TIME-SERIES FORECASTING TECHNIQUES FOR SCHEDULING OF MULTIPROCESSOR COMPUTER JOBS

A THESIS

Presented to

The Faculty of the Division of Graduate

Studies and Research

By

Albert Sleder, Jr.

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Operations Research

Georgia Institute of Technology

August, 1974

TIME-SERIES FORECASTING TECHNIQUES FOR SCHEDULING

OF MULTIPROCESSOR COMPUTER JOBS

Approved:

Donovan B. Young, Chairman

Douglas C. Montgomery

Russell G. Heikes

Date approved by Chairman: 8-27-74

ACKNOWLEDGMENTS

I would like to express my appreciation to my thesis advisor, Professor Donovan B. Young for his guidance, encouragement, and invaluable advice.

I would also like to thank the members of my reading committee, Professors Montgomery and Heikes, for their most helpful suggestions and assistance.

Special thanks go to Mrs. Carolyn Piersma for typing the thesis on such short notice.

To my wife, Martha, and son Alex, whose love and affection carried my spirits through the most difficult of times, I will never be able to adequately express the extent of my gratitude. This thesis is dedicated to them.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SUMMARY

In executive-request scheduling for increased throughput in a multiprocessor computer system, choice of a method of forecasting execution times is complicated by the high cost of tracing actual program tasks, by the difficulty of defining and obtaining a truly representative sample of jobs processed by a computer center, by the lack of theory for selecting appropriate forecasting methods for these series that have a special structure reflecting computer programming practices, and finally by uncertainty as to the cost/accuracy tradeoff in using the forecasts in a scheduling algorithm.

Previously, a 'level-reset' forecasting method developed by Young had been found by Raynor to be more accurate and less costly than standard forecasting methods, when the forecasts were used in Raynor's specific scheduling algorithm applied to a very limited sample of real program tasks. The present work extends Raynor's empirical sample, establishes a theoretical basis for forecasting (based on assumptions concerning piecewise constant time series and empirical verification of piecewise constant structure), derives extensions of level-reset forecasting, and empirically compares level-reset forecasting and extensions to alternative forecasting methods. An improved criterion for evaluating forecast errors is derived and applied. A less costly and perhaps more accurate version of Raynor's level-reset forecasting is developed and is recommended as the method of choice for scheduling of multiprocessors.

CHAPTER I

FORECASTING FOR MULTIPROCESSOR SCHEDULING

Today's computer industry stands at the threshold of a new and exciting generation of electronic computer systems, the multiprocessor computer. In the thirty years preceding 1974, the industry has proceeded from the vacuum tube, through the transistor, to the modern-day central processing units (CPUs) composed of modules of printed circuitry. The result has been a significant reduction in the size of computer systems, as well as an increase in both efficiency and reliability of such systems. The next logical step is to unite many of these modern CPUs into a complex system linked together by both hardware (physical equipment) and software (supervisory programs, data banks, etc.).

Such a system would have several inherent assets. First, there would be a consolidation of the large data files (subroutines, special libraries, etc.) that would otherwise have been duplicated in the separate system concept. Along with the multiplicity of the CPUs would be the replication of the many peripheral devices associated with a computer system. Such replication (which is being considered on a large scale [8][14][16][37]) would make it worthwhile to maintain an inventory of repair parts and probably an in-house repairman at the facility. This should conceivably reduce the down time on those devices, enhancing the efficiency of the entire computer system. Although M processors cannot do M times as much work as one processor, cost savings stem from the fact that far less than M times as much peripheral equipment is necessary. The savings are amplified by the fact that the cost of processors has decreased much faster than the cost of peripheral equipment [2].

Efficient design of a multiprocessor system presents challenging difficulties. The most significant is the need to assemble the system in such a way that all components are efficiently utilized. In other words, the jobs to be processed by the system must somehow be scheduled into each processor in such a way that the processors do not interfere with each other's operation. Madnick [23] showed that such interference, called multiprocessor lockout, is indeed a significant factor to be dealt with. For example, with no scheduling algorithm to reduce lockout, it was demonstrated under real operating loads that if there were 15 processors in the system, an average of one would be idle. The reason for this idleness is that the supervisor is busy assigning a job to another processor. The supervisor can schedule only one processor at a time. Any other processor needing the supervisor is put in a queue until the supervisor becomes available. An increase to 40 processors results in 19 idle processors, while 41 processors results in 20 idle. In other words, the 41st processor has zero marginal effectiveness! (See Figure 1.) Thus, before systems beyond the research level are produced, a scheduling algorithm must be developed to minimize mutual interference among the processors. The first steps have already been taken in this area. Most recently Pass [28] and Raynor [29] at Georgia Tech have pursued this matter and offer excellent references for the most up-to-date literature such

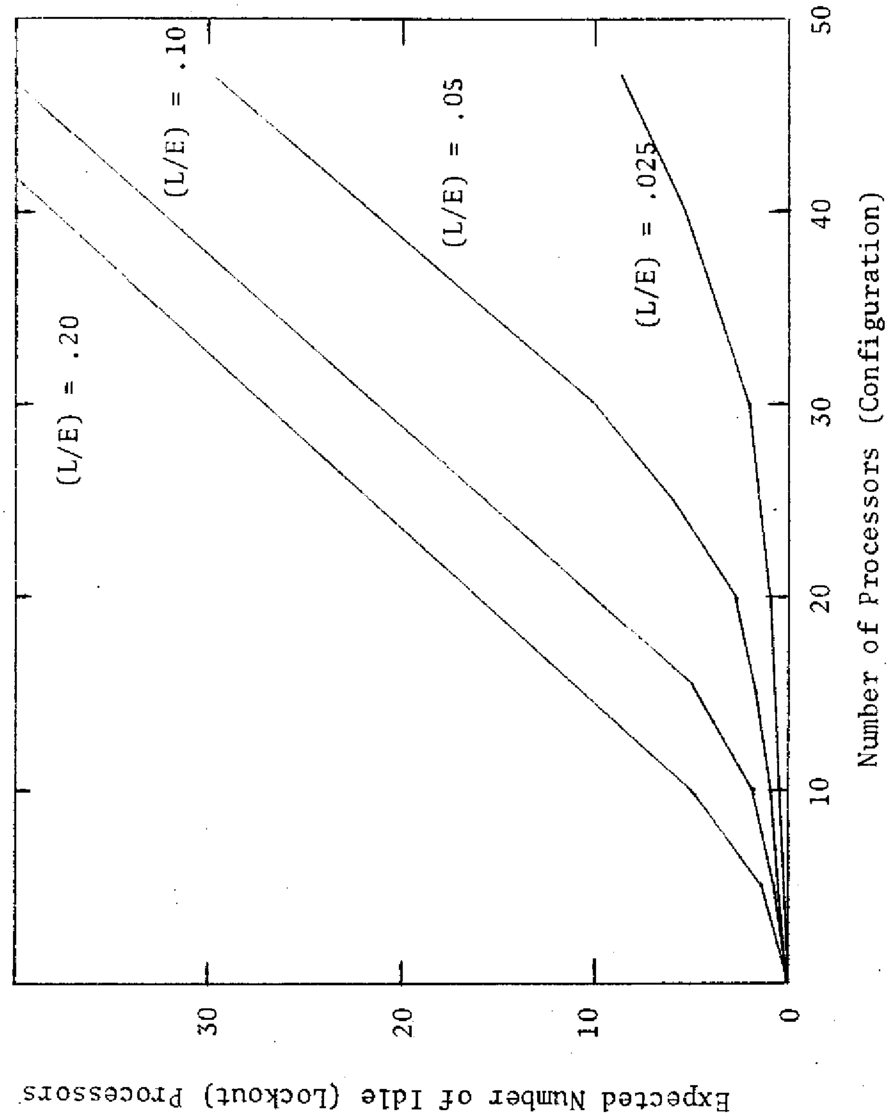


Figure 1. Idle Processors

as that of Lampson [19] and Sherman, Baskett, and Browne [32]. They also provide valuable initial results from which to continue development and refinement of the needed scheduling algorithms.

One of the necessary assumptions for the algorithm development is the assumption of being able to forecast the times between input and output (I/O) interrputs. These interrupts characterize the jobs generated by the system's workload. We will use the symbol ER (for executive request) interchangeably with I/O interrupts, following the terminology employed by the staff of the Georgia Tech computer center. It is not necessary for a program being computed by the system to be completed from start to finish. Instead the program is done in segments (jobs) which are separated by I/O interrupts. Forecasting accuracy was demonstrated to have a definite effect on the amount of work that can be processed through a multiprocessor system. Table 1 shows such effect when using the Raynor algorithm for scheduling in a multiprocessor environment [29].

Objective of the Research

Forecasting of the times between successive I/O interrupts is the subject of this research. Certain preliminary results obtained by Pass and Raynor will serve as the starting point for our research efforts. These preliminary results will be discussed in the following chapter as part of the survey of forecasting techniques.

It is the objective of this research to determine to what extent and precision it is possible to forecast times between successive I/O interrupts generated by actual computer programs. It is not enough to say we can forecast, we must know whether or not our

Table 1. Forecasting Errors Effect

Standard Deviation of Error Distribution*	Average Throughput	Percent Increase in Throughput
0	6.78	10.04
5%	6.73	9.24
10%	6.66	8.10
15%	6.57	6.64
20%	6.57	6.64
35%	6.53	5.99
50%	6.48	5.18

*As a percentage of the true value.

forecasts are acceptably accurate and if so at what cost (the forecasts themselves use computer time). Forecasts must be timely as well as accurate and efficient; for example, it is useless to forecast if the times between interrupts are smaller than the time it takes to forecast. In such a case the answer would arrive too late to be of any value.

Summary of the Chapters

Chapter II will present a survey of the literature as to the types of forecasting techniques currently employed today with emphasis on some of the results of Pass and Raynor. Chapter III will explain the specific techniques of forecasting that were examined. Also included will be a section on how the actual time series were generated, for the question of what kind of series best represents actual workloads at an operating computer center remains unresolved. Chapters IV and V will present the results and conclusions of the research and suggestions for further research.

CHAPTER II

SURVEY OF THE PREVIOUS RELATED WORK

Many examples of forecasting systems are found in the literature. Most of the current literature is concerned primarily with forecasting systems that have evolved from the basic writings of Brown [9] on moving-average and exponential smoothing techniques and Box and Jenkins [7] on linear filtering. Many efforts have been made to extend these techniques for more powerful use in specific applications in industry and business [5][15][18][31].

Need for Self-Adaptive Systems

In the context of the technical literature in forecasting, to forecast means to assign estimates of future values--forecasts--of a random variable whose values are assumed to constitute a non-stationary stochastic process. Forecasting systems vary as to what information is formally taken into account and as to the assumed structure of the stochastic process, but many forecasting techniques may be viewed as including a smoothing constant, $0 \leq \alpha \leq 1$, or its equivalent.

The choice of smoothing constant chosen is extremely important since regardless of the model chosen, the ability to detect changes in the time series depends on the value of α. If the constant is large, say close to one, more weight will be placed on the more recent observations. When it is close to zero, it will give more weight to the historical data. Exponential smoothing also requires an initial value of the smoothing statistics to start the smoothing process. Much of the literature concerns development of an adaptive technique, a system to adapt to changes in the time series and to correct for an improperly chosen initial smoothing constant. Wichern [36] at the University of Wisconsin showed that even when the proper model is used for a given time series, if an improper value of α is chosen, the variance of the forecast errors will be significantly underestimated. The result is not only to fail to minimize the variance of the forecast errors, but also to fail to get an accurate estimation of the actual variance.

Review of Some Self-Adaptive Systems

Let us now examine some systems that have been developed to try to deal with this problem of smoothing parameters. Such systems are called "self-adaptive" in that they examine themselves and make the appropriate change in the smoothing constant when the system appears not to forecast the monitored time series adequately. This often occurs when there is a large change in the underlying stochastic process. If the forecasting parameters were fixed it might take an unacceptably long time for the system to readjust itself.

Box [5][6], in his articles on evolutionary operations (EVOP) proposed a method of using a factorial experimental design such as that used in response surface analysis to determine when and how to modify the independent variables of an experiment or process to obtain a desired change in the dependent variable. Such a method consists of setting up the design in such a way that the effect of changing each variable can be determined and action taken according to established rules.

Roberts and Reed [30] developed a self-adaptive forecasting technique (SAFT) which combines exponential smoothing with a response surface analysis technique to test the forecast accuracy of various smoothing parameters in a forecasting model. The technique is a specific application of Box's evolutionary operations technique.

Chow [12] proposed a technique of establishing a high, normal, and low value of the smoothing constant to be utilized in the exponential smoothing technique. The constants are initially chosen arbitrarily, but are modified as the time series progresses. Whenever, on the basis of an error criterion, one of the "outer" forecasts turns out better than the normal forecast, the next period's forecast is made based on the new "best" value. At the same time new high and low values are introduced around the reset normal value. This is in reality a one-parameter version of the evolutionary operation design of Roberts and Reed.

Montgomery [25] has also used an evolutionary operation scheme for an adaptive forecasting system. However, he proposed the use of an orthogonal, first order experimental design called the simplex. His procedure involves the changing of the exponential smoothing parameters each period by the sequential application of the simplex design. A new simplex is determined each period by deleting the worst parameter combination (that which gives the worst forecast error) and creating a new point according to fixed relationships. These relationships generally create a point geometrically opposite of the deleted point. An example in two-space is shown in Figure 2.

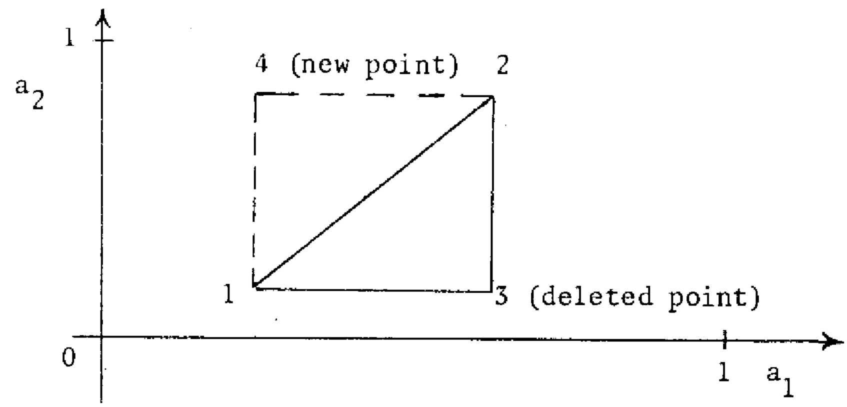


Figure 2. Montgomery's Simplex Design for Forecasting

Brown [9] proposed the use of either the tracking signal or the mean absolute deviation (used as an approximation of standard deviation) of the forecast errors as the criterion for monitoring the forecasting technique to determine when it goes out of control. The tracking signal is the sum of recent forecast errors, which, if the system is under control, should oscillate around a mean value of zero. If the signal significantly moves away from zero, the system is to be considered as out of control and corrections to the parameters are made.

Burgess [11] proposes an automatic adaptive system using the tracking signal as the out-of-control indicator. The smoothing parameter is defined as $\alpha = 1/(1 + M)$ where M is the number of time periods to the midpoint of an exponentially smoothed moving average. For each period that the system is in control, M is incremented by 1 up to a value of M = 20 (which corresponds to α of approximately .05). This heavily weights historical data when the system is in control.

When the system goes out of control, a constant value is subtracted from the current value of M. This effectively increases the value of α, putting more weight on the most recent information.

Trigg and Leach [35] proposed a method of equating the smoothing constant to the modulus of the tracking signal.

Pass [28] used a modification of double exponential smoothing which used a relative error (e_{t-1}) and a threshold value (τ) as the means of determining when the system is out of control.

$$e_{t-1} = \frac{\hat{x}_{t-1} - x_{t-1}}{t_{t-1}} \tag{1}$$

where $\hat{x}_{t-1}$ is the forecast of the actual observation x_{t-1}. If e_{t-1} is greater than τ and the sign of e_{t-1} is the same as the sign of e_{t-2}, it is assumed that the system was not responsive enough; α is changed by a small fixed increment according to appropriate rules.

Raynor [29] used a similar measure of error, but did not use it as a means of updating α. Instead, when it was determined that the system was out of control, the smoothed value used for the next forecast is reset to the value of the most recent observation. This is an example of the <u>level-reset</u> class of methods to be discussed in Chapter III. In equations we would write:

$$\hat{x}_t = \alpha x_{t-1} + (1-\alpha)\hat{x}_{t-1} \quad \text{when} \quad \frac{|x_{t-1} - \hat{x}_{t-1}|}{x_{t-1}} < \tau$$

$$= x_{t-1} \qquad \text{otherwise}$$

We are in effect setting α equal to one when out of control and equal to a predetermined value when in control.

Results of Raynor's Research

Results of comparison among Raynor's, Pass', current-observation forecasting (Raynor's with $\tau = 0$), and double moving average techniques indicated that Raynor's method surpassed the others in forecasting the times between ERs. Table 2 is from Raynor's work.

Table 2. Forecast Technique Comparison

Forecasting Technique	Average Percent of Forecasts within $\pm 15\%$ of the Observation
Double Moving Average	43.0
Pass' Method	44.5
Raynor's Method	74.4
Current Observation ($\hat{x}_t = x_{t-1}$)	62.5

This result is not unrealistic. It is not surprising that the $\tau = 0$ version of single exponential smoothing, which is merely current-observation forecasting, did well. Computers are built to handle repetitious data. The routines that accomplish this digestion contain loops which tend to cause times between ERs to form an approximately constant series with jumps from one level to another as we proceed from one loop to another. Raynor's results suggest our research should include methods of adapting a constant forecasting scheme that resets data to the new level when the process is out of control.

With this method we hope to reduce the time it takes for our forecasting system to reset to the new level and thus increase forecasting accuracy.

We will, therefore, concentrate on a constant model and utilize techniques to determine when to reset to a new level. Methods for adapting both single exponential smoothing and moving average will be tested. Moving average will be discussed more fully in the next chapter.

CHAPTER III

DESCRIPTION OF THE RESEARCH

Raynor's work [29] showed that there exists at least one scheduling algorithm, using forecasts of times between successive I/O requests, that is capable of significantly increasing throughput in a multiprocessor computer system. For his scheduling algorithm, which considered CPU time in rather coarse blocks of 200μ-sec, several forecasting methods were found to perform adequately. He reported a version of "level-reset" forecasting as both lowest-cost and highest-benefit for the programs he ran and the scheduling algorithm he used, but two important considerations were beyond the scope of his study. First, Raynor did not make a systematic study, either theoretical or empirical, of appropriate forecasting methods, and second, his sample of programs was so small as to leave in doubt whether they were typical of programs submitted to a computer center.

The present research attempts to make a systematic study of available forecasting methods for times between successive I/O requests. It was hoped the results would (1) either provide a better forecasting method or verify Raynor's selection, and (2) provide additional samples of typical I/O-request time series. This work should be useful for scheduling by any method (Raynor evaluated forecasting methods only as applied to his own scheduling algorithm).

The research consisted of three parts: (1) data generation from typical programs submitted to the Georgia Tech computer center,

(2) theoretical work to derive appropriate forecasting techniques, and (3) evaluation of the forecasting methods.

Data Generation

All the electronic calculations for this research were carried out on the Univac 1108 computer. Within the Univac System Library, there exists a program trace routine called SNOOPY. SNOOPY provides an account of every instruction executed and its effect. Univac affiliated programming personnel are familiar with this trace routine and are capable of modifying the routine's output in several ways.

Figure 3 below is representative of the type of information that may be generated as output by SNOOPY. The first line of output indicates that a command from the program called TEST1 is beginning to

```
1     TEST1,$(1)
           076   002           FM
           076   002           FM
           001   000           SA
           074   013     J     LMJ
2     NEXP2$,$(1)
           006   001           SX,H2
           005   000           SZ
           010   016           LA,U
           010   016           LA,U
      NEXP6$,$(1)
3          073   012           LSSL
           074   004     J     J
           055   000           TG
           055   000     S     TG
-------------------------------------
 000001000001
-------------------------------------
4         0015                 ER
```

Figure 3. SNOOPY Output

be processed (traced) by SNOOPY. The line could be an equation, logic statement, or any other FORTRAN instruction. The second type of output line is one that represents a breakdown of the first line into computational jobs such as addition or subtraction. For example, the equation Y = X**2 + 2*W*X + W**2 would be broken down into six jobs of exponentiation, addition, and multiplication. This type of output is expressed as the second underlined line in Figure 3. Under each of the two previously mentioned outputs are found a third type (numbered 3) which indicates every individual step the computer goes through to solve the problem it is given. Output that would normally result from the program being traced is separated from the SNOOPY output by a dashed line (-------) above and below. By examining the type-one or type-three lines, the researcher can determine how far SNOOPY has progressed through the traced program. The final line in the figure is representative of that output generated when an ER is initiated by the computer.

All of the output mentioned can be turned off by program modification of SNOOPY. This can be done by sending the information to a subroutine to be analyzed rather than to memory to be printed in the output, or by simply flagging the output so that it is not routed to any location. In the present research, a subroutine was written to examine each line as it was sent to determine the time it took to execute each instruction. The times are determined according to specific rules found in the Exec 8 Handbook distributed by Univac. A running total of time is maintained until an ER line is sent. The time on hand is then printed and the running total reset to zero to begin the process again until the next ER. This continues until the program being traced has completed its run or the maximum allowable computation time on the computer has been reached.

The exact method of setting up a program for the use of SNOOPY is found in Appendix 1A. A copy of the subroutine used is found in Appendix 1B. A copy of SNOOPY is too lengthy to be contained herein, but is contained in the Univac Executive 8 Library.

The system of routines and subroutines offers an excellent means of obtaining accurate times between I/O interrupts. However, the necessity of screening a line for many possible values and the movement of logic into and out of many subroutines utilizes large quantities of CPU time. As a result, one must have access to large amounts of CPU time for at the maximum run time all computations cease whether or not the process is completed. Thus one must be careful to insure enough run time is used to complete at least one full cycle of the program as a minimum and to insure that an adequate number of times are generated. This generation of an adequate number of times is important for the proper analysis of any forecasting technique that is proposed. In general, one should attempt to get a minimum of 100 times in the series. With less data, it would be presumptuous to speak of analyzing its structure as a non-stationary stochastic process.

Piecewise Constant Time Series

Multiprocessor computer systems are designed for flexible simultaneous handling of many computing jobs submitted by many users, such as is the situation at large university computing centers. Experience shows that the available job mix is generally dominated by tasks from "large" programs full of repetitive "number crunching" [22].

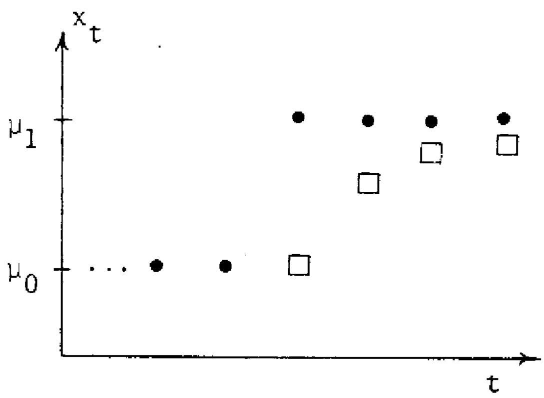
Large programs exhibit a strongly repetitive structure consisting of loops, in each of which an identical set of instructions is executed many times. The most commonly encountered loop structure contains one executive request in each execution of the loop (for example, one READ statement or one WRITE statement), and uses approximately a constant time for the execution between successive requests. This motivates the piecewise constant structure of the series of execution times expected in processing a program.

Variations among the successive execution times in a single loop are generally of two distinctive kinds. There are small highly-autocorrelated fluctuations caused by very small variations in the time required for each arithmetical, logical or transferral operation. These variations are dwarfed by program logic variations within a loop, which are also usually highly autocorrelated and which can range from less than 1.0μ-sec to any amount whatsoever. Conditional control transfers (IF statements) are the most commonly encountered program logic variations found within a loop. The computation time between two executive requests varies anywhere from less than 1μ-sec up to about 10,000μ-sec, but the variability cannot be shown to increase significantly with computation time. This independence of variability and level has convenient implications in choosing forecast parameters. Its cause is apparently that the main difference between a longer interval between I/O statements and a shorter interval is that the longer interval is packed with more number crunching of almost zero variance. In other words, this phenomenon is apparently an artifact of programming practice.

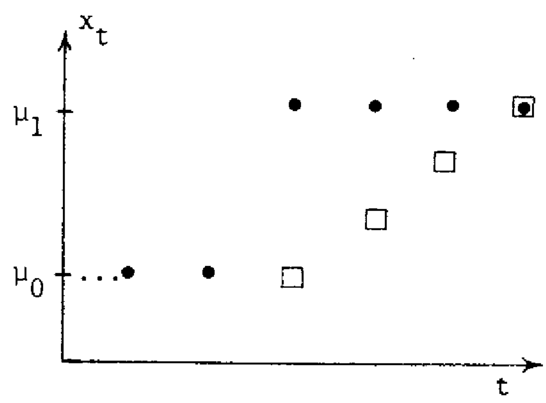
The following arguments are adapted from Young [39].

Let us postulate a <u>piecewise constant</u> time series, in which each observation x_t is either (Event A) a further observation from the current constant process whose mean is μ_o or (Event A') the first observation from a new constant process whose mean is μ_1. We assume that the standard deviation of x_t under Event A, denoted σ_A, is far smaller than $|\mu_1 - \mu_o|$, i.e., that the variation of observations in any one single constant process is far smaller than the variation of observations from two different processes.

In forecasting a piecewise constant series there are obviously two separate kinds of error: ordinary forecast errors (A-errors) within a single process and much larger process-change errors (A'-errors) incurred when the process changes levels from μ_o to μ_1. From our assumption $\sigma_A << |\mu_1 - \mu_o|$, we see that avoidance of A'-errors is paramount, and hence that standard methods such as exponential smoothing, moving average and linear filtering will incur large errors. In fact, exponential smoothing forecasts with smoothing constant α will incur a total A'-error approaching $|\mu_1 - \mu_o|(1-\alpha)/\alpha$ in the first few forecasts after a change in level from μ_o to μ_1, and moving average forecasts of length N will incur a total A'-error approaching $|\mu_1 - \mu_o|(N+1)/2$. This is easily seen by referring to Figure 4, where ● denotes an observation with the smaller A-error suppressed and □ denotes a forecast calculated one period earlier:



Exponential smoothing
with α = .6



Moving average
with N = 3

Figure 4. A'-errors in Forecasting a Piecewise Constant Time Series by Exponential Smoothing and Moving Average

To reduce the large A'-error in forecasting a piecewise constant time series to its theoretical minimum of $|\mu_1 - \mu_o|$, which corresponds to immediate recovery, we can set $\alpha = 1$ in exponential smoothing or set $N = 1$ in moving average forecasting, in either case obtaining the simple forecasting method $\hat{x}_t = x_{t-1}$, i.e., the forecast calculated for time t equals the observation obtained at time t-1. Raynor [Ref. 29, page 112] found this method to outperform all others for multiprocessor scheduling except the level-reset method to be described below.

A natural extension, after reducing A'-error to its theoretical minimum, would be to attempt to reduce A-error without sacrificing the feature of immediate recovery from a process level change. From our assumption $\sigma_A << |\mu_1 - \mu_o|$, we can almost always distinguish whether an observation x_t signals Event A or Event A'; when $|x_t - \hat{\mu}_o|$

is small enough to be comparable to σ_A, Event A is likely, otherwise Event A'. (Here $\hat{\mu}_o$ represents the current estimate of the process level.) If Event A' is indicated, the next forecast should certainly be x_t, which is the best and only estimate available for the new level μ_1; on the other hand, if Event A is indicated, we are free to forecast by any appropriate method that assumes continuation of a constant process. Thus a promising class of forecasting methods for piecewise constant series includes all those constant-model methods that reset the level of the forecast when an outlying observation is received. Members of this class can be called level-reset methods.

