NASA CONTRACTOR REPORT 165948

(NASA-CR-165948) TOWARD AUTONOMOUS SPACECRAFT Final Report (Decision Science, Inc.) 125 p HC A06/MF A01 CSCL 22B

N82-31403

Unclas
G3/18 28759

TOWARD AUTONOMOUS SPACECRAFT

LAWRENCE J. FOGEL, PHILIP G. CALABRESE,

MICHAEL J. WALSH, AND ALVIN J. OWENS

DECISION SCIENCE, INC.
SAN DIEGO, CA 92117

Contract NAS1-16621

June, 1982

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

NASA CONTRACTOR REPORT 165948

TOWARD AUTONOMOUS SPACECRAFT

LAWRENCE J. FOGEL, PHILIP G. CALABRESE,
MICHAEL J. WALSH, AND ALVIN J. OWENS

DECISION SCIENCE, INC.
SAN DIEGO, CA 92117

Contract NAS1-16621
June, 1982

National Aeronautics and
Space Administration

Langley Research Center
Hampton Virginia 23665

CONTENTS

INTRODUCTION

Modern spacecraft represent a significant investment. The return on this investment can only be realized if these craft adequately perform the assigned missions in spite of the stringent environment of outer space and unexpected circumstance. The latest advances in engineering technology are brought to bear to overcome the challenge posed by the environment, and yet the designer recognizes his inability to foresee all possible circumstance. Therefore, to compensate for the unexpected, the control of modern spacecraft remains largely in human hands.

Attempts have been made to automate those lower level routine functions that must be performed within the system. The human operator is only called upon to manage these by exception... to assume control whenever there is an indication of inadequate performance in some particular regard. He can compensate for failures by reconfiguring the mechanism, or if necessary, altering the mission. The higher levels of decision making remain solely the prerogative of the human operator. The need for close control of the developing situation dramatizes the potential value of having man aboard the spacecraft.

Yet two important missions stand in need of taking new steps toward autonomous spacecraft. The first of these concerns exploration of deep space. Here, it is clearly unsuitable to include the human operator aboard the vehicle. Further, remote control becomes inadequate in view of the significant communication delays. Such spacecraft must be intelligently capable of evaluating a variety of opportunities and coping with unexpected threats, for only then can the mission be completed with the greatest degree of success.

So long as the intent of the exploration is defined in a strict sense of what is sought, the findings are unduly constrained. The value of having a man onboard largely rests upon

his ability to understand the mission in some broader sense so that he can take advantage of "targets of opportunity" and modify the mission to explore previously unforeseen avenues. By the same token, he would be in a position to more properly evaluate unanticipated dangers and, recognizing these, alter the course of the mission or take other appropriate actions. In essence, the challenge is to devise logical processes that can perform this sophisticated function. We must incorporate a decision-making mechanism that simulates some of the essential features of human intelligence, at least to the extent of referencing the broad scope of the mission intent and optimal selection of response behavior in the light of that purpose and the developing situation.

The second mission concerns the retrieval of spacecraft that require refurbishment or may have failed in orbit. The space shuttle is equipped to accomplish this for low altitude target objects. It seems reasonable to expect similar retrieval devices for synchronous orbit target objects in the near future.

Control of the retrieval mechanism requires prediction of the relative orientation and motion of the target object, thus making it possible to safely approach, contact, and bring that object aboard the shuttle. But, the target object may behave in an erratic manner. Contact with it may cause a change of its internal state and perhaps activation of its propulsion system. Further, the retrieval mechanism may behave in a complex manner when coupled with such a target object. Even if the target is passive, perfect reliability of the retrieval mechanism is never assured. It is therefore essential to design for "graceful failure" in that inappropriate retrieval might prove disasterous for the shuttle.

Both of these missions encompass the more general problem wherein it's desirable to approach and investigate or evade some particular object in space. Some target objects of interest might be less well-known than our own failed satellites. The

very presence of the "shuttle" may affect the behavior of such an object . . . causing the object to adopt a collision course (through a gravitational attraction or as a result of the programmed propulsion of a space mine). Alternatively, the object might take an evasive course (in the case of a foreign satellite programmed to avoid being captured).

The situation becomes even more complex if the target object can operate at some higher level of intelligence. For example, it may be purposive within some context that includes the friendly spacecraft. Note that if there is adequate remote monitoring and close control capability, this intelligence need not be onboard the target object. In the case of space exploration, a number of friendly spacecraft may be assigned to cooperate by performing complementary functions. Here their interactive behavior requires gaming in an effective manner so that they benefit one another and operate collectively to best support the accomplishment of the mission. Under certain circumstances a single spacecraft of the consort might be assigned a terminal mission, this in order to gain and transfer the knowledge required to increase the likelihood of success for those that remain.

The purpose of this investigation has been to explore some ways in which autonomous behavior can be extended to treat situations wherein close control by the human operator may not be appropriate or even possible.

DISCUSSION

Intelligent behavior begins with holding a concise understanding of what is to be accomplished. Ordinarily, purpose is depicted only in terms of the most desirable outcome. . . with some consideration being given to those alternative futures that are considered most undesirable. In point of fact, purpose becomes well defined if, and only if, it consists of a statement of the relative worth of each of the significantly different futures . . . this being expressed in the form of a hierarchic valuated state space and appropriate normalizing function.

Intelligent behavior also requires an adequate sensing system. There is some chance that the environment will be as desired, but it is more likely to experience the contrary. Opportunities may be some distance off-course, and there may be stumbling blocks or greater dangers directly in the way. The sensor system must allow observation in such a way as to enable pattern recognition. . . re-cognition, knowing again what has been known before. Simply stated, pattern recognition consists of comparing the observed environment to similar templates referenced from memory. A decision is made as to which of these templates is most like the present observation. Note that this process yields only a limited understanding . . . one restricted to the vocabulary of templates; most endeavors to improve pattern recognition are concerned with selecting a most suitable nearness metric and finding convenient means for computing the error.

More sophisticated purposive behavior involves classification . . . the discovery of useful templates. Here the intent is to characterize the observed environment in terms of the existing regulations. Mathematical techniques in clustering, factor analysis, and discriminant function analysis provide meaningful ways to group data points within a predefined state space. Here again, the task is to define that space in terms of axes and a distance metric.

Decisions are made in sequence. It is therefore essential to find temporal regularities and extend these for the sake of predicting the environment. Purposive behavior hinges on an ability to predict the environment in order to anticipate opportunities and avoid threats. Lacking an ability to predict one's environment generally precludes intelligent behavior.

Although the process of prediction can be identified with the scientific method, it is more convenient to briefly state that forecasting requires definition of prediction span (the time of future concern), an ability to retrieve the sensed data from memory, and a criterion that specifically indicates the relative worth of each of the possible correct and incorrect predictions. This allows developing a model of those regularities that can then be extended to yield a most appropriate forecast on the basis of what is known and the criterion of predictive performance. Note that this process requires pattern recognition in the sense that the recorded data base must be referenced in terms of the given criterion (payoff matrix, error cost function, predictive goal). It also requires classification in the sense that the regularities already experienced must be identified before these can be considered in consort and collectively extended to yield a forecast. Indeed, the process of prediction is necessarily inductive and therefore cannot be performed with perfect certainty. Efficient structuring of a useful model is the very essence of creativity.

Although the literature is replete with numerous methods for prediction, most of these treat the process of forecasting only with respect to the least mean squared error criterion. This tradition has grown in view of analytic procedures which are made far more tractable with this criterion. For example, Regression Analysis and Fourier Analysis provide methods for forecasting time series on the basis of the least mean squared error criterion. But in the real world, equally correct interpretations are <u>not</u> of equal worth, and the various errors of forecasting

usually are attributed widely different costs. Appendix A indicates a specific method for extracting cyclic components from an arbitrary environment with respect to the usual criterion and with respect to an arbitrary criterion. Such a method is essential for treating environments wherein the best prediction is not simply the most likely future, but rather that future which reflects the underlying purpose of the prediction.

The credibility of any predictive model can only be determined by examining the validity of its forecasts over time. In general, the same model is used again and again so long as its predictions are of sufficient worth. If, on the other hand, the model proves untrue (that is, sufficiently costly), its credibility is degraded. It then becomes worthwhile to introduce other uncorrelated data for the sake of generating a new model worthy of testing. In point of fact, models (theories, conjectures, hypotheses, rules, laws, and so forth) must include information beyond that contained within the data base. Ordinarily, such additional information comes from prior experience using different types of models in similar problem domains. Without such learning, the creative process is reduced to a selection upon randomness.

The credibility of each predictive model can be estimated by comparing the forecast of that model to the most recently experienced data. It is tempting, but invalid, to extend the mathematical model into the recent past to yield postdictions to be compared point by point with respect to the error criterion, for here the same data base is being used twice. The proper procedure is to truncate the data at, say, a point in recent time comparable in span to the time interest of the forecast, then generate a new model based on the truncated data, then evaluate its predictions against the recent past. Presumably, the same kind of model would have a similar credibility.

Prediction is a basis for control. Each prediction is based on a model that represents the underlying logic of the

environment. If the environment is responsive to stimulation, then a prediction of its response in the light of a recent sequence of stimuli is based on a model of the transduction. Control theory was developed with the intent of causing a linear plant to behave in a desired manner. Some treatment was then given to certain particular nonlinearities in that all real world transduction is nonlinear. The problem of identification arises when it becomes necessary to characterize an unknown plant. More carefully stated, the task is to select from the available resources that stimulus which is most likely to cause the unknown environment to yield the desired response. And this is a simplistic view, for it is important to understand when the specific desire cannot be realized and, if it cannot be, then what stimulus is most desirable in the sense of yielding a worthwhile, if not most desirable, response. Modern control theory does not treat this problem in complete generality but rather offers particular approximations on the basis of limiting assumptions. In general, there should be no such presumption. The unknown plant may be linear or nonlinear, passive or active, possibly even intelligently cooperative, ambivalent or competitive.

Evolutionary programming provides a general approach for prediction and control in this regard. Preliminary findings on such programming have been reported in the literature. However, a new program was written wherein finite state machines are scored in their ability to predict each data point in the most recent portion of the experienced data stream, this with respect to an arbitrary criterion. An original machine is chosen at random or on the basis of assumptions concerning the underlying regularities within the environment. This parent machine is then mutated in a random manner to yield an offspring which is then similarly scored in terms of its ability to forecast each next point in the recent past. If the score for this offspring is less than that of the parent, that offspring is discarded, and a new offspring is generated. If, however, the offspring is superior to the parent, this offspring becomes a new parent.

Such nonregressive evolution proceeds in fast time until a higher level criterion is reached. For example, the predictive file may reach a sufficient level, or the computational time or space may run out. The resulting machine is then exercised to yield the required forecast.

This prediction is then compared with the actual next state of the environment, and the question is raised as to the next symbol. Here the machine used for the last prediction becomes the progenitor of the next evolutionary exploration, for surely some useful regularity must have been found, even if the current prediction may be in error.

It is convenient to include a cost for complexity in the structure indicating the worth of each evaluated machine. In essence, this embodies the Maxim of Parsimony. If this factor is small, the evolving machines grow in complexity to express each regular aspect of the environment in the light of the criterion. If this factor is large, the machines are reduced to an over-simplified view of these regularities. Note that a periodic environment of arbitrary cycle can be perfectly represented by a single state machine. The program written for such evolutionary prediction and modeling permits a variable alphabet size, arbitrary predictive criteria and includes an inner loop scored in terms of its ability to forecast each next point in the recent past. If the score for this offspring is less than that of the parent, that offspring is discarded, and a new offspring is generated. If, however, the offspring is superior to the parent, this offspring becomes a new parent. Such nonregressive evolution proceeds in fast time until a higher level criterion is reached. For example, the predictive file may reach a sufficient level, or the computational time or space may run out. The resulting machine is then exercised to yield the required forecast.

This prediction is then compared with the actual next state of the environment, and the question is raised as to the next

symbol. Here the machine used for the last prediction becomes the progenitor of the next evolutionary exploration, for surely some useful regularity must have been found, even that allows the nature of the mutation noise to be a function of the prior success of that kind of noise in the evolutionary process. This method for prediction provides a significant advantage with respect to modeling and closing the control loop for an arbitrary environment.

Each prediction is compared with the next actual output to yield a measure of the credibility of the identification process. When sufficient credibility has been reached, the model of the plant can be used as a basis for closing the loop in an appropriate manner. In essence, successful prediction confirms the model as a replica of the plant.

Here is a critical aspect of control loop design and yet a straightforward logic permits determination of each next optimal resource assignment. The logic references the control goal (a valuated state space that portrays each of the significantly different futures and their relative worth), the allocable resources at that moment in time, and the finite-state machine feedback from the predictor, reference Figure 1.

Examine only the present state of that machine. Each of the transitions from that state is examined in an exhaustive manner to determine if any transition indicates the more desired output from the plan. If so, the related input is noted, and there is a test of the inventory of resources to determine if such a stimulus can be invoked to yield the desired response. If either the output symbol or the required resource is not found, reference is made to the next most desirable state in terms of the corresponding output symbol and required resource assignment. The process continues until a commitment is made or there is a determination that no meaningful options are open. For example, if the model is the machine shown in Figure 2, then reference to the present state, K, indicates that an output of four cannot be

obtained, regardless of the input. The desire is to have the largest output response, then obviously this is nine, provided five is an allocable resource. Note that, in general, the plant may be controllable only in certain states and even controllable to a different degree as a function of the state. Here is the essence of the control theory without the usual restricting assumptions.

The problem of control occurs in many different regimes. The task of retrieving objects from space by means of the shuttle is difficult for several reasons. First, the retrieval arm is not a rigid body. The equations of motion for such a flexible structure are quite complex. Nevertheless, modern structural analysis programs on large digital computers provide a means for performing structural analysis by finite element methods, taking only the first few bending modes into account.

Second, the shuttle does not provide a stable base in inertial space. Therefore, the geometric problems associated with the retrieval of an object from space are an order of magnitude more complicated than those faced by designers of terrestrial robots, where it is reasonable to expect an inertially fixed base for the robot.

Third, the exact size and mass of the object to be retrieved (the spacecraft) may not be known; and therefore, a prediction of the motion once the object to be retrieved has been "grabbed" by the retrieving system may not be possible with the desired degree of reliability for a successful retrieval operation.

Fourth, the initial motion, and therefore the initial movement and moment of momentum of the spacecraft, may be unknown or only approximately known. It is also not known if the spacecraft is spinning, whether it may be retrieved while in that state or first the motion must be reduced to some degree, or it may need to be stopped completely (at some additional cost in size and weight

Fifth, the forces and movements acting on the spacecraft may not be known. Frequently, of course, the spacecraft is a passive body, exerting no forces or movements by itself. On the other hand, it may be a satellite with an attitude control system which may or may not be functioning properly. One could even consider the case of a satellite whose attitude control system may have been struck, so that it might seem that the satellite is passive, but the shock of the capture might make it active once again.

These five areas of uncertainties (and there are more, such as temperature effects and others) should indicate that an off-line study and simulation of a particular retrieval task will generally provide only a baseline model from which first-order approximations to guidance laws may be derived and around which certain sensitivity studies may be performed. If the total amount of the uncertainties is relatively small, and if no unforeseen effects take place during the retrieval process, such an off-line model and simulation may be adequate. If, however, many uncertainties exist, a real-time systems' identification and an adaptive control system may be required to prevent a catastrophic failure.

There exists many ways to address the problem of adaptive control and on-line system identification. In some cases, one assumes a certain structure or topology of the system (for example, a linear, second or higher order system with constant coefficients), and the on-line identification process then merely consists of estimating these coefficients such that a certain error (usually in some least mean square sense) between observed and modeled behavior of the system is minimized. In the case at hand, such a parameter might be the mass, or the moment of inertia, or the rotational frequency of the spacecraft.

Taking a radically different approach, the spacecraft is not modeled as a linear system, nor as a system described by a set of linear or nonlinear differential equations, but rather,

in terms of finite state machines. These machines may be used to perform any one of the following three functions: prediction of each next input symbol from a sequence of observed past symbols; transformation of a sequence of input symbols into a sequence of output symbols; or classification of a given sequence of symbols. All three of these functions may be used in the control of the retrieval operation. The first property may be used to predict the motion (position, attitude, and their derivatives) of the spacecraft and of the end effector. The transformation property can be used to effect the control of the actuators of the retriever. Finally, the classification property may be helpful to identify certain classes of spacecraft.

Various approaches may be taken to create finite state machines capable of performing these three tasks, the most general one is the method of evolutionary programming. That it is, indeed, feasible to use finite state machines for tasks of prediction and identification in control systems has previously been shown by key personnel of Decision Science, Inc.*

Using finite-state machines as a controller is, of course, not a panacea that resolves all the space shuttle's control problems. One of the difficulties when using finite-state machines lies in the alphabet size, which is a reflection of the required resolution. In the past, evolutionary programmers have created finite-state machines within an alphabet of up to 64 symbols. This corresponds with a resolution of 6 bits. For identification and control tests, machines with a resolution of 8 to 9 bits appear to be desirable, thus, the alphabet size would be 256 or 512.

This gives an angular resolution of about one degree. The problem with such a large alphabet size is the long past history

---

*"Finite-State Machines as Elements in Control Systems" by G. Burgin and M. Walsh, 1971, IEEE Systems, Man and Cybernetics Group, Convention Record, IEEE 1971, pp. 241-246.

required to evolve finite-state machines. This means that considerable resources are required in terms of memory and CPU capability. In the past, such large alphabet sizes were uneconomical; but with the present availability of 16-bit microprocessors, and the recent introduction of INTELS iAPX 432 chips, which provide, with only four chips, a 32-bit processor comparable to a medium sized IBM 370 model, CPU capability should be no limitation for an on-line identification and adaptive control using evolutionary programming. There remains the question of sufficient memory. Here again, 64k bit RAMs are now available, and magnetic bubble memories allow even greater densities. About a half-dozen manufacturers will have bubble memories commercially available by the end of this year.

