCH
C  DATA FILE
      OPEN(1,FILE='KALMAN.DATA')
      OPEN(3,FILE='SPECULAT.DATA')
C*****************************************************
```

```
      COUNT = 1
 1050 READ(1,1100,END=1200)Z(COUNT),XFORE(COUNT),OPEN
      OPENPR(COUNT) = .FALSE.
      IF (OPEN.EQ.'OPEN') THEN
        OPENPR(COUNT)=.TRUE.
      ENDIF
      COUNT = COUNT + 1
      GOTO 1050
 1100 FORMAT(F6.2,2X,F8.4,2X,A4)
      WRITE(6,40)
   40 FORMAT(1X)
 1200 COUNT = COUNT - 1
      XFORE(COUNT + 1) = Z(COUNT)
      WRITE(6,1225)COUNT
 1225 FORMAT(1X,I5,' DATA POINTS')
C*****************************************************
C READ IN PARAMETERS
      READ(3,550)CMPCNT
  550 FORMAT(I5)
C
      READ(3,675)LOAD
  675 FORMAT(F6.2)
      WRITE(6,680)LOAD
  680 FORMAT(1H0,'LOAD FACTOR = ',F6.2,' %')
      LOAD = LOAD/100.0
C
      WRITE(6,700)
  700 FORMAT(1H0,' BIAS',3X,' BUY',3X,'SELL',3X,' CLIP',3X,
     $  'ASSETS($)',3X,'SHARES($)',3X,' BROKER')
C
  600 READ(3,625,END=6000)BIAS,BFACTR,SFACTR,CLIP
  625 FORMAT(4(F8.4,4X))
C
C*****************************************
C  INITIALIZATION
      CASH   =  5000.00
      OLDCSH = 5000.00
      OLDSHR = 0.0
      PERCNT = 0.50
      BROKER = 0.0
      SHARES = CASH/Z(CMPCNT)
      OLDSHR = SHARES
C*************************************
C READ IN MEASUREMENTS, PERFORM FILTER OPS, AND STORE DATA
      DO 1900 CNT = 1,COUNT
      IF (OPENPR(CNT)) THEN
        CASH = OLDCSH
        SHARES = OLDSHR
      ENDIF
C
```

```
C************************************
C  BUY/SELL STRATEGY
      IF (CNT.LT.CMPCNT) GOTO 1550
        ASSETS = CASH + SHARES*Z(CNT)
        OLDCSH = CASH
        OLDSHR = SHARES
        DELTAX = XFORE(CNT+1) - Z(CNT)
        ABSDIF = ABS(DELTAX)
C
        IF (ABSDIF.LE.BIAS) THEN
          PERCNT = 0.50
        ELSE
          IF (DELTAX.GE.CLIP) THEN
            PERCNT = BFACTR
          ELSEIF (DELTAX.LE.-1.0*CLIP) THEN
            PERCNT = SFACTR
          ELSEIF (DELTAX.GT.BIAS) THEN
            IF (OPTION.EQ.'S') THEN
              PERCNT = BFACTR
            ELSEIF (OPTION.EQ.'L') THEN
              PERCNT=(BFACTR-0.5)*((DELTAX-BIAS)/(CLIP-BIAS))
     1                      + 0.5
            ELSEIF (OPTION.EQ.'Q') THEN
              PERCNT = (BFACTR-0.5)*(((DELTAX-BIAS)/
     1           (CLIP-BIAS))**2)+0.5
            ENDIF
          ELSEIF (DELTAX.LT.-1.0*BIAS) THEN
            IF(OPTION.EQ.'S') THEN
              PERCNT = SFACTR
            ELSEIF(OPTION.EQ.'L') THEN
              PERCNT = (SFACTR-0.5)*((-BIAS-DELTAX)/
     1                 (CLIP-BIAS)) + 0.5
            ELSEIF (OPTION.EQ.'Q') THEN
              PERCNT=(SFACTR-0.5)*(((-BIAS-DELTAX)/
     1                (CLIP-BIAS))**2)+0.5
            ENDIF
          ENDIF
        ENDIF
C
        DLTSHR = PERCNT*ASSETS/Z(CNT) - SHARES
        FEE =  ABS(LOAD*DLTSHR*Z(CNT))
        DLTSHR = DLTSHR * (1.0-LOAD)
        CASH = CASH - FEE  - DLTSHR * Z(CNT)
        SHARES = SHARES +DLTSHR
        BROKER = BROKER + FEE
C
 1550 CONTINUE
 1900 ASSETS = SHARES*Z(CNT) + CASH
C
C****************************************
```

```
      WRITE(6,5000)BIAS,BFACTR,SFACTR,CLIP,ASSETS,
     1        SHARES*Z(CNT),BROKER
 5000 FORMAT(1H0,F5.3,3X,F4.2,3X,F4.2,3X,F5.3,3X,F9.2,3X,
     1    F9.2,3X,F7.2)
      GOTO 600
C
C**********************
 6000 CONTINUE
      END
```