Level-Reset Forecasting

Level-reset forecasting differs from the variety of useful methods that dynamically adjust the smoothing constant. The latter methods apply especially well to highly autocorrelated series that exhibit changes in variability, and they focus mainly on reacting to changes in the relative sizes of permanent and temporary errors. By contrast, level-reset forecasting is specifically intended for piecewise constant time series, in which permanent errors are far larger than temporary errors. Application of both methods to a piecewise constant series is shown in Figure 5. On the left, the level-reset method forecasts the new level after a large change. On the right, following Brown [Ref. 9, page 296, and proprietary IBM forecasting software], α is reduced after two successive outliers, accelerating the recovery. Of course, the simple forecast $\hat{x}_t = x_{t-1}$ is a special case of both methods.

The level-reset forecasting method is as follows:

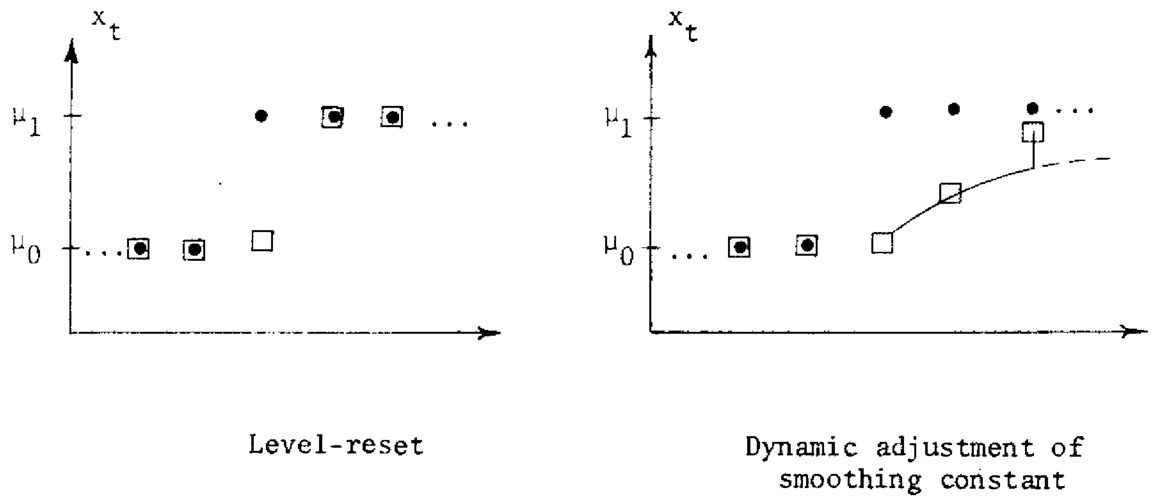


Figure 5. A'-errors in Forecasting a Piecewise Constant Time Series by Level-reset and by Dynamic Adjustment of the Smoothing Constant

$$\hat{x}_t = \begin{cases} \alpha x_{t-1} + (1-\alpha)\hat{x}_{t-1} & \text{if} \quad g(x_{t-1},\hat{x}_{t-1}) < T \\ x_{t-1} & \text{otherwise.} \end{cases} \tag{1}$$

Level-reset forecasting has two parameters: α is the usual smoothing constant used when the process is judged not to have changed levels, and T is a "gate" or maximum error function that represents the highest value of the current forecast error function $g(x_{t-1},\hat{x}_{t-1})$ that is considered not to signal a level change. In the definitions to follow, g is an increasing function of forecast error, and is also normalized so that T = 0 means "always reset" ($\hat{x}_t = x_{t-1}$), and T = ∞ means "never reset" (exponential smoothing).

There are three forms of the forecast error function $g(x_{t-1},\hat{x}_{t-1})$ of special interest. Raynor [29] and Pass [28] have used a relative error (or percentage error if expressed in percentage), so that $g(x_{t-1},\hat{x}_{t-1}) < T$ in Equation 1 becomes specifically

$$\frac{|x_{t-1}-\hat{x}_{t-1}|}{x_{t-1}} < T \tag{1a}$$

Relative error is meaningful in the context of using the forecasts for scheduling, but its use introduces a bias that makes the parameter T difficult to choose; as a matter of empirical fact, large relative errors are rare when x_t is large and common when x_t is small, so that a given value of the gate T cannot be satisfactorily related to the probability that an error signals a change in level.

From a probabilistic point of view it would seem more logical to use the relative squared error:

$$\frac{(t_{t-1} - x_{t-1})^2}{x_{t-1}} < T \tag{1b}$$

The relative squared error criterion can be justified by assuming the execution time to be a sum of independent execution times. However, computer programming practices seem to favor loops that contain only one or two highly variable statements (such as conditional control transfers), with the remainder being made up of number-crunching statements with very low variance. Thus in actual practice a long loop actually has about the same execution-time variability as a short one, leading to the most truly appropriate error function for forecasting execution times:

$$|x_{t-1} - \hat{x}_{t-1}| < T \tag{1c}$$

The experimental work in the present study uses level-reset forecasting with two error functions: that of Inequality 1a for comparison with previous work, and the more appropriate one of Inequality 1c. (The error function of Inequality 1b would be applicable for piecewise constant time series in more general contexts, but it is not useful here.)

Evaluation of Forecast Errors

In earlier work [Ref. 28, Ref. 29] forecasts were evaluated directly in terms of the increase in work throughput that was achieved by scheduling based on the forecasts. From Raynor's empirical results given in Table 1, Chapter I, perfect forecasting gave a 10 per cent increase in throughput, "ballpark" forecasting (68 per cent of the forecasts falling between half and twice the true execution time) gave a 5 per cent increase in throughput, and of course completely random forecasting would have given no increase in throughput. Such results suggest that the usual evaluation of forecasts on the basis of variance of forecast error is quite inappropriate in this application context. The paradox of variance versus usefulness is illustrated repeatedly in the six actual time series studied herein. The variance depends most strongly on the largest errors whereas the usefulness depends most strongly on the smallest errors.

Figure 6 shows a time series (with A-errors suppressed) illustrating a type-1 pathology which is the commonly occurring case of a piecewise constant time series interrupted by one outlier. The observations (●) are forecast by level-reset (□) and exponential smoothing (△); parameters of the level-reset forecast are $0 < \alpha < 1$,

$0 < T < \mu_1 - \mu_0$, $|x_{t-1} - \hat{x}_{t-1}| < T$; the exponential smoothing constant is $\alpha = .5$; and with the chosen parameters Raynor's empirical results would predict roughly an 8 per cent increase in throughput by either method.

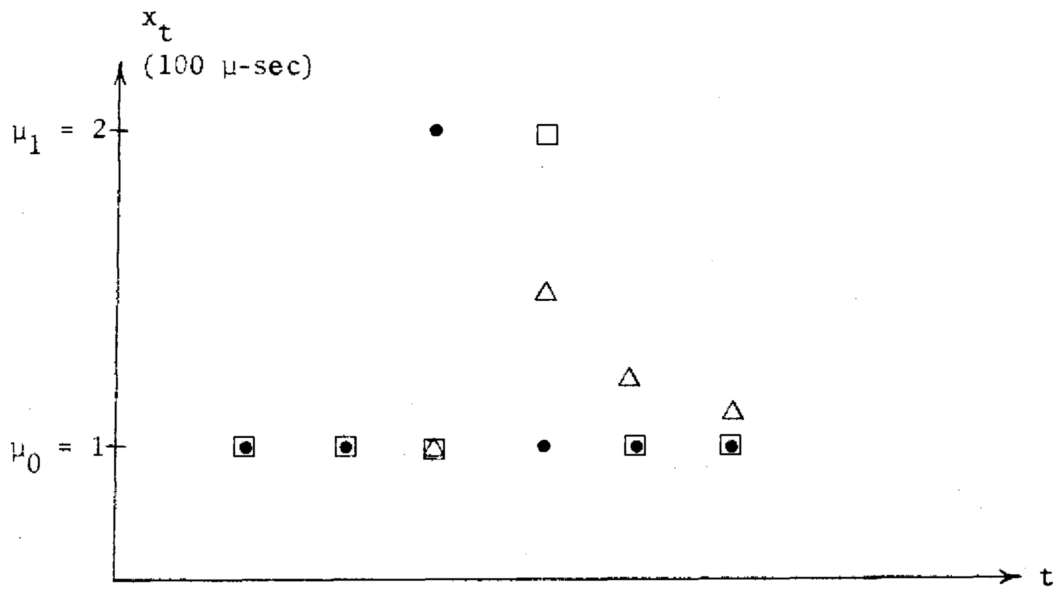


Figure 6. A'-errors in Forecasting a Piecewise Constant Time Series with a Type-1 Pathology, Using Level-reset and Exponential Smoothing

Directly from Figure 6 we can calculate the variance of forecast errors, which for the six observations shown is $(0 + 0 + 1^2 + 1^2 + 0 + 0)/6 = 2/6$ with level-reset forecasting and $(1 + .25 + .0625 + .015625) = 1.33/6$ with exponential smoothing. If we compare mean absolute deviations, we get 2/6 for level-reset forecasting and 1.875/6 for exponential smoothing. Since the forecasts were chosen specifically as those yielding approximately equal usefulness, we can conclude that unfortunately neither variance nor mean absolute deviation gives an appropriate measure of forecast usefulness.

Raynor [Ref. 29, page 112] used the average percentage of forecasts lying between 85 per cent and 115 per cent of the true value as his measure of forecast performance. This criterion was apparently selected over variance, over mean absolute deviation, and over other functions of relative error for its ability to rank the tested forecasting methods in the same order as the throughput increases obtained by their use in scheduling. It is uncertain whether this criterion would be appropriate when used in conjunction with scheduling algorithms other than Raynor's. Certainly the bias of relative error, as discussed earlier, suggests that a criterion based on some absolute rather than relative error would be more appropriate. For discrete scheduling in blocks of W μ-sec, a criterion that suggests itself is the <u>percentage of forecasts with error less than W μ-sec</u>. Under Raynor's scheduling algorithm, this criterion at W = 200 μ-sec gives the approximate percentage of <u>essentially perfect forecasts</u>--those where the actual execution time falls within one 200-μ-sec block the forecast.

Generally, errors in smaller ranges (see Table 1) should be weighted more heavily in ranking forecast methods than errors in larger ranges. The question of exactly what weights to give to errors in various ranges can be sidestepped, as the actual results reported in the next chapter fortunately rank various methods in the same order for all values of W small enough to provide significant improvements in scheduling (although variance, with its overwhelmingly large weighting of the largest errors, gives rankings that differ).

Description of the Adaptive Systems Tested

The methods tested were based, as mentioned previously, on an adaptive system that resets the past data to the new level (level-reset) of the constant model. Both moving average and single exponential smoothing techniques were modified to do this. Each of the techniques tested under each of the two main categories differ from the other only in the rules by which we determine whether or not to reset to the new level.

Standard Constant Model Techniques

As a reference point we begin by using a single exponential smoothing technique in which the value of the smoothing constant α is examined at six levels. We use exponential smoothing since we know that the expected value of the smoothed value is equal to the expected value of the coefficient of a constant model (see below). In single exponential smoothing we express the next forecast by

$$S_t(x) = \alpha x_t = (1-\alpha)S_{t-1}(x) \tag{2}$$

where α = the smoothing constant

$S_t(x)$ = the smoothed value of x at time t

x_t = the observation of x at time t

In general form we have

$$\begin{aligned} S_t(x) &= \alpha x_t+(1-\alpha)[\alpha x_{t-1}+(1-\alpha)S_{t-2}(x)] \\ &= \alpha x_t+\alpha(1-\alpha)x_{t-1}+(1-\alpha)^2[\alpha x_{t-2}+(1-\alpha)S_{t-3}(x)] \qquad (3) \\ &= \alpha x_t+\alpha(1-\alpha)x_{t-1}+\alpha(1-\alpha)^2 x_{t-2}+\ldots+\alpha(1-\alpha)^n x_{t-n}+\ldots+(1-\alpha)^t x_0 \end{aligned}$$

$$S_t(x) = \alpha \sum_{k=0}^{t-1} (1-\alpha)^k x_{t-k} + (1-\alpha)^t x_0 \tag{4}$$

That is, $S_t(x)$ is a linear combination of all past observations. The expected value of $S(x)$ is shown below.

$$E[S(x)] = \sum_{k=0}^{\infty} \beta^k E[x_{t-k}] \tag{5}$$

$$= E[x]\alpha \sum_{k=0}^{\infty} \beta^k = \frac{\alpha}{1-\beta} E[x] = E[x] \tag{6}$$

since $1-\beta = \alpha$.

Since the expectation of the smoothed value is equal to the expectation of the data, we have a method of estimating a value of our constant model.

A moving average of length N is similar to exponential smoothing. In this case rather than weighting the past observations geometrically, the N most recent observations are given a weight of 1/N and the remaining observations a weight of zero. The moving average is computed as follows:

$$M_t = M_{t-1} + \frac{x_t - x_{t-N}}{N} \tag{7}$$

where M_t is the current moving average

M_{t-1} is the previous moving average

x_t is the current observation

x_{t-N} is the observation N periods ago

Level-Reset Techniques

Two modifications of single exponential smoothing were developed to determine when the system goes out of control. The first method is that developed by Young (Raynor's best method) which consists of resetting to the new level when the latest observation is outside some specified percentage limit. We express this modification as

$$\hat{x}_t = \begin{cases} \alpha x_{t-1} + (1-\alpha)\hat{x}_{t-1} & \text{if } \dfrac{|x_{t-1} - \hat{x}_{t-1}|}{x_{t-1}} < \tau \\ x_{t-1} & \text{otherwise} \end{cases} \qquad (8)$$

This is the same method derived earlier herein from theoretical considerations assuming a piecewise constant time series, and given in Equation (1) and Inequality (1a). When the system is out of control we wish to reset to the new level and then continue smoothing at some fixed value of α until the system goes out of control again. Table 3 demonstrates this technique with $\tau = .5$ and $\alpha = .1$.

Table 3. Example of SAES Method ($\tau = .5$, $\alpha = .1$)

t	x_t	$\hat{x}_{t-1}$	UL (upper limit)	LL (lower limit)	In Control?
...					
46	110.0	100.0	150.0	50.0	yes
47	110.0	101.0	151.5	50.5	yes
48	50.0	101.9	152.85	50.95	no
49	52.0	50.0	75.0	25.0	yes

Graphically we would have

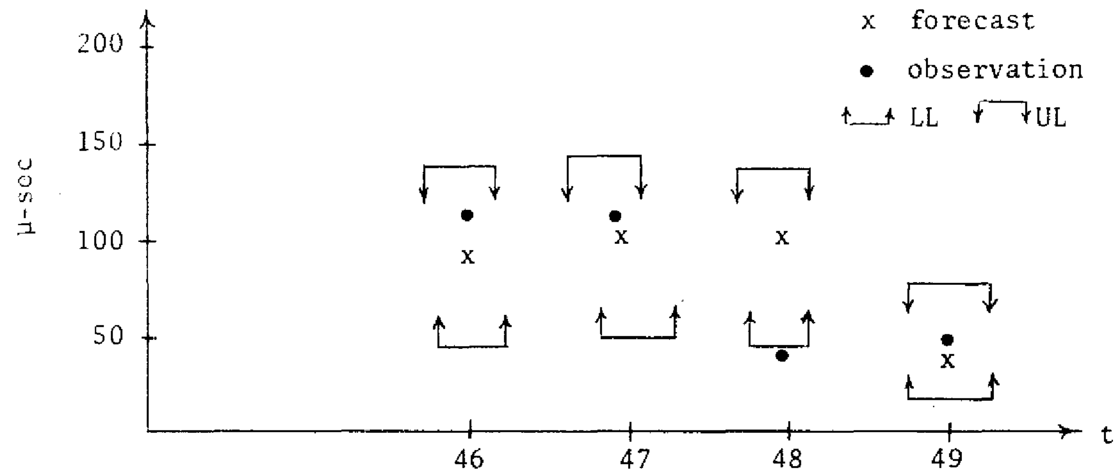


Figure 7. Graphical Representation of Table 3.

The second modification is similar to the first except that rather than setting $|x_t - \hat{x}_{t-1}|/x_{t-1} < \tau$ we set the criterion as $|x_t - \hat{x}_{t-1}| < \Delta$ where Δ is some fixed constant. That is, rather than changing the width of the acceptance region according to the time level, we will keep the region a fixed width at all levels.

Two rules were used to set the acceptance region for the two moving average level-reset methods. First a percentage rule similar to SAES was used. The moving average was computed as follows:

$$M_t = \begin{cases} \dfrac{\sum_{k=0}^{N-1} x_{t-k}}{N} & \text{if} \quad \dfrac{|x_{t-1} - \hat{x}_{t-1}|}{x_{t-1}} < \tau \\ x_{t-1} & \text{otherwise} \end{cases}$$

Calculations would proceed as in Table 4.

Table 4. Example of SAMA Method ($\tau = .1$)

t	x_t	Total	$\hat{x}_t$	UL	LL	In Control?	N_{old}	N_{new}
46	...	1000	100	...	...	yes	9	10
47	106	1106	105.45	110.0	90.0	yes	10	11
48	90	90	90	115.9	94.9	no	11	1

The second level-reset moving average consists of the rule in which the acceptance region is of a fixed width no matter at what level the time series is located. The only difference between this method and the second modification for exponential smoothing is the substitution of moving average in place of exponential smoothing. Thus the six methods used to forecast the real time series were:

1. Single Exponential Smoothing (ES)
2. Single Moving Average (MA)
3. Self-Adaptive Exponential Smoothing (SAES(τ))
4. Self-Adaptive Moving Average (SAMA(τ))
5. Self-Adaptive Exponential Smoothing (SAES(Δ))
6. Self-Adaptive Moving Average (SAMA (Δ))

Description of the Time Series Used

The question of what kind of series best represents the actual workloads at an operating computer center remains unanswered. No one computer program or set of programs has been developed that is representative of the majority of programs processed at a computer center. Thus the time series were generated from a random sampling of programs in an attempt to reduce bias of the results of the research.

Unfortunately, due to computer time limitations, we were somewhat restricted in that the programs chosen had to be of fairly short execution time themselves (that is, when not being traced). Also, due to the number of observations (I/O times) needed, the programs had to generate considerable input and output in a short run time.

However, within these restrictions, it is felt that a representative sample was achieved of the types of programs processed at the Georgia Tech computer center. No two programs were written by the same person, thus eliminating the possible bias of results due to one person's programming technique. Also, the six programs used were accumulated from five different schools (academic departments) at Georgia Tech. This should help eliminate duplication of possible types of problems that might be processed by the computer center.

Time Series 1 (COBOL)

Time series 1 (TS-1) was generated by a COBOL program of the types employed by students in the School of Industrial Management at Georgia Tech. This type of program is similar to those used by the business world and would be commonly used at a central computer facility used by many businesses. Figure 8 is a graph of this time series.

Time Series 2 (DIFFER)

The second time series (TS-2) was generated from a program written by a mathematics student. This program was used to examine two methods for approximating a differential equation. This program used a FORTRAN FUNCTION which is similar to a FORTRAN subroutine in its use. The graph of this time series is Figure 9.

Time Series 3 (METHANE)

A chemistry program, comparing several techniques for determining the pressure of methane gas at several temperatures, was used to generate the third time series (TS-3). This program read no input and contained one basic DO LOOP for incrementing the temperature. Figure 10 depicts this series of times.

Time Series 4 (OUT-OF-KILTER)

Time series 4 (TS-4) was generated from the OUT-OF-KILTER algorithm program from the School of Industrial and Systems Engineering program library. This program is representative of the linear programming problems found. The program reads in all its data, has several DO LOOPS (some within the loop of other DO LOOPS) and prints all of its output at one time at the end of the program versus at each iteration calculated by the program. Figure 11 is a plot of the times from this series.

Time Series 5 (SIM)

A FORTRAN simulation was the program used to generate the fifth series (TS-5). It is representative of programs written by students in the Information and Computer Science Department at Georgia Tech. This program specifically describes the operation of a computer system designed by the programmer. This program differs from programs one and two in that it contains several FORTRAN subroutines. Time series five is depicted in Figure 12.

Time Series 6 (NLS)

The sixth time series (TS-6) was generated from a program that conducted a simple coordinate search of a non-linear programming problem in industrial engineering. This is a simple, repetitious program that reads in the initial data and proceeds to calculate until specific criteria are met. Each calculation is printed as the program progresses. It contains no standard DO LOOP, but does repetitious operations due to IF statements that recycle when specified criteria are not met. Another feature of this program is the additional END = ____ statement within the READ command that abruptly terminates the program if there is no more input data. This again is another instance where a DO LOOP was not used but the program cycles are similar to those in a DO LOOP. Figure 13 is a graph of the time series.

Where time series (TS-1 and TS-5) were available from earlier work by Raynor [28, page 104], they were given in units truncated down to the next lower 200 μ-sec. These were randomized by replacing each observation x_t by $(x_t + R)200$, where R is a pseudo-random variate from a uniformly distributed population on the interval (0,1). This allowed approximate calculation of forecast errors within the range of 200 μ-sec. Of course, all results depending on errors in this range were checked for consistency with errors in larger ranges, because the randomization could introduce a bias in the smaller range. Appendix 3 contains listings of the times for each of the six time series.

Visual examination of each of the time series provides us with two useful conclusions. First, time series have specific structure that can be exploited in forecasting. Basically, all the programs displayed varying degrees of the piecewise constant structure mentioned previously. It was possible to relate the individual time series observations to programming statements in all time series. From doing this, one obvious conclusion was that type-1 pathologies (one outlier within a series) could often be avoided by improved programming practice. The large errors at the beginning of the OUT-OF-KILTER program were a result of unnecessary line skipping between lines of output as were the large deviations in the non-linear search program. Corrections to programs such as these would remove those small line skip interrupts, which add nothing in the way of useful information to the programmer and cause the program to compute longer because of (1) the additional commands necessary for output of a blank line, and (2) the need to reschedule even this small task since it is an I/O-interrupt which breaks the program into even smaller jobs. The second conclusion is that variance of times is not related to the times themselves (that is, their level). There is no noticeable significant increase in variance of the times with an increase in time level. The programming practices mentioned on pages 17-18 explain this phenomenon. The concept of relative error is not really meaningful. In fact, as was demonstrated, unnecessary forecast errors are encountered when the level is very low or very high, since the acceptance region is too narrow or too wide, respectively.