Advantage should be taken of this phenomenal progress in hardware, both CPU and memory, together with a drastic reduction in size and weight, to solve the space shuttle's retrieval problems. Evolutionary programming is a powerful method but was previously limited in its applicability to real-life problems because of hardware limitations. Now, it seems, the time has come to demonstrate the usefulness of this method.

Automatic control of robotic devices requires the solution of two problems. First, the goal of the robotic device must be defined, and that requires a precise formulation of what the robot is supposed to do. This is a problem pertaining to the field of artificial intelligence. If the robotic device is simply a mechanical manipulator, the question reduces to: "To which point in space and along which path is the robot's end effector supposed to move, and what should its attitude be at the terminal point?"

The second problem attempts to find forces and moments acting on the manipulator such that it will perform the motion specified above. In other words, the question is: "How do we force the robot to do what it is supposed to do?" This is a problem of applied automatic control.

Appendix C of this report addresses this problem. First, the different control schemes, which have been proposed in the recent open literature, are reviewed. Then, a novel scheme of controlling robots of unknown physical properties is proposed wherein the control system's gains are adjusted by means of finite-state machines. To develop such a control system, a computer simulation of a manipulator is required. Appendix C offers a detailed account of such a simulation for a specific, commercially available manipulator, the PUMA 250. A control system capable of controlling this manipulator's upper and lower arm from any arbitrary point in space to any other, physically reachable point, is designed. The response of this control system is quite satisfactory for payloads ranging from zero to five times nominal payload.

Briefly stated, prediction is the basis for control. The predictive model is a first cut representation of the logic underlying the environment of interest. Predicting the environmental response to the sequence of stimuli allows the prediction process to forecast each next response, but more importantly, this yields an up-to-date representation of the transduction. In other words, prediction of each response based on prior stimulus/response pairs can be used to resolve the identification problem. Once this is accomplished, the remaining task is to determine how to close the loop.

The literature is replete with techniques for the control of linear and certain non-linear systems with respect to relatively simple control goals (the criterion that specifies the worth of each correct control response and the cost of the alternative errors). The problem remains unresolved for the general situation of a non-linear, potentially active, and even intelligent plant to be controlled with respect to an arbitrary control goal. The present task is to explore a generalized technique in this regard.

Having a capability to model the environment sets the stage for self-modeling. Here the same process may be applicable. But, there must be recognition of fundamental limitations concerning self-referential systems. Self-diagnostic routines already exist. In a sense, these are self-referential but limited in scope to the designer's prior knowledge of the alternatives that might arise. More sophisticated "self-awareness" requires continual modeling of the self as exhibited under the developing situation together with an ability to reference such models for the sake of improving the likelihood of correct response in the goal seeking interaction with the environment. To be meaningful, such "consciousness" must be sufficiently precise (that is of adequate specificity), sufficiently accurate (demonstrate a continuing correspondence with reality), adaptable (capable of updating as new aspects of the self come into view), and readily referencable (suitable for immediate data retrievable and updating in view of the required response time). Once such a capability has been realized, it is of interest to enquire as to the possibility of still higher level modeling of the self. Here the process is analogous to knowing that you exist and knowing that you know that you exist. With such an onboard capability, spacecraft may become more adaptive, taking into account not only the changing environment but their own remaining capabilities. Such artificial consciousness sets the stage for intelligent interaction among spacecraft and other autonomous inanimate entities.

The purpose of another entity can only be inferred from its behavior and the presumption of similarity between the "organisms". Such higher level modeling sets the stage for interacting with other intelligent creatures in a meaningful manner. Note that the game can be played at different levels at the same time. "What resource am I willing to expend to learn more about the other player's goal so that I might better direct my efforts in future moves?" A more detailed understanding of this process

should provide new insight into the nature of coalition building and generalized gaming.

Lastly, the essential concern for the very mechanism of purpose cannot be avoided. How does purpose arise? Is there a minimum complexity of logical structure required for the generation of the survival "instinct" . . . the paramount purpose of all living creatures? The approach to this problem requires a careful definition of the nature of purpose and reference to logical means for generating such representations within mechanisms. It seems reasonable to suspect that inanimate machines can be designed having sufficient complexity to generate, then seek their own purposes. The "human animal" is a demonstration of one such mechanism. The task remaining is to understand the logic that allows such mechanisms to operate. An understanding in this regard might open the door to the design of highly sophisticated spacecraft that construct and deploy "offspring". Monitoring their activity should prove to be instructive.

CONCLUSION

It is easy to envision the requirement for retrieval, repair, and replacement of satellites. Here prediction and control processes are required. These can be approached by classical means, but it is time to examine an alternative approach. . . one that can treat an arbitrary data stream, predicting each next data point with respect to an arbitrary payoff matrix. The criterion for linking spacecraft is more complex than least mean squared error. Final closure is at great cost if, say, the angular difference is excessive. Success is assured only if the angular difference is within the acceptable bounds. The task is to predict closure windows with just such a criterion in mind.

The problem of control can also be addressed by classical means; however, it might be more suitable to represent the alternative states of the closing vehicles and determine each next move by reference to the prospective transition among such states. We learn to ride a bicycle not by solving the equations of motion but by remembering how best to actuate the controls from the given situation to reach a more desirable state. Computation for control of the shuttle might be benefited in this same regard.

Finite-state machines provide a natural means for representing the logic which may underlie a sequence of data derived from a sensed environment or for depicting the transduction between stimulus and response of such an environment. Such representation permits expansion of the logic in terms of arbitrary input and output languages so long as these are expressed within finite alphabets. Further, the machines may be of arbitrary specificity so long as they have only a finite number of states. Thus, no unnatural constraint is imposed, as is so often the case when a sequence of data is expressed in terms of a linear difference or differential equation.

Exploratory spacecraft require onboard computation that predicts dangers and opportunities, then responds to these through appropriate control actions. Clearly, it is desirable to avoid colliding with meteors and other space debris. It is essential to avoid space mines that may seek to collide on the basis of IR, radar, or other sensed information. It is desirable to come sufficiently close and interact with various interesting objects or regions so that these can be suitably investigated. Such interaction may involve the cooperative construction of space based platforms or other facilities. It may involve close control onboard the robots with strategic control reserved to the human operator, or in the more distant future, autonomous spacecraft interacting with each other on their own behalf.

Decision Science, Inc.

San Diego, California, June 1982

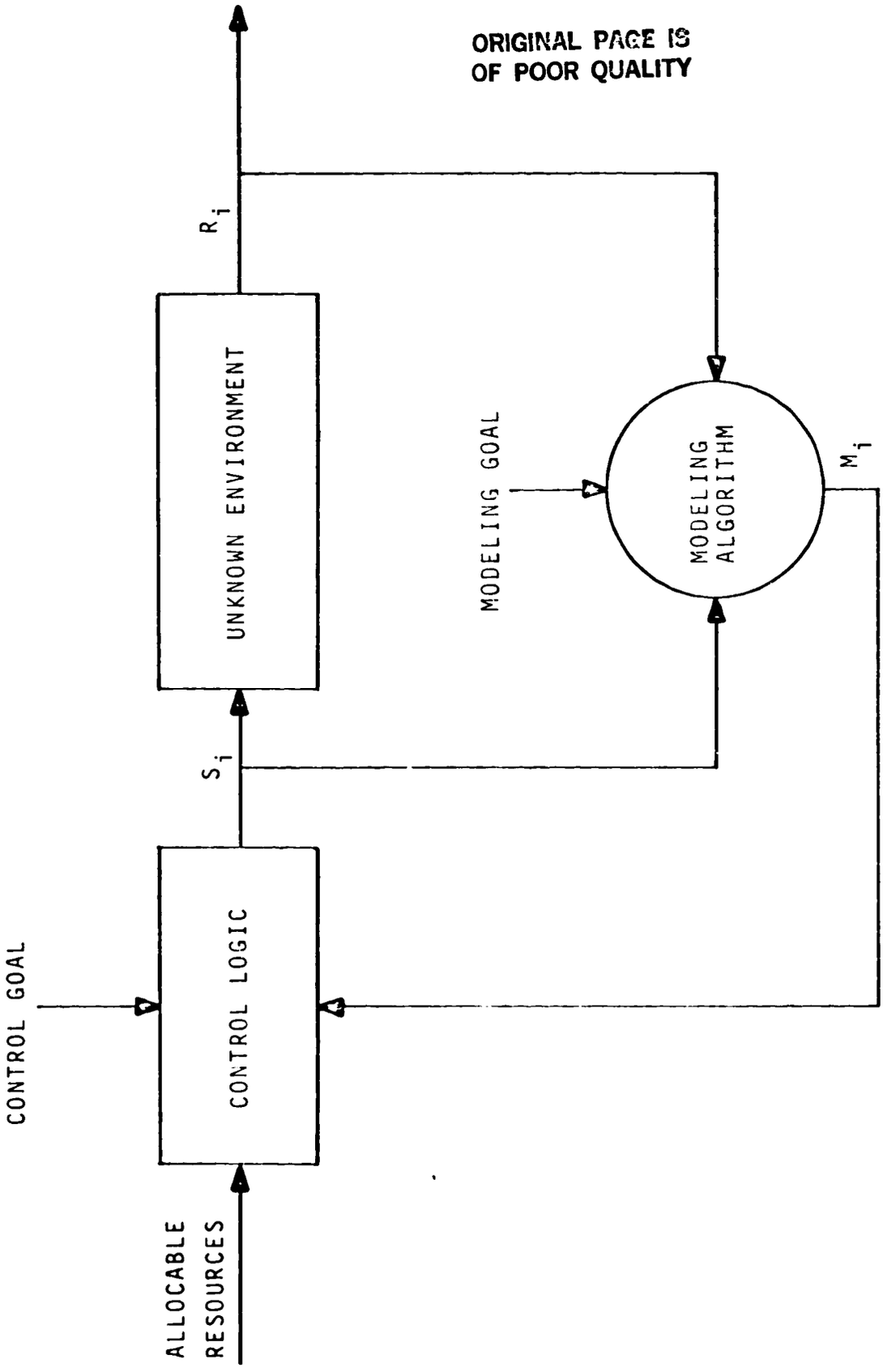


Figure 1.

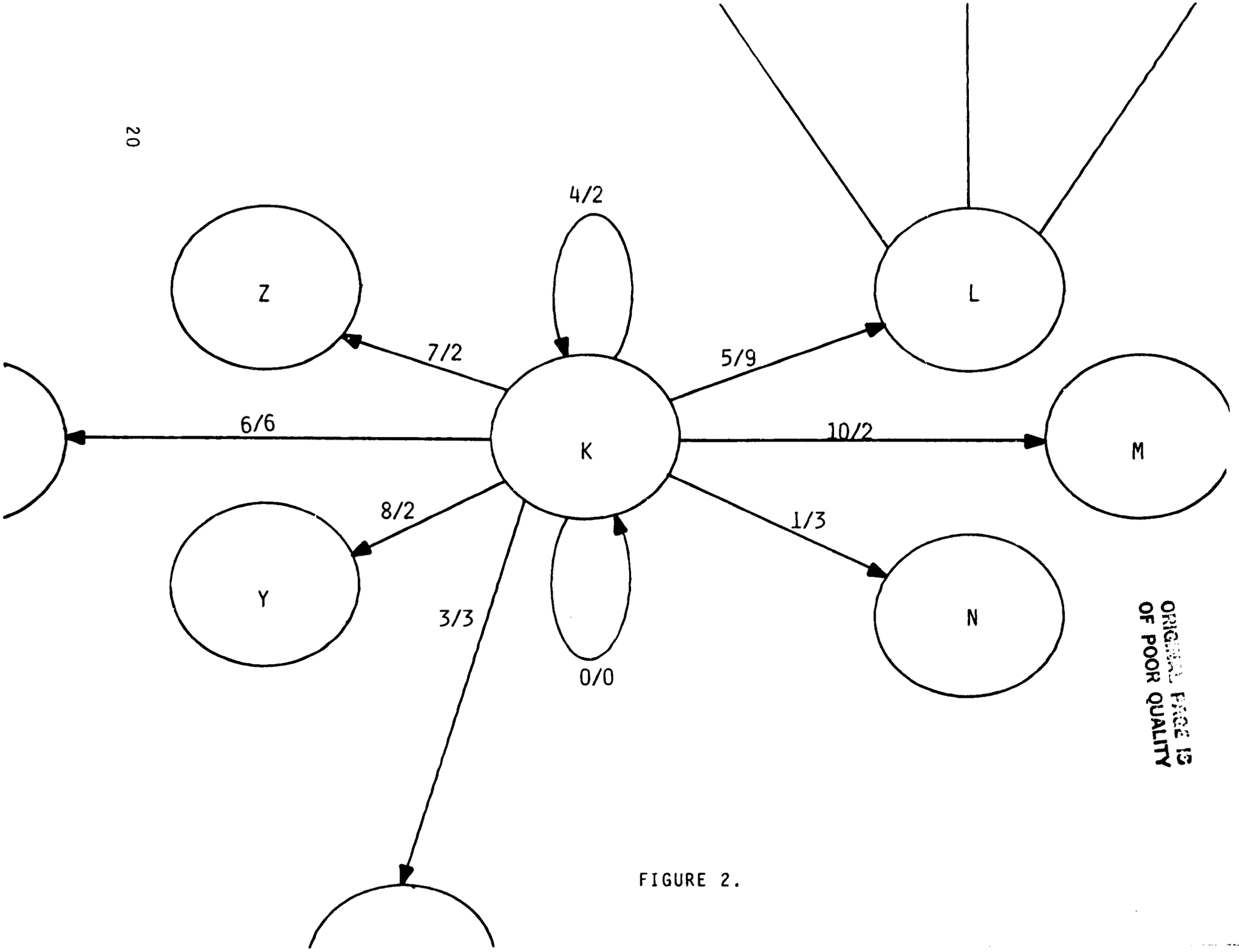


FIGURE 2.

APPENDIX A:

Page

DETERMINING PERIODS OF NOISY SEQUENTIAL DATA

APPENDIX A

DETERMINING THE PERIODS OF NOISY SEQUENTIAL DATA

I INTRODUCTION

Many phenomena give rise to periodic sequences of data values. Economic indicators, celestial events, human processes, electrical circuits, tumbling spacecraft,--the list is endless. Given an arbitrary sequences of data values, the problem is to determine the smallest period. If the data is "noisy," that is, if the sequence is only approximately periodic, then the task becomes more interesting and the result more practical. The purpose here is to explicate in detail a general procedure for determining those periodic sequences which best fit an arbitrary sequence of data values. Statistical measures are developed by which to define this best periodic sequence and for determining confidence limits on the error when subsequent values in the given data sequence are estimated by the corresponding values in the periodic sequence.

II THE SMALLEST PERIOD OF A PERIODIC SEQUENCE

Suppose that $X(1), X(2), \ldots, X(n)$ is a finite sequence of n real numbers. By definition, the sequence (X) is <u>periodic with period p</u>, if and only if for all positive integers k, $X(k + p) = X(k)$. Equivalently, for all non-negative integers i and all positive integers k between 1 and p inclusive,

$$X(k + pi) = X(k).$$

In other words, if $(1 \leq k \leq p)$ then

$$X(k) = X(k + p) = X(k + 2p) = \ldots$$

If both p and q are periods of the sequence (X) then it follows that (p,q), the greatest common positive divisor of the integers p and q is also a period of (X). To show this recall that if d is the greatest common divisor of p and q, then there exist two integers, a and b, such that

$$d = pa + qb.$$

Therefore, for all positive integers k,

$$X(k + d) = X(k + pa + qb).$$

Since d is positive, a and b cannot both be negative and so either $(k + pa)$ or $(k + qb)$ is positive. If $k + pa$, say, is positive, then

$$X(k + pa + qb) = X(k + pa)$$

because (X) has period q. But since p is also a period,

$$X(k + pa) = X(k)$$

Therefore for all positive integers k,

$$X(k + d) = X(k)$$

That is, (X) has period d.

This implies that there is a smallest period for a given sequence (X) and that this smallest period is a divisor of all other periods for (X). That is, all other periods are multiples of the smallest period. It suffices, therefore, to determine the smallest period of a sequence, all other periods being multiples.

III THE METHOD

If (X) is exactly periodic with smallest period p, then there are just p data values to the sequence, subsequent values being repetitions. These values are X(1), X(2), ... , X(p) and they constitute one cycle of the periodic sequence. If on the other hand (X) is merely approximately periodic, then, for example, the data values X(1), X(1 + p), X(1 + 2p), ... will only be approximately equal. Similarly for $(1 \leq k \leq p)$, the values X(k), X(k + p), X(k + 2p), ... will only be approximately equal. X(1), X(1 + p), X(1 + 2p), ... are all the first data values of all the cycles; similarly X(k), X(k + p), X(k + 2p), ... are all the kth data values of the cycles of the sequence.

The method used here to determine a periodic sequence (Y) that best fits the given approximately periodic sequence (X) is to determine, for each possible period p, the average (arithmetic mean), Y(k), of all the kth data values of the cycles of the sequence (X). That is,

$$Y(k) = \frac{1}{T+1} \sum_{i=0}^{T} X(k + pi)$$

where T is the number of cycles. Here, T can be expressed in closed form as

$$T = \text{Int}((n - k)/p),$$

where $\text{Int}(j)$ is the greatest integer less than or equal to j. This formula for T takes into consideration the likelihood that the finite data sequence (X) may end in a partial cycle.

In order to measure the degree of fit of the periodic sequence (Y) to the sequence (X), the sample variance, $V(k)$, for each of the p values of the cycle $(k = 1, 2, \ldots, p)$ is determined by

$$V(k) = (1/T) \sum_{i=0}^{T} \left[X(k + pi) - Y(k)\right]^2$$

Assuming that T is at least 30 or that deviations of the data values $X(k + pi)$, $(i = 0, 1, 2, \ldots, T$ are normally distributed, it is then appropriate to calculate 95% confidence error bounds for $Y(k)$ when $Y(k)$ is used as an estimate of any individual value $X(k + pi)$, $i = 1, 2, \ldots, T, \ldots$

Depending upon the particular application, there are various criteria for determining the best fit periodic sequence $Y(k)$. For many purposes the standard least squares criterion is appropriate and so this criterion is

developed first. But it is easy to imagine situations where, for example, the cost of error is asymmetric, thus making it optimal to fit the sequence (X) with a periodic sequence (Y) which is not optimal in the least squares sense. This more general case is considered later.

The most straightforward criterion for determining the best fit periodic sequence (Y) is the average 95% confidence error tolerance, E, for the p values Y(k), k = 1, 2, ..., p. E is given by

$$E = (1/p) \sum_{k=1}^{p} E(k) ,$$

where, by well-known statistical methods,

$$E(k) = \frac{t\sqrt{V(k)}}{\sqrt{(T + 1)}} ,$$

t being the appropriate t- distribution value for T degrees of freedom.

Each possible period p generates a periodic sequence (Y) and an average 95% confidence error bound E for (Y). That sequence (Y) whose average error bound E is smallest is deemed the best least squares fit periodic sequence approximating the given sequence (X).

IV COMPUTER PROGAM

The procedure outlined in the previous section would hardly be practical without the aid of a high speed

computer. (Indeed many procedures once thought impractical can now be resurrected and applied in place of more analytic techniques.)

A computer program has been written and demonstrated that takes an arbitrary sequence of real numbers (X) and determines the ten best periods and corresponding periodic sequences (Y). Ninety-five percent confidence error bounds are given for each data value $Y(k)$, $k = 1, 2, \ldots, p$. As indicated previously, the best periodic sequence (Y) is the one whose average error bound over the p values of any one cycle is minimal.

The program is interactive and allows convenient input, storage, recall, and display of all relevant parameters. (See pages A - 12 - 19).

V SAMPLE RUN OF THE COMPUTER PROGRAM

When the program is executed, the user is asked an initial question. By entering a single letter the user may 1) input a new sequence of real numbers, or 2) recall from storage a previously input data sequence, or 3) list the data sequence, or 4) correct individual members of the data sequence, or 5) analyze the data sequence for periodicities, or 6) display this period analysis, or 7) store the data sequence for future retrieval, or 8) end the session.

If the user wishes to input a data sequence and enters the appropriate letter, then values are accepted in groups

of 20 or less. After data input is completed, the program returns to the initial question.

At this time individual data values may be listed and corrected by entering the appropriate letter and then entering the particular index and corresponding sequence value. Multiple corrections can be made without return to the initial question.

Return to the initial question allows the user to analyze the data sequence for periodicities. When the analysis is completed, the total number of data values is given together with the period which best fits the data, the number of complete cycles and the predicted next value in the data sequence. The user is now asked whether he wishes to analyze a particular potential period, or rank the periods according to goodness of fit or end the display and return to the initial question. If the user wishes to analyze some particular period p, he enters that period and quickly sees a table containing an in-depth analysis including error estimates and statistical measures. The user may then immediately analyze any other period.

By entering a single letter the user may also display a table ranking all possible periods from 1 to 10 according to their average 95% confidence error. This is the average error when sequence values are estimated by the associated periodic sequence constructed by averaging the various representatives coming from the different cycles of the data sequence.

After analysis and display is completed, the user may store the data sequence for future retrieval.

Three different data sequences were generated to test the program. (See the end of this appendix.) The program determined the best fit periodic sequences and successfully carried through the other steps explicated in the previous paragraphs of this section.

VI OTHER CRITERIA FOR OPTIMALITY

One of the advantages of the method outlined above is that the criterion for optimality can be easily generalized without extensive alteration of the procedure.

Suppose that instead of the least squares criterion, there is defined a cost function, C, that assigns to each ordered pair (x,y) the worth of estimating y when in fact the actual value is x. Clearly, C will have its smallest entries on the diagonal, where the estimate is exactly correct. Off-diagonal entries may be arbitrarily assigned depending upon the context. C may be quite non-symmetric.

In the least squares case, the average

$$Y(k) = \frac{1}{T+1} \sum_{i=0}^{T} X(k + ip)$$

can be easily shown to be the best estimate of the data values $X(k + ip)$, $i = 0, 1, \ldots, T$. But for an arbitrary cost function C, the best estimate for these data values

depends upon C. Therefore it is necessary first to determine Y(k) for each k = 1, 2, ..., p by finding that value Y(k) for which the average cost

$$F(k) = \frac{1}{T+1} \sum_{i=0}^{T} C(X(k+pi), Y(k))$$

attains a minimum.

If C is a continuously differentiable function defined on an open (possibly infinite) domain in the xy-plane, then partial differentation of F(k) with respect to Y(k) and setting equal to zero yields the necessary condition

$$\sum_{i=0}^{T} C_2 (X(k+pi), Y(k)) = 0 ,$$

where $C_2(x,y) = \frac{\partial}{\partial y} C(x,y)$.

For example, if $C(x,y) = (x-y)^2$, then the least square criterion leads to

$$\sum_{i=0}^{T} (-2) \left[X(k+pi) - Y(k)\right] = 0 .$$

That is,

$$\sum_{i=0}^{T} X(k+pi) - (T+1) Y(k) = 0 .$$

That is,

$$Y(k) = \frac{1}{T + 1} \sum_{i = 0}^{T} X(k + pi) .$$

In other words in this case, $Y(k)$ is just the average of the $(T + 1)$ values $X(k + pi)$, $i = 0, 1, \ldots, T$.

If C is defined discretely as a cost matrix, then the cost $F(k)$ must be minimized directly by determining for which column of the matrix the entries $C(X(k + pi), Y(k))$ have smallest sum for $i = 0, 1, 2, \ldots, T$.

Once $Y(k)$ has been found for each k, the average cost, F, for the given period p can be determined by

$$F = (1/p) \sum_{k = 1}^{p} F(k) .$$

Now the most straightforward criterion for determining the least cost periodic sequence (Y) is the average 95% confidence cost error bound, E, for the p values $Y(k)$, $k = 1, 2, \ldots, p$. (This step is necessary; otherwise, the least cost periodic sequence will always turn out to be the whole sequence of data values. That is, $T = 1$.)

The average 95% confidence cost error bound, E, is given by

$$E = (1/p) \sum_{k = 1}^{p} E(k) ,$$

where,

$$E(k) = t \sqrt{\frac{V(k)}{T + 1}} + F(k)$$

Here, t is the appropriate t-distribution value for T degrees of freedom and $V(k)$ is the sample variance in costs given by

$$V(k) = (1/T) \sum_{i = 0}^{T} \left[C(X(k + pi), Y(k)) - F(k) \right]^2$$

Each possible period p generates a periodic sequence (Y) and an average 95% confidence cost error bound E for (Y). That sequence (Y) whose average cost error bound E is smallest is deemed the best fit periodic sequence with respect to the cost matrix C.

LISTING OF PERIOD ANALYSIS PROGRAM

```
10 DEFINT X
20 DIM X(300),Y(150),V(150),VA(150),EA(150),E(150),RP(10),RE(10),TD(300)
28 REM T-DISTRIBUTION VALUES (95% CONFIDENCE LEVEL)
30 FOR I=1 TO 30
32    READ TD(I)
34 NEXT I
40 DATA 12.7,4.3,3.2,2.8,2.6,2.4,2.4,2.3,2.3,2.2,2.2,2.2
41 DATA 2.2,2.1,2.1,2.1,2.1,2.1,2.1,2.1,2.1,2.1,2.1,2.1
42 DATA 2.1,2.1,2.1,2,2,2
45 FOR I=31 TO 300:TD(I)=2:NEXT I
90 CLS
100 PRINT"PROGRAM TO DETERMINE THE PERIOD OF A SEQUENCE OF"
200 PRINT" DISCRETE VALUES AND PREDICT THE NEXT VALUE"
300 PRINT:PRINT
400 PRINT"ENTER"
500 PRINT"    I (INPUT DATA VALUES) OR"
550 PRINT"    C (CORRECT DATA VALUES OR NUMBER OF VALUES) OR"
575 PRINT"    L (LIST DATA VALUES) OR"
600 PRINT"    A (ANALYZE DATA VALUES) OR"
620 PRINT"    D (DISPLAY PERIOD ANALYSIS) OR"
650 PRINT"    S (STORE DATA VALUES ON TAPE #-1) OR"
660 PRINT"    R (RECALL DATA VALUES FROM TAPE #-1) OR"
700 INPUT"    E (END)";S$
900 IF S$="E" THEN END
950 IF S$="C" THEN GOTO 3900
975 IF S$="L" THEN GOTO 4400
1000 IF S$="A" THEN GOTO 2300
1025 IF S$="D" THEN GOTO 7620
1050 IF S$="S" THEN GOTO 5000
1060 IF S$="R" THEN GOTO 5600
1100 IF S$<>"I" THEN GOTO 300
1200 REM INPUT DATA VALUES ROUTINE
1300 PRINT:PRINT
1350 INPUT "LAST INDEX N (0 TO START)";N
1500 INPUT"HOW MANY (MORE) DATA VALUES M (-1 < M < 21)";M
1600 IF M=0 THEN GOTO 300
1700 FOR I=1 TO M
1800    PRINT"X(";N+I;")=";
1850    INPUT X(N+I)
```

```
1900 NEXT I
2100 PRINT:FOR I=1 TO M
2120 PRINTX(N+I);
2130 NEXT I
2150 N=N+M
2190 IF NMAX < N THEN NMAX=N
2195 PRINT
2200 GOTO 1500
2300 REM ANALYZE DATA ROUTINE
2310 CLS:PRINT @ 465,"PATIENCE, I AM ANALYZING."
2312 PRINT:PRINT TAB(19) "DO NOT DISTURB ME!"
2320 FOR K=1 TO 10:RE(K)=1000:NEXT K
2400 FOR P=1 TO INT(NMAX/2)
2590 REM ROUTINE TO ANALYZE PERIODS P
2600 VA(P)=0:EA(P)=0
2630 FOR K=1 TO P
2660    S=0:R=0
2690    T=INT((NMAX-K)/P)
2720    FOR I=0 TO T
2740       L=K+P*I
2750       S=S+X(L)
2760       R=R+X(L)*X(L)
2780    NEXT I
2800    IF T=0 THEN V(K)=0:GOTO 3020
2810    V(K)=R-S*S/(T+1)
2840    V(K)=V(K)/T
2870    E(K)=TD(T)*V(K)[(0.5)
2930    EA(P)=EA(P)+E(K)
2950    IF (EA(P)/P)>RE(10) THEN GOTO 7610
2970    VA(P)=VA(P)+V(K)
3020 NEXT K
3050 VA(P)=VA(P)/P
3075 EA(P)=EA(P)/P
3080 :IF S$="A" THEN GOTO 6300 ELSE GOTO 7700
3300 REM DISPLAY ROUTINE
3400 PRINT:PRINT
3420 P=RP(1)
```

```
3450 PRINT"NUMBER OF DATA VALUES IS NMAX=";NMAX
3500 PRINT"BEST LEAST SQUARES FIT PERIOD IS P=";P
3600 PRINT"NUMBER OF COMPLETE CYCLES IS";INT(NMAX/P)
3700 PRINT"NEXT DATA VALUE, X(";NMAX+1;"), SHOULD BE";Y((NMAX+1-P*INT((NMAX+1)/P
)))
3800 GOTO 7620
3900 REM CORRECT DATA VALUES ROUTINE
4000 PRINT:PRINT
4020 PRINT"ENTER  C (CHANGE NUMBER, NMAX, OF DATA VALUES) OR"
4040 INPUT"       N (NO CHANGE IN NMAX)";S$
4050 IF S$="C" THEN PRINT:INPUT"NMAX = ";NMAX
4060 IF S$<>"N" THEN GOTO 4020
4080 PRINT
4100 PRINT"TO CORRECT EXISTING DATA VALUES,"
4120 INPUT"ENTER I,X(I) OR 0,0 TO ESCAPE";I,X(I)
4200 IF I=0 THEN GOTO 300
4300 PRINT:GOTO4100
4400 REM LIST DATA VALUES ROUTINE
4500 PRINT:PRINT
4600 FOR I=1 TO NMAX
4700    PRINT I;X(I),
4800 NEXT I
4900 GOTO 300
5000 REM ROUTINE TO STORE DATA VALUES ON TAPE #-1
5020 PRINT:INPUT "SET TAPE #-1 TO RECORD AND HIT ENTER";S$
5050 X(0)=NMAX
5100 POKE 16526,79:POKE 16527,191
5200 B%=2000:R%=USR(B%)
5300 POKE 16526,101:POKE 16527,191
5400 C%=VARPTR(X(0))
5500 D%=USR(C%)
5510 CLS
5520 PRINT:PRINT"DATA STORED ON TAPE #-1"
5550 GOTO 300
5600 REM ROUTINE TO RECALL DATA VALUES FROM TAPE #-1
5620 PRINT:INPUT "SET TAPE #-1 TO PLAY AND HIT ENTER";S$
5700 POKE 16526,79:POKE 16527,191
5800 B%=2000:R%=USR(B%)
5900 POKE 16526,175:POKE 16527,191
```

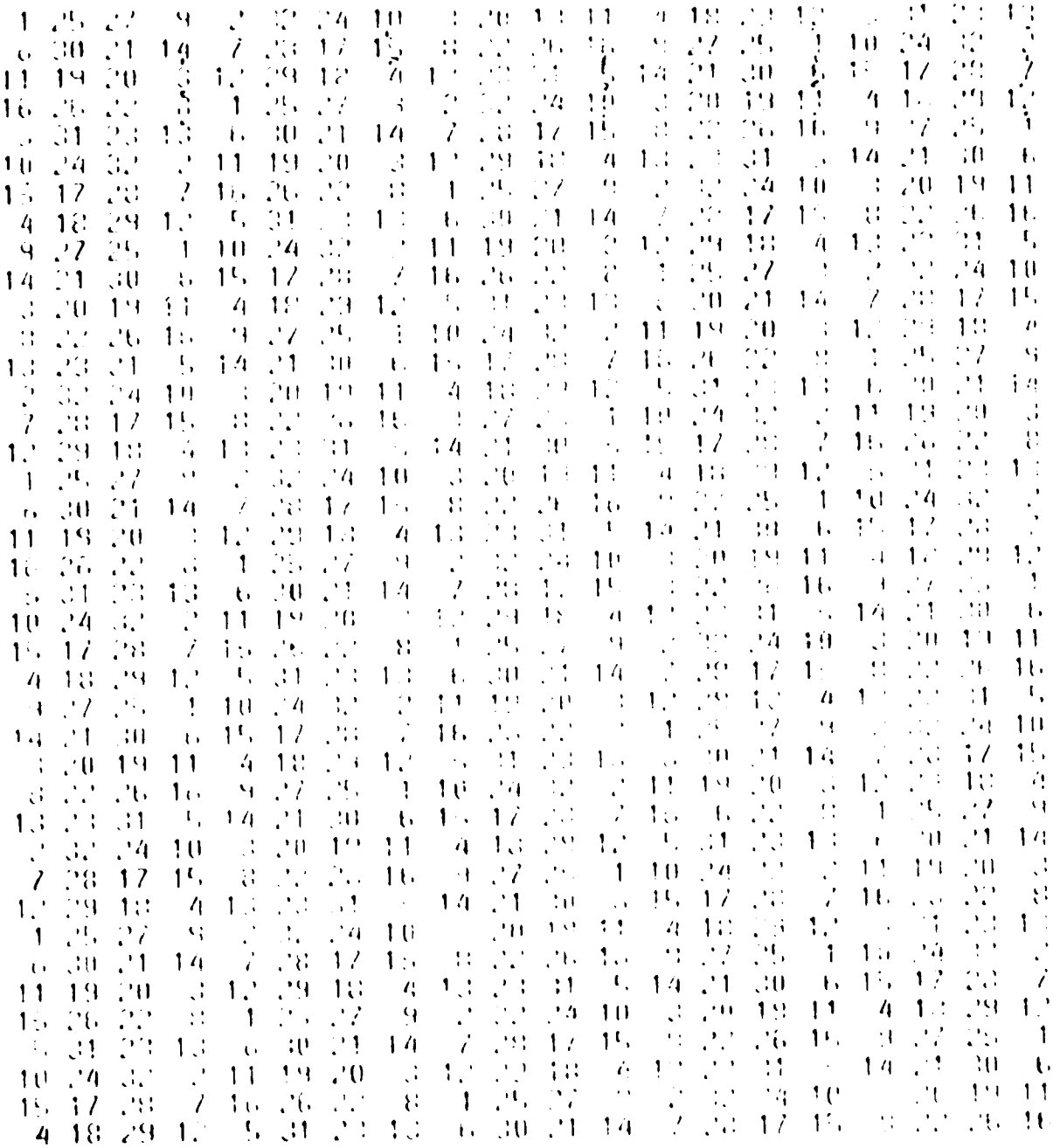
```
6000 B%=VARPTR(X(0)):E%=USR(B%)
6100 IF E%<>0 THEN PRINT:PRINT"BAD DATA":PRINT"CHECKSUM=";E%
6150 NMAX=X(0)
6170 CLS
6180 PRINT:PRINT"DATA RECALLED FROM TAPE #-1"
6200 GOTO 300
6300 REM ROUTINE TO RANK PERIODS
6400 FOR J=1 TO 10
6500    IF EA(P)<RE(11-J) THEN IF J=10 THEN J=11 ELSE GOTO 7600
6600    IF J=1 THEN GOTO 7610
6650    IF J=2 THEN GOTO 7100
6700    FOR I=0 TO J-3
6800       RP(10-I)=RP(9-I)
6900       RE(10-I)=RE(9-I)
7000    NEXT I
7100 RP(12-J)=P
7200 RE(12-J)=EA(P)
7550 GOTO 7610
7600 NEXT J
7610 :NEXT P
7611 CLS:PRINT"ANALYSIS OF PERIODS COMPLETED"
7612 P=RP(1):FOR K=1 TO P
7613    S=0:T=INT((NMAX-K)/P)
7614    FOR I=0 TO T
7615       S=S+X(K+P*I)
7616    NEXT I
7617    Y(K)=S/(T+1)
7618 NEXT K:Y(0)=Y(P)
7619 GOTO 3300
7620 REM DISPLAY QUESTION
7630 PRINT:PRINT"TO DISPLAY ENTER"
7632 PRINT"    P (PERIOD P ANALYSIS) OR"
7634 PRINT"    R (RANK PERIODS FOR BEST FIT) OR"
7636 INPUT"    E (END DISPLAY)";S$
7638 IF S$="E" THEN GOTO 300
7640 IF S$="R" THEN GOTO 9000
7642 IF S$<>"P" THEN GOTO 7630
7650 REM PERIOD P QUESTION
7652 PRINT:INPUT"FOR WHICH PERIOD P";P
7660 GOTO 2600
7700 PRINT:PRINT
```

```
7700 PRINT:PRINT
7800 PRINT TAB(15),"PERIOD = ";P
7900 PRINT:PRINT" K    Y(K)   V(K)   E(K)";
8000 PRINT TAB(26)"X(K+I*P)"
8050 PRINT"-------------------------------------------------"
8100 FOR K=1 TO P
8200    PRINT USING "##";K;
8300    PRINT USING "####.#";Y(K);V(K);E(K);:PRINT"   ";
8400    FOR I=0 TO INT((NMAX-K)/P)
8500       PRINT USING "###";X(K+I*P);
8600    NEXT I
8620 PRINT
8700 NEXT K
8800 PRINT:PRINT"AVERAGE VARIANCE VA(";P;")=";VA(P)
8820 PRINT"AVERAGE 95% CONFIDENCE ERROR =";EA(P)
8900 GOTO 7620
9000 REM  DISPLAY BEST FIT PERIODS
9100 CLS
9200 PRINT:PRINT:PRINT TAB(17) "BEST LEAST SQUARE FIT PERIODS"
9300 PRINT TAB(15) "(DATA INCLUDES AT LEAST 2 CYCLES)"
9400 PRINT
9500 PRINT "RANK J","PERIOD P","AVERAGE ERROR - 95% CONFIDENCE"
9600 PRINT"------------------------- ------------------------------"
9700 FOR J=1 TO 10
9800    PRINT USING "####":J,:PRINT"             ";
9900    PRINT USING "######";RP(J),:PRINT"          ";
10000    PRINT"       ";:PRINT USING "######.#";RE(J)
10100 NEXT J
10200 INPUT "TO ESCAPE HIT ENTER";S$
10300 GOTO 7620
```