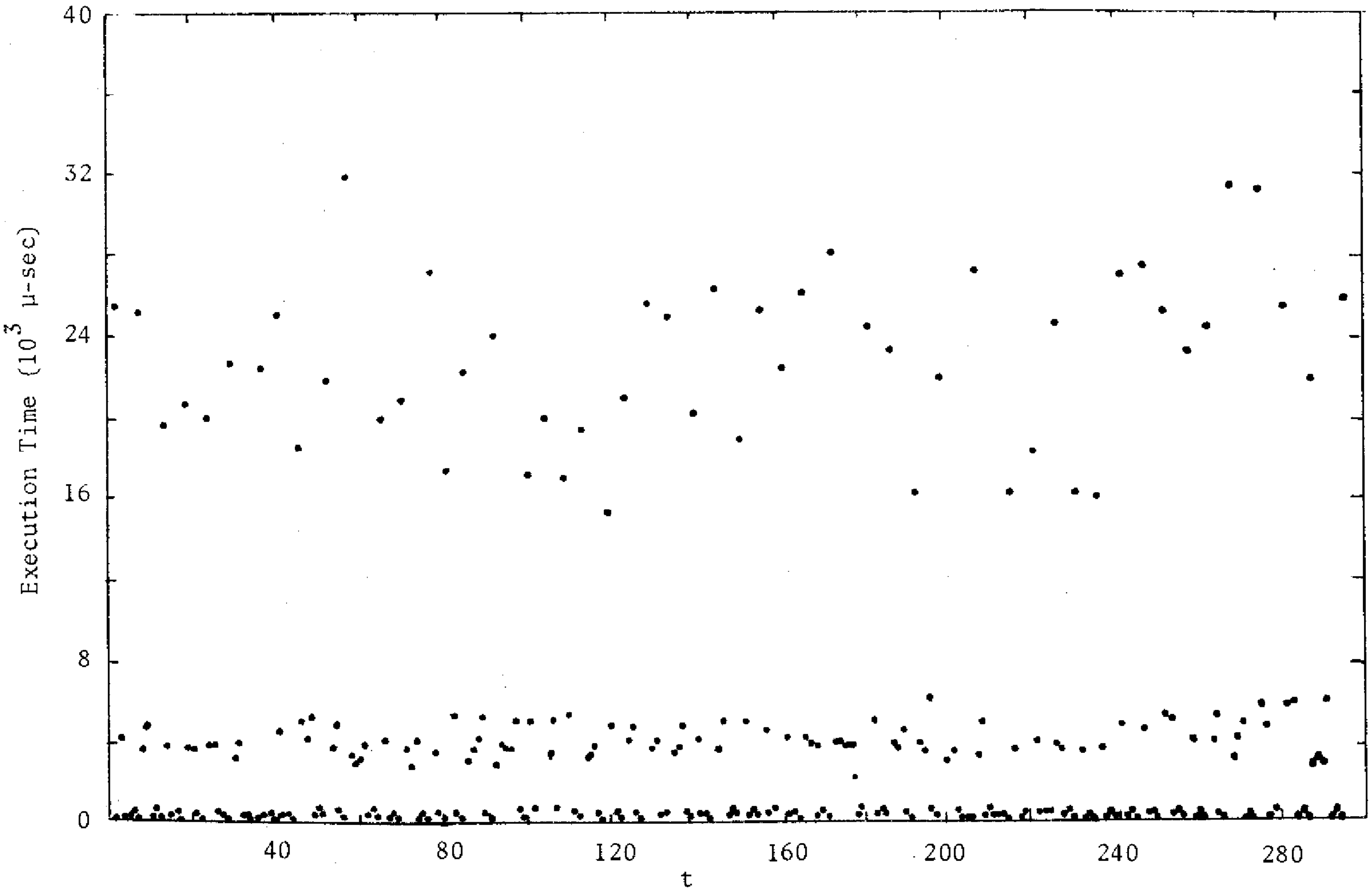


Figure 8. Time Series I

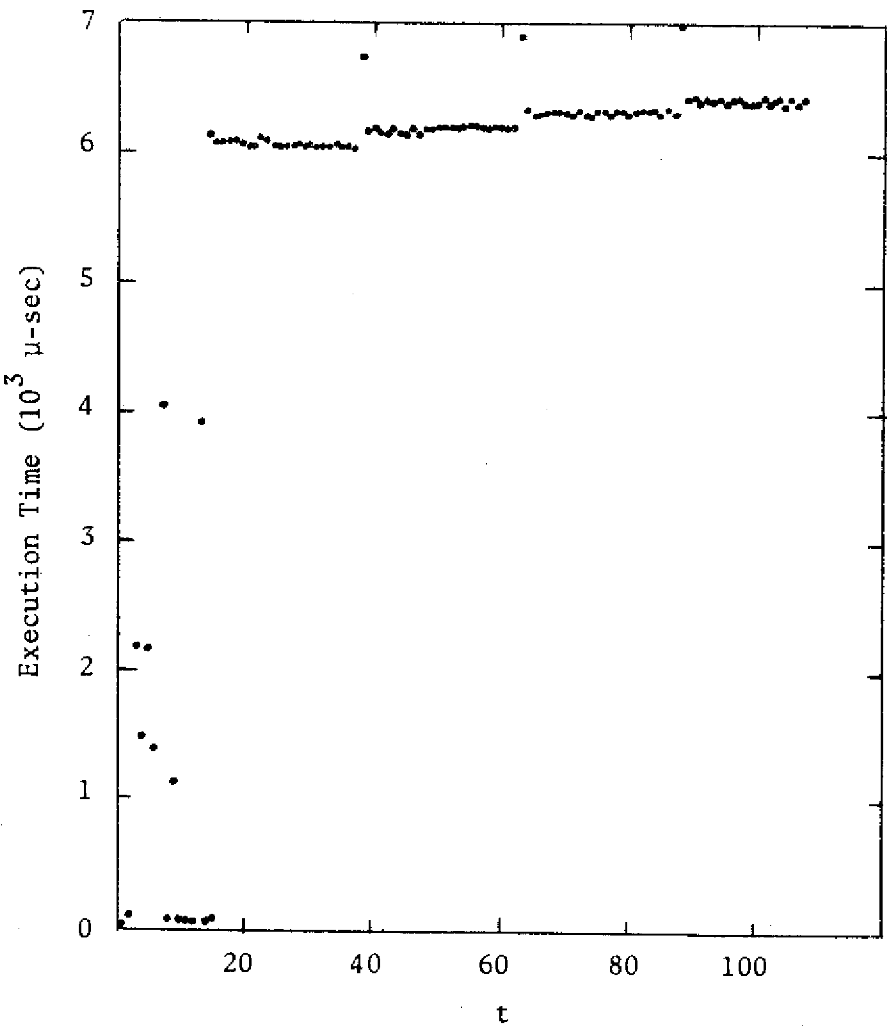


Figure 9. Time Series 2

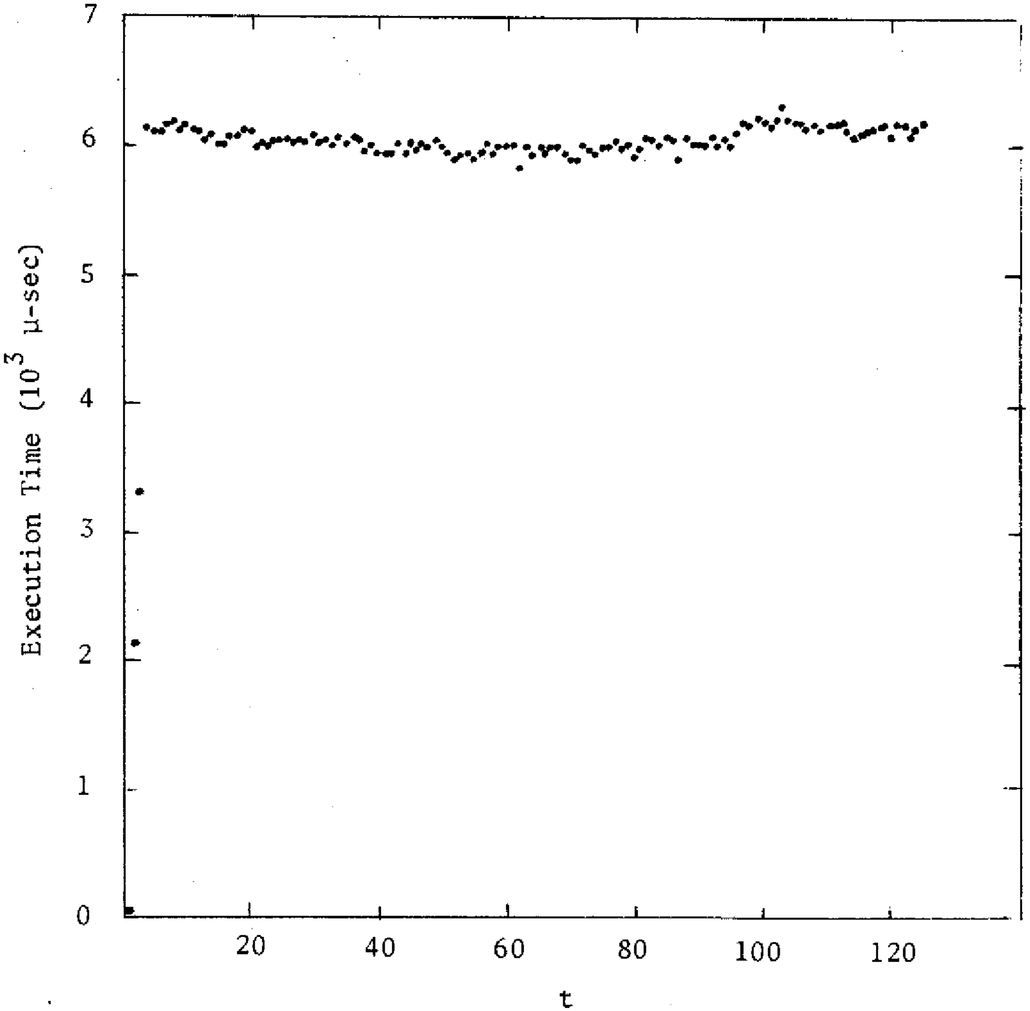


Figure 10. Time Series 3

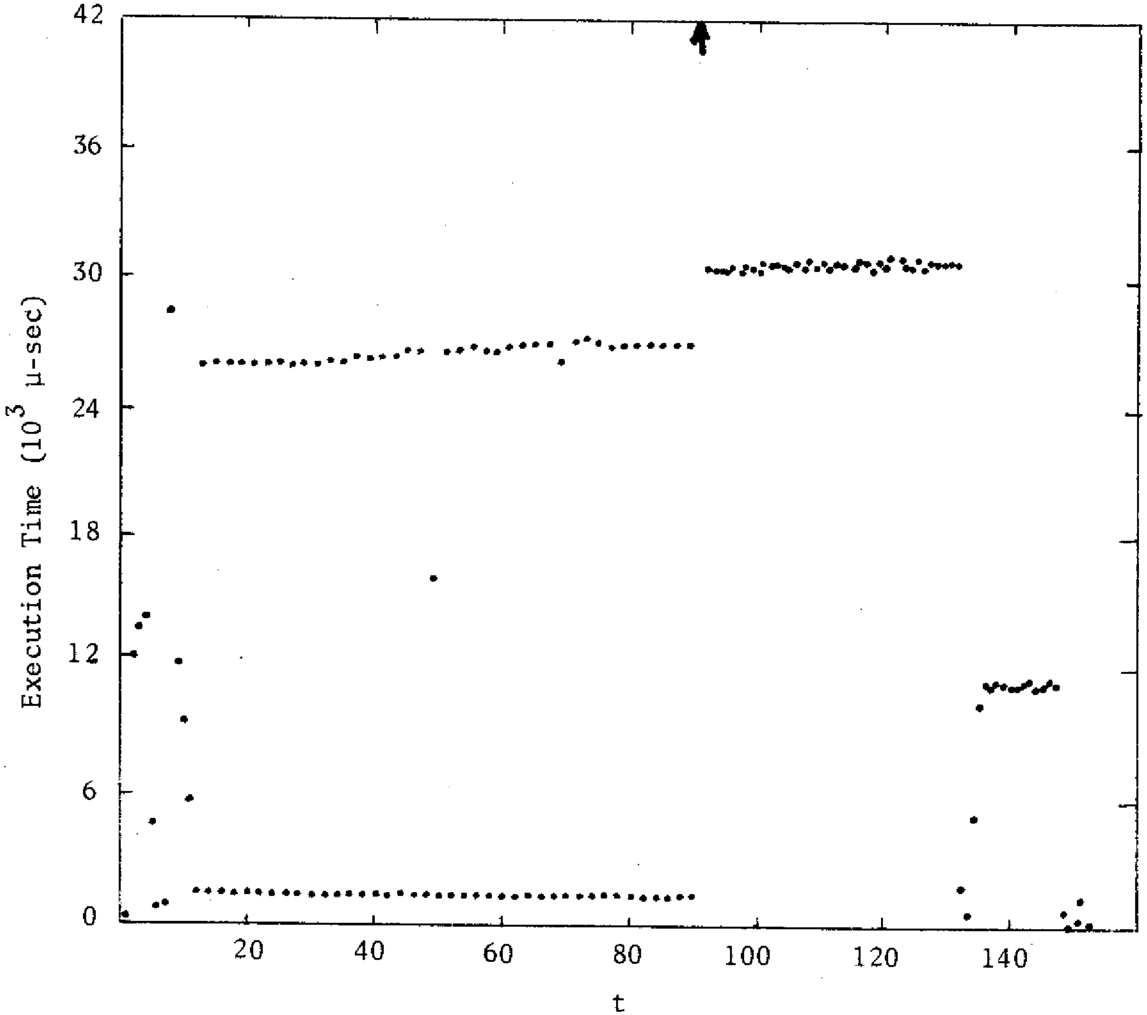


Figure 11. Time Series 4

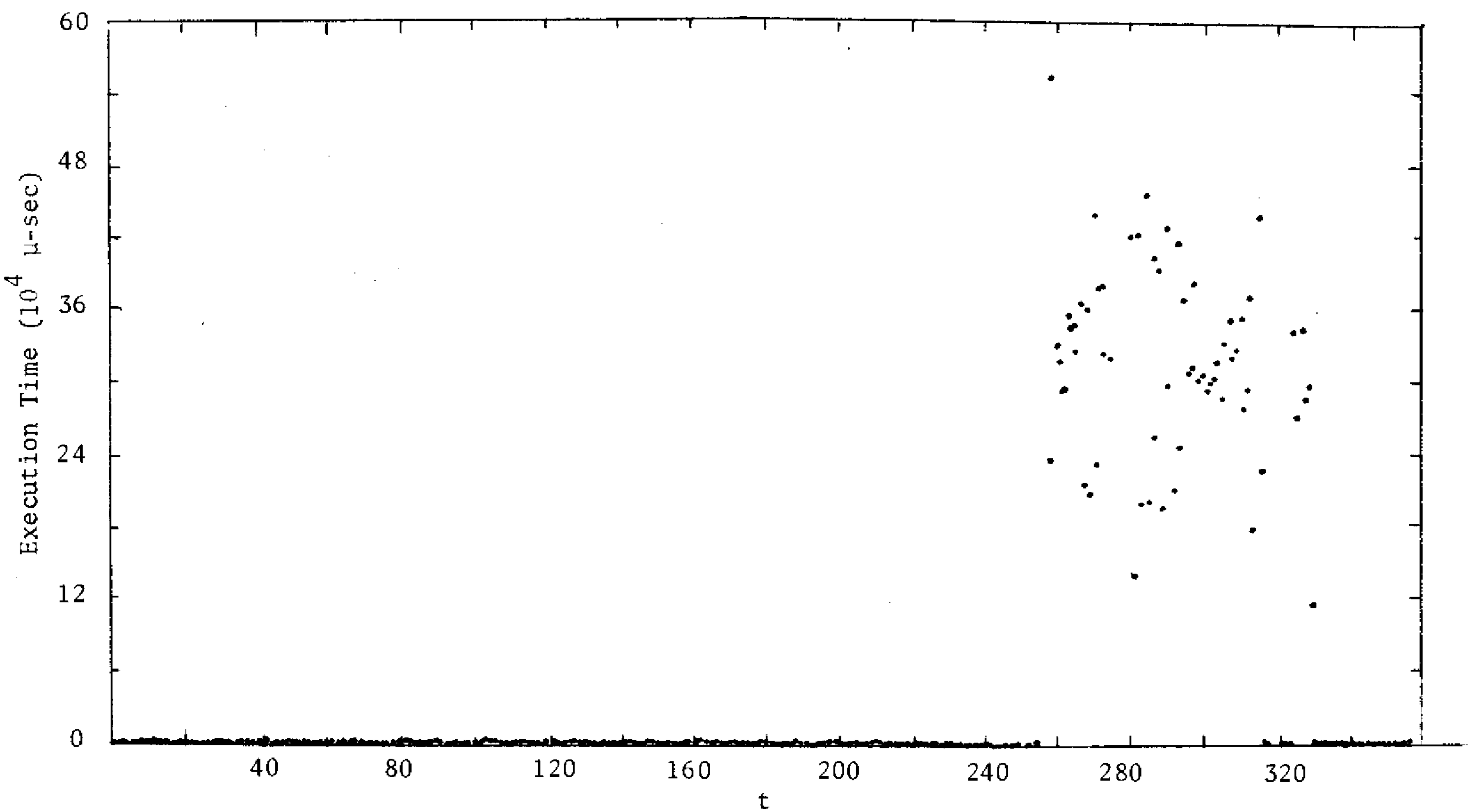


Figure 12. Time Series 5

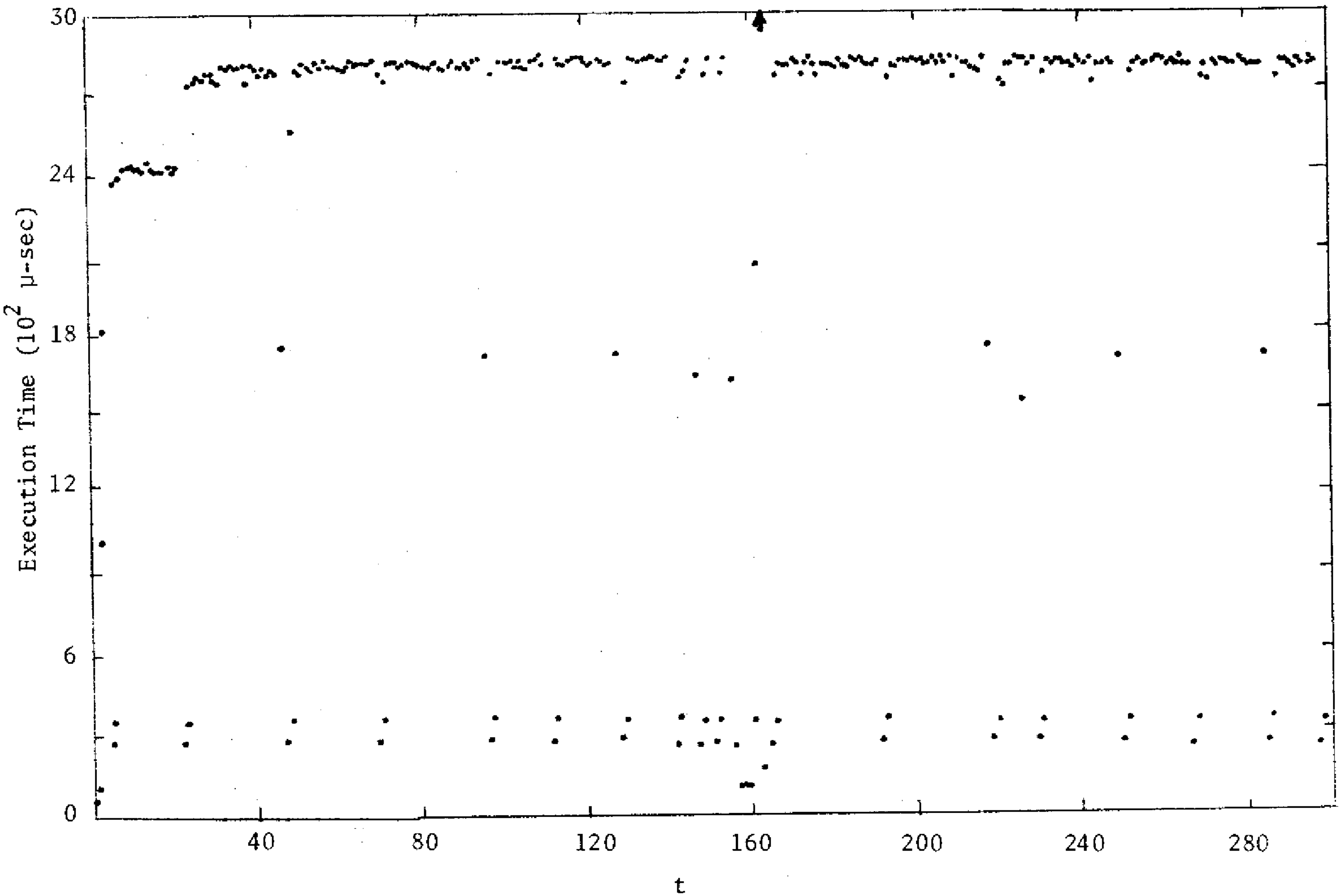


Figure 13. Time Series 6

CHAPTER IV

RESULTS AND CONCLUSIONS

The forecasting techniques described in Chapter III were applied to the six series TS-1 to TS-6. A search for optimal parameters in each forecasting technique was made to identify the best version of each technique when applied to each series separately and when applied to the combined series. The criterion for "best" was the number of forecast errors within $\pm W$ μ-sec, with $W = 200$ showing the most discrimination among various parameters and methods--a fortunate coincidence, since this is the smallest W allowed by the data (recall that numbers of errors in the smallest range are most important in determining actual throughput increases achieved by scheduling based on the forecasts). Among the techniques found to be relatively accurate, the parameter choices using larger values of W are identical (as will be shown in Tables 8 through 13 below). The searches for optimal parameters were limited to the following parameter values: α from .1 to 1 in increments of .1, N from 1 to 9 in increments of 1, τ from .1 to .9 in increments of .1, and Δ from 200 to 1200 in increments of 200 and also at 250, 300, and 350 for those series (TS-1 and TS-5) where the original data had been truncated to the next lower 200 μ-sec.

Best Forecasting Parameters

Table 5 summarizes the forecasting results using the best parameters for each forecasting technique when applied to each

Table 5. Performance of All Tested Forecasting Methods on Each Series, Using Parameters Found Best for Each Series Separately

Forecasting Technique	TS-1 (COBOL) (298 errors)	TS-2 (DIFFER) (107 errors)	TS-3 (METHANE) (122 errors)	TS-4 (OOK) (150 errors)	TS-5 (SIM) (358 errors)	TS-6 (NLS) (298 errors)
	No. of forecast errors within ± 200 μ-sec of observation					
ES Exponential Smoothing	60 $\alpha=1.0$	90 $\alpha=1.0$	120 $\alpha=1.0$	59 $\alpha=1.0$	212 $\alpha=1.0$	247 $\alpha=1.0$
MA Moving Average	60 N=1	90 N=1	120 N=1	59 N=1	212 N=1	247 N=1
SAMA(τ) Self-Adaptive Moving Average	65 $\tau=.6$	94 $\tau=.5-.9$	120 any τ	57 $\tau=.1-.8$	241 $\tau=.9$	247 $\tau=.1-.6$
SAES(τ) Self-Adaptive Exponential Smoothing	68 $\alpha=.1$ $\tau=.5$	94 $\alpha=.9$ $\tau=.5-.9$	120 $\alpha=.1$ $\tau=.5$	59 $\alpha=.1$ $\tau=.5$	224 $\alpha=.9$ $\tau=.9$	247 $\alpha=.1$ $\tau=.5-.9$
SAMA(Δ) Self-Adaptive Moving Average	69 $\Delta=800$	94 $\Delta=800$	120 $\Delta=600-1000$	60 $\Delta=200-800$	274 $\Delta=800$	248 $\Delta=600-800$
SAES(Δ) Self-Adaptive Exponential Smoothing	71 $\alpha=.1$ $\Delta=600-1000$	95 $\alpha=.1$ $\Delta=1200$	120 $\alpha=.1$ $\Delta=800$	59 $\alpha=.1$ $\Delta=200$	274 $\alpha=.1$ $\Delta=800-1200$	248 $\alpha=.1$ $\Delta=200-800$

series separately.

The best version of ES (exponential smoothing) and of MA (moving average) is the special case of current-observation forecasting ($\alpha = 1$ in ES and $N = 1$ in MA). This is true for every series and hence also true for the combined series.

The best version of SAMA(τ) (self-adaptive moving average with level-reset criterion based on relative error) is that with $\tau = .6$ for each series except TS-5, for which $\tau = .9$ is best.

The best version of SAES(τ) (self-adaptive exponential smoothing with level-reset criterion based on relative error) is that with $\alpha = .1$ and $\tau = .5$ for four of the series, and that with $\alpha = .9$ and $\tau = .9$ for TS-2 and TS-5.

The best version of SAMA(Δ) (self-adaptive moving average with level-reset criterion based on absolute error) is that where the level is reset after an error exceeding $\Delta = 800$ μ-sec.

The best version of SAES(Δ) (self-adaptive exponential smoothing with level-reset criterion based on absolute error) is that with $\alpha = .1$ for every series, but the best value of Δ varies slightly from series to series. For TS-2 and for TS-4, resetting the level upon encountering errors exceeding 1200 and 200 μ-sec, respectively, gives slightly better forecasting (one extra forecast error within $W = 200$ μ-sec in each case) than resetting using $\Delta = 800$ μ-sec. For the remaining four series, $\Delta = 800$ μ-sec was best.

Appendix 2 contains histograms of the best versions of each technique for each time series. The time series and technique (with its parameters) are listed on each histogram. The vertical axis numbered from -4 to +4 indicates the number of standard deviations each group is from the mean of the forecast errors.

Table 6 summarizes the forecasting results using the best parameters for each forecasting technique when applied to the combined series. For every technique, the set of parameters that is best for the majority of the individual series is also best for the combined series.

We conclude that the empirical evidence indicates that unmodified exponential smoothing and moving average techniques are not appropriate (except in their trivial versions that collapse to current-observation forecasting), that $\alpha = .1$ is an appropriate smoothing constant within each piece of a piecewise constant series and that $\Delta = 800$ μ-sec is an appropriate forecast error beyond which to assume a change in level.

Best Forecasting Techniques

Choice of forecasting techniques depends both on accuracy and cost. Table 7 gives accuracy information summarized from Table 6 for each forecasting technique and also gives the cost of a single forecast by each technique in terms of the actual UNIVAC 1108 computation time required (as measured by SNOOPY). The same information is presented graphically in Figure 14.

We conclude that two techniques, current-observation and SAES(Δ), are dominant over the other techniques in terms of being significantly more accurate or less costly or both. The choice between current-observation forecasting and SAES(Δ) forecasting would depend on the scheduling algorithm being used, because of doubt as to

Table 6. Performance of All Tested Forecasting Methods on Each Series, Using Parameters Found Best for the Combined Series

Forecasting Technique & Parameters	TS-1 (COBOL) (298 errors)	TS-2 (DIFFER) (107 errors)	TS-3 (METHANE) (122 errors)	TS-4 (OOK) (150 errors)	TS-5 (SIM) (358 errors)	TS-6 (NLS) (298 errors)
	No. of forecast errors within ± 200 μ-sec of observation					
ES Exponential Smoothing, $\alpha=1$	60	90	120	59	212	247
MA Moving Average, N=1	60	90	120	59	212	247
SAMA(τ) Self-Adaptive Moving Average, $\tau=.6$	65	94	120	47	218	247
SAES(τ) Self-Adaptive Exponential Smoothing, $\alpha=.1$ $\tau=.5$	68	94	120	59	196	247
SAMA(Δ) Self-Adaptive Moving Average, $\Delta=800$ μ-sec	69	94	120	60	274	248
SAES(Δ) Self-Adaptive Exponential Smoothing, $\alpha=.1$, $\Delta=800$ μ-sec	71	94	120	58	274	248

Table 7. Forecasting Results for Combined Series TS-1 through TS-6

Forecasting Technique	Parameters Found Best for Combined Series	Errors Within ±200 μ-sec/ No. of Errors	Percentage Within ±200 μ-sec	Computation Time Above Minimum Possible, μ-sec
ES Exponential Smoothing	$\alpha = 1$ (Current Observation)	788/1339	58.8	0.00 (Would be 10.25 for $\alpha < 1$)
MA Moving Average	N = 1 (Current Observation)	788/1339	58.8	0.00 (Would be 16.25 for N > 1)
SAMA(τ) Self-Adaptive Moving Average	$\tau = .6$	801/1339	59.8	38.75
SAES(τ) Self-Adaptive Exponential Smoothing	$\alpha = .1$ $\tau = .5$	784/1339	58.6	25.00
SAMA(Δ) Self-Adaptive Moving Average	$\Delta = 800$ μ-sec	865/1339	64.6	33.50
SAES(Δ) Self-Adaptive Exponential Smoothing	$\alpha = .1$ $\Delta = 800$ μ-sec	865/1333	65.00	18.75

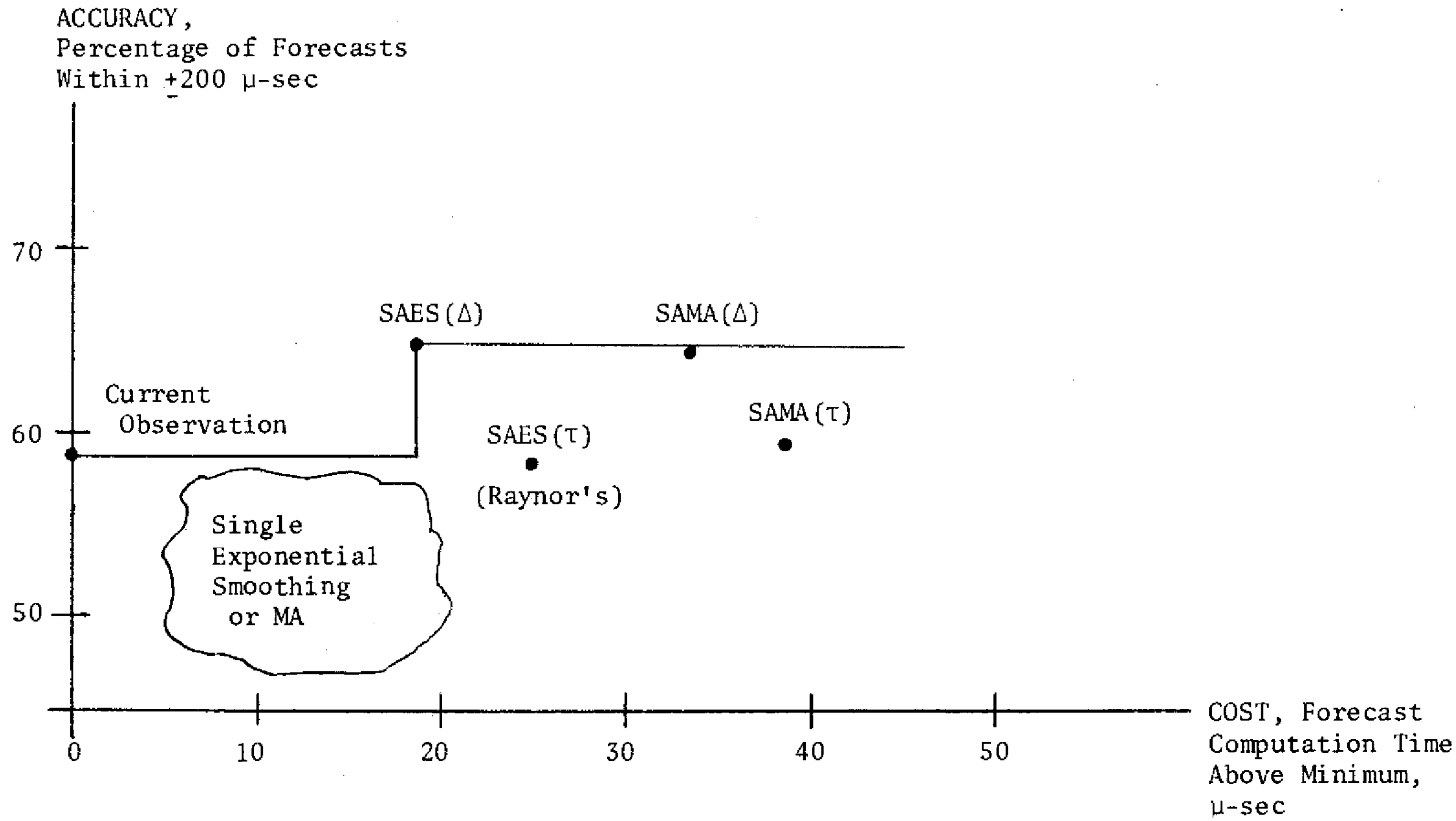


Figure 14. Dominance Graph of Forecasting Methods (Data from Table 7).

the relative contribution (to reducing supervisor queuing) of better scheduling versus reduced supervisor computation time. SAES(Δ) gave forecast errors within ± 200 μ-sec in 65 per cent of all forecasts, and current-observation forecasting in 58.8 per cent. In testing the null hypothesis that the two methods are equally accurate against the hypothesis that SAES(Δ) is more accurate, the advantage of SAES(Δ) over current-observation forecasting is statistically significant at the .001 level. The accuracy advantage of SAES(Δ) over SAMA(Δ) is not significant, but the cost difference is substantial. The accuracy advantage of SAES(Δ) over SAES(τ) (which is the method found best by Raynor of those tested by him) is significant at the .001 level, and the cost difference is also substantial.

We find SAES(τ) and current-observation forecasting to be equally accurate when applied to the six time series. This does not corroborate Raynor's finding that SAES(τ) was slightly but significantly more accurate than current-observation forecasting. However, Raynor's conclusion was based on the series TS-1 and TS-5 only, and as discussed earlier, his accuracy measure was biased.