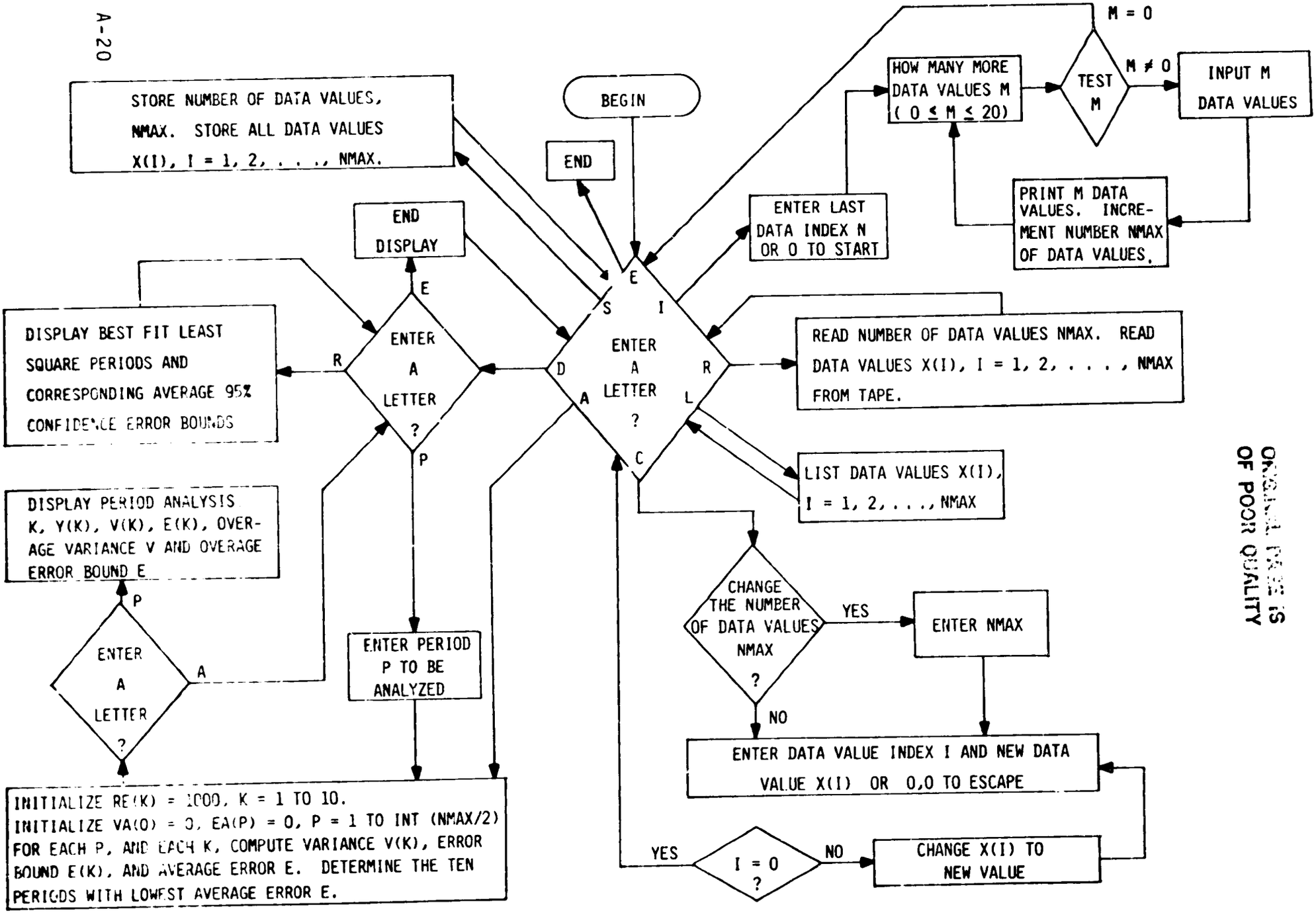
16	21	24	26	29	29	31	31	31	28	26	24	21	17	14	12	9	9	8	7
8	7	7	10	11	12	12	13	15	16	15	16	17	17	15	17	18	19	20	20
23	23	24	25	25	25	25	24	22	20	18	14	12	8	6	4	3	1	0	1
3	5	8	13	16	20	23	26	28	31	32	32	31	28	26	25	21	18	15	13
9	8	8	7	6	8	8	9	10	11	13	14	14	15	16	16	16	16	17	17
17	18	20	21	21	24	25	26	26	25	24	23	22	21	16	14	12	9	6	4
2	1	1	2	4	6	9	13	15	19	24	27	29	30	31	32	31	29	27	23
21	19	16	12	11	8	7	7	6	7	8	8	10	11	12	14	15	15	15	16
15	16	17	17	18	19	20	21	22	23	24	25	26	25	25	24	21	20	17	14
11	8	5	3	1	1	0	1	4	5	10	11	16	19	23	26	28	30	31	32
29	29	26	24	21	18	15	13	10	8	6	7	7	7	8	8	11	11	12	15
14	15	16	15	15	17	17	15	17	17	20	20	21	23	23	25	25	26	24	23
21	19	17	14	12	8	5	3	1	0	1	2	3	6	8	13	17	19	23	26
28	30	31	30	29	29	27	24	22	18	16	13	10	9	7	7	6	7	8	9

1	26	22	9	2	17	28	10	3	21	30	11	4	23	31	12	5	29	18	13
6	19	20	14	7	24	32	15	8	27	25	16	9	26	22	1	10	17	28	2
11	21	30	3	12	23	31	4	13	29	18	5	14	19	20	6	15	24	32	7
16	27	25	8	1	26	22	9	2	17	28	10	3	21	30	11	4	23	31	12
5	29	18	13	6	19	20	14	7	24	32	15	8	27	25	16	9	26	22	1
10	17	28	2	11	21	30	3	12	23	31	4	13	29	18	5	14	19	20	6
15	24	32	7	16	27	25	8	1	26	22	9	2	17	28	10	3	21	30	11
4	23	31	12	5	29	18	13	6	19	20	14	7	24	32	15	8	27	25	16
9	26	22	1	10	17	28	2	11	21	30	3	12	23	31	4	13	29	18	5
14	19	20	6	15	24	32	7	16	27	25	8	1	26	22	9	2	17	28	10
3	21	30	11	4	23	31	12	5	29	18	13	6	19	20	14	7	24	32	15
8	27	25	16	9	26	22	1	10	17	28	2	11	21	30	3	12	23	31	4
13	29	18	5	14	19	20	6	15	24	32	7	16	27	25	8	1	26	22	9
2	17	28	10	3	21	30	11	4	23	31	12	5	29	18	13	6	19	20	14
7	24	32	15	8	27	25	16	9	26	22	1	10	17	28	2	11	21	30	3
12	23	31	4	13	29	18	5	14	19	20	6	15	24	32	7	16	27	25	8
1	26	22	9	2	17	28	10	3	21	30	11	4	23	31	12	5	29	18	13
6	19	20	14	7	24	32	15	8	27	25	16	9	26	22	1	10	17	28	2
11	21	30	3	12	23	31	4	13	29	18	5	14	19	20	6	15	24	32	7
16	27	25	8	1	26	22	9	2	17	28	10	3	21	30	11	4	23	31	12
5	29	18	13	6	19	20	14	7	24	32	15	8	27	25	16	9	26	22	1
10	17	28	2	11	21	30	3	12	23	31	4	13	29	18	5	14	19	20	6
15	24	32	7	16	27	25	8	1	26	22	9	2	17	28	10	3	21	30	11
4	23	31	12	5	29	18	13	6	19	20	14	7	24	32	15	8	27	25	16
9	26	22	1	10	17	28	2	11	21	30	3	12	23	31	4	13	29	18	5
14	19	20	6	15	24	32	7	16	27	25	8	1	26	22	9	2	17	28	10
3	21	30	11	4	23	31	12	5	29	18	13	6	19	20	14	7	24	32	15
8	27	25	16	9	26	22	1	10	17	28	2	11	21	30	3	12	23	31	4
13	29	18	5	14	19	20	6	15	24	32	7	16	27	25	8	1	26	22	9
2	17	28	10	3	21	30	11	4	23	31	12	5	29	18	13	6	19	20	14
7	24	32	15	8	27	25	16	9	26	22	1	10	17	28	2	11	21	30	3
12	23	31	4	13	29	18	5	14	19	20	6	15	24	32	7	16	27	25	8
1	26	22	9	2	17	28	10	3	21	30	11	4	23	31	12	5	29	18	13
6	19	20	14	7	24	32	15	8	27	25	16	9	26	22	1	10	17	28	2
11	21	30	3	12	23	31	4	13	29	18	5	14	19	20	6	15	24	32	7
16	27	25	8	1	26	22	9	2	17	28	10	3	21	30	11	4	23	31	12
5	29	18	13	6	19	20	14	7	24	32	15	8	27	25	16	9	26	22	1
10	17	28	2	11	21	30	3	12	23	31	4	13	29	18	5	14	19	20	6
15	24	32	7	16	27	25	8	(1)	26	22	9	2	17	28	10	3	21	30	11
4	23	31	12	5	29	18	13	6	19	20	14	7	24	32	15	8	27	25	16

1



FLOW CHART OF PERIOD ANALYSIS PROGRAM



APPENDIX B:

THE EVOLUTIONARY PROGRAM

APPENDIX B

THE EVOLUTIONARY PROGRAM

An evolutionary program was written for use in this study. Input in addition to the data points of concern consists of the error matrix, the number of states in the initial finite state machine, the start state of the initial machine, the maximum number of states allowed, the number of symbols in the data alphabet, the length of the initial history to be used and the length of the window over which the machines are exercised. The window can have either a fixed length, or it can be the total history. The alphabet consists of the integers from one to the number of different symbols.

The program constructs, by random assignment of next state and output symbol, a finite-state machine with the initial number of states and the initial state. It then exercises the machine over the window; and, for those state-input pairs exercised, assigns that output which minimizes the error. The resulting machine is the initial parent machine. It is then exercised over the window, the error score is computed, and the prediction of the next data point is made.

Up to five offspring are constructed and scored over the window. If an offspring has a better score than the parent machine, it replaces the parent machine and its prediction of the next data point is the accepted prediction. If none of the five offspring have a better score, the parent machine is retained. The available history is advanced one data point and the procedure is repeated.

In forming an offspring, the program randomly chooses one of the following mutations: 1. Change up to five randomly chosen next state transistions. 2. Add a state. 3. Delete a state, and 4. Change the start state. In each case, the mutated machine is exercised over the window and those state-input pairs

which are exercised are assigned that output which minimizes the error. The resulting machine is the offspring machine. As before, whether or not the resulting machine is retained as a new parent depends upon its score being better than that of its parent.

For debugging purposes, the program was tested over three different environments, one cyclic with a short cycle, one with a long cycle, and another which was random. In each a four symbol alphabet was used, each had 96 data points with an initial history of length 50 and a window of length 50. Each entry in the error matrix was the square of the difference between predicted value and actual value. For the short cycle, the program after four to six mutations achieved a perfect score and made all predictions correctly. For the long cycle case the error score improved from .796 to .82 in one run and from .632 to .82 in another. For the random environment, the error score fluctuated between 1.06 and .449 in one run and between .816 and .612 in another.

A listing of the program and flowcharts follows.