The forecasting results for each series using SAES(τ) and current-observation forecasting are given in Tables 8 through 13. Since these two techniques are the best found by this research, we present these tables to demonstrate the differences between the two techniques for each error range examined. We can compare forecasting accuracies using the best parameters for each individual series with those using the best parameters for the combined series. Note that SAES(Δ) forecasting was significantly more accurate than the second-best

Table 8. Forecasting Results for Series TS-1 (COBOL), Based on 298 Forecast Errors, Using SAES(Δ) and Current-Observation Forecasting

	No. of forecast errors less than W μ-sec						Error σ, μ-sec
	W=200	W=400	W=600	W=800	W=1000	W=1200	
SAES(Δ) Best level-reset parameters for TS-1: α=.1, Δ=800	71	107	121	123	129	132	12531.5
Best level-reset parameters for combined series: α=.1, Δ=800	71	107	121	123	129	132	12531.5
Current Observation (ES α=1) (MA N=1)	60	70	120	123	128	133	12578.1

Table 9. Forecasting Results for Series TS-2 (DIFFER), Based on 107 Forecast Errors, Using SAES(Δ) and Current-Observation Forecasting

	No. of forecast errors less than W μ-sec						Error σ, μ-sec
	W=200	W=400	W=600	W=800	W=1000	W=1200	
SAES(Δ) Best level-reset parameters for TS-2: α=.1, Δ=1200	95	95	95	100	100	101	934.8
Best level-reset parameters for combined series: α=.1, Δ=800	94	94	94	99	99	101	943.3
Current Observation (ES α=1) (MA N=1)	90	90	92	99	99	101	965.2

Table 10. Forecasting Results for Series TS-3 (METHANE), Based on 122 Forecast Errors, Using SAES(Δ) and Current-Observation Forecasting

	No. of forecast errors less than W μ-sec						Error σ, μ-sec
	W=200	W=400	W=600	W=800	W=1000	W=1200	
SAES(Δ) Best level-reset parameters for TS-3: α=.1, Δ=800	120	120	120	120	120	121	282.8
Best level-reset parameters for combined series: α=.1, Δ=800	120	120	120	120	120	121	282.8
Current Observation (ES α=1) (MA N=1)	59	62	65	65	66	67	282.8

Table 11. Forecasting Results for Series TS-4 (OUT-OF-KILTER), Based on 150 Forecast Errors, Using SAES(Δ) and Current-Observation Forecasting

	No. of forecast errors less than W μ-sec						Error σ, μ-sec
	W=200	W=400	W=600	W=800	W=1000	W=1200	
SAES(Δ) Best level-reset parameters for TS-4: α=.1, Δ=200	59	63	65	65	66	67	3937.4
Best level-reset parameters for combined series: α=.1, Δ=800	58	62	62	63	66	67	3934.7
Current Observation (ES α=1) (MA N=1)	59	62	65	65	66	67	3937.7

Table 12. Forecasting Results for Series TS-5 (SIM), Based on 358 Forecast Errors, Using SAES(Δ) and Current-Observation Forecasting

	No. of forecast errors less than W μ-sec						Error σ, μ-sec
	W=200	W=250	W=300	W=400	W=600	W=800	
SAES(Δ) Best level-reset parameters for TS-5: α=.1, Δ=800	274	290	291	292	293	293	68123.9
Best level-reset parameters for combined series: α=.1, Δ=800	274	290	291	292	293	293	68123.9
Current Observation (ES α=1) (MA N=1)	212	248	275	290	293	293	68127.0

Table 13. Forecasting Results for Series TS-6 (NLS), Based on 293 Forecast Errors, Using SAES(Δ) and Current-Observation Forecasting

	No. of forecast errors less than W μ-sec						Error σ, μ-sec
	W=200	W=400	W=600	W=800	W=1000	W=1200	
SAES(Δ) Best level-reset parameters for TS-6: α=.1, Δ=800	248	248	248	248	250	258	863.0
Best level-reset parameters for combined series: α=.1, Δ=800	248	248	248	248	250	258	863.0
Current Observation (ES α=1) (MA N=1)	247	248	248	248	250	257	851.9

method of current-observation forecasting in individual series TS-1, TS-2, and TS-5 according to the W criterion. The variance of forecast errors failed to indicate this except in the case of TS-2, and in the case of TS-6 the variance falsely indicates a reverse-order accuracy ranking. Also note that in every case, including the two series with truncated data (TS-1 and TS-5), the results using W = 200 are corroborated by similar results using higher values of W.

Recapitulation of Results

The purpose of this research was to develop an improved technique for forecasting execution times between I/O interrupts, so that throughput of a multiprocessor computer system could be increased by using the forecasts in a scheduling algorithm to reduce queueing of processors attempting to obtain jobs. Previous work by Pass and Raynor had developed a method that gives essentially perfect forecasts for 59 per cent of all jobs, giving an assumed 6.6 per cent increase in throughput. The present work has developed a method that gives essentially perfect forecasts for 65 per cent of all jobs, and furthermore uses only three-fourths as much computation time as previous methods. Reasoning from Raynor's results, the improvement of our method over Raynor's should boost the throughput increase to 7.0 per cent or higher. The forecasting method, SAES(Δ), is

$$\begin{aligned}\hat{x}_t &= .1x_{t-1} + .9\hat{x}_{t-1} \quad \text{when } |x_{t-1} - \hat{x}_{t-1}| < 800\ \mu\text{-sec} \\ &= x_{t-1} \qquad\qquad\qquad \text{otherwise}\end{aligned}$$

Our results, based on Raynor's 656 observations from two computer programs plus 683 additional observations from four additional programs of widely varying types, corroborate and strengthen previous suggestions that scheduling based on forecasts can significantly increase the throughput of future multiprocessor computer systems.

CHAPTER V

RECOMMENDATIONS FOR FURTHER RESEARCH

Six areas of further research could continue the work done for this thesis. The first two deal with the generation of the real time series. The next two pertain to the actual utilization of the results and conclusions of this thesis. The fifth area considers forecasting before a program is run in the computer. Finally, further extensions of forecasting methods could be investigated.

First, it is quite apparent that a more efficient method of tracing the programs to generate the time series is needed. Simply too much time and effort are expended in generation of these times. This is not only important for our purposes, but also such research might provide the software that will be needed when multiprocessor systems actually are put into operation in more than just a research configuration.

The second area is that area which at the start of this research was ambiguous and remains so, that is, the search for a program or set of programs that is representative of those habitually processed at a computer center. The more programs that are analyzed, the broader the basis for the results and conclusions enumerated by the researcher.

This thesis dealt with the work of Raynor and his specific scheduling algorithm. Further research is needed to utilize the proposed forecasting techniques in other scheduling algorithms since it is the scheduling algorithm that establishes the accuracy desired from the forecasts. In one algorithm, it may be that a more costly forecasting technique is needed in order to obtain the desired accuracy, whereas in another algorithm not designed to use such great accuracy, a less costly technique might be more satisfactory.

The fourth area for further research is the actual application of the forecasting techniques proposed. That is, the best technique should be put into the computer system, and its performance measured. Since these techniques were developed with Raynor's work in mind, the logical use would be to apply Raynor's scheduling algorithm to a multi-processor system with the best technique as the forecasting routine.

The fifth area for further research was beyond the scope of this thesis. It appears possible that when a program is compiled by the computer, that the computer could at that time tag each computer job with a guessed time to next I/O-interrupt based on the FORTRAN statements between requests for input or output.

As the sixth area for further research, there are at least two classes of time-series forecasting methods that show some promise but have not been fully investigated.

One of these classes includes methods that dynamically re-adjust the criterion for deciding whether or not a time series has changed levels. Preliminary examination was made into a level-reset technique that used $|x_{t-1} - \hat{x}_{t-1}| < k\hat{\sigma}$ as a reset criterion, where $\hat{\sigma}$ was an estimate of the standard deviation of forecast error and k is a constant, say 2.0. It is not yet clear whether $\hat{\sigma}$ should be reset when the level is reset.

Another class of methods would exploit the repetitive structure of loops explicitly. When an observation or series is encountered that closely matches an earlier observation or series, then the forecast would assume continuation of the previous pattern.

APPENDIX 1A

APPENDIX 1A

SET-UP OF THE PROGRAM FOR A SNOOPY TRACE

This appendix is presented under the assumption that the reader has a basic knowledge of FORTRAN programming and Univac 1108 control techniques.

Before a trace can be run, a file (we will call it FILE) must be catalogued containing the following elements.

Element	Where located
1. RELOCATABLE TRA$ER.	. EXEC 8 LIBRARY
2. RELOCATABLE SNOOPY.	. EXEC 8 LIBRARY
3. RELOCATABLE PROGRAM TO BE TRACED	PROGRAMMER
4. RELOCATABLE SUBROUTINE TO PRODUCE TIMES.	PROGRAMMER
5. RELOCATABLE DUMMY ELEMENT	.SEE BELOW

The relocatable DUMMY element is produced through a mapping command as below.

```
@MAP,R    ,FILE.DUMMY
IN FILE.TRA$ER
IN FILE.SNOOPY
IN FILE.SUBROUTINE
DEF TRON
LIB  SYS$*RLIB$.
END
```

Then the executable absolute of the program is produced by mapping

```
@Map,N    ,FILE.PROGRAM
IN FILE.PROGRAM
IN FILE.DUMMY
END
```

Once the absolute has been produced, the program can be executed from either batch (cards) or demand. For short tests demands can be used, but for the actual runs batch is necessary due to the large number of pages of output generated. Figures 15 and 16 depict the commands and the check set up for batch.

```
@RUN CARD
@PWRD CARD
@COL 9 (if used 029 key punch)
@ASG,A FILE.
@XQT FILE.PROGRAM
 [DATA            ]
 [CARDS,IF ANY    ]   or @ADD DATAFILE.
 @EOF
 @FIN
                          DATAFILE is a file with
                          your data previously entered
```

Figure 15. Batch Deck for SNOOPY

```
> RESPONSES TO GET ON
>     TERMINAL
> XCTS (must be in EXEC MODE)
> @ASG,A FILE.
> @XQT FILE.PROGRAM                      or respond to
                                         first > with
> RLIB A                                 @ADD DATAFILE.
> GO
> DATA AS REQUESTED
  BY COMPUTER FOR YOUR
  PROGRAM (TERMINAL WILL PRINT > sign
  AND WAIT FOR YOUR DATA)
> @EOF
> @FIN
```

Figure 16. Demand Commands for SNOOPY

Note: DO NOT @@CQUE since you need to know when computer is requesting information from you.

Due to slowness of demand terminal output, you probably will not be able to let program run more than a short time. Use of the demand should be limited to execution of the program to see that everything is in working order. Once you can establish that fact, terminate the run with normal control procedures.

APPENDIX 1B

Subroutine for Use with SNOOPY

```
FTN4 ... COUNTR
      SUBROUTINE COUNTR(A)
      DIMENSION A(25),ICODE(14),B(8)
      IF(LL.GT.0) GO TO 55
      DEFINE FILE 3(1000,10,F,IFLE),4(1,29,F,JFLE),5(1,30,F,KFLE)
      TIME=0.000
      DO 10 I=1,14
      ICODE(I)=0
   10 CONTINUE
      INDEX1=000000
      INDEX2=000000
      LL=100
      WRITE(4'1,3,ERR=90)TIME,INDEX1,INDEX2
    3 FORMAT(F10.3,2I7)
      INTEGER A,B
   55 WRITE(5'1,506,ERR=90)(A(I),I=1,5)
  506 FORMAT(5A6)
      READ(8'1,507,ERR=90)(B(J),J=1,8)
  507 FORMAT(3X,3A1,4X,2A1,5X,1A1,6X,2A1)
      IF(B(1) .EQ. '0' .OR. B(7) .EQ. 'E')GO TO 6
      RETURN
    6 IF(B(7) .EQ. 'E' .AND. B(8) .EQ. 'R')GO TO 7
      READ(4'1,11,ERR=710)TIME
      IF (B(6) .EQ. 'S')GO TO 8
      IF (B(6) .EQ. 'J')GO TO 9
      GO TO 51
   11 FORMAT(F10.3)
    7 READ(4'1,11,ERR=510)TIME
      WRITE(3'IFLE,11,ERR=510)TIME
      WRITE(6,11)TIME
      TIME=0.0
      WRITE(4'1,11,ERR=510)TIME
  710 FORMAT('YOU BLEW IT HERE')
      RETURN
   51 READ(9'1,600,ERR=510)(ICODE(I),I=1,14)
  600 FORMAT(10X,14I1)
      IF(B(2) .EQ. '2' .AND. B(3) .EQ. '2')GO TO 900
      IF(B(2) .EQ. '6' .AND. (B(3) .NE. '0' .OR. B(3)
     1.NE. '1'))GO TO 901
      IF (B(2) .EQ. '7' .AND. B(3) .EQ. '1' .AND. B(4)
     1.EQ. '0')GO TO 902
      GO TO 50
  900 IF(ICODE(1) .EQ. 0)GO TO 118
      GO TO 99
  901 IF(B(3) .EQ. '2' .AND. ICODE(2) .EQ. 0)GO TO 119
      IF(B(3) .EQ. '2' .AND. ICODE(2) .NE. 0)GO TO 99
      IF(B(3) .EQ. '3' .AND. ICODE(3) .EQ. 0)GO TO 120
      IF(B(3) .EQ. '3' .AND. ICODE(3) .NE. 0)GO TO 99
      IF(B(3) .EQ. '4' .AND. ICODE(4) .EQ. 0)GO TO 121
      IF(B(3) .EQ. '4' .AND. ICODE(4) .NE. 0)GO TO 99
      IF(B(3) .EQ. '5' .AND. ICODE(5) .EQ. 0)GO TO 122
      IF(B(3) .EQ. '5' .AND. ICODE(5) .NE. 0)GO TO 99
      IF(B(3) .EQ. '6' .AND. ICODE(6) .EQ. 0)GO TO 123
      IF(B(3) .EQ. '6' .AND. ICODE(6) .NE. 0)GO TO 99
      IF(B(3) .EQ. '7' .AND. ICODE(7) .EQ. 0)GO TO 124
      IF(B(3) .EQ. '7' .AND. ICODE(7) .NE. 0)GO TO 99
  902 IF(B(5) .EQ. '1' .AND. ICODE(8) .EQ. 0)GO TO 125
```

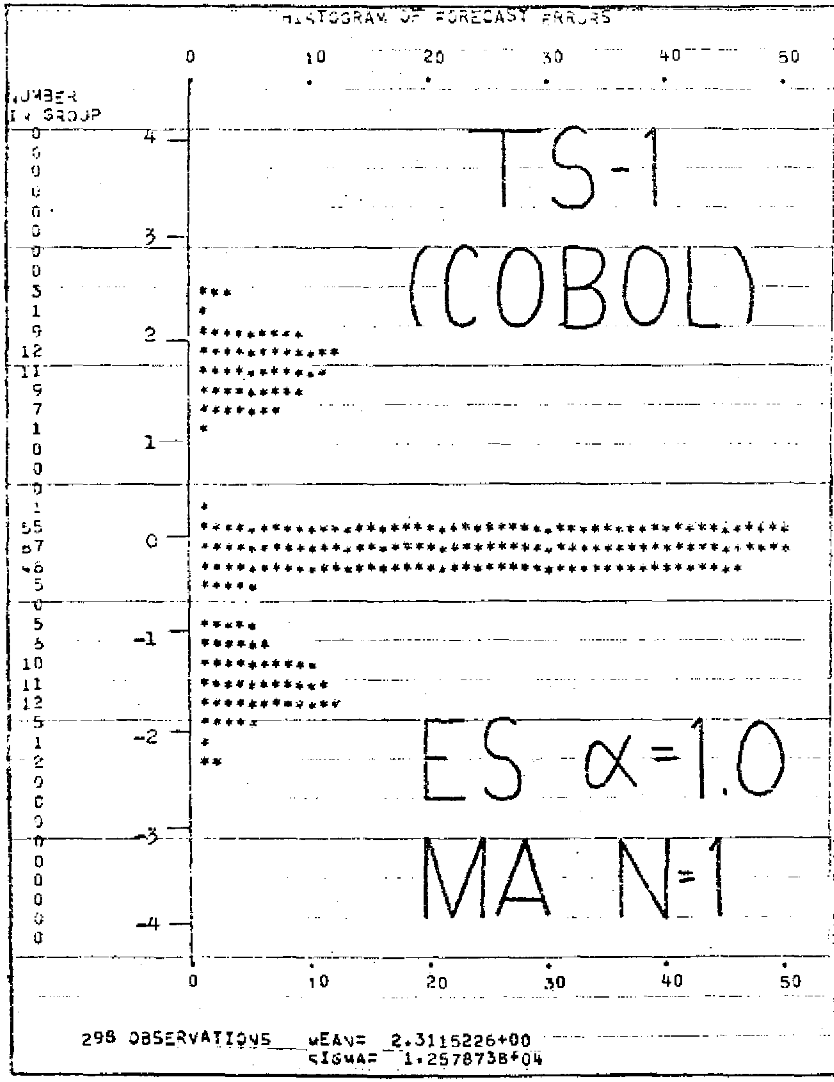
```
      IF(B(5) .EQ. '1' .AND. ICODE(8) .NE. 0)GO TO 99
      IF(B(5) .EQ. '2' .AND. ICODE(9) .EQ. 0)GO TO 126
      IF(B(5) .EQ. '2' .AND. ICODE(9) .NE. 0)GO TO 99
      IF(B(5) .EQ. '3'.AND. ICODE(10) .EQ. 0)GO TO 127
      IF(B(5) .EQ. '3' .AND. ICODE(10) .NE. 0)GO TO 99
      IF(B(5) .EQ. '4' .AND. ICODE(11) .EQ. 0)GO TO 128
      IF(B(5) .EQ. '4' .AND. ICODE(11) .NE. 0)GO TO 99
      IF(B(5) .EQ. '5' .AND. ICODE(12) .EQ. 0)GO TO 129
      IF(B(5) .EQ. '5' .AND. ICODE(12) .NE. 0)GO TO 99
      IF(B(5) .EQ. '6' .AND. ICODE(13) .EQ. 0)GO TO 130
      IF(B(5) .EQ. '6' .AND. ICODE(13) .NE. 0)GO TO 99
      IF(B(5) .EQ. '7' .AND. ICODE(14) .EQ. 0)GO TO 131
      IF(B(5) .EQ. '7' .AND. ICODE(14) .NE. 0)GO TO 99
    8 IF(B(2) .EQ. '4' .AND. B(3) .EQ. '4')GO TO 100
      IF(B(2) .EQ. '4' .AND. B(3) .EQ. '5')GO TO 100
      IF(B(2) .EQ. '4' .AND. B(3) .EQ. '7')GO TO 101
      IF(B(2) .EQ. '5' .AND. (B(3) .EQ. '6' .OR. B(3) .EQ. '7'
     1))GO TO 101
      IF(B(2) .EQ. '6' .AND. (B(3) .EQ. '0' .OR. B(3) .EQ.
     1 '1'))GO TO 103
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '1' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '7')GO TO 104
      IF(B(2) .EQ. '5' .AND. (B(3) .EQ. '0' .OR. B(3)
     1 .EQ. '1' .OR. B(3) .EQ. '2' .OR. B(3) .EQ. '3' .OR.
     2 B(3) .EQ. '4' .OR. B(3) .EQ. '5'))GO TO 102
      GO TO 51
    9 IF(B(2) .EQ. '7' .AND. B(3) .EQ. '1' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '7')GO TO 105
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '0')GO TO 99
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '4' .AND.
     1 B(4) .EQ. '1' .AND. B(5) .NE. '3')GO TO 103
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '4' .AND. B(4) .EQ.
     1 '0' .AND. (B(5) .EQ. '0' .OR. B(5) .EQ. '1' .OR.
     2 B(5) .EQ. '2' .OR. B(5) .EQ. '3'))GO TO 103
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '2' .AND. B(4) .EQ.
     1 '0' .AND. (B(5) .EQ. '2' .OR. B(5) .EQ. '3'))
     2 GO TO 103
      GO TO 51
   50 IF(B(2) .EQ. '7' .AND. B(3) .EQ. '2' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '1') GO TO 106
      IF(B(2) .EQ. '4' .AND. (B(3) .EQ. '4' .OR. B(3) .EQ. '5'
     1))GO TO 107
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '6' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '2')GO TO 108
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '5' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '3')GO TO 109
      IF((B(2) .EQ. '4' .OR. B(2) .EQ. '5') .AND. (B(3) .EQ.
     1 '6' .OR. B(3) .EQ. '7'))GO TO 110
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '6' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '6')GO TO 110
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '6' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '7')GO TO 105
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '1' .AND. B(4) .EQ.
     1 '1' .AND. (B(5) .EQ. '0' .OR. B(5) .EQ. '1' .OR.
     2 B(5) .EQ. '7'))GO TO 105
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '6' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '4')GO TO 103
```

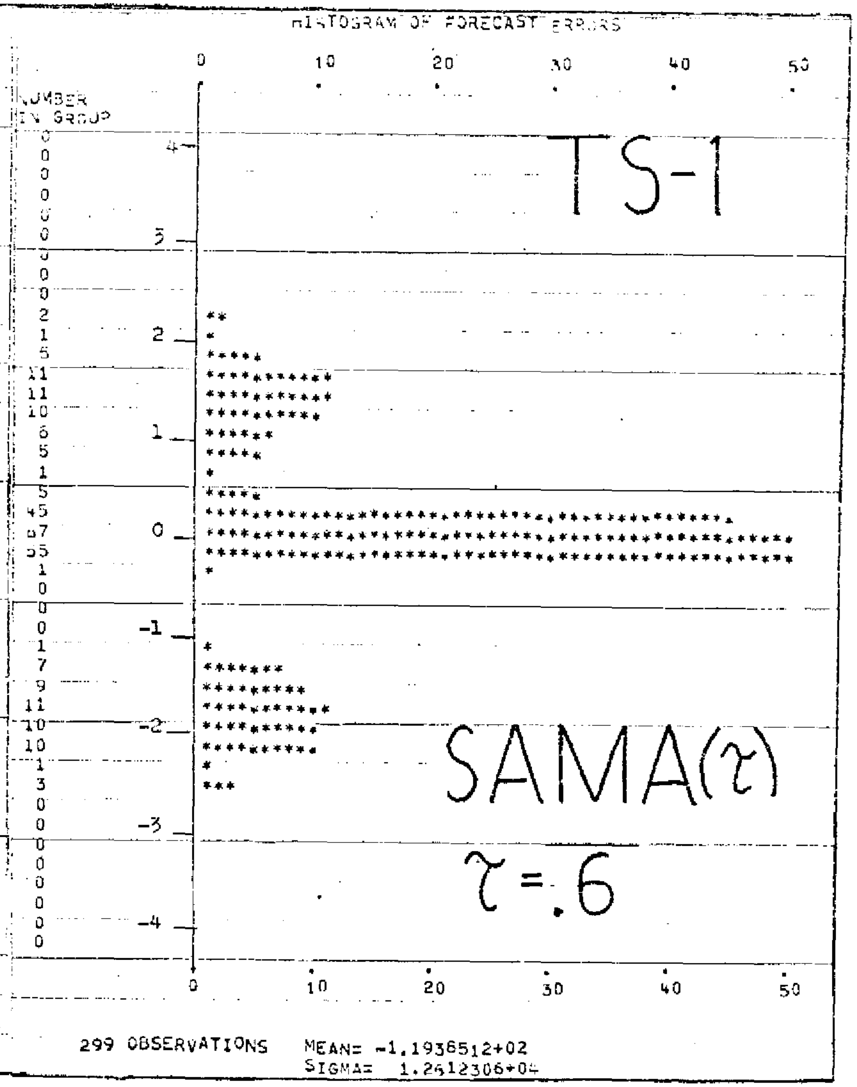
```
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '1' .AND. B(4) .EQ.
     1 '1' .AND. (B(5) .EQ. '2' .OR. B(5) .EQ. '3' .OR. B(5)
     2 .EQ. '4' .OR. B(5) .EQ. '5'))GO TO 103
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '0')GO TO 103
      IF(B(2) .EQ. '3' .AND. (B(3) .EQ. '0' .OR. B(3) .EQ.
     1 '1' .OR. B(3) .EQ. '2'))GO TO 104
      IF(B(2) .EQ. '7'.AND. B(3) .EQ. '5' .AND. B(4) .EQ.
     1 '0' .AND. (B(5) .EQ. '0' .OR. B(5) .EQ. '1')) GO
     2 TO 111
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '3' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '7')GO TO 100
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '6' .AND. B(4) .EQ.
     1 '0' .AND. B(5) .EQ. '3')GO TO 112
      IF(B(2) .EQ. '3' .AND. (B(3) .EQ. '4' .OR. B(3) .EQ. '5'
     1 .OR. B(3) .EQ. '6'))GO TO 113
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '3' .AND. B(4) .EQ.
     1 '0' .AND. B(5) .EQ. '6')GO TO 114
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '6' .AND. B(4) .EQ.
     1 '0' .AND. B(5) .EQ. '5')GO TO 114
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '6' .AND. B(4) .EQ.
     1 '0' .AND. B(5) .EQ. '2')GO TO 115
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '6' .AND. B(4) .EQ.
     1 '1' .AND. (B(5) .EQ. '0' .OR. B(5) .EQ. '1'))
     2 GO TO 115
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '2' .AND. B(4) .EQ.
     1 '0' .AND. B(5) .EQ. '1')GO TO 116
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '3' .AND. B(4) .EQ.
     1 '0' .AND. B(5) .EQ. '7')GO TO 116
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '6' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '5')GO TO 116
      IF(B(2) .EQ. '2' .AND. B(3) .EQ. '6')GO TO 117
      IF(B(2) .EQ. '5' .AND. (B(3) .EQ. '0' .OR. B(3) .EQ. '1'
     1 .OR. B(3) .EQ. '2' .OR. B(3) .EQ. '3' .OR. B(3) .EQ.
     2 '4' .OR. B(3) .EQ. '5'))GO TO 117
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '1' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '6')GO TO 117
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '3' .AND. (B(4) .EQ.
     1 '0' .OR. B(4) .EQ. '1') .AND. (B(5) .EQ. '1' .OR. B(5)
     2 .EQ. '3' .OR. B(5) .EQ. '5'))GO TO 117
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '3' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '6')GO TO 117
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '4' .AND. B(4) .EQ.
     1 '1' .AND. B(5) .EQ. '3')GO TO 117
      IF((B(2) .EQ. '2' .OR. B(2) .EQ. '6') .AND. B(3) .EQ.
     1 '2')GO TO 118
      IF(B(2) .EQ. '6' .AND. (B(3) .EQ. '3' .OR. B(3) .EQ.
     1 '4' .OR. B(3) .EQ. '5' .OR. B(3) .EQ. '6' .OR. B(3)
     2 .EQ. '7'))GO TO 118
      IF(B(2) .EQ. '7' .AND. B(3) .EQ. '1' .AND. B(4) .EQ.
     1 '0')GO TO 118
      GO TO 199
   99 TIME=TIME+.750
      WRITE(4'J,11,ERR=510)TIME
      RETURN
  100 TIME=TIME+2.000
      GO TO 201
  101 TIME=TIME+1.750
```

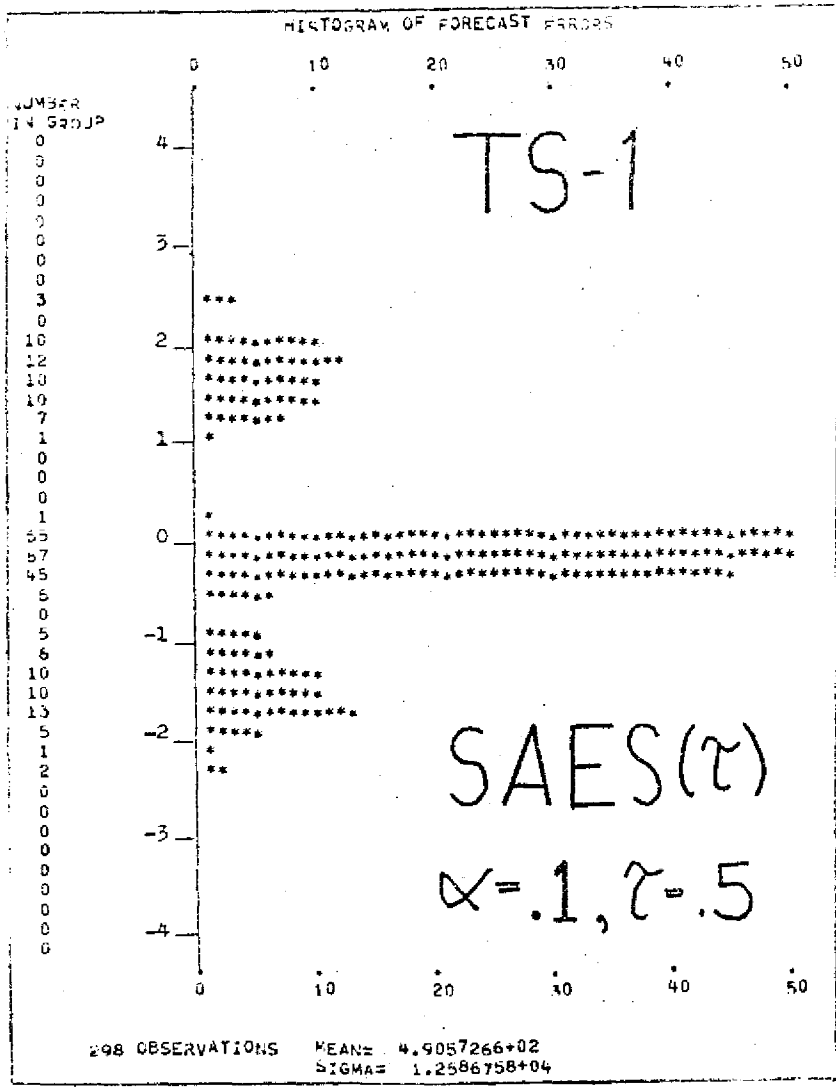
```
      GO TO 201
  102 TIME=TIME+1.625
      GO TO 201
  103 TIME=TIME+1.500
      GO TO 201
  104 TIME=TIME+2.375
      GO TO 201
  105 TIME=TIME+1.625
      GO TO 201
  106 TIME=TIME+1.375
      GO TO 201
  107 TIME=TIME+1.250
      GO TO 201
  108 TIME=TIME+4.250
      GO TO 201
  109 TIME=TIME+17.250
      GO TO 201
  110 TIME=TIME+1.000
      GO TO 201
  111 TIME=TIME+1.875
      GO TO 201
  112 TIME=TIME+8.250
      GO TO 201
  113 TIME=TIME+10.125
      GO TO 201
  114 TIME=TIME+1.125
      GO TO 201
  115 TIME=TIME+2.625
      GO TO 201
  116 TIME=TIME+2.125
      GO TO 201
  117 TIME=TIME+.875
      GO TO 201
  118 TIME=TIME+2.250
      INDEX1=1000000
      INDEX2=0000000
      GO TO 203
  119 TIME=TIME+2.250
      INDEX1=0100000
      INDEX2=0000000
      GO TO 203
  120 TIME=TIME+2.250
      INDEX1=0010000
      INDEX2=0000000
      GO TO 203
  121 TIME=TIME+2.250
      INDEX1=0001000
      INDEX2=0000000
      GO TO 203
  122 TIME=TIME+2.250
      INDEX1=0000100
      INDEX2=0000000
      GO TO 203
  123 TIME=TIME+2.250
      INDEX1=0000010
      INDEX2=0000000
      GO TO 203
```

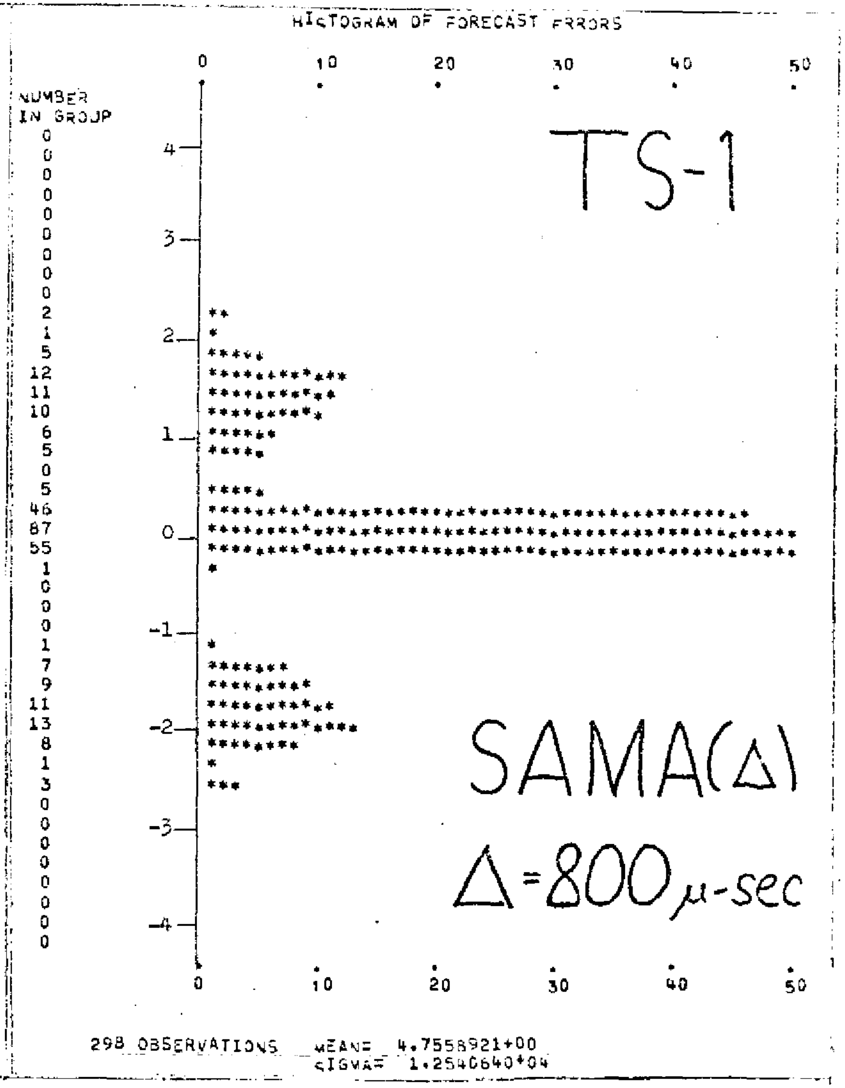
```
  124 TIME=TIME+2.250
      INDEX1=0000001
      INDEX2=0000000
      GO TO 203
  125 TIME=TIME+2.250
      INDEX1=0000000
      INDEX2=1000000
      GO TO 203
  126 TIME=TIME+2.250
      INDEX1=0000000
      INDEX2=0100000
      GO TO 203
  127 TIME=TIME+2.250
      INDEX1=0000000
      INDEX2=0010000
      GO TO 203
  128 TIME=TIME+2.250
      INDEX1=0000000
      INDEX2=0001000
      GO TO 203
  129 TIME=TIME+2.250
      INDEX1=0000000
      INDEX2=0000100
      GO TO 203
  130 TIME=TIME+2.250
      INDEX1=0000000
      INDEX2=0000010
      GO TO 203
  131 TIME=TIME+2.250
      INDEX1=0000000
      INDEX2=0000001
      GO TO 203
  199 TIME=TIME+.750
  201 INDEX1=0000000
      INDEX2=0000000
  203 WRITE(9'1,602,ERR=510)TIME,INDEX1,INDEX2
  602 FORMAT(F10.3,2I7)
      RETURN
      ENTRY CLEANP
  510 WRITE(6,511)
  511 FORMAT('YOU DID AND ERR EXIT')
      GO TO 90
  710 WRITE(6,711)
   90 STOP
      END
@PRT HISTO.PLOT
```

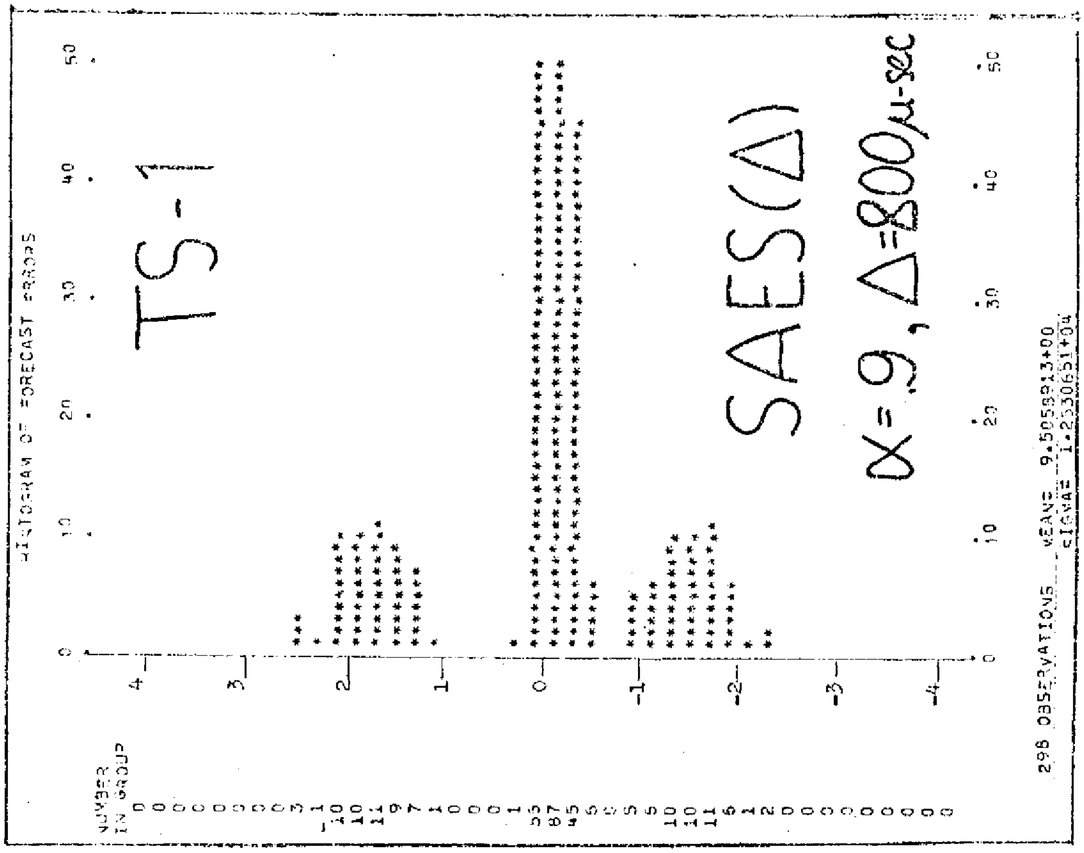
APPENDIX 2

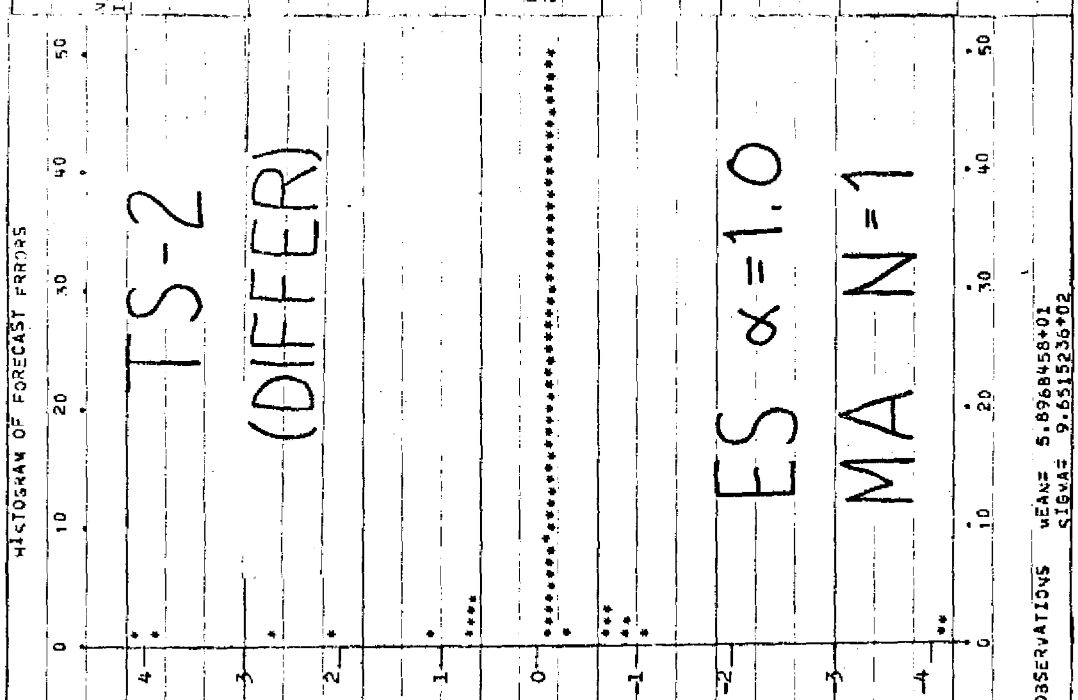

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-1
(COBOL)
ES α=1.0
MA N=1
298 OBSERVATIONS
MEAN= 2.3115226+00
SIGMA= 1.2578738+04

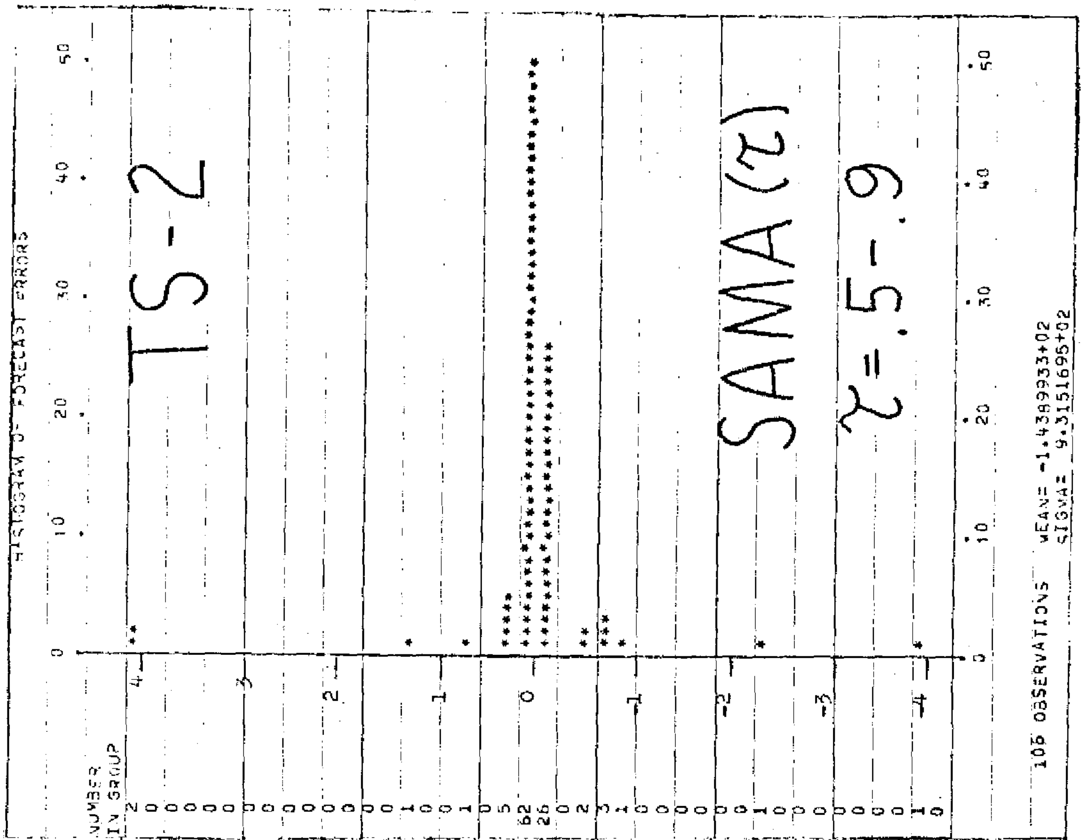

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-1
SAMA(τ)
τ=.6
299 OBSERVATIONS
MEAN= -1.1936512+02
SIGMA= 1.2612306+04

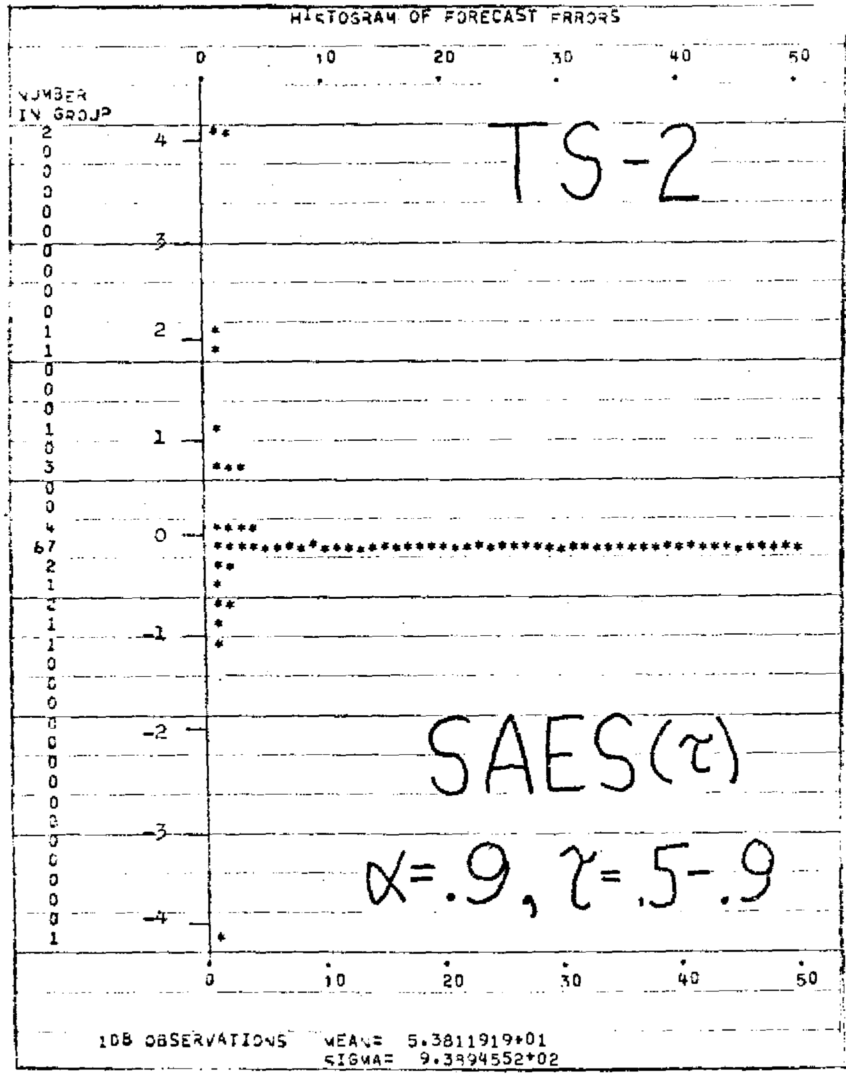

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-1
SAES(τ)
α=.1, τ=.5
298 OBSERVATIONS
MEAN= 4.9057266+02
SIGMA= 1.2586758+04

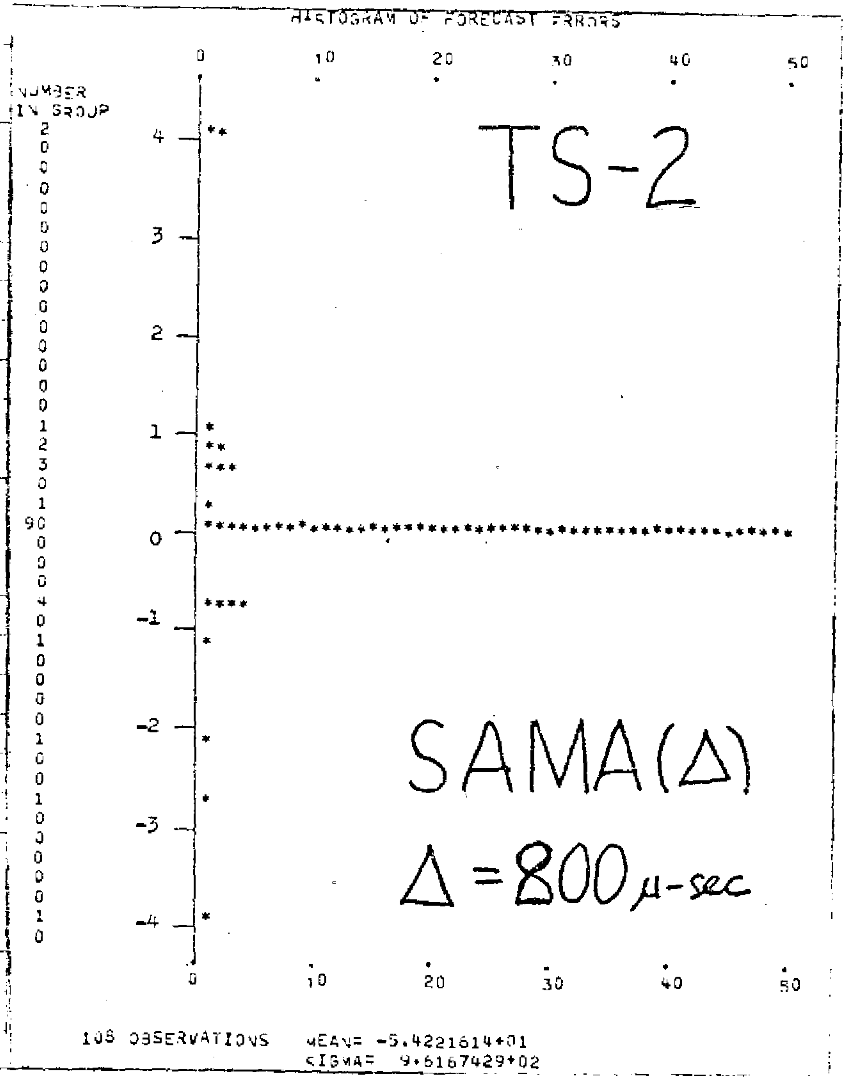

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-1
SAMA(Δ)
Δ=800 μ-sec
298 OBSERVATIONS
MEAN= 4.7558921+00
SIGMA= 1.2540640+04