LISTING OF THE EVOLUTIONARY PROGRAM

```
?? p999
      PROGRAM FSM(INPUT,OUTPUT,TAPE1,TAPE3,TAPE6)
C
C     THIS IS MAIN DRIVER PROGRAM
C
      COMMON/IODEF/INPFSM,INPDAT,IOUFSM
      COMMON/FSMPA/IERMTX,IHS,IMCSN,ISS,LWIN,MXNS,NDP,NHS,NIN
      DIMENSION IFSM(60), MFSM(60), IHS(200), IERMTX(30)
C
C     SET INPUT OUTPUT TAPES
      INPFSM = 1
      INPDAT = 3
      IOUFSM = 6
      REWIND INPFSM
      REWIND INPDAT
      REWIND IOUFSM
C     READ IN FSM INITIAL PARAMETERS, ERROR MATRIX AND DATA
      CALL INRD
C     INITIALIZE FINITE STATE MACHINE (FSM)
      CALL INIT(NIN,IMCSN,IFSM)
C     INITIALIZED MUTATE ROUTINE
      NHSS=NHS
      INFL = 0
      MWSS = 1
      MCSN = 1
      CALL MUTAT(MFSM,INFL,MWSS,MCSN)
      IWSS = ISS
      NPRR = 0
      NPRD = 0
      INFL = 1
      KPRED = 0
C     SET OUTPUTS OPTIMALLY
  100 IF (KPRED.EQ.1) GO TO 105
      CALL STOPT(LWIN,NHS,IWSS,NIN,IFSM,IHS,IERMTX)
  105 NMAC = 0
      CALL SCOR(IFSM,IWSS,SCORI,IPRED,IPERR)
      PRINT*,"SCORI= ",SCORI," IPRED= ",IPRED," IPERR= ",IPERR
      PRINT*,"IWSS = ",IWSS," IMCSN = ",IMCSN
      IF(SCORI.EQ.0) GO TO 130
C     SET UP FOR OFFSPRING MACHINE
      M2 = 2*IMCSN*NIN
      DO 112 K=1,5
      DO 110 I=1,M2
        MFSM(I) = IFSM(I)
  110 CONTINUE
C     MUTATE PARENT MACHINE TO FORM OFFSPRING
      MWSS = IWSS
      MCSN = IMCSN
```

```
      CALL MUTAT(MFSM,INFL,MWSS,MCSN)
    SET OUTPUTS OF OFFSPRING OPTIMALLY
      CALL STOPT(LWIN,NHS,MWSS,NIN,MFSM,IHS,IERMTX)
      CALL SCOR(MFSM,MWSS,SCORM,MPRED,MPERR)
      IF(SCORM.LT.SCORI) GO TO 115
  112 CONTINUE
  115 PRINT*,"SCORM= ",SCORM," MPRED= ",MPRED," MPERR= ",MPERR
      PRINT*,"MWSS = ",MWSS," MCSN = ",MCSN
    DOES OFFSPRING HAVE BETTER SCORE
      IF(SCORI.LE.SCORM) GO TO 130
    OFFSPRING HAS BETTER SCORE SO REPLACE PARENT BY IT
      IMCSN = MCSN
      IWSS = MWSS
      M2 = 2*IMCSN*NIN
      DO 120 I=1,M2
        IFSM(I) = MFSM(I)
  120 CONTINUE
      NMAC = 1
  130 CALL UPDATE(IFSM,IWSS)
    UP COUNTER ON CORRECT PREDICTIONS
      KPERR=IPERR
      KPRED = IPRED
      IF(NMAC.EQ.0) GO TO 118
      KPERR=MPERR
      KPRED=MPRED
  118 NPRR=NPRR+KPERR
      NPRD = NPRD+KPRED
      IF(NHS.LT.NDP) GO TO 100
      PRINT*,"NBR OF CORRECT PREDICTIONS = ",NPRD
      APRR=NPRR
      APRR=APRR/(NDP-NHSS)
      PRINT*,"AVERAGE PREDICTION ERROR= ",APRR
      PRINT 900, (IFSM(I),I=1,M2)
  900 FORMAT (/,8I4)
      STOP
      END
      SUBROUTINE INIT(NINT,JCSN,JFSM)
C
C   SUBROUTINE SETS UP INITIAL FINITE STATE MACHINE
C
      DIMENSION JFSM(60)
      DO 100 J=1,JCSN
        DO 100 I=1,NINT
C   COMPUTE POSITION IN FSM TABLE
          NCS = 2*((J-1)*NINT+I)
C   RANDOMLY SELECT NEXT STATE AND STORE IN FSM TABLE
          X = RANF(N)
          FNS = JCSN*X+1.
          INS = FNS
          JFSM(NCS-1) = INS
C   RANDOMLY SELECT OUTPUT AND STORE IN FSM TABLE
          X = RANF(N)
          FOT = NINT*X+1.
          IOT = FOT
          JFSM(NCS) = IOT
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INRD

C     SUBROUTINE READS IN DATA FOR INITIAL MACHINE,
C     ERROR MATRIX AND DATA TO BE OPERATED ON

      COMMON/IODEF/INPFSM,INPDAT,IOUFSM
      COMMON/FSMPA/IERMTX,IHS,IMCSN,ISS,LWIN,MXNS,NDP,NHS,NIN
      DIMENSION IERMTX(30), IHS(200)

C     READ DATA FOR INITIAL FSM
      READ(INPFSM,*)IMCSN,ISS,LWIN,MXNS,NHS,NIN

C     READ IN ERROR MATRIX
      KK = NIN*NIN
      READ(INPFSM,*)(IERMTX(I),I=1,KK)

C     READ IN DATA POINTS
      I = 1
100   READ(INPDAT,*) IHS(I)
      IF(IHS(I).LT.0) GO TO 110
      I = I+1
      GO TO 100
110   NDP = I-1
      RETURN
      END
```

```
      SUBROUTINE MUTAT(JFSM,INFLT,JWSS,JCSN)
C
C     THIS SUBROUTINE CHANGES NEXT STATE ASSIGNMENTS, ADDS STATE,
C     AND CHANGES START STATE-ALL RANDOMLY
C
        COMMON/FSMPA/IERMTX,IHS,IMCSN,ISS,LWIN,MXNS,NDP,NHS,NIN
        DIMENSION JFSM(60), IERMTX(30), IHS(200)
        IF(INFLT.EQ.1) GO TO 100
C     INITIALIZE
        MUCNT = 0
        RETURN
C     NORMAL ENTRY
  100   MUCNT = MUCNT+1
        JCSN = IMCSN
C     CHECK IF TIME FOR POSSBILY ADDING OR DELETING A STATE
C     OTHERWISE GO TO CHANGE NEXT STATE ASSIGNMENT
        IF(MUCNT.LT.10) GO TO 150
C     POSSIBLY ADD STATE
        MUCNT = 0
        X = RANF(N)
        IF(X.GT.0.5) GO TO 200
C     ADD STATE UNLESS MAXIMUM NBR OF STATES ALREADY REACHED
        IF(IMCSN.GE.MXNS) GO TO 200
C     ADD STATE
  105   JCSN = IMCSN+1
C     RANDOMLY ASSIGN NEXT STATE AND OUTPUT FOR EACH STATE/
C     INPUT PAIR IN NEW STATE
        DO 110 I=1,NIN
          X = RANF(N)
          FSN = JCSN*X+1.
          NNS = FNS
          NCS = 2*((JCSN-1)*NIN+I)
          JFSM(NCS-1) = NNS
          X = RANF(N)
          FOT = NIN*X+1.
          NOT = FOT
          JFSM(NCS) = NOT
  110   CONTINUE
C     RANDOMLY CHANGE THE NEXT STATE TO NEW STATE FOR FROM 1 TO NIN
C     INPUT/STATE PAIRS FROM ORIGINAL MACHINE
        X = RANF(N)
        FK = NIN*X+1.
        KK = FK
        MM = (JCSN-1)*NIN
        DO 120 I=1,KK
          X = RANF(N)
          FSI = 2.*(MM*X+1.)
          NSI = FSI
          JFSM(NSI-1) = JCSN
  120   CONTINUE
        RETURN
```

```
C     SELECT WHETHER TO CHANGE STATE OR NEXT STATE FOR
C     RANDOMLY CHOSEN SET OF STATE/INPUT PAIRS
  150 X = RANF(N)
      IF(X.GT.0.1) GO TO 170
C     RANDOMLY CHANGE START STATE
      X = RANF(N)
      FWSS = JCSN*X+1.
      JWSS = FWSS
      RETURN
C     CHANGE NEXT STATE RANDOMLY FOR RANDOM SET OF STATE/INPUT PAIRS
  170 MM = JCSN*NIN
      X = RANF(N)
      FN = 5.*X+1.
      NN = FN
      DO 180 I=1,NN
        X = RANF(N)
        FNS = JCSN*X+1.
        NNS = FNS
        X = RANF(N)
        FCS = 2.*(MM*X+1.)
        NCS = FCS
        JFSM(NCS-1) = NNS
  180 CONTINUE
      RETURN
C
CC    THIS SECTION DELETES STATE IF MORE THAN ONE STATE,
C     OTHERWISE GO TO ADD STATE
  200 IF(JCSN.EQ.1) GO TO 105
C
CC    THIS PORTION SELECTS DELETES A STATE FROM THE FINITE STATE
CC    MACHINE. HOWEVER, THE START STATE IS NEVER DELETED
C
C       JCSN   - NUMBER OF STTES IN FINITE STTE MACHINE
C       JFSM   - TABLE CONTAINING FINITE STATE MACHINE
C       NIN    - NUMBER OF INPUT ALPHABET SYMBOLS
C       NSTBD  - STATE TO BE DELETED (DETERMINED IN ROUTINE)
C       NSTBM  - NUMBER OF STATES TO BE MOVED
C       NWTBM  - NUMBER OF WORDS TO BE MOVED
C       IWFRM  - FROM POSITION MINUS ONE
C       IWTOO  - TO POSITION MINUS ONE
C       JWSS   - INITIAL START STATE
C
C     DECIDE WHICH STATE TO DELETE
  210 X = RANF(N)
      F = X*JCSN+1.0
      NSTBD = F
C     START STATE IS NOT DELETED
      IF(NSTBD.EQ.JWSS) GO TO 210
      IF(NSTBD.EQ.JCSN) GO TO 270
C     CALCULATE NUMBER OF WORDS TO MOVE IN TABLE MFSM
      NSTBM = JCSN NSTBD
      NWTBM = 2*NIN*NSTBM
```

```
C     MOVE STATES TO CLOSE GAP BUT FIRST GET ADDRESSES
        IWTOO = 2*NIN*(NSTBD-1)
        IWFRM = IWTOO+2*NIN
        DO 260 I=1,NWTBM
          JFSM(IWTOO+I) = JFSM(IWFRM+I)
  260 CONTINUE
C     DECREMENT NUMBER OF STATES
  270 JCSN = JCSN-1
C     GET SET TO TEST ALL NEXT STATE REFERENCES
        ITEST = JCSN*NIN*2
C     TEST WHETHER OR NOT TO CHANGE NEXT STATE REFERENCE
        DO 290 I=1,ITEST,2
          IF(JFSM(I).LT.NSTBD) GO TO 290
          IF(JFSM(I).GT.NSTBD) GO TO 280
          X = RANF(N)
          F = X*JCSN+1.0
          II = F
          JFSM(I) = II
          GO TO 290
  280     JFSM(I) = JFSM(I)-1
  290 CONTINUE
C     CORRECT START STATE, IF NECESSARY
        IF(JWSS.LT.NSTBD) GO TO 300
        JWSS = JWSS-1
  300 CONTINUE
        RETURN
        END
```

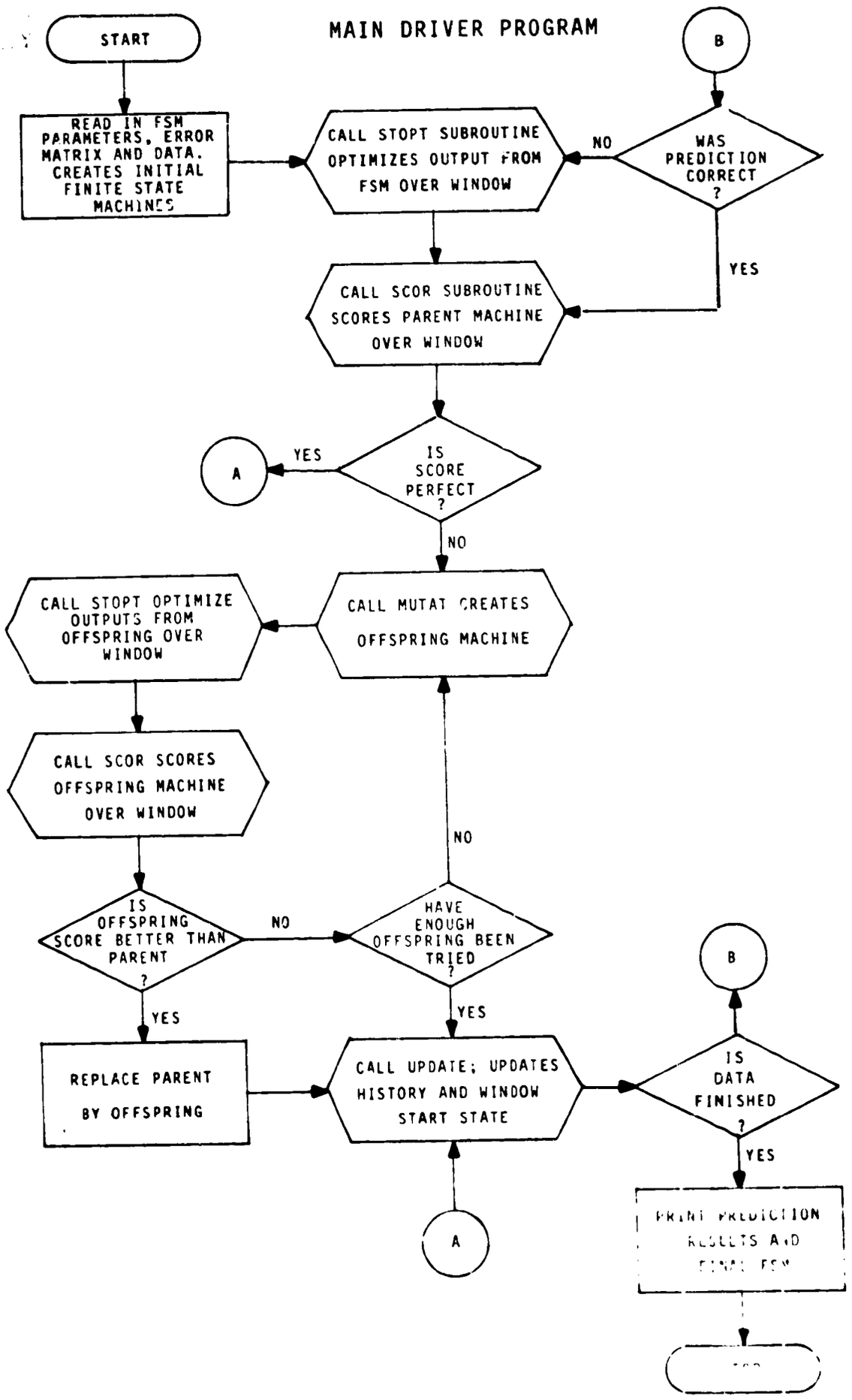
```
      SUBROUTINE SCOR(JFSM,JWSS,SCORX,JPRED,JPERR)
C
C     THIS SUBROUTINE SCORES THE MACHINE OVER THE HISTORY
C     IN THE WINDOW
C
      COMMON/FSMPA/IERMTX,IHS,IMCSN,ISS,LWIN,MXNS,NDP,NHS,NIN
      DIMENSION IHS(200),JFSM(60),IERMTX(30)
      SCORX = 0
      ISN = JWSS
      KK = 1
      ANP = NHS-1
C     IS WINDOW TO BE HISTORY?
      IF(LWIN.LT.0) GO TO 100
C     WINDOW IS FIXED LENGTH.  IS IT LONGER THAN HISTORY?
      IF(LWIN.GE.NHS) GO TO 100
C     SET UP FOR FIXED LENGTH WINDOW
      KK = NHS-LWIN+1
      ANP = LWIN-1
C     SCORE OVER WINDOW
  100 NHS1 = NHS-1
      DO 120 I=KK,NHS1
        IIN = IHS(I)
        NCS = 2*((ISN-1)*NIN+IIN)
        INS = JFSM(NCS-1)
        IOM = JFSM(NCS)
        IOD = IHS(I+1)
C     GET ERROR VALUE FROM ERROR MATRIX
        NCS = ((IOM-1)*NIN+IOD)
        SCORX = SCORX+IERMTX(NCS)
        ISN = INS
  120 CONTINUE
      SCORX = SCORX/ANP
C     CHECK IF PREDICTION IS CORRECT
      IIN = IHS(NHS)
      NCS = 2*((ISN-1)*NIN+IIN)
      IOM = JFSM(NCS)
      IOD = IHS(NHS+1)
      JPRED = 0
      IF(IOM.EQ.IOD) JPRED=1
      NCS=((IOM-1)*NIN+IOD)
      JPERR=IERMTX(NCS)
      RETURN
      END
```