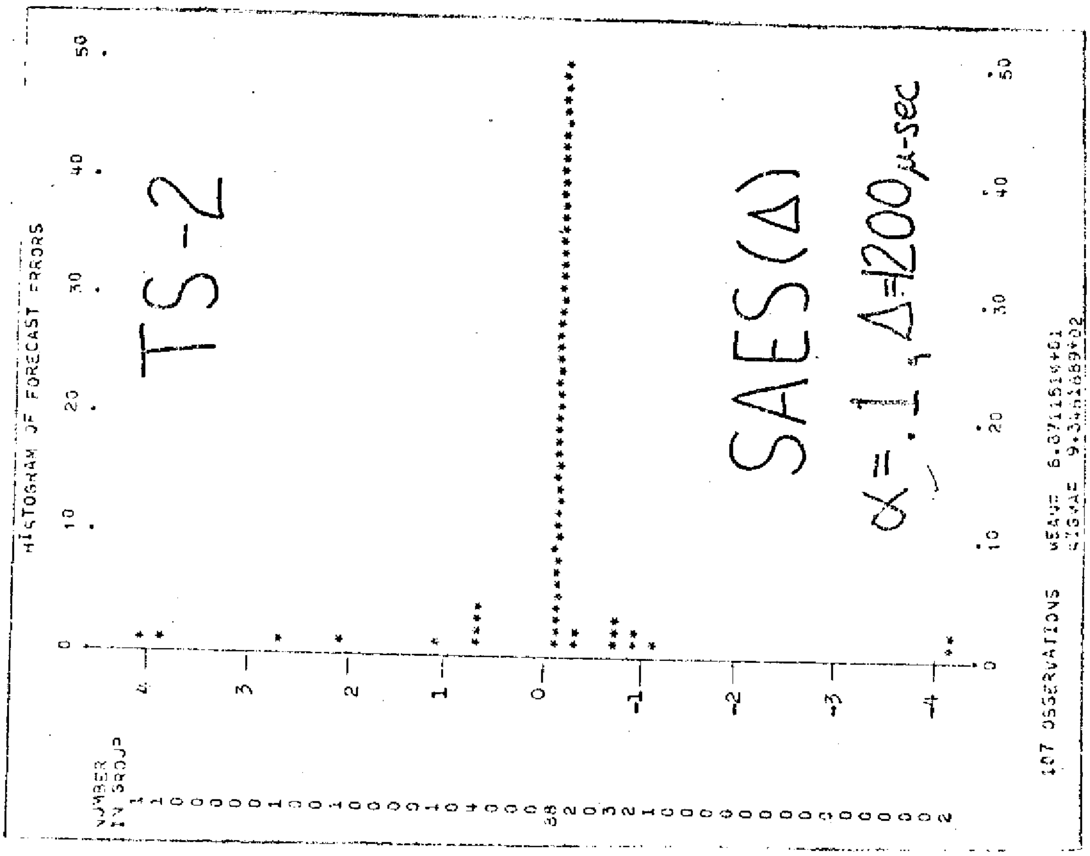

HISTOGRAM OF FORECAST ERRORS
TS-1
SAES(Δ)
α=.9, Δ=800μ-sec
298 OBSERVATIONS

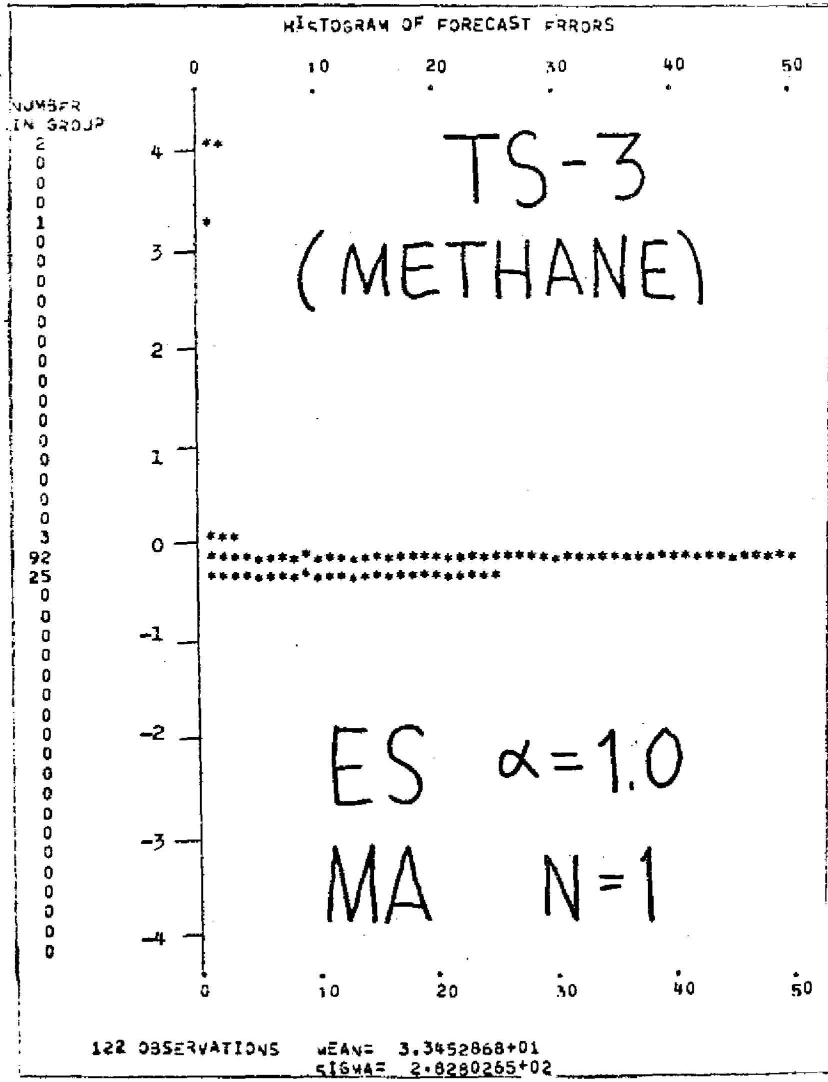

HISTOGRAM OF FORECAST ERRORS
TS-2
(DIFFER)
ES α=1.0
MA N=1
107 OBSERVATIONS
MEAN= 5.8968458+01
SIGMA= 9.6515236+02

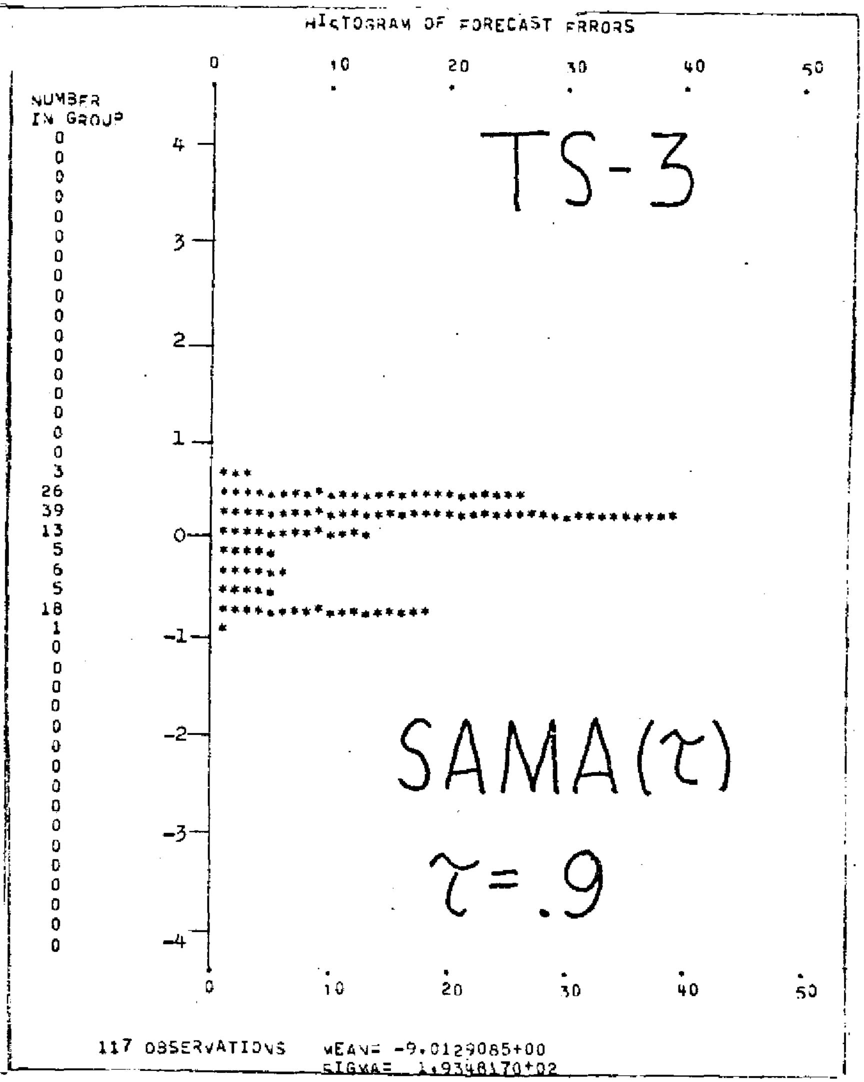

HISTOGRAM OF FORECAST ERRORS
TS-2
SAMA(τ)
τ=.5-.9
106 OBSERVATIONS
MEAN= -1.4389933+02
SIGMA= 9.3151695+02

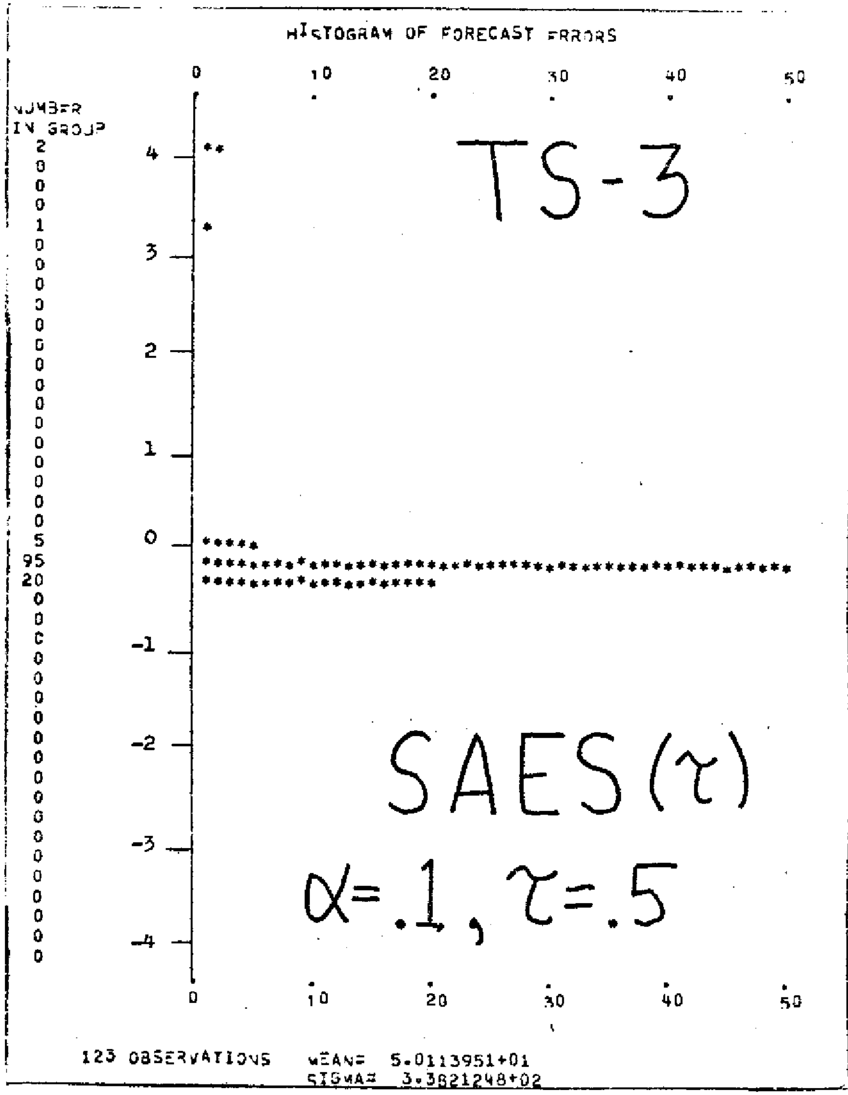

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-2
SAES(τ)
α=.9, τ=.5-.9
108 OBSERVATIONS
MEAN= 5.3811919+01
SIGMA= 9.3894552+02

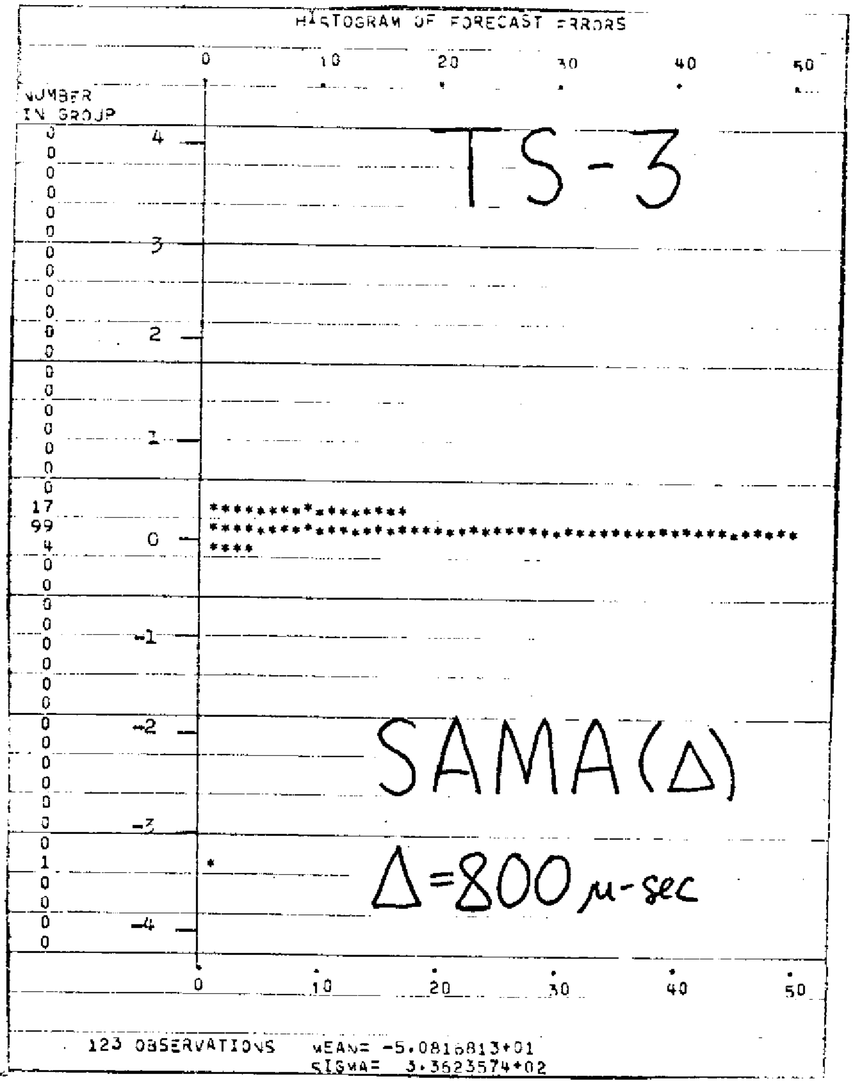

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-2
SAMA(Δ)
Δ=800μ-sec
108 OBSERVATIONS
MEAN= -5.4221614+01
SIGMA= 9.6167429+02

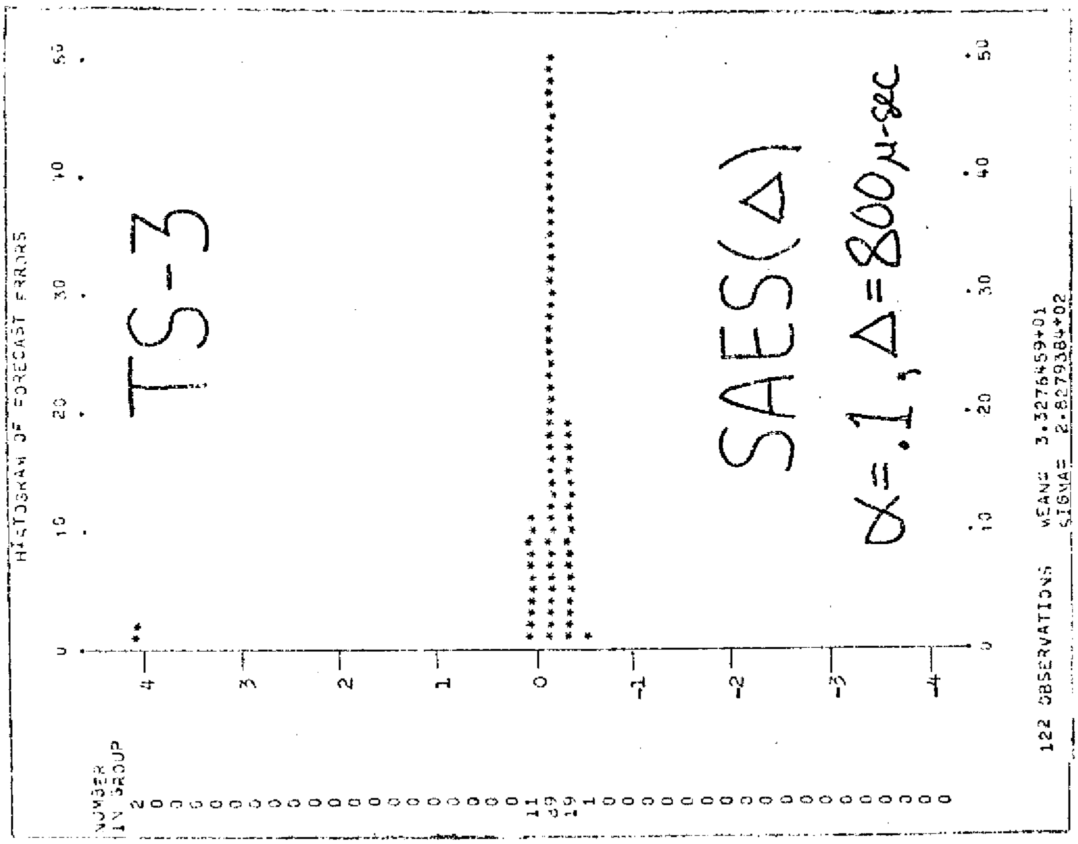

HISTOGRAM OF FORECAST ERRORS
TS-2
SAES(Δ)
α = .1 , Δ=1200 μ-sec
NUMBER IN GROUP
157 OBSERVATIONS

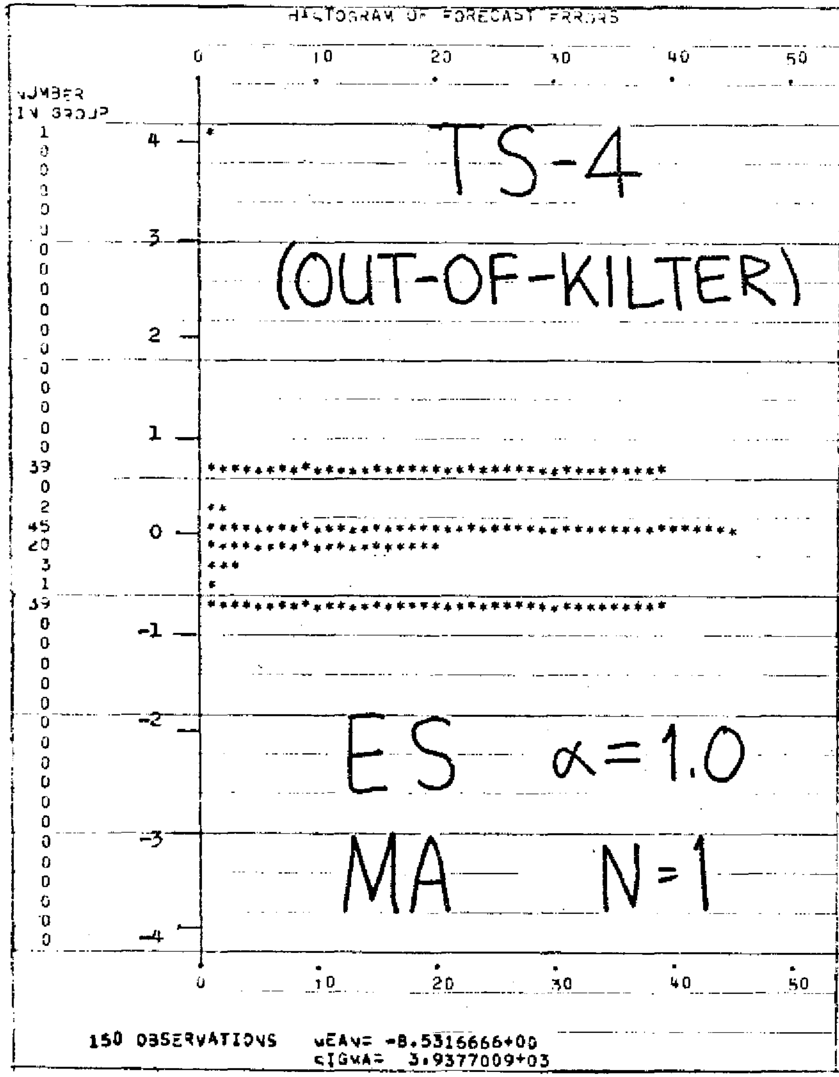

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-3
(METHANE)
ES α=1.0
MA N=1
122 OBSERVATIONS
MEAN= 3.3452868+01
SIGMA= 2.8280265+02

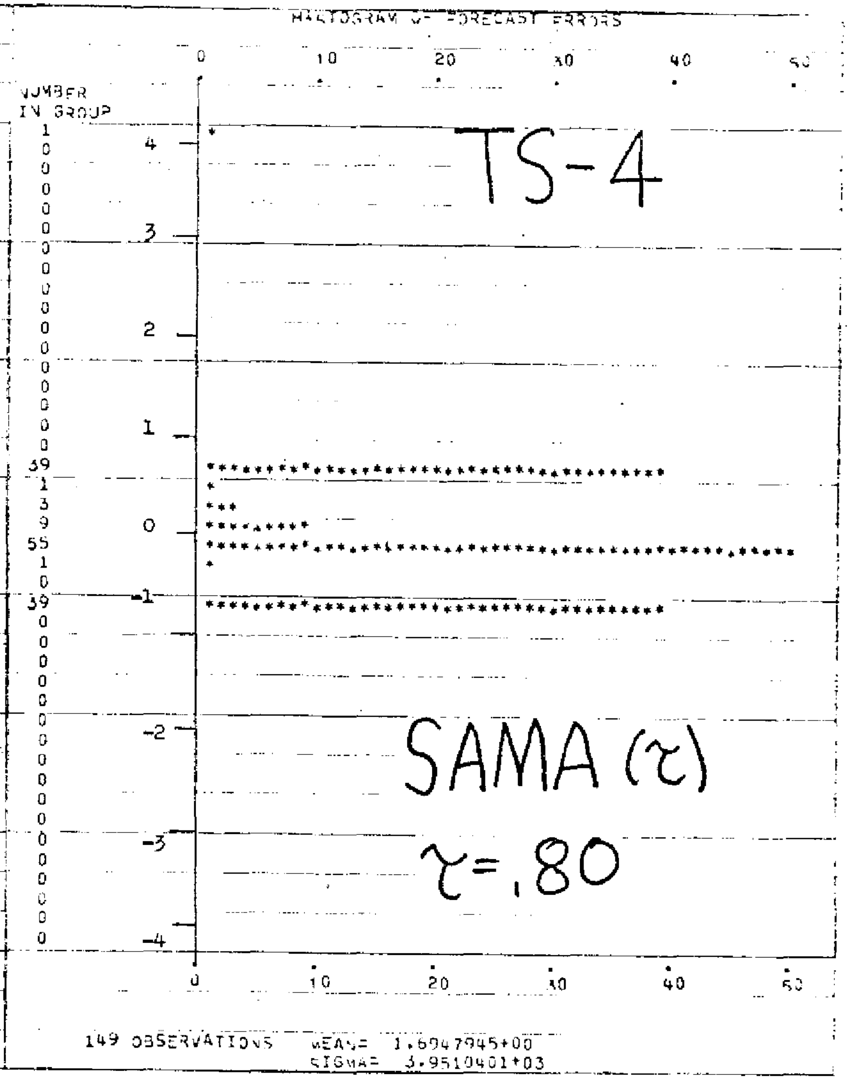

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-3
SAMA(τ)
τ=.9
117 OBSERVATIONS
MEAN= -9.0129085+00
SIGMA= 1.9348170+02

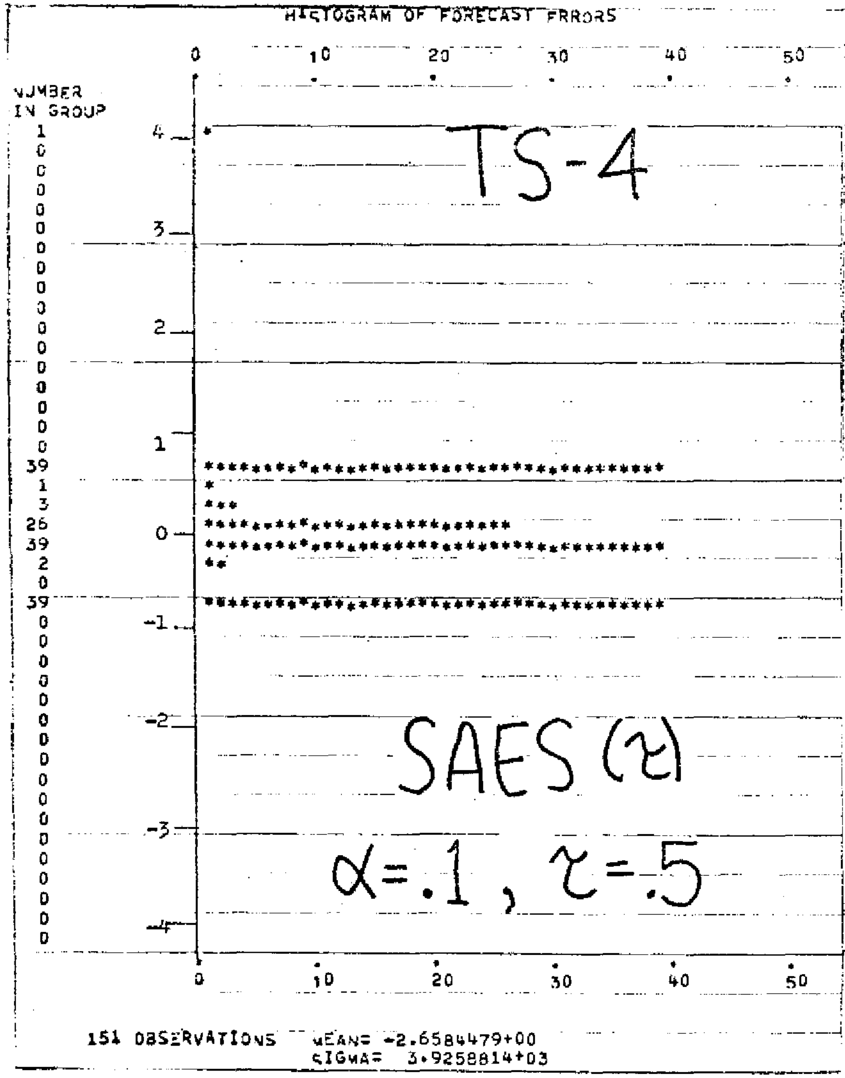

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-3
SAES(τ)
α=.1, τ=.5
123 OBSERVATIONS
MEAN= 5.0113951+01
SIGMA= 3.3821248+02

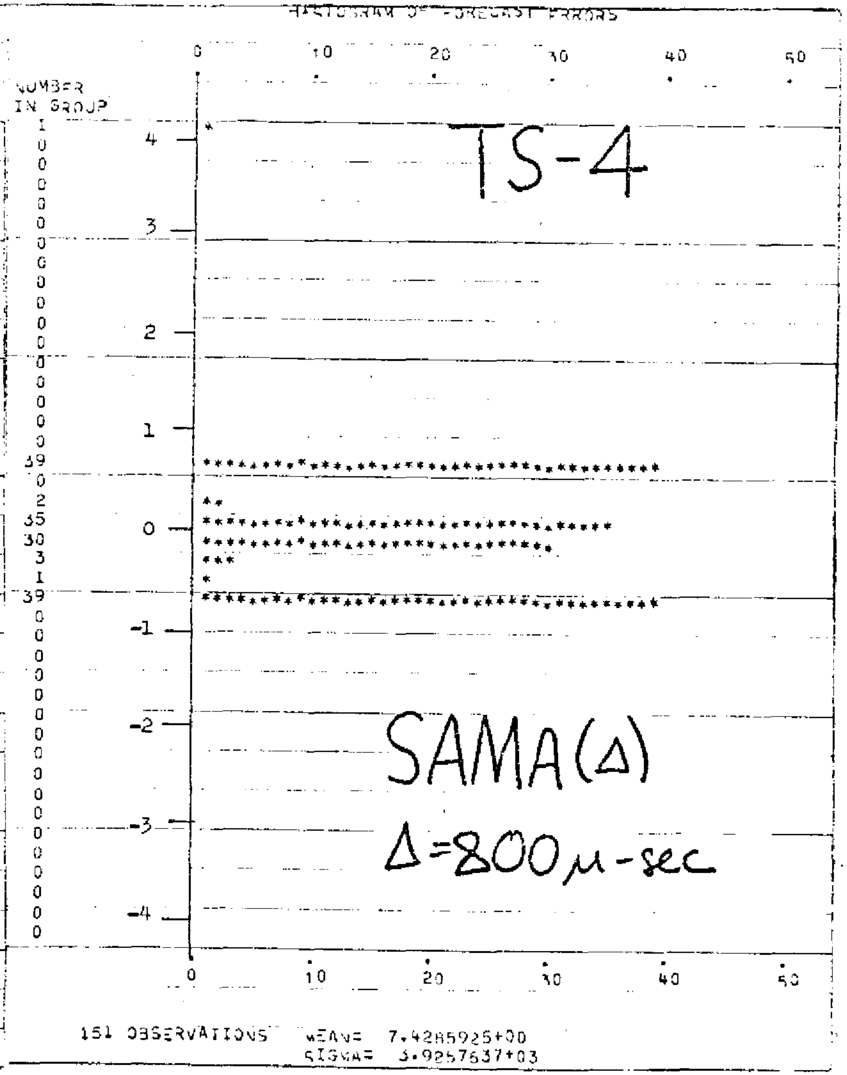

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-3
SAMA(Δ)
Δ=800 μ-sec
123 OBSERVATIONS
MEAN= -5.0816813+01
SIGMA= 3.3623574+02

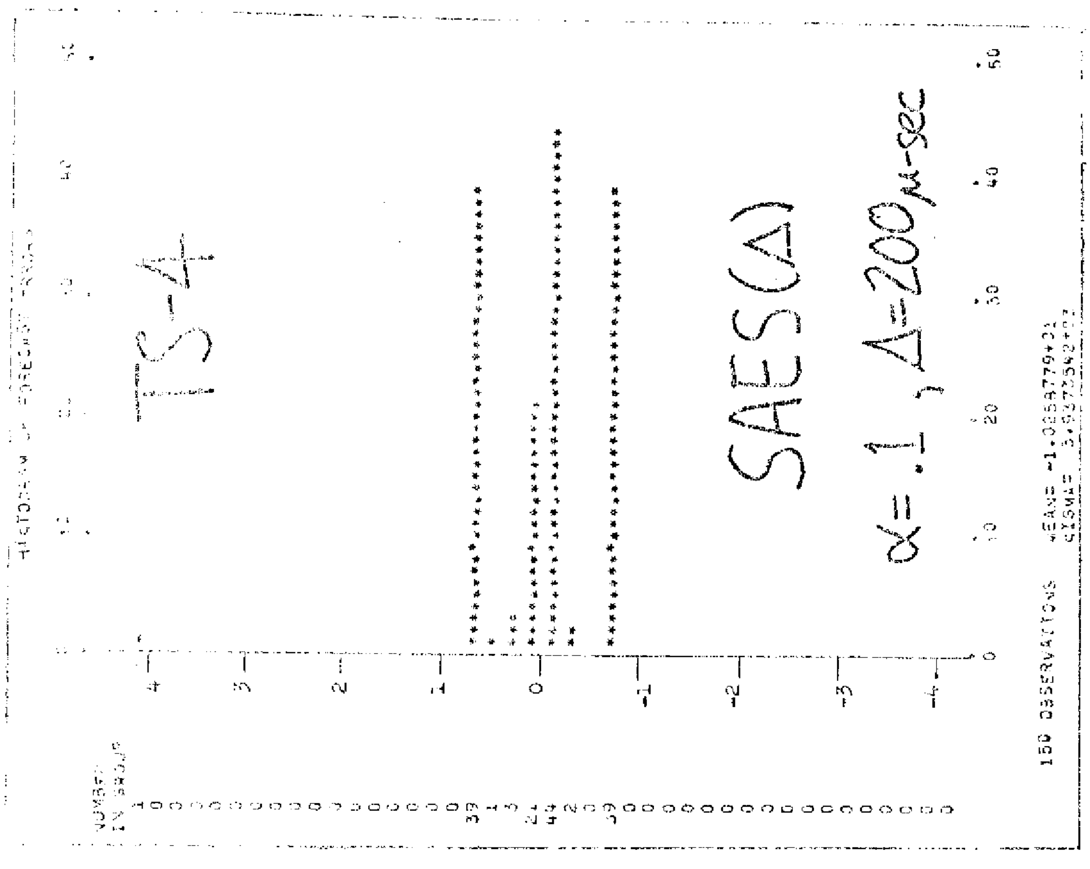

HISTOGRAM OF FORECAST ERRORS
TS-3
SAES(Δ)
α=.1, Δ=800 μ-sec
NUMBER IN GROUP
122 OBSERVATIONS
MEANS 3.3276459+01
SIGMA= 2.8279384+02