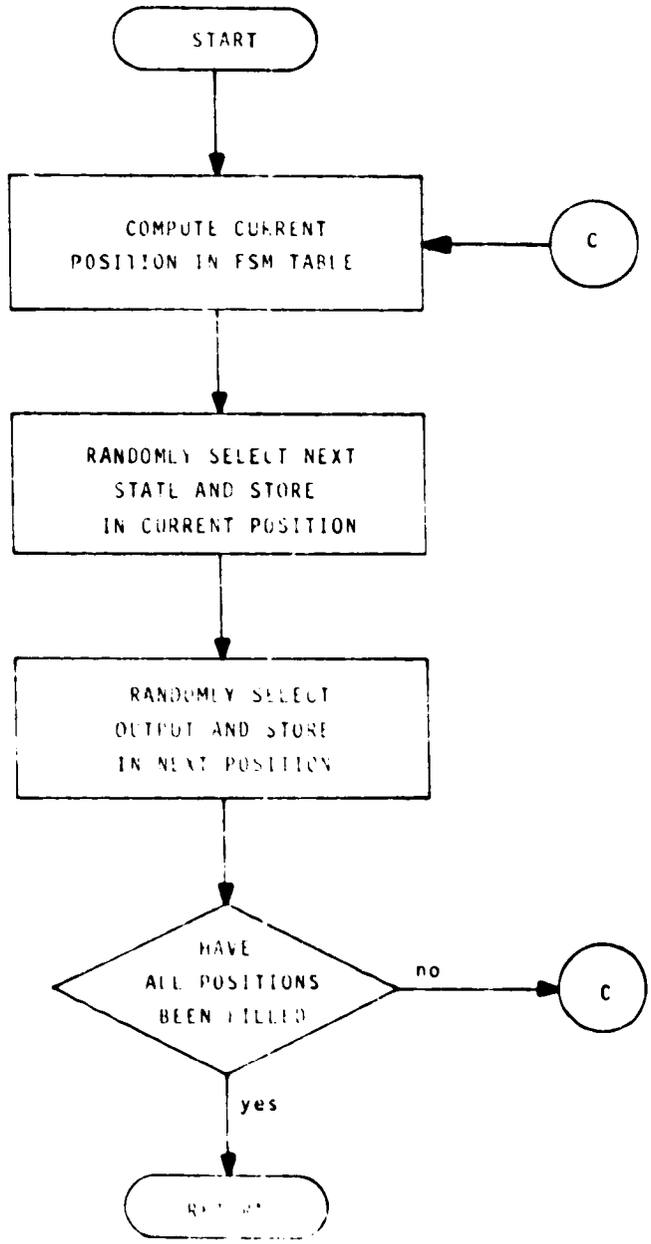
```
      SUBROUTINE STOPT(LWINT,NHST,IWSST,NINT,IFSMT,IHST,IERMT)
C
C   LWINT - LENGTH OF THE WINDOW
C   NHST  - CURRENT POSITION IN THE HISTORY (IHS IN COMMON)
C   IWSST - WINDOW START STATE
C   NINT  - ALPHABET SYMBOL SIZE
C   IFSMT - FINITE STATE MACHINE FOR SETTING OUTPUT
C   IERMT - ERROR MATRIX
C   IHST  - ADDRES OF HISTORY DATA
C
C   ISIPR(I,J) - STATE-INPUT TABLE WHERE
C                  I IS THE POSITION IN THE TABLE
C                  J IS 1, 2 OR 3 WHERE
C                    (I,1) CONTAINS THE STATE NUMBER
C                    (I,2) CONTAINS THE INPUT NUMBER
C                    (I,3) IS TOTAL NUMBER OF OUTPUTS ASSOCIATED
C                          WITH THIS STATE-INPUT PAIR
C   IACOP(I)    - THE MACHINE OUTPUT-DESIRED OUTPUT TABLE WHERE
C                   I IS THE POSTION IN THE TABLE
C   ISIPK       - THE COUNT OF UNIQUE STATE-INPUT PAIRS IN TABLE ISIPR
C   IAOPK       - THE COUNT OF TOTAL ENTRIES IN TABLE IACOP
      DIMENSION ISIPR(25,3),IACOP(50),IHST(200),IFSMT(60),IERMT(30),
     1          IERR(10)
C   CACULATE THE FIRST INPUT POSITION OF THE WINDOW
      NHSWP = NHST-LWINT+1
C   SET LOOP TO LENGTH OF WINDOW MINUS TWO
      ILOOP = LWINT-2
C   ZERO STATE-INPUT PAIR COUNT
      ISIPK = 0
C   ZERO OUTPUT COUNT
      IAOPK = 0
C   SET ISNT TO WINDOW START STATE
      ISNT = IWSST
C   NOW BUILD STATE-INPUT-OUTPUT TABLES
C   FIRST STATE-INPUT-OUTPUT ENTRY HANDLED AS A SPECIAL CASE
      ISIPR(1,1) = ISNT
      IINT = IHST(NHSWP)
      ISIPR(1,2) = IINT
C   SET OUTPUT COUNT FOR THE STATE-INPUT PAIR TO ONE
      ISIPR(1,3) = 1
C   GET NEXT STATE
      II = 2*((ISNT-1)*NINT+IINT)
      ISNT = IFSMT(II-1)
C   INCREMENT INPUT POINTER
      NHSWP = NHSWP+1
C   GET NEXT INPUT
      IINT = IHST(NHSWP)
C   STORE DESIRED OUTPUT
      IACOP(1) = IINT
C   SET OUTPUT COUNT TO ONE
      IAOPK = 1
C   SET COUNT OF UNQIUE STATE-INPUT PAIRS TO ONE
      ISIPK = 1
```

```
:   NOW BUILD THE BALANCE OF THE STATE-INPUT-OUTPUT TABLES
      DO 500 K=1,ILOOP
:   IS THIS STATE-INPUT PAIR IN THE TABLE?
        DO 100 IJK=1,ISIPK
          IF(ISNT.NE.ISIPR(IJK,1)) GO TO 100
          IF(IINT.NE.ISIPR(IJK,2)) GO TO 100
;   ARRIVE HERE, STATE-INPUT PAIR IS ALREADY IN THE TABLE
          IJKS = IJK
          GO TO 200
  100   CONTINUE
;   APPEND NEW STATE-INPUT-OUTPUT TO TABLE
;   INCREMENT COUNT
  110   ISIPK = ISIPK+1
        ISIPR(ISIPK,1) = ISNT
        ISIPR(ISIPK,2) = IINT
        ISIPR(ISIPK,3) = 1
;   GET NEXT STATE
        II = 2*((ISNT-1)*NINT+IINT)
        ISNT = IFSMT(II-1)
;   INCREMENT INPUT POINTER
        NHSWP = NHSWP+1
;   GET NEXT INPUT
        IINT = IHST(NHSWP)
;   INCREMENT OUTPUT COUNT
        IAOPK = IAOPK+1
;   STORE DESIRED OUTPUT
        IACOP(IAOPK) = IINT
        GO TO 500
;   CHECK IF THE IDENTIFIED STATE-INPUT PAIR IS LAST IN THE TABLE
  200   IF(IJKS.NE.ISIPK) GO TO 300
;   ARRIVE HERE, STATE-INPUT IDENTIFIED IS LAST IN THE TABLE
;   GET NEXT STATE AND OUTPUT
        II = 2*((ISNT-1)*NINT+IINT)
        ISNT = IFSMT(II-1)
C   INCREMENT OUTPUT COUNT FOR THIS STATE-INPUT PAIR
        ISIPR(ISIPK,3) = ISIPR(ISIPK,3)+1
C   ICREMENT INPUT POINTER
        NHSWP = NHSWP+1
C   GET NEXT INPUT
        IINT = IHST(NHSWP)
C   INCREMENT OUTPUT COUNT
        IAOPK = IAOPK+1
C   STORE DESIRED OUTPUT
        IACOP(IAOPK) = IINT
        GO TO 500
C   ARRIVE HERE, NECESSARY TO CREATE SPACE IN TABLE FOR OUTPUT
C   DETERMINE WHERE SPACE SHOULD BE IN TABLE IACOP
  300   ISUM = 0
        DO 320 III=1,IJKS
          ISUM = ISUM+ISIPR(III,3)
  320   CONTINUE
C   MOVE IACOP ENTRIES TO MAKE SPACE
        IMOVE = IAOPK-ISUM
        INEW = IAOPK+2
        IOLD = IAOPK+1
```

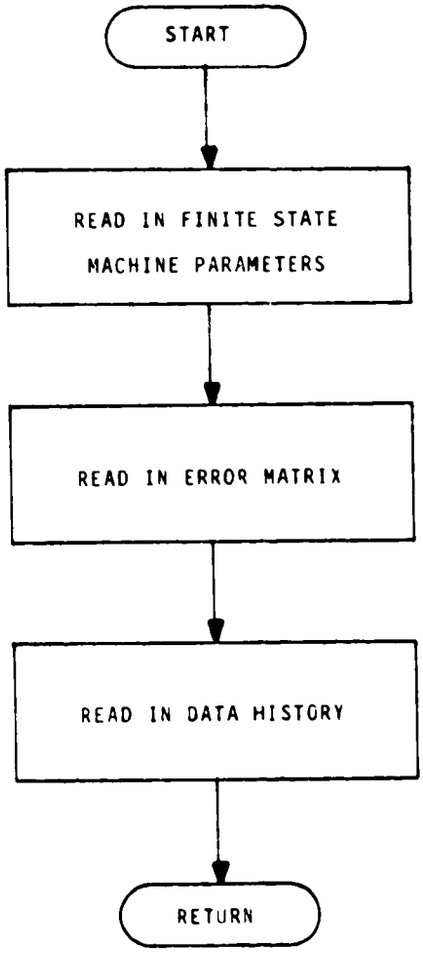
```
      DO 330 I=1,IMOVE
        IACOP(INEW-I) = IACOP(IOLD-I)
330   CONTINUE
  GET NEXT STATE
      II = 2*((ISNT-1)*NINT+IINT)
      ISNT = IFSMT(II-1)
  INCREMENT INPUT POINTER TO GET NEXT INPUT
      NHSWP = NHSWP+1
      IINT = IHST(NHSWP)
  STORE DESIRED OUTPUT
      IACOP(ISUM+1) = IINT
  INCREMENT OUTPUT COUNT
      IAOPK = IAOPK+1
  INCREMENT OUTPUT COUNT FOR THIS STATE-INPUT PAIR
      ISIPR(IJKS,3) = ISIPR(IJKS,3)+1
500 CONTINUE
  NOW THAT TABLES ISIPR AND IACOP ARE CONSTRUCTED, MACHINE
  OUTPUT CAN BE DETERMINISTICALLY SET TO MINIMIZE ERROR

  SET LOOP TO SET OUTPUT FOR THOSE STATE-INPUT PAIRS EXERCISED
600 DO 800 I=1,ISIPK
      IF(I.NE.1) GO TO 604
      IACKN=0
      GO TO 608
604   IACKN=IACKN+ISIPR(I-1,3)
608   DO 610 L=1,NINT
        IERR(L) = 0
610   CONTINUE
  SET ILOOP TO NUMBER OF TIMES THIS STATE-INPUT PAIR EXERCISED
      ILOOP = ISIPR(I,3)
  SET LOOP TO TRY ALL THE ALPHABET
      DO 700 J=1,NINT
        DO 700 K=1,ILOOP
  CALCULATE POSITION IN ERROR MATRIX TO OBTAIN ERROR FOR
  THIS TENTATIVE OUTPUT AND THE EXPERIENCES OUTPUT
          M = (J-1)*NINT+IACOP(K+IACKN)
          IERR(J) = IERR(J)+IERMT(M)
700   CONTINUE
  NOW FIND THE OUTPUT PRODUCING THE LEAST ERROR AND SELECT IT
      ICOMP = IERR(1)
  SET ISOP = 1 (TENTATIVE OUTPUT)
      ISOP = 1
      DO 750 IJ=2,NINT
        IF(IERR(IJ).GT.ICOMP) GO TO 750
        ICOMP = IERR(IJ)
        ISOP = IJ
750   CONTINUE
  CALCULATE WHERE DETERMINED OUTPUT IS TO BE STORED
      ISO = 2*((ISIPR(I,1)-1)*NINT+ISIPR(I,2))
  SET DETERMINISTIC OUTPUT
      IFSMT(ISO) = ISOP
800 CONTINUE
    RETURN
    END
```

```
      SUBROUTINE UPDATE(JFSM,JWSS)
C
C    THIS SUBROUTINE UPDATES START STATE. CURRENT HISTORY
C    LENGTH
C
      COMMON/FSMPA/IERMTX,IHS,IMCSN,ISS,LWIN,MXNS,NDP,NHS,NIN
      DIMENSION JFSM(60), IHS(200), IERMTX(30)
C    UPDATE HISTORY LENGTH
      NHS = NHS+1
C    IS WINDOW EQUAL HISTORY LENGTH
      IF(LWIN.LT.0) GO TO 100
C    IS WINDOW LONGER THAN HISTORY
      IF(LWIN.GE.NHS) GO TO 100
C    SET UP FOR FIXED WINDOW
      KK = NHS-LWIN
      IIN = IHS(KK)
      ISN = JWSS
C    UPDATE START STATE
      NCS = 2*((ISN-1)*NIN+IIN)
      INS = JFSM(NCS-1)
      JWSS = INS
  100 RETURN
      END
END OF FILE
```

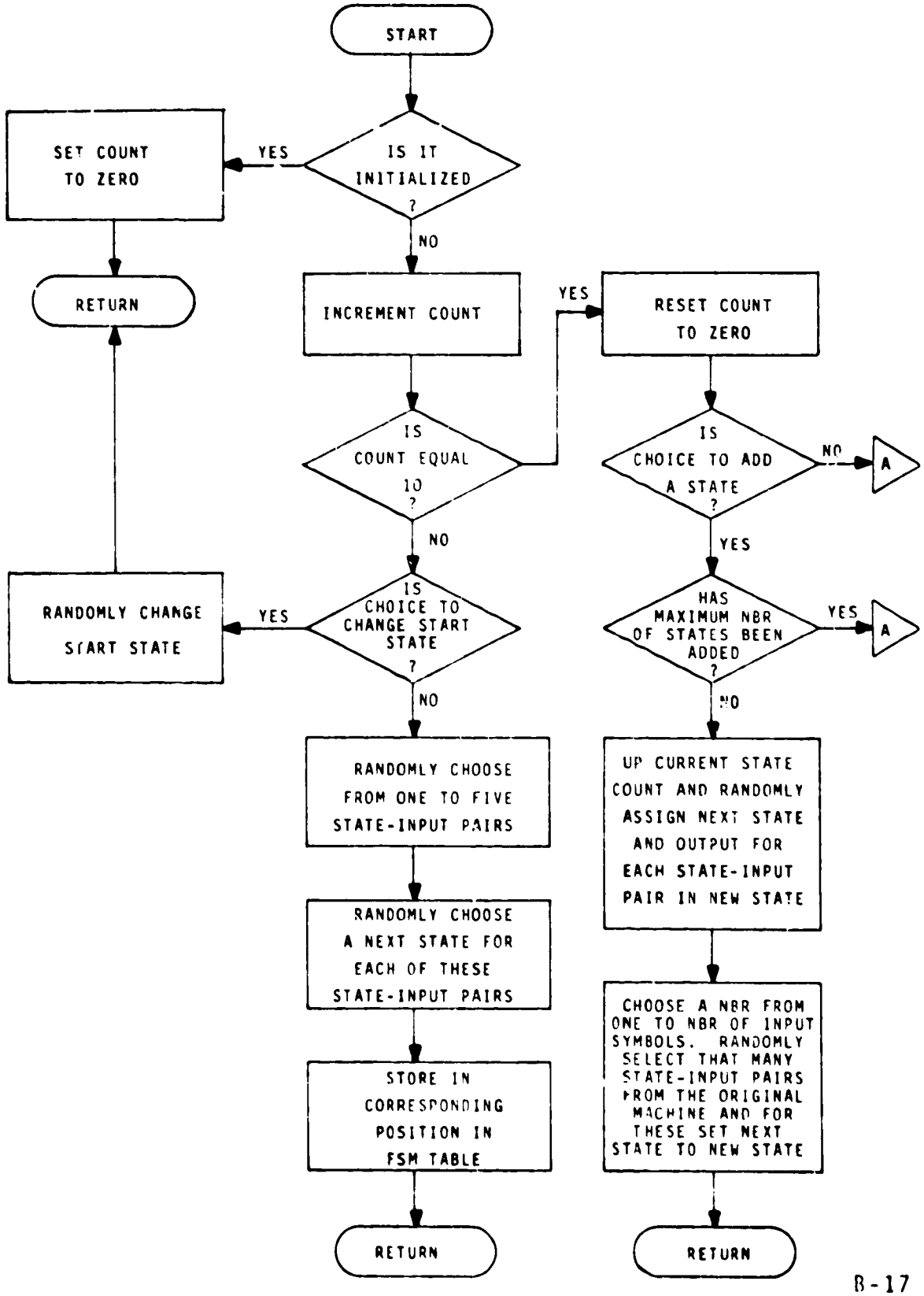

MAIN DRIVER PROGRAM
START
READ IN FSM PARAMETERS, ERROR MATRIX AND DATA. CREATES INITIAL FINITE STATE MACHINES
CALL STOPT SUBROUTINE OPTIMIZES OUTPUT FROM FSM OVER WINDOW
NO
WAS PREDICTION CORRECT ?
B
YES
CALL SCOR SUBROUTINE SCORES PARENT MACHINE OVER WINDOW
IS SCORE PERFECT ?
YES
A
NO
CALL MUTAT CREATES OFFSPRING MACHINE
CALL STOPT OPTIMIZE OUTPUTS FROM OFFSPRING OVER WINDOW
CALL SCOR SCORES OFFSPRING MACHINE OVER WINDOW
IS OFFSPRING SCORE BETTER THAN PARENT ?
NO
HAVE ENOUGH OFFSPRING BEEN TRIED ?
NO
YES
YES
REPLACE PARENT BY OFFSPRING
CALL UPDATE; UPDATES HISTORY AND WINDOW START STATE
IS DATA FINISHED ?
B
YES
A
PRINT PREDICTION RESULTS AND FINAL FSM



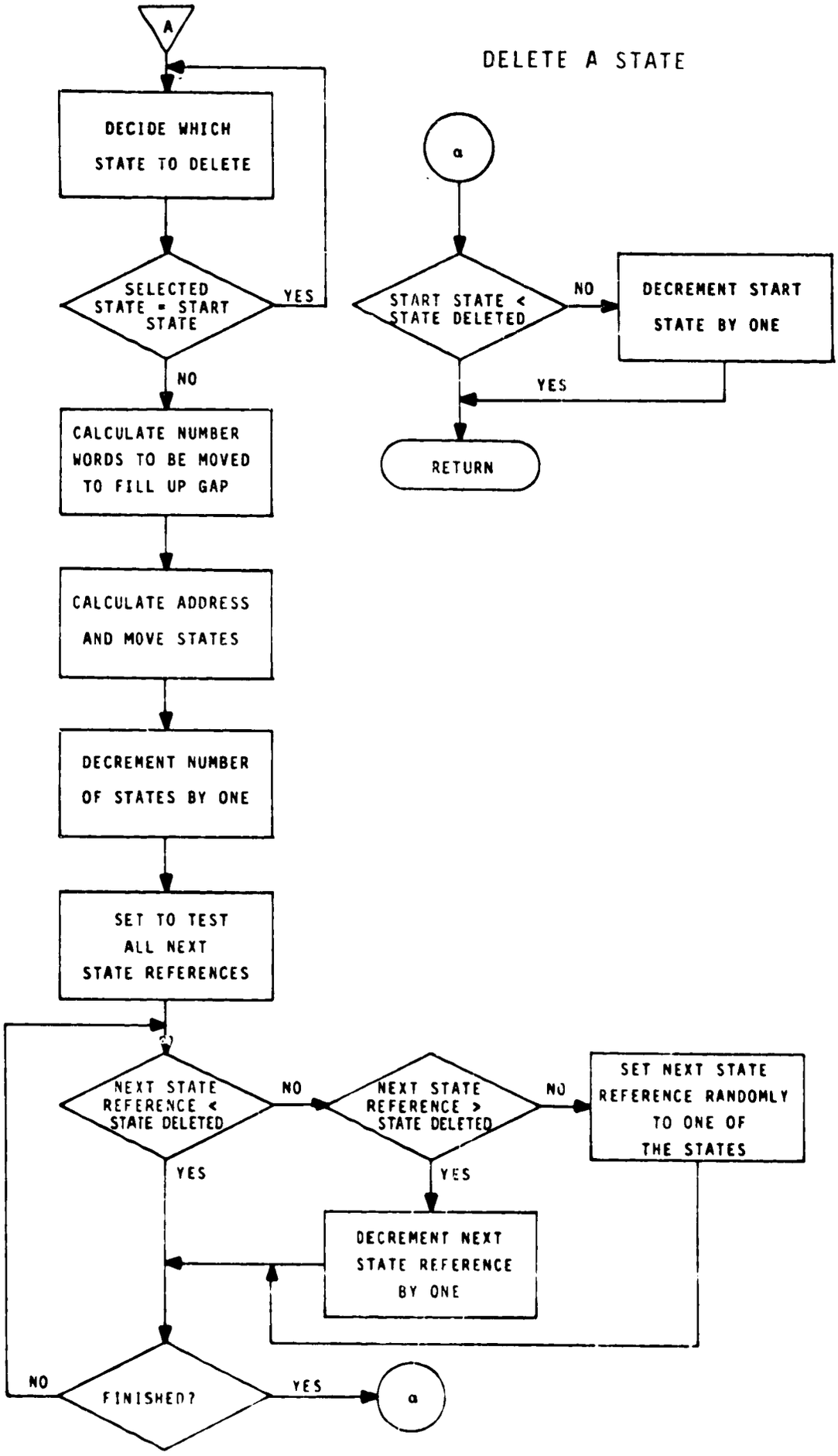
SUBROUTINE INRD



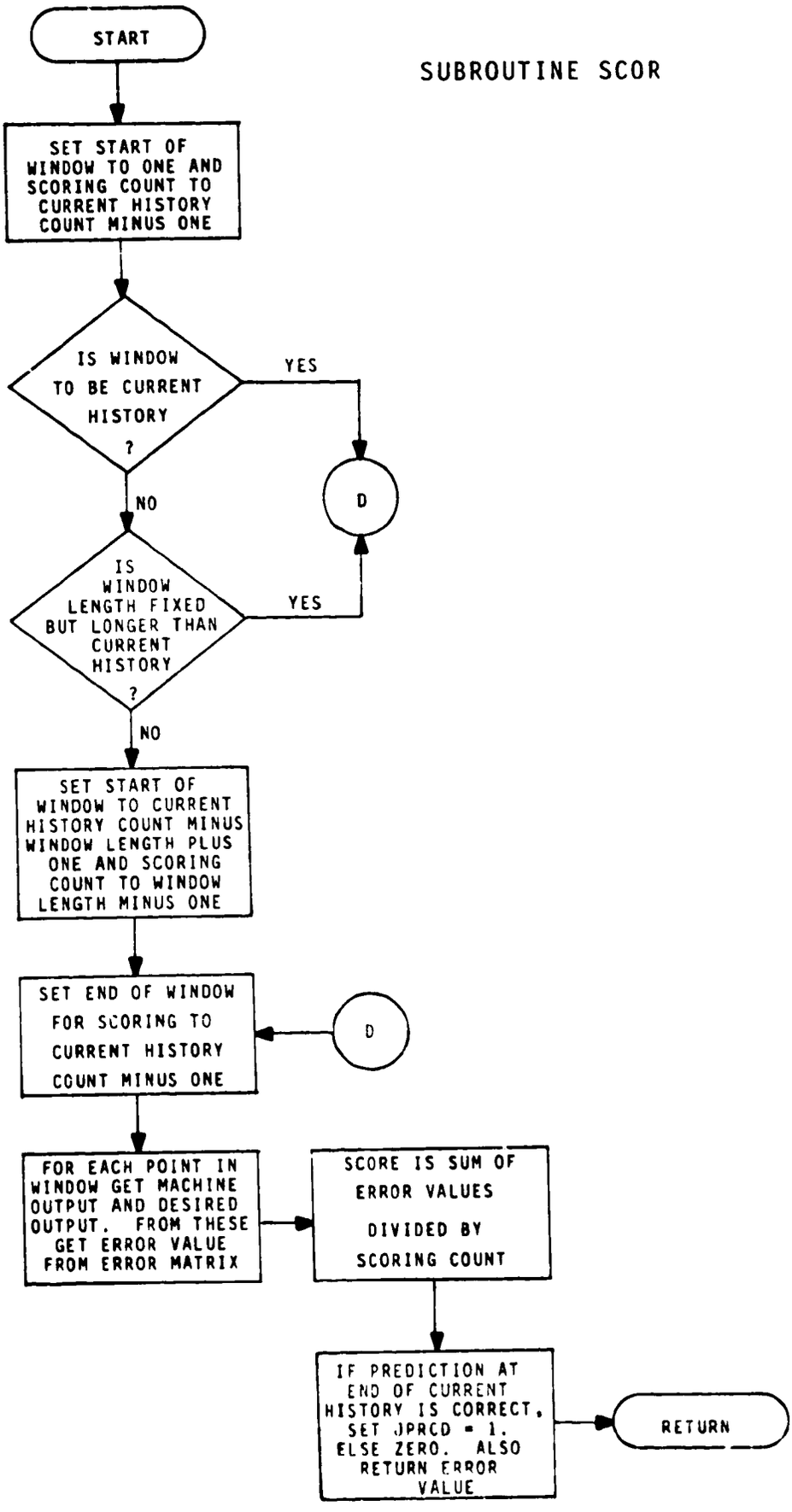
SUBROUTINE MUTAT



DELETE A STATE

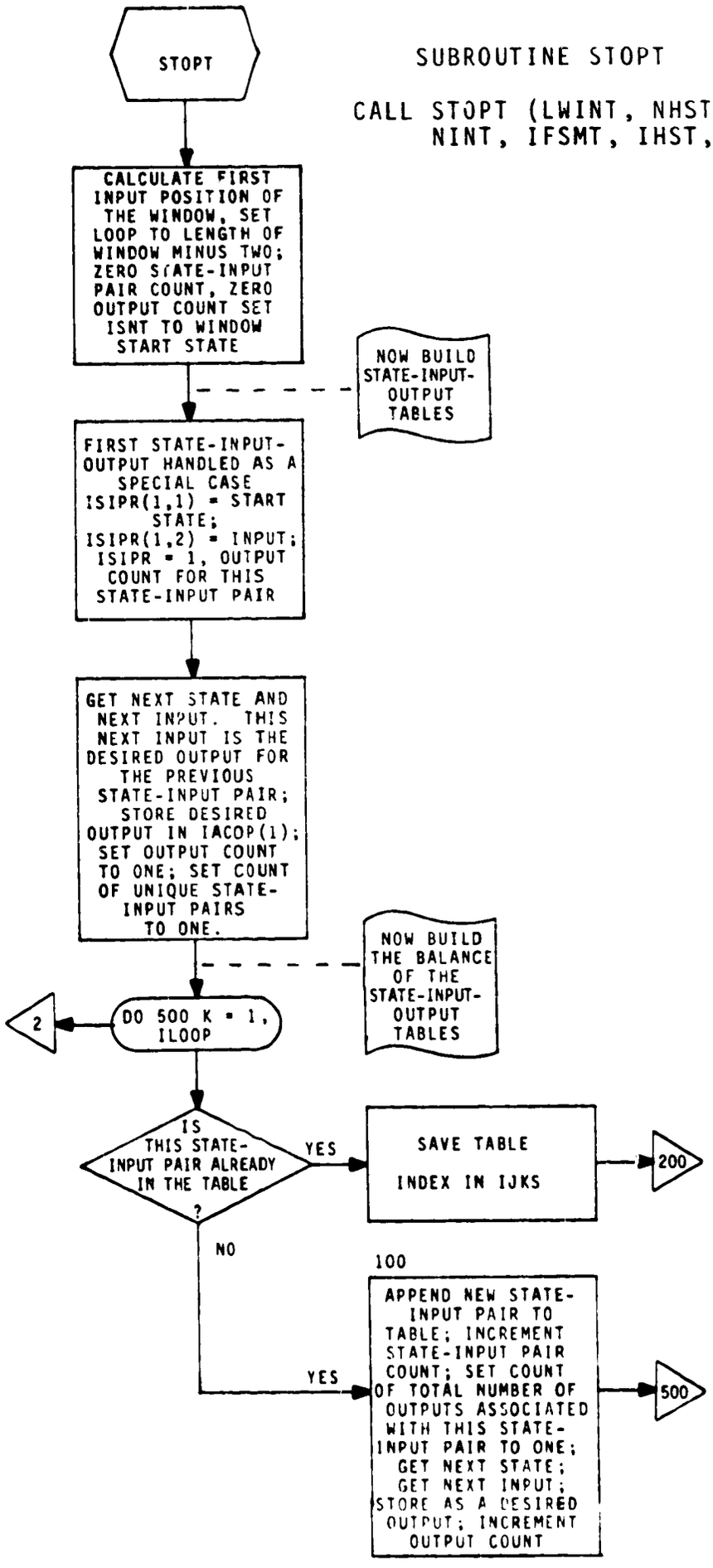


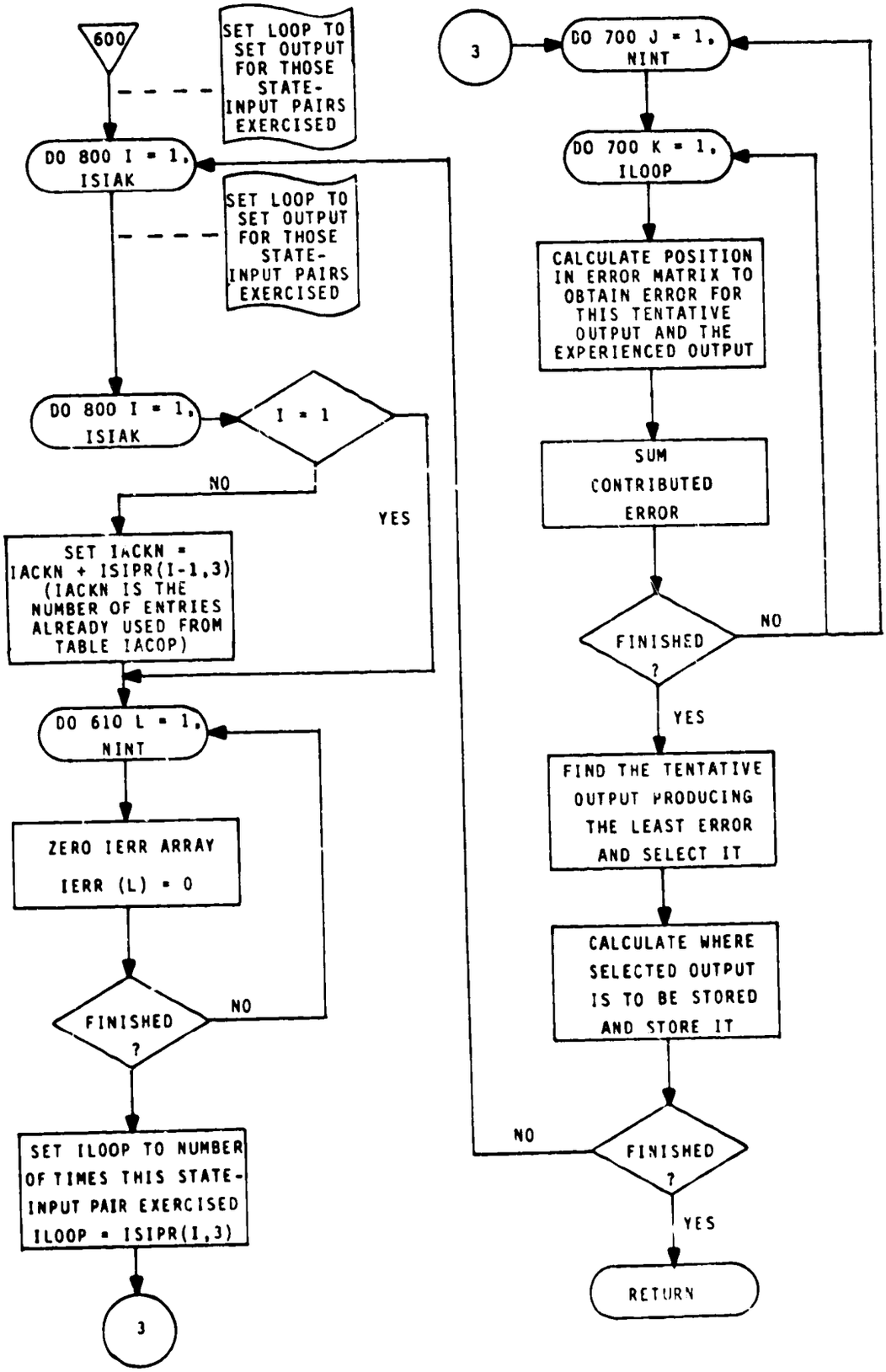
SUBROUTINE SCOR

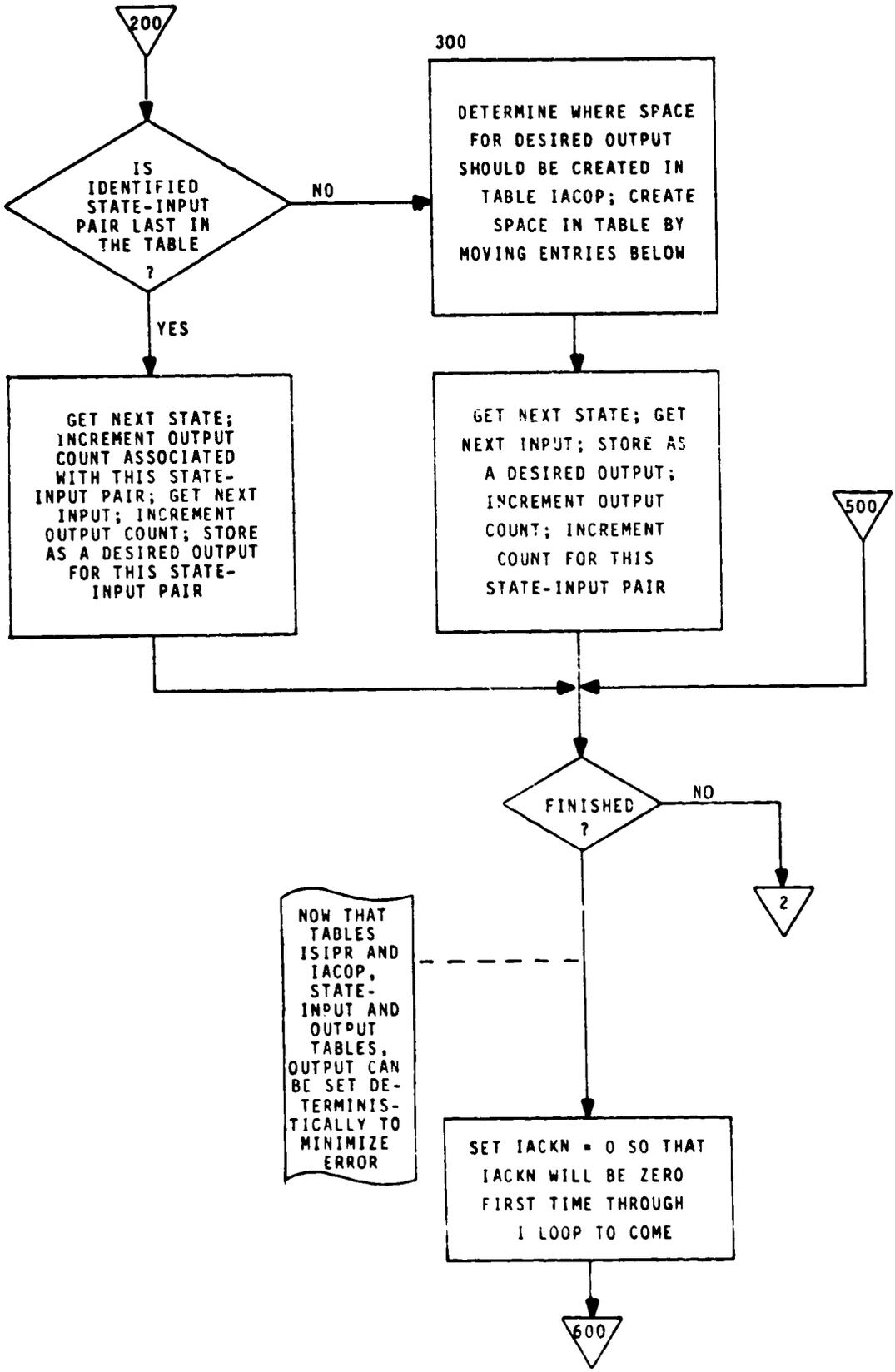


SUBROUTINE STOPT

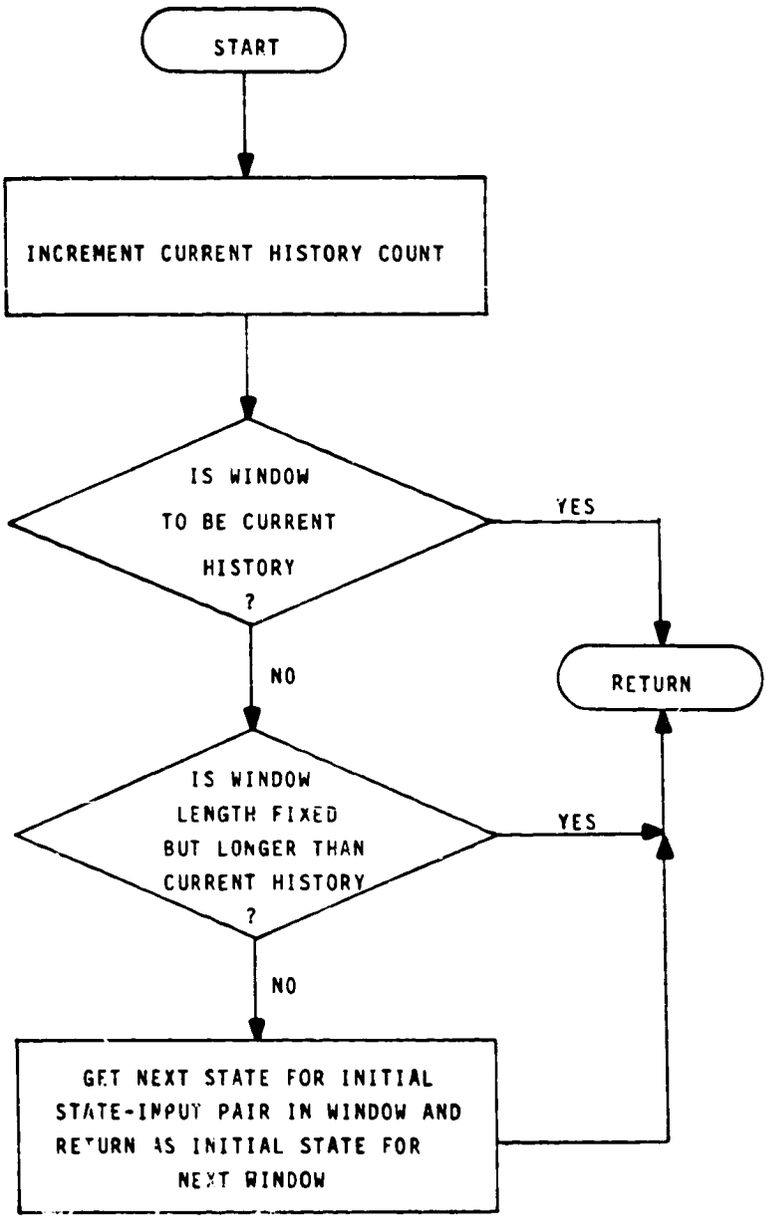
CALL STOPT (LWINT, NHST, IWSST,
NINT, IFSMT, IHST, IERMT)






200
IS IDENTIFIED STATE-INPUT PAIR LAST IN THE TABLE ?
NO
YES
300
DETERMINE WHERE SPACE FOR DESIRED OUTPUT SHOULD BE CREATED IN TABLE IACOP; CREATE SPACE IN TABLE BY MOVING ENTRIES BELOW
GET NEXT STATE; INCREMENT OUTPUT COUNT ASSOCIATED WITH THIS STATE-INPUT PAIR; GET NEXT INPUT; INCREMENT OUTPUT COUNT; STORE AS A DESIRED OUTPUT FOR THIS STATE-INPUT PAIR
GET NEXT STATE; GET NEXT INPUT; STORE AS A DESIRED OUTPUT; INCREMENT OUTPUT COUNT; INCREMENT COUNT FOR THIS STATE-INPUT PAIR
500
FINISHED ?
NO
2
NOW THAT TABLES ISIPR AND IACOP, STATE-INPUT AND OUTPUT TABLES, OUTPUT CAN BE SET DE-TERMINIS-TICALLY TO MINIMIZE ERROR
SET IACKN = 0 SO THAT IACKN WILL BE ZERO FIRST TIME THROUGH I LOOP TO COME
600

SUBROUTINE UPDATE



APPENDIX C

ON THE CONTROL OF ROBOTIC DEVICES

By

George H. Burgin

CONTENTS

CONTROL OF ROBOTIC DEVICES

By George H. Burgin, Decision Science, Inc.

SUMMARY

The control of robotic devices is a challenging and important task. Two primary factors contribute to that challenge. 1. The differential equations describing the behavior of a multilink manipulator turn out to be very complex and tedious to derive. 2. The manipulator is a highly non-linear device so that conventioanl control system design techniques are applicable only to a limited extent.

The presently applied or proposed control techniques are reviewed, particularly the "inverse problem" technique, the computed torque technique and the model-referenced adaptive control technique, and a new technique, which uses finite state machine gain adjusters, is proposed.

Next, the differential equations for a representative two-link model (upper arm and a lower arm) are derived and a basic linear feedback control system, operating about some linearized position of the system is designed. The entire system is simulated on a digital computer and representative time histories of responses to step inputs are shown, first for a control system which uses fixed gains over the entire operating envelope.

Next, a finite state machine (FSM) gain adjuster for the shoulder joint is designed. It receives an 8-symbol input alphabet, which encodes information about error and error rate. The outputs of the finite-state machine are the gains for the proportional and rate feedback signals.

A few representative sample reponses show the superiority of this control system over the one with fixed gains.

It is recommended to implement such a control system on the actual Puma hardwave and to compare its performance with the one predicted in this report.

INTRODUCTION

For this discussion, we will limit ourselves to mechanical manipulators which represent a subset of robotic devices. We exclude, therefore, such devices which might be used to explore--remotely controlled or autonomously-- such things as surfaces of planets.

Mechanical manipulators are chains of linkages connected by joints. Joints may be rotational or translational. Without loss of generality, we will assume that each joint has one degree of freedom; multiple rotational degrees of freedom can be represented by links of zero masses and length, each one with one single degree of freedom.

A typical industrial manipulator has six joints, seven links and a gripper (also called endeffectors). Figure 1 shows a commercially available manipulator, the PUMA 250, manufactured by Unimation, Inc. Two such manipulators are in operation for research purposes at the NASA Langley Research Center. Table I summarizes some of the key specifications of this manipulator.

Mechanical manipulators of this type have become increasingly important in recent years, and a great deal of effort has been spent for research in the area of simulation and control of industrial manipulators. During the 1981 joint automatic control conference in Charlottsville, not less than eighteen papers were presented on this subject! (1)

SYMBOLS

A,B,C,D	geometrical constants of system
g	acceleration due to earth gravity (9.81 m/sec^2)
I_1	moment of inertia of upper arm about its center of mass
$\underline{\underline{J}}$	inertia matrix

K_I	gain of integral feedback path
K_F	gain in forward loop
K_p	gain of proportional feedback path
K_r	gain of rate-feedback rate
ℓ_1	distance between shoulder joint and elbow joint
m_p	mass of payload
m_1	mass of upper arm
m_2	mass of lower arm
q_i	i-th generalized coordinate
Q_i	i-th generalized force (moment)
r_1	distance from shoulder joint to center of mass of upper arm
r_2	distance between elbow joint and center of mass of lower arm
RS	magnitude of step-input at elbow joint
RS_1	magnitude of step-input at shoulder joint
s	Laplace operator
T	kinetic energy
V	potential energy
$\alpha,\beta,\gamma,\delta,\varepsilon$	system variables which are functions of (ϕ_1,ϕ_2 and A,B,C,D)
ζ	damping factor
ϕ_{1_c}	commanded angle ϕ_1
ϕ_{2_c}	commanded angle ϕ_2
τ	vector of generalized input forces (moments)
ω_n	natural frequency

$\Delta\phi_{1_{Th}}$ = Threshold value for absolute value in error of ϕ_1 for encoding for input symbol to FSM

$\dot{\phi}_{1_{Th}}$ = Threshold value for error rate for encoding for input symbol to FSM.

ROBOT CONTROL BACKGROUND

Equations of Motion

Mechanical manipulators are highly nonlinear devices. The primary cause for the nonlinear nature of manipulators is the changing moments of inertia of the various links. In the expressions for the moments of inertia, trigonometric functions of the generalized coordinates appear. Since the generalized coordinates vary over a wide range (sometimes over a full 360°), linearization of the trigonometric functions cannot be performed over the full operating range of the manipulator. Other important nonlinear terms are products of derivatives of generalized coordinates with trigonometric functions of generalized coordinates. In addition to these nonlinear effects, which are due to the changing physical configuration of the manipulator as it moves through space, there are the usual nonlinearities associated with any device with moving parts linked by joints: nonlinear friction effects, hysteresis, and so forth.

Much of the literature on robotic manipulators is concerned with the formulation of the equations of motion. This problem is by no means trivial. Walker and Orin (2) point out that for mechanisms with only two or three degrees-of-freedom, these equations can usually be derived manually but that for mechanisms with more than three degrees-of-freedom, a separate computer program is required to symbolically derive the equations of motion.

The Different Control Schemes

In the available literature, a number of schemes to control mechanical manipulators have been proposed.

The Inverse Problem Technique: Here, the required input torque for each joint is computed as a function of the desired joint acceleration, $\ddot{q}_d$, the joint velocity $\dot{q}_d$ and the joint position q_d

and the actual $\ddot{q}$, $\dot{q}$ and q. To explain this method, consider the general form of the equations of motion for a six joint manipulator to be:

$$\underline{\underline{J}}(q)\ddot{q} + \underline{\underline{V}}\dot{q} + \underline{f}(\dot{q},q) + \underline{g}(q) = \underline{\tau} \qquad (1)$$

where $\underline{\underline{J}}(q)$ is a 6 by 6 inertia matrix

$\underline{\underline{V}}$ is a 6 by 6 viscuous friction matrix

$\underline{f}(\dot{q},q)$ is a 6 by 1 vector defining Coriolis and centrifugal force terms

$\underline{g}(q)$ is a 6 by 1 vector defining the gravity forces

$\underline{\tau}$ is a 6 by 1 vector of the generalized input forces (moments)

Then, the desired input torque vector is computed as:

$$\underline{\tau} = \underline{\underline{J}}_c(q)\left\{\ddot{q}_d + K_1(\dot{q}_d - \dot{q}) + K_2(q_d - q)\right\} + \underline{\underline{V}}_c\,\dot{q} + \underline{f}_c(\dot{q},q) + \underline{g}_c(q) \qquad (2)^*$$

where K_1 and K_2 are some gains. (Luh, et al in (3) assume K_1 and K_2 as being scalar gain constants. One of the purposes of this investigation is to determine whether the manipulator performance could be substantially improved by replacing these scalar gain constants by automatically adjusted [by finite state machines]) gain vectors.

Ideally, we would like to have $\underline{q}$ approaching $\underline{q}_d$, then we would have

$$\underline{\underline{J}}_c(q) = \underline{\underline{J}}(q)$$

$$\underline{\underline{V}}_c = \underline{\underline{V}}$$

$$\underline{f}_c(\dot{q},q) = \underline{f}(\dot{q},q)$$

$$\underline{g}_c(q) = \underline{g}(q)$$

---

*The subscript c indicates values computed by the control program.

If these four conditions were satisfied, the equation could be written as

$$\tau = J(q)\left\{\ddot{q}_d+K_1(\dot{q}_d-\dot{q})+K_2(q_d-q)\right\} +V\dot{q}+f(\dot{q},q)+g(q) \quad (3)$$

Now, we can equate (1) and (3) and we obtain

$$J(q)\left\{\ddot{q}_d-\ddot{q}+K_1(\dot{q}_d-\dot{q})+K_2(q_d-q)\right\} = 0$$

If we call $q-q_d$ the position error e_q, and keep in mind that the intertia matrix $J(q)$ is nonsingular, we obtain

$$\ddot{e}_q + K_1\dot{e}_q + K_2e_q = 0$$

This then leads to a control system of the form as shown in Figure 2.

This method appears to be restricted to those applications where the trajectory of the hand is preplanned, which makes it possible to know exactly, all the way along the manipulator's path, q, $\dot{q}$ and $\ddot{q}$. (In other words, these quantities become q_d, $\dot{q}_d$ and $\ddot{q}_d$ in the above equations.)

Reference (3) mentions that "proper choice of values for K_1 and K_2 guarantees the convergency of errors. It does not coordinate the speed of convergence for all six joints. Thus, some joint may converge faster than the others." Reference (3) is very vague on the proper choice of K_1 and K_2, which seems to be a key problem.

The Computed Torque Method: Earlier work performed in the field of manipulator control system design compared the computed torque method with conventional position servo control (4). This is a Jet Propulsion Laboratory internal report and could not be made available in time for the preparation of this report. Important for the computed torque method (and any other method which requires the real-time calculation of the generalized moments and forces) is the efficiency by which these torques may be computed. This efficiency, in turn, depends on how the equations of motion, forming the basis for these torque calculations, are formulated. Much of the modern literature on robot control is devoted to this problem, reference (2) presents a good summary of this subject. At the present time, the consensus among researchers in this field seems to be that the

formulation by the Newton-Euler method yields computationally more efficient equations than the formulation by Lagrange's method. It is usually claimed that in the Newton-Euler approach, the computation time grows linearly with the number of links, whereas in the Lagrange approach, it grows with the fourth power of the number of links. Therefore, any method relying on calculating the torques on-line in real-time, almost has to use Newton-Euler for deriving the equations unless the number of links is very small.

Very recently, however, Silver (5) in a benchmark paper has shown that it is possible to overcome some of the difficulties generally attributed to the Lagrangian method. He uses a recursive Lagrangian formulation such that there is no longer a fundamental difference in the computational efficiency between Lagrangian and Newton-Euler formulations.

The Model-Referenced Adaptive Control Technique: - This technique was primarily developed by Professor Dubowsky at UCLA and was first described for the continuous system in (6), and was recently expanded to the Discrete-Time case as described in (7). The requirement for robots which deliver uniformly high performances over a wide range of systems operating conditions precludes the exclusive use of classical linear control systems. Adaptive model-referenced control system can "learn" to compensate for nonlinearities arising from the various geometrical configurations of the manipulator, and they may also be designed to adapt for changing payload characteristics. This is especially important for manipulators employed to retrieve satellites or parts of satellites of unknown mass.

Figure 3, reproduced from (7), shows the block diagram of a continuous model referenced adaptive control system. It uses a linear, second order reference model and the adaptation occurs on the gains $K_p(i)$ (positional feedback gain for all joints, that is, $i = 1...6$ for a 6 joint manipulator), and the rate feedback gains $K_v(i)$. An interesting finding of this paper was that for parameters of common industrial manipulators, K_v and K_p do not need to be varied independently.

Gains Adjusted by Finite-State Machines: This study proposes a new approach to solve the robot control problem. Rather than employing an algorithm which adjusts the gains $K_p(i)$ and $K_v(i)$ based on the observed difference between the robot's response to a command input and the reference model's response to the command input, the gains will be adjusted based on the output of a finite-state machine which receives as input the robot's response. A similar approach was used previously to adjust the gains of an aircraft stability augmentation control system (8). To derive the algorithm for gain adjustment, a specific manipulator was selected.

DEVELOPMENT OF A ROBOT CONTROL SYSTEM

Model Definition

The Puma 250 manipulator was chosen as a device to be controlled for two reasons: First, it is a manipulator typically representing today's commercially available manipulator, and second, two Puma's 250 are presently in use at the NASA Langley Research Center. This made it possible to obtain physical data about the manipulator which are, in general, not available from a manipulator's manufacturer.

It has been shown, for example in (7), that for most applications, the motion of the wrist joints have little effect on the dynamic performance of the lower joints. It is, therefore, justified to assume the wrist joints (endeffector joints) of the Puma to be locked with respect to the lower arm.

Another simplifying assumption was made for this study. It is assumed that the waist rotation is decoupled from the shoulder and elbow rotation. This is the case if the manipulator does not move simultaneously around the waist joint, shoulder joint and elbow joint, but keeps the shoulder and elbow joint in a locked position while it is moving around the waist joint. The study, therefore, concentrates on analyzing simultaneous motions about shoulder and elbow joint. The manipulator model, therefore, is a two link and two joints system as shown in Figure 4.

Derivation of the Equations of Motion for the Selected Model

In the previous section, Newton-Euler and Lagrange were compared primarily with respect to the computational efficiency of the resulting equations of motion. It is important to realize that in the proposed control system, there is no solution (and not even a formulation) of the manipulator's equations of motion required. The only reason why we need to know the equations of motion and why we have to solve them by some numerical method is for the purpose of simulating the closed-loop system of manipulator and control system. There exists no requirement to perform this simulation in real time. Computational efficiency is, therefore, of no practical importance. This is quite in contrast to the "inverse plant" and the "computed torque" technique, where torques have to be computed online and in real time. We can, therefore, compare the two techniques to obtain the equations of motion simply based on their relative merits of simplicity of the derivation.

Newton's-Euler's Method: Applying this method to the system shown in Figure 4 results in two translational and one rotational equation for each one of the links, resulting in 6 second order differential equations. But, obviously, the system has only two degrees of freedom; therefore, we have to formulate kinetic constraint equations; for example, the coordinates of the elbow joint in inertial space must be the same for the upper arm as for the lower arm. It is our experience that considerable skill is required to formulate exactly the right number of constraint equations, certainly a drawback of the Newton-Euler Method. On the other hand, proper formulation in the Newton-Fuler method will provide internal reaction forces and moments, presenting an important advantage in robot design.

Lagrangian Method - Once appropriate generalized coordinates have been defined, the Lagrange's method is relatively straight forward: Express the system's kinetic energy in terms of the generalized coordinates and their derivatives, let the kinetic energy be T. Express the potential energy (including all conservative forces) as V, then Lagrange's equation states:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial \dot{q}_i} + \frac{\partial V}{\partial q_i} = Q_i \qquad i=1 \ldots n$$

n=number of degrees of freedom

Thus, a system of n second order differential equations will result. Note that Q_i is the generalized force (or moment) for the i-th equation, all nonconservative forces and moments must be included in Q_i, specifically, the externally applied moments at the individual joints. Table II summarizes some of the advantages and disadvantages of the two methods.

Lagrange's Method for Two-Link System: - A natural choice for the two generalized coordinates in the two-link system are the two angles ϕ_1 and ϕ_2 as shown in Figure 4.

$$T = 1/2\dot{