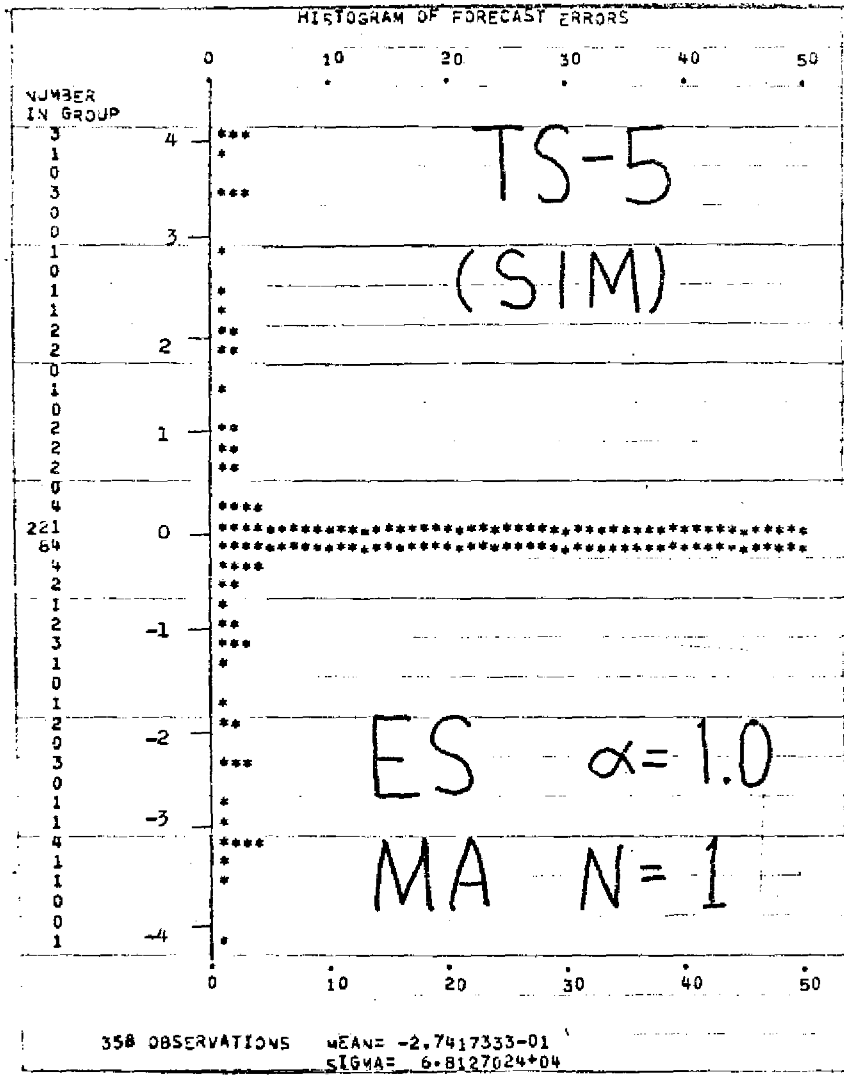

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-4
(OUT-OF-KILTER)
ES α=1.0
MA N=1
150 OBSERVATIONS
MEAN= -8.5316666+00
SIGMA= 3.9377009+03

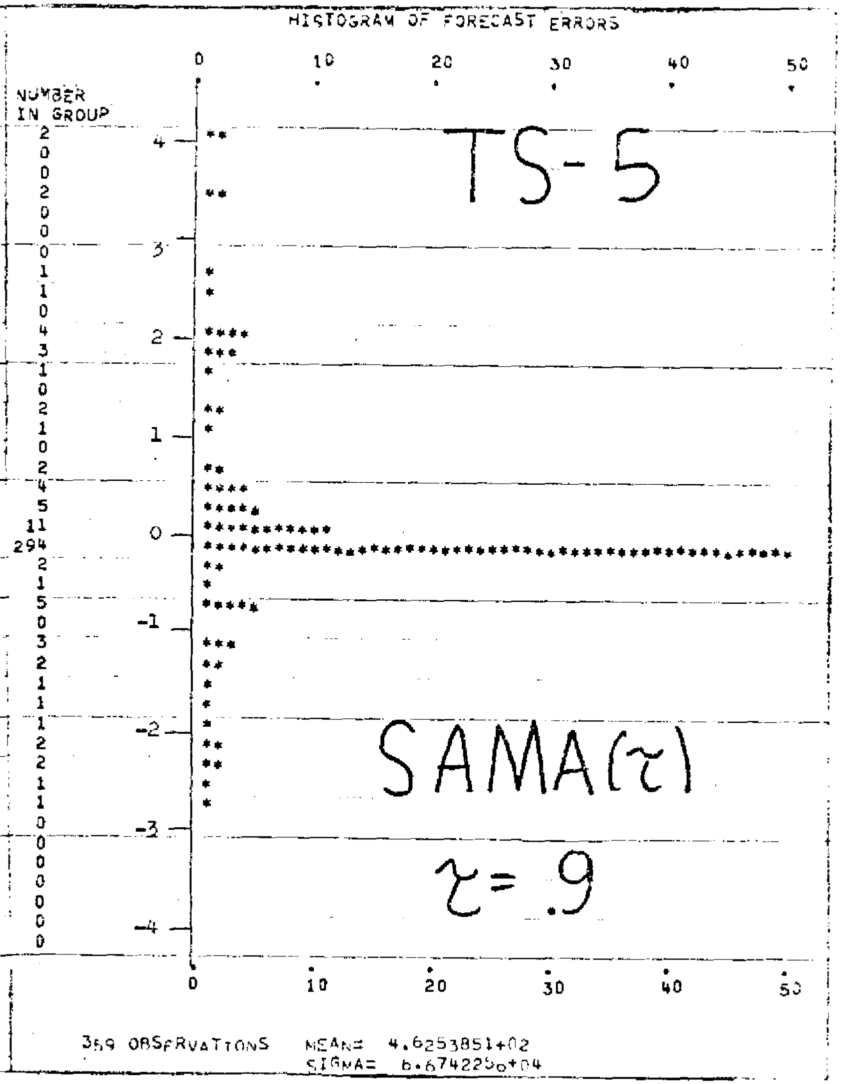

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-4
SAMA (τ)
τ=.80
149 OBSERVATIONS
MEAN= 1.6947945+00
SIGMA= 3.9510401+03

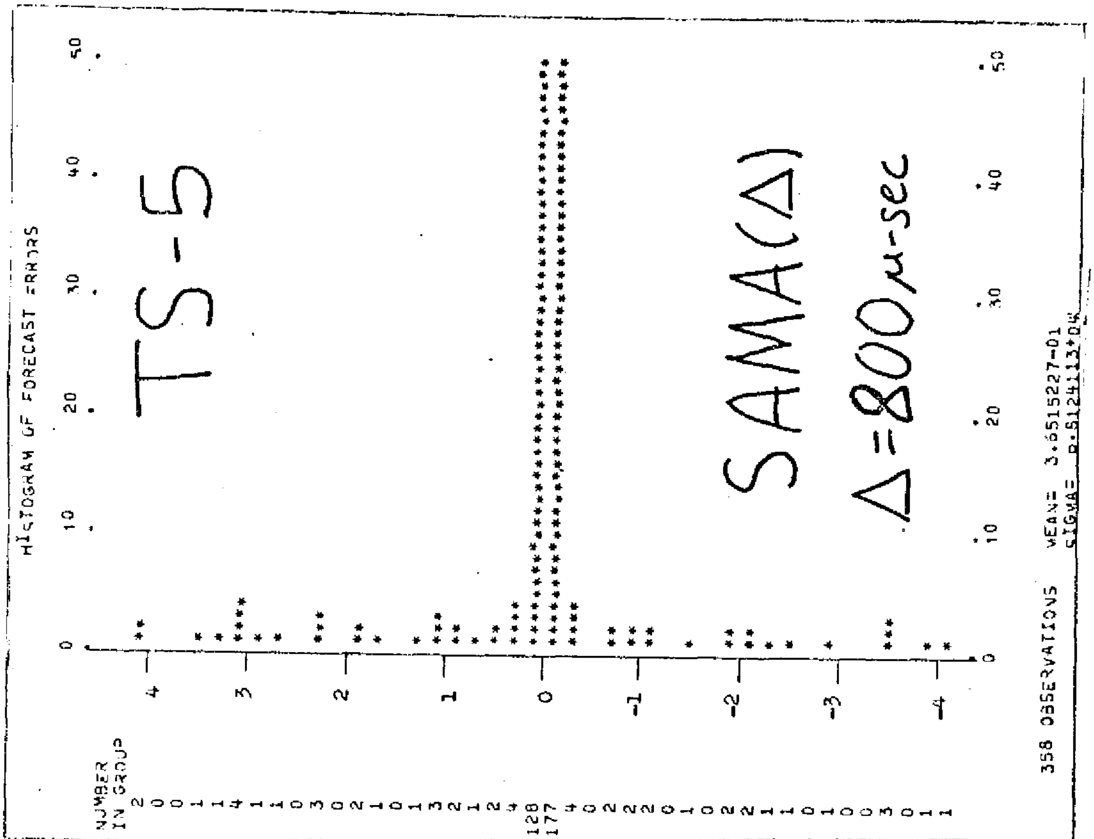

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-4
SAES (τ)
α=.1 , τ=.5
151 OBSERVATIONS
MEAN= -2.6584479+00
SIGMA= 3.9258814+03

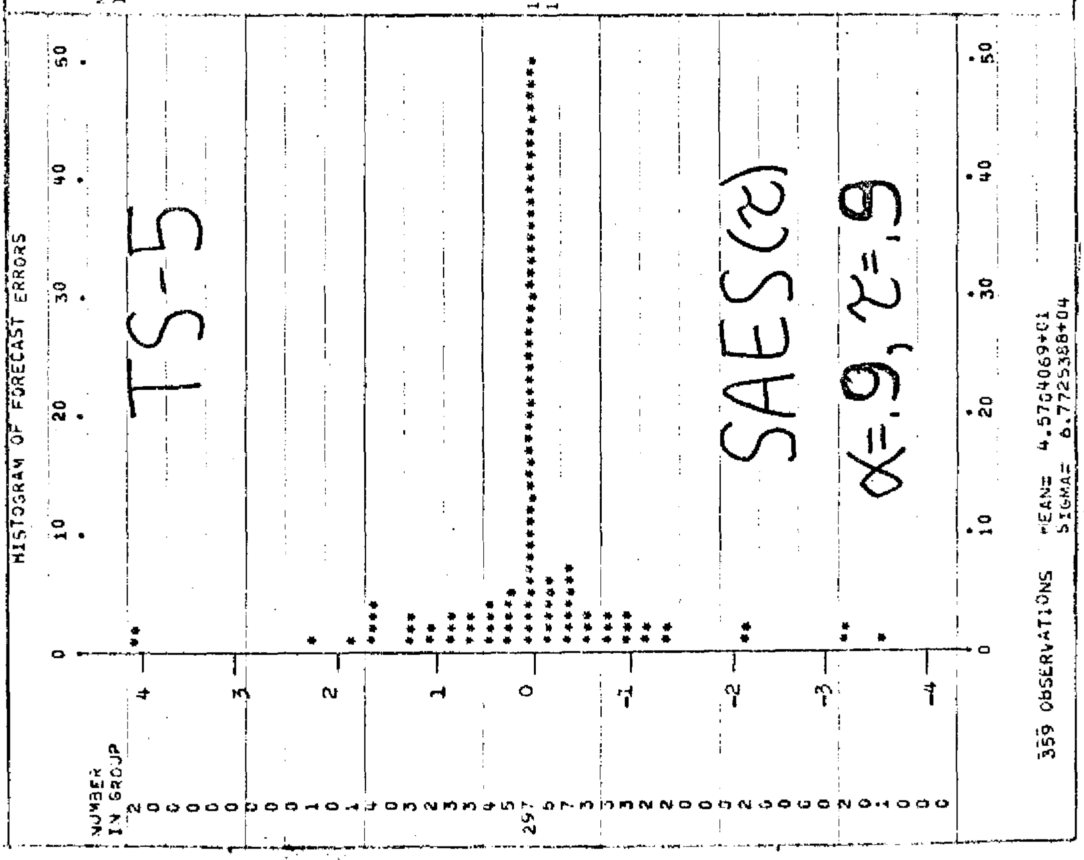

NUMBER IN GROUP
TS-4
SAMA(Δ)
Δ=800μ-sec
151 OBSERVATIONS
MEAN= 7.4285925+00
SIGMA= 3.9257637+03

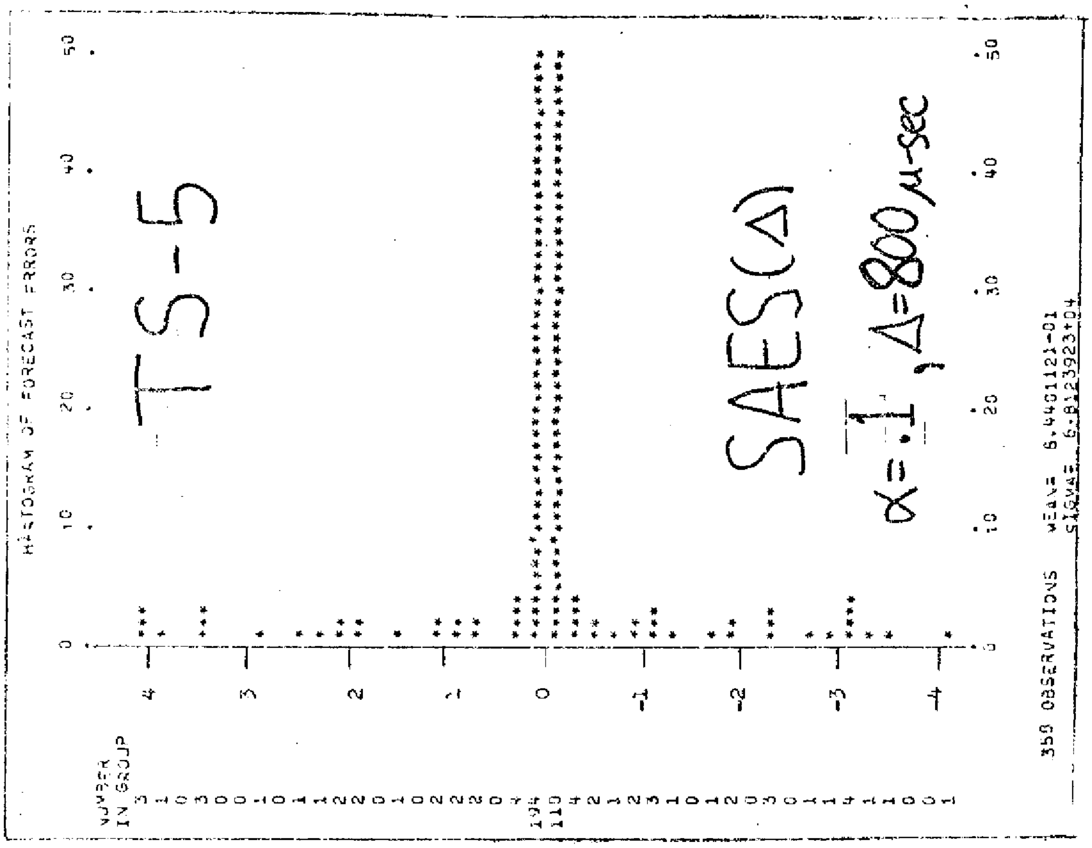

TS-4
SAES(Δ)
α = .1 , Δ = 200 μ-sec
150 OBSERVATIONS

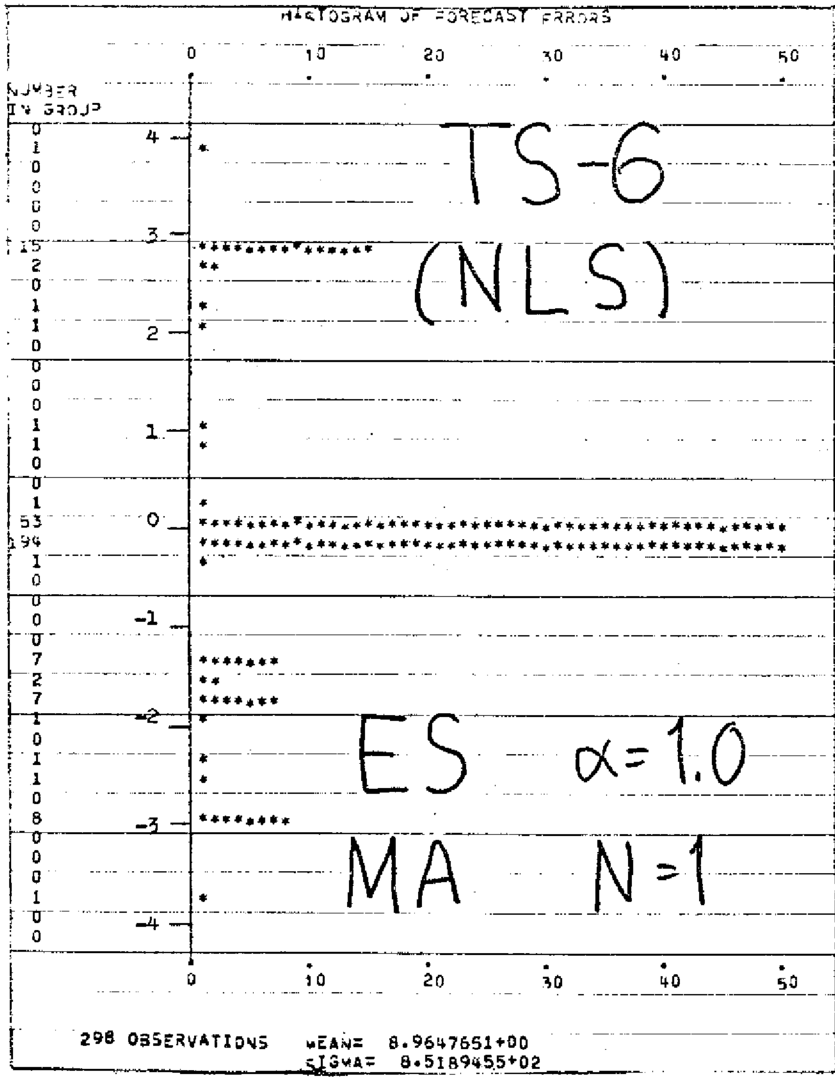

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-5
(SIM)
ES α=1.0
MA N=1
358 OBSERVATIONS
MEAN= -2.7417333-01
SIGMA= 6.8127024+04

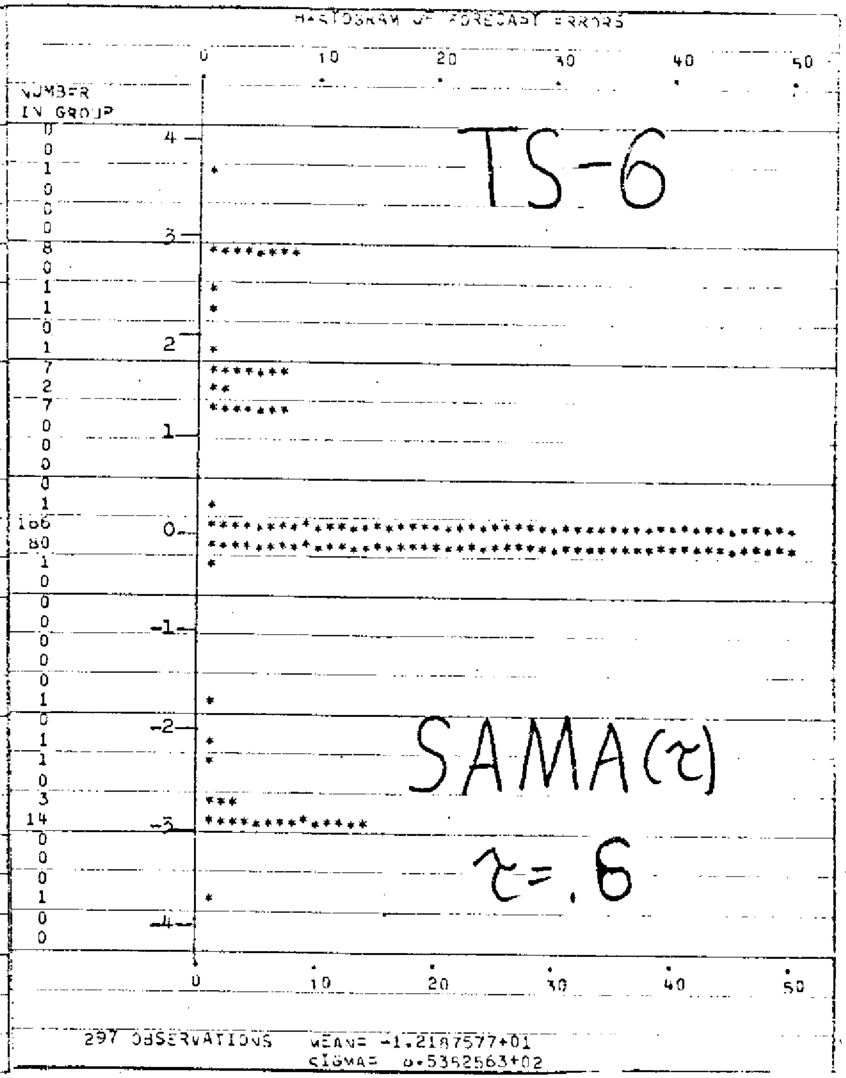

HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-5
SAMA(τ)
τ=.9
359 OBSERVATIONS
MEAN= 4.6253851+02
SIGMA= 6.6742256+04

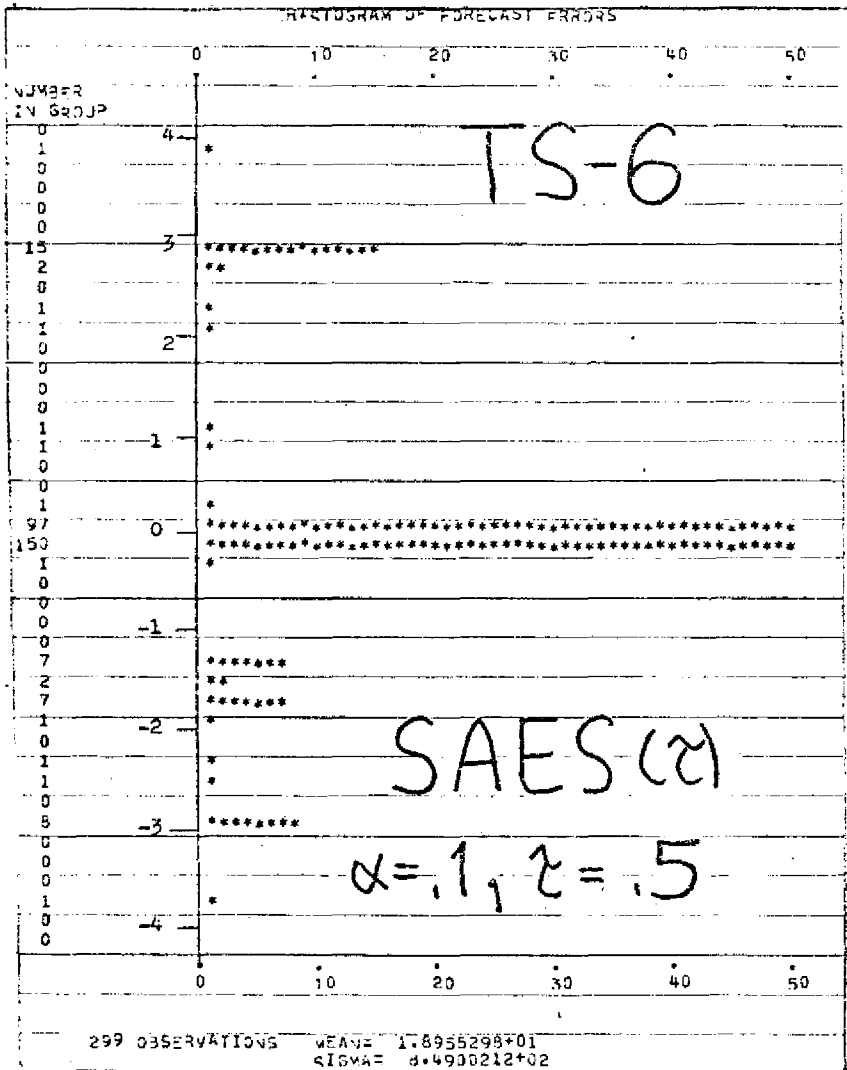

HISTOGRAM OF FORECAST ERRORS
TS-5
SAMA(Δ)
Δ=800μ-sec
358 OBSERVATIONS

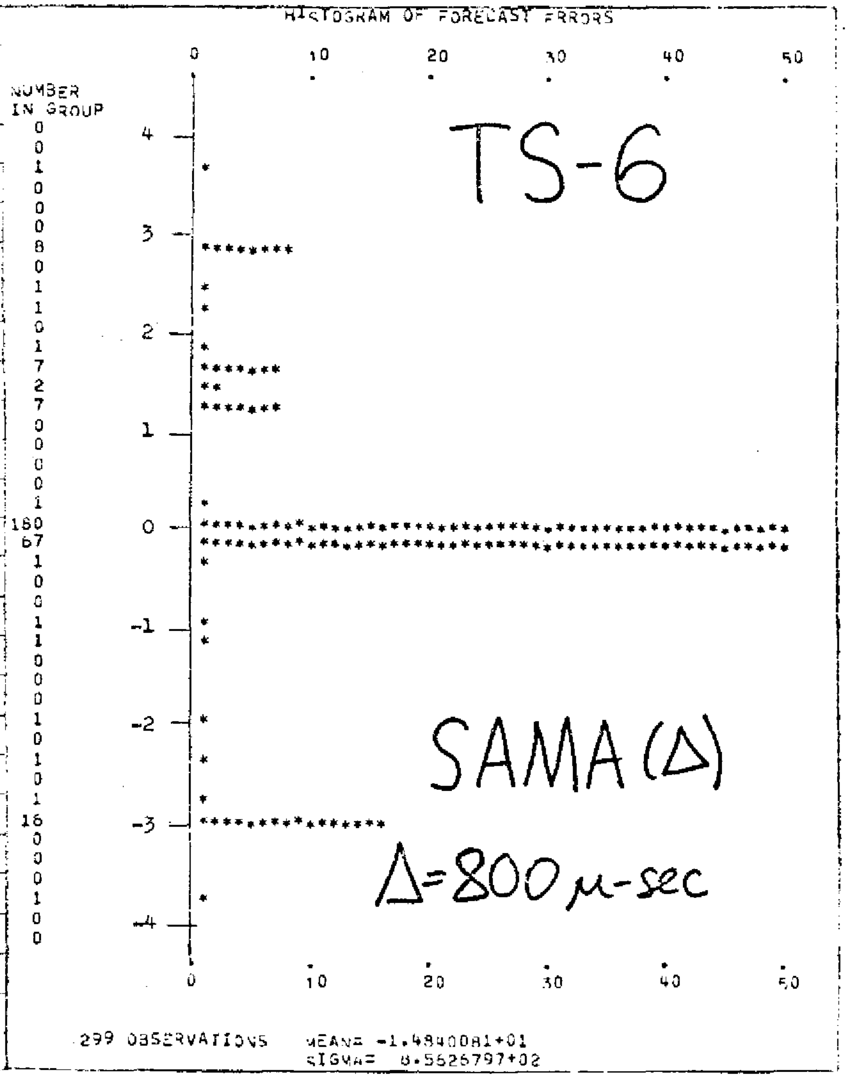

HISTOGRAM OF FORECAST ERRORS
TS-5
SAES(τ)
α=.9, τ=9
359 OBSERVATIONS
MEAN= 4.5704069+01
SIGMA= 6.7725388+04

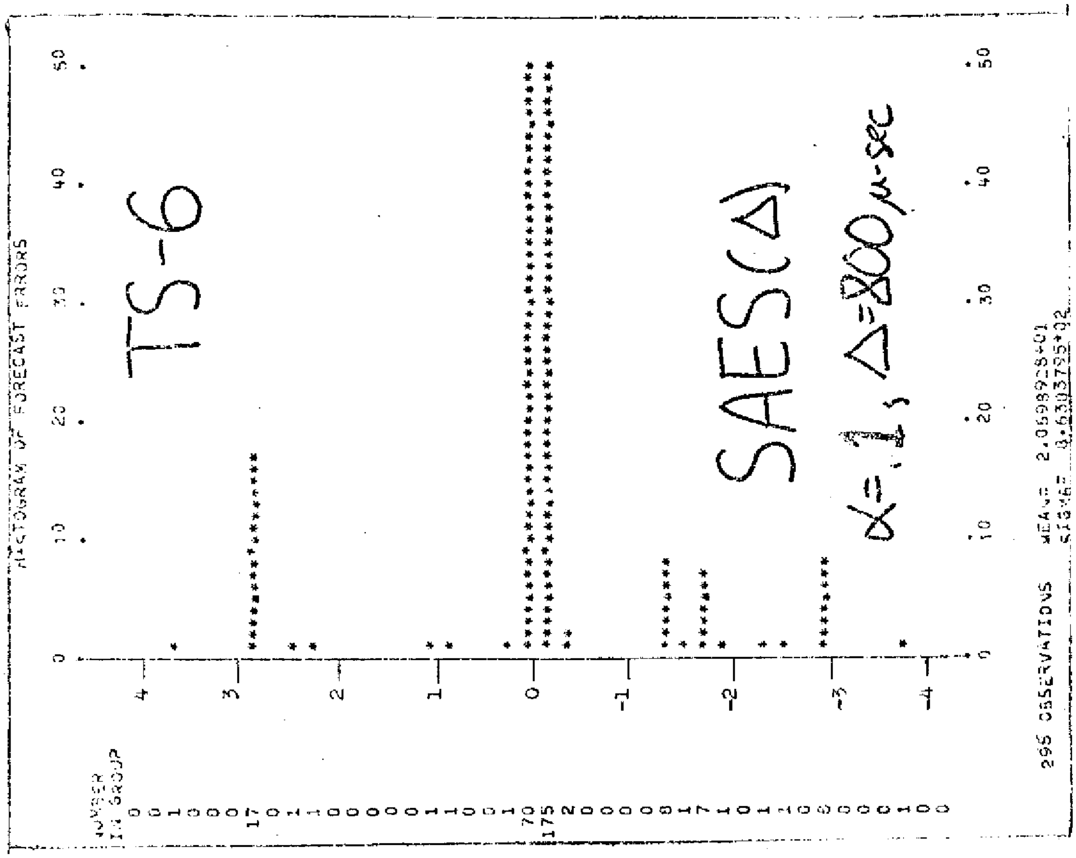

HISTOGRAM OF FORECAST ERRORS
TS-5
SAES(Δ)
α=.1, Δ=800 μ-sec
NUMBER IN GROUP
359 OBSERVATIONS
MEAN= 6.4401121-01
SIGMA= 6.8123923+04


HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-6
(NLS)
ES α=1.0
MA N=1
298 OBSERVATIONS
MEAN= 8.9647651+00
SIGMA= 8.5189455+02


HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-6
SAMA(τ)
τ=.6
297 OBSERVATIONS
MEAN= -1.2187577+01
SIGMA= 6.5352563+02


HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-6
SAES(τ)
α=.1, τ=.5
299 OBSERVATIONS
MEAN= 1.8955298+01
SIGMA= 8.4900212+02


HISTOGRAM OF FORECAST ERRORS
NUMBER IN GROUP
TS-6
SAMA(Δ)
Δ=800μ-sec
299 OBSERVATIONS
MEAN= -1.4840081+01
SIGMA= 8.5626797+02


HISTOGRAM OF FORECAST ERRORS
TS-6
SAES(Δ)
295 OBSERVATIONS

APPENDIX 3

ACTUAL TIMES FOR TS-I(COBOL) UNITS: μ-sec
(Randomized from data originally grouped into 200 μ-sec blocks).

Read Down Each Column

849.311	604.511	5059.454	26488.140
25847.931	770.081	701.814	3683.359
4295.861	684.381	283.483	5191.908
493.181	21049.910	17214.462	839.755
659.650	3362.329	5094.750	735.915
335.510	4925.058	724.349	533.363
25380.678	737.096	303.257	19079.160
3775.157	298.445	19831.475	5174.248
4918.946	31609.103	3509.003	818.546
612.044	3469.071	5135.841	612.353
654.452	2378.349	711.988	555.370
446.170	3235.936	237.445	25447.597
19787.198	3944.834	17112.212	4680.334
3577.535	402.041	5336.289	380.280
517.183	608.558	509.576	320.535
705.140	364.384	232.372	310.102
244.407	20069.521	19504.378	22598.978
20731.983	4123.957	3325.594	4337.163
3768.870	327.723	3496.320	979.659
3755.066	480.789	3816.255	561.454
490.572	383.165	485.500	397.579
375.388	20834.850	304.055	26383.003
20009.513	3635.845	15271.920	4317.738
3992.948	2785.150	4989.095	3801.782
3929.694	4085.765	655.579	3835.136
607.748	334.690	271.373	817.800
439.113	532.924	21036.477	749.774
219.788	280.468	4150.891	231.057
22749.771	27377.322	4814.615	28261.550
3229.066	3423.486	427.648	4041.553
4057.570	418.959	389.991	4170.766
435.583	363.742	301.644	3849.258
562.807	17457.835	25762.606	3877.121
239.340	5301.238	3772.879	3854.263
265.183	493.951	4132.461	580.715
22440.335	235.973	441.353	555.476
564.798	22327.305	499.555	281.548
439.570	3167.947	25107.066	24655.929
261.652	3757.899	3463.868	5179.520
25034.044	4097.160	3770.019	452.621
4555.746	5385.732	4825.460	674.931
426.757	423.613	430.211	496.552
447.079	210.804	384.271	23567.482
216.710	24147.304	20287.641	3837.722
18735.650	2833.115	4140.322	3657.272
5003.901	3868.235	542.312	4626.131
4221.461	3652.565	493.611	544.300
5388.331	3785.405	394.221	211.779

ACTUAL TIMES FOR TS-1(COBOL) UNITS: μ-sec
(Randomized from data originally grouped into 200 μ-sec blocks)

Read Down Each Column

549.311	604.511	5059.954	26444.140
25847.931	770.001	701.814	3643.369
4295.861	584.301	283.483	5191.903
493.101	21043.910	17214.462	439.756
639.650	3562.329	5094.750	735.915
335.510	4025.058	724.349	533.363
25380.578	737.096	303.257	19079.160
3775.157	298.445	19831.475	5174.243
4918.946	31609.103	3509.003	418.646
412.044	3469.071	5135.841	612.353
654.952	2578.349	711.988	555.370
445.170	3235.936	237.445	25497.597
19787.198	3944.834	17112.212	4689.334
3577.535	402.041	5335.289	480.280
517.183	608.558	509.576	620.535
705.140	364.384	232.372	310.102
244.407	20069.521	19504.378	22598.978
20731.983	4123.957	3325.594	4337.163
3768.870	327.723	3496.320	474.669
3755.066	480.789	3816.255	561.464
490.572	383.165	485.500	527.579
375.348	20834.850	304.055	26383.903
20007.513	3635.845	15271.920	4317.738
3992.948	2785.150	4989.095	3301.782
3925.594	4085.765	655.579	3835.156
607.748	334.690	271.373	417.808
439.113	532.924	21036.477	749.779
219.788	280.468	4150.891	231.057
22749.771	27377.322	4814.615	28261.650
3229.066	3423.486	427.648	4041.553
4057.570	418.959	389.991	4170.755
435.583	363.742	301.644	3849.258
562.807	17657.835	25762.606	3877.121
239.340	5301.238	3772.879	3854.263
265.183	493.951	4132.461	580.715
22440.335	235.973	441.353	555.476
564.798	22327.305	499.555	281.548
439.570	3167.947	25107.066	24655.929
261.652	3757.899	3463.888	5179.620
25034.044	4097.160	3770.019	452.621
4555.746	5385.732	4825.460	679.931
426.757	423.613	430.211	495.552
447.079	210.804	384.271	23567.482
216.710	24147.304	20287.641	3837.722
18735.650	2833.115	4140.322	3657.272
5003.901	3858.235	542.312	4626.131
4221.461	3652.565	493.611	544.300
5388.331	3786.405	394.221	211.779

TS-1 Times (Continued)

15428.568	320.966	27650.195	246.258
3994.667	13519.428	4689.023	31506.449
3710.075	4063.201	667.159	5915.951
6374.793	651.283	794.606	4874.761
788.821	405.676	271.362	582.882
552.159	401.378	25497.428	640.312
22064.807	24945.389	5472.804	247.052
3326.764	3838.711	5397.490	25403.102
3338.031	3681.342	471.486	5908.962
3698.608	973.283	694.791	5163.131
608.995	614.534	467.406	567.110
267.691	305.095	23589.330	720.399
276.198	16584.965	4060.565	222.998
234.014	3734.145	881.110	22674.906
27641.139	272.535	650.964	7076.125
3597.575	560.435	370.128	226.653
5003.320	397.545	24638.601	526.491
558.376	16383.964	4056.385	375.639
662.741	3919.693	5423.478	22174.096
516.415	204.732	539.881	3121.863
519.400	639.081	205.594	3218.941
471.694	422.739	31620.617	3065.327
373.298	27155.708	3384.949	6261.024
16424.212	4437.986	4298.592	205.030
3624.436	269.573	5161.544	700.347
373.969	650.471	373.806	343.973
472.813	380.679	535.377	25136.908

ACTUAL TIMES FOR TS-2(DIFFER)
Units: μ-sec

Read Down Each Column

55.125	6038.125	6286.875
97.000	6043.500	6311.375
2168.375	6019.000	6299.125
1427.625	6745.250	6299.125
2168.375	6160.375	6317.500
1369.125	6191.000	6311.375
4003.375	6178.750	6299.125
70.125	6172.625	6305.250
1105.125	6191.000	6317.500
70.125	6178.750	6293.000
70.125	6160.375	6311.375
70.125	6191.750	6305.250
3899.250	6179.500	6305.250
70.125	6185.625	6311.375
87.375	6173.375	6293.000
6216.500	6179.500	6318.250
6060.125	6185.625	6299.875
6054.000	6173.375	7001.625
6060.125	6179.500	6399.750
6072.375	6173.375	6405.875
6054.750	6179.500	6387.500
6034.250	6179.500	6405.875
6034.250	6173.375	6393.625
6046.500	6173.375	6412.000
6060.125	6185.625	6393.625
6038.125	6197.875	6400.500
6032.000	6180.250	6406.625
6032.000	6174.125	6394.375
6025.875	6871.000	6394.375
6038.125	6304.500	6394.375
6038.125	6286.125	6418.875
6032.000	6298.375	6382.125
6025.875	6292.250	6406.625
6032.000	6304.500	6382.125
6038.125	6304.500	6418.875
6056.500	6304.500	6394.375
		6406.625

ACTUAL TIMES FOR TS-3(METHANE)
Units: μ-sec

Read Down Each Column

63.375	6052.875	5993.750	6111.375
2146.125	6040.625	6012.125	6093.000
3299.250	6052.875	5993.750	6122.000
6215.000	6040.625	6012.125	6122.000
6197.500	6052.875	6012.125	6134.250
6197.500	5999.875	5999.875	6115.875
6203.625	6024.375	5993.750	6115.875
6222.000	5993.750	5993.750	6144.875
6209.750	5999.875	6012.125	6186.125
6222.000	5993.750	6006.000	6202.875
6203.625	6006.000	5999.875	6190.625
6203.625	5999.875	6006.000	6233.500
6175.125	6030.500	6006.000	6202.875
6193.500	5999.875	6012.125	6209.000
6128.250	6018.250	6006.000	6196.750
6109.875	6012.125	6006.000	6202.875
6122.125	6018.250	5987.625	6215.125
6122.125	6006.000	6012.125	6227.375
6128.250	5999.875	6035.000	6202.875
6128.250	5981.500	6035.000	6165.000
6081.375	5999.875	6028.875	6202.875
6087.500	5999.875	6035.000	6209.000
6069.125	5987.625	6035.000	6209.000
6087.500	5999.875	5984.875	6215.125
6087.500	6024.375	6070.125	6215.125
6093.625	5993.750	6064.000	6196.750
6081.375	6012.125	6051.750	6215.125
6087.500	6012.125	6064.000	6215.125
6081.375	6012.125	6070.125	6190.625
6087.500	5943.625	6051.750	6196.750
6046.750	6006.000	6051.750	6227.375

TS-4(OUT-OF-KILTER)
Units: μ-sec

Read Down Each Column

55.125	2642.875	2685.375	3063.125
1293.625	156.500	156.500	3079.500
1369.625	2642.875	2694.750	3079.500
1410.875	156.500	156.500	3063.125
479.875	2642.875	2685.375	3079.500
92.000	156.500	156.500	3079.500
97.000	2656.000	2694.750	3095.875
2827.250	156.500	156.500	3095.875
1221.125	2660.750	2694.750	3079.500
954.625	156.500	156.500	3063.125
590.875	1607.750	2694.750	3079.500
156.500	156.500	156.500	3063.125
2590.000	2655.500	2694.750	3079.500
156.500	156.500	156.500	3079.500
2594.750	2669.250	31776.875	3079.500
156.500	156.500	3047.875	3079.500
2603.250	2673.000	3048.250	3081.000
156.500	156.500	3046.750	197.875
2608.000	2668.375	3046.750	70.125
156.500	156.500	3046.750	526.250
2608.000	2668.375	3048.250	1051.125
156.500	156.500	3048.250	1144.750
2611.750	2681.500	3046.750	1137.750
156.500	156.500	3046.750	1144.000
2616.500	2690.000	3063.125	1144.000
156.500	156.500	3063.125	1136.250
2616.500	2685.375	3063.125	1136.250
156.500	156.500	3063.125	1144.000
2620.250	2694.750	3064.625	1160.375
156.500	156.500	3063.125	1144.000
2634.375	2634.375	3063.125	1144.000
156.500	156.500	3079.500	1154.125
2634.375	2704.125	3079.500	1152.625
156.500	156.500	3081.000	76.750
2638.125	2704.125	3063.125	.750
156.500	156.500	3079.500	6.375
2642.875	2694.750	3079.500	150.000
156.500	156.500	3063.125	13.875

ACTUAL TIMES FOR TS-5(SIM) Units: μ-sec
(Randomized from data originally grouped into 200 μ-sec blocks)

Read Down Each Column

749.311	304.511	69.454	44.140	29.568
247.951	370.001	301.814	43.369	394.567
95.951	384.801	83.483	391.908	310.075
93.181	148.910	214.452	289.755	173.794
39.550	262.329	94.750	136.915	188.821
135.510	125.058	324.349	333.383	352.159
180.579	337.095	103.257	270.161	64.807
175.157	298.445	31.475	174.248	326.754
118.946	9.103	309.903	219.645	338.031
12.049	269.071	335.841	12.353	98.608
254.452	279.349	311.988	355.370	8.495
245.170	36.936	37.445	47.697	67.591
187.198	144.834	112.212	289.334	276.198
77.535	202.041	336.289	280.280	239.014
117.183	208.558	309.676	20.536	141.140
106.140	364.384	232.372	310.102	397.575
44.907	269.521	104.378	143.978	203.321
331.983	323.967	125.694	337.163	158.376
369.870	327.723	96.320	274.659	262.741
355.066	80.789	16.255	161.464	316.415
290.572	183.165	285.500	397.579	319.400
375.388	234.850	104.055	183.003	71.634
209.513	235.845	271.920	317.738	373.298
392.948	386.150	189.095	1.762	224.212
325.694	285.765	255.579	235.135	224.436
7.748	134.690	71.373	217.800	373.959
239.113	132.924	36.477	149.774	72.813
219.768	280.468	150.891	31.057	320.966
349.772	177.322	214.515	61.650	118.428
229.066	223.486	27.648	241.553	65.201
257.670	218.959	389.991	370.766	361.263
235.583	363.742	301.644	249.268	206.676
362.807	57.835	352.607	277.121	1.378
239.340	301.238	372.879	254.253	345.389
265.183	293.951	132.461	180.715	239.711
240.335	235.973	41.353	256.476	81.342
364.798	127.305	299.555	81.548	273.283
238.570	167.947	107.067	55.929	214.534
261.552	357.899	263.888	179.520	105.095
234.044	97.160	170.019	252.521	144.955
355.746	385.732	225.460	274.931	134.145
226.757	23.513	30.211	246.552	272.635
247.079	210.804	384.271	367.462	160.435
216.710	347.304	287.642	237.722	197.545
135.550	33.115	140.322	57.272	383.964
203.901	63.235	342.312	226.131	319.693
221.461	52.505	93.611	344.300	4.732
189.331	386.405	394.221	211.779	239.081

TS-5 (Continued)

22.739	228561.541	335079.152	175.570
355.708	378773.801	353970.703	345.419
237.986	319335.375	318511.570	66.577
259.573	317046.254	329001.746	536.046
50.471	423706.445	356341.230	154.824
380.679	145115.949	279030.023	122.912
260.196	424274.754	296868.129	240.310
89.023	216582.879	375455.543	307.017
267.159	455440.305	180192.270	723.035
194.606	217847.051	439878.301	288.362
71.362	393803.098	225113.643	2.999
297.428	259108.459	2099.301	266.945
272.804	400563.125	232.264	80.202
197.490	194167.109	115.536	242.768
5071.485	431920.395	149.119	354.544
294.791	302622.996	330.011	15.830
67.406	215874.904	261.213	226.326
557989.320	410076.121	341.725	385.131
206260.564	249225.650	371.547	295.246
335081.109	373126.488	344550.676	153.671
320550.961	311775.537	274279.113	361.406
296970.125	315774.090	338956.867	318.450
358638.598	385321.859	284583.926	224.805
342056.383	305618.937	302160.297	80.469
328823.477	312665.324	115885.981	285.443
369939.879	291861.020	160.972	239.726
218405.594	303206.027	185.272	143.320
363420.513	307300.344	158.862	195.223
209384.947	317543.969	81.801	398.436
441098.590	288935.906	354.031	149.959

ACTUAL TIMES FOR TS-6(NLS)
Units: μ-sec

Read Down Each Column

55.125	343.375	1715.000	2774.625
97.000	2751.500	259.625	2794.375
999.500	2778.125	343.375	1643.000
1824.875	2772.000	2774.000	259.625
257.250	2794.500	2794.500	343.375
343.375	2788.375	2794.500	2767.875
2364.750	2778.875	2794.500	2795.250
2391.375	2795.250	2794.500	277.000
2401.625	2788.375	2789.125	343.375
2407.750	2794.500	2795.250	2774.625
2418.000	2789.125	2789.125	2794.375
2408.500	2789.125	2788.375	1581.000
2408.500	2788.375	2794.500	255.875
2401.625	2788.375	2795.250	92.125
2424.125	2789.125	2801.375	92.125
2408.500	2795.250	2789.125	92.125
2402.375	2789.125	277.000	92.125
2401.625	2794.500	343.375	343.375
2402.375	2794.500	2793.000	2068.750
2414.625	2794.500	2787.500	155.000
2401.625	2795.250	2787.500	3408.750
2402.375	277.000	2793.625	257.250
277.000	343.375	2793.625	343.375
343.375	2774.625	2794.375	2767.875
2719.375	2749.750	2788.250	2788.375
2738.375	2787.500	2787.500	2788.375
2760.875	2793.625	2788.250	2788.375
2754.750	2793.625	2794.375	2794.500
2771.125	2787.500	2788.250	2789.125
2771.125	2788.250	2787.500	2772.000
2761.625	2794.375	2788.250	2789.125
2728.000	2794.375	1715.000	2800.625
2778.000	2793.625	259.625	2789.125
2771.875	2793.625	343.375	2772.000
2778.000	2788.250	2730.125	2789.125
2777.250	2788.250	2794.500	2789.125
2777.250	2793.625	2794.500	2789.125
2778.000	2793.625	2788.375	2788.375
2727.250	2788.250	2789.125	2794.500
2778.000	2787.500	2789.125	2789.125
2777.250	2788.250	2795.250	2789.125
2771.875	2793.625	2795.250	2794.500
2777.250	2788.250	2788.375	2794.500
2771.875	2788.250	2788.375	2795.250
2777.250	2794.375	2795.250	2789.125
2771.875	2787.500	277.000	2788.375
1731.375	2793.625	343.375	2788.375
259.625	2794.375	2744.625	2789.125

TS-6 (Continued)

277.000	2789.125	2789.125
343.375	2795.250	2800.625
2768.500	277.000	2789.125
2787.500	343.375	2789.125
2787.500	2774.625	277.000
2793.625	2793.625	343.375
2793.625	2787.500	2768.500
2793.625	2793.625	2787.500
2800.500	2793.625	2737.500
2794.375	2788.250	2793.625
2794.375	2788.250	2788.250
2787.500	2787.500	2794.375
2787.500	2794.375	2793.625
2794.375	2794.375	2787.500
2788.250	2793.625	2788.250
2794.375	2799.750	2794.375
2787.500	2788.250	2788.250
2793.625	2788.250	2787.500
2788.250	2787.500	2787.500
2794.375	2787.500	2788.250
2793.625	2794.375	2788.250
2793.625	2788.250	1715.000
2788.250	1715.000	259.625
2788.250	259.625	343.375
2787.250	343.375	2767.875
2794.375	2774.000	2794.500
1731.375	2788.375	2794.500
259.625	2794.500	2789.125
343.375	2788.375	2795.250
2767.875	2789.125	2794.500
2788.375	2789.125	2794.500
2788.375	2794.500	2795.250
2788.375	2795.250	2789.125
2794.500	2795.250	277.000
2795.250	2788.375	343.375
2795.250	2794.500	2768.500

BIBLIOGRAPHY

1. Batson, A., Ju, S. M., and Wood, D. C., "Measurements of Segment Size," Comm. of ACM (March), pp. 155-159.

2. Bell, C. G., Chen, R., and Rege, S., "Effect of Technology on Near Term Computer Structures," Computer, Vol. 5, No. 2 (1972), p. 29.

3. Blackman, R. B., Linear Data-Smoothing and Prediction in Theory and Practice Reading. Addison-Wesley Publishing Company, Inc., 1965.

4. Boer, J. L., "A Survey of Some Theoretical Aspects of Multiprocessing," ACM Computing Surveys, Vol. 5, No. 1 (1973), p. 31.

5. Box, G. E. P., and Draper, N. R., Evolutionary Operation. John Wiley & Sons, Inc., New York, 1969.

6. Box, G. E. P., "Evolutionary Operation: A Method for Increasing Industrial Productivity," Applied Statistics, Vol. 6, No. 2 (June, 1957), pp. 81-101.

7. Box, G. E. P., and Jenkins, G. M., Time Series Analysis. Holden-Day, Inc., San Francisco, 1970.

8. Bright, H. S., "A Philco Multiprocessing System," AFIPS SJCC, Vol. 18 (1964), p. 97.

9. Brown, R. G., Smoothing, Forecasting, and Prediction of Discrete Time Series. Prentice-Hall, Inc., Englewood Cliffs, 1963.

10. Brown, R. G., and Meyer, R. F., "The Fundamental Theorem of Exponential Smoothing," J. Oper. Res. Soc. Amer., Vol. 9 (1961), pp. 673-686.

11. Burgess, J. T., "Adaptive Forecasting," Presented at the 13th Annual TIMS Meeting, Philadelphia, Penn., Sept. 6-8, 1966.

12. Chow, W. M., "Adaptive Control of the Exponential Smoothing Constant," Journal of Industrial Engineering, Vol. 16, No. 5 (Sept.-Oct., 1965), pp. 314-317.

13. Cox, D. R., "Prediction by Exponentially Weighted Moving Average and Related Methods," Journal of Royal Statistical Society, Vol. B23, No. 2 (1961), pp. 414-422.

14. Critchlow, A. J., "Generalized Multiprocessing and Multiprogramming Systems," AFIPS FJCC, Vol. 24 (1963), p. 109.

15. Deutsch, S. J., Time Series Modeling. Manuscript distributed to ISyE 6402 students, Georgia Tech, 1973.

16. Devereanz, J. A., "An Application-Oriented Multiprocessor System-Control Program Feature," IBM Systems Journal, Vol. 6, No. 2 (1967), p. 95.

17. Fagan, T. L., and Wilson, M. A., "Exponential Smoothing for Prediction of Reliability Growth," Industrial Quality Control, Vol. 23, No. 7 (January, 1967), pp. 319-323.

18. Ferguson, D. E., The Development of a Self-Adaptive Prediction and Control System. M.S. Thesis, Georgia Tech, Atlanta, Georgia, 1971.

19. Lampson, B. W., "A Scheduling Philosophy for Multiprocessing Systems," Comm. of ACM, Vol. 2, No. 5 (May, 1968), pp. 347-360.

20. Leland, O. M., Practical Least Squares. McGraw-Hill Book Company, Inc., New York, 1921.

21. Lindgren, B. W., and McElrath, G. W., Introduction to Probability and Statistics. MacMillan Company, New York, 1969.

22. Luehrmann, A. W., and Nevison, J. M., "Computer Use Under a Free-Access Policy," Science, Vol. 184, No. 4140 (May, 1974), pp. 957-961.

23. Madnick, S. E., "Multi-processor Software Lockout," Proc. ACM National Conference (1968), pp. 19-24.

24. Montgomery, D. C., "An Introduction to Short-Term Forecasting," Journal of Industrial Engineering, Vol. 19 (November 10, 1968), pp. 500-504.

25. Montgomery, D. C., "Adaptive Control of the Exponential Smoothing Parameters by Evolutionary Operation," AIIE Transactions, Vol. 2, No. 3 (September, 1970), pp. 268-269.

26. Muth, J. F., "Optimal Properties of Exponentially Weighted Forecasts," Amer. Stat. Assoc. Journal, Vol. 55 (1960), pp. 299-306.

27. Nelder, J. A., and Mead, R., "Adaptive Control of Exponential Smoothing Parameters by Evolutionary Operation," AIIE Transactions, Vol. 2, No. 3 (September, 1970), pp. 308-313.

28. Pass, E. M., An Adaptive Michroscheduler for a Multiprogrammed Computer System. Ph.D. dissertation, Georgia Tech, Atlanta, Georgia, 1973.

29. Raynor, R. S., Minimization of Supervisor Conflict for Multiprocessor Computer Systems. Ph.D. dissertation, Georgia Tech, Atlanta, Georgia, 1974.

30. Roberts, S. D., and Reed, R., Jr., "The Development of a Self-Adaptive Forecasting Technique," AIIE Transactions, Vol. 1, No. 4 (December, 1969), pp. 314-322.

31. Roos, C. F., "Survey of Economic Forecasting Techniques," Econometrica (October, 1955), pp. 363-395.

32. Sherman, S., Baskett, F., and Browne, J. C., "Trace-Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System," Comm. of ACM, Vol. 15, No. 12 (December, 1972), pp. 1063-1069.

33. Stewart, K. B., "A New Weighted Average," Technometrics, Vol. 12 (May, 1970), pp. 247-257.

34. Stratonovich, R. L., Topics in the Theory of Random Noise, Vol. 1 translated by Richard A. Silverman. Gordon and Breach Science Publishers, Inc., New York, 1963.

35. Trigg, D. N., and Leach, A. G., "Exponential Smoothing with an Adaptive Response Rate," Operational Research Quarterly, Vol. 18, No. 1 (March, 1967), pp. 53-59.

36. Wichern, D. W., "The Effect of Estimation Error on a Geometric Moving Average," Technometrics, Vol. 14, No. 3 (August, 1972), pp. 745-755.

37. Witt, B. I., "M65MP: An Experiment in OS/360 Multiprocessing," Proc. 1968 ACM National Conference (1968), p. 691.

38. Yap, Y., A Comparison of Two Adaptive Prediction Systems. M.S. thesis, Georgia Tech, Atlanta, Georgia, 1973.

39. Young, D., Unpublished class notes, Georgia Tech Graduate Course ISyE 6305, "Forecasting Systems," 1973.

40. Zadeh, L. A., and Ragazinni, J. R., "An Extension of Wiener's Theory of Prediction," Journal of Applied Physics, Vol. 21 (July, 1950), pp. 645-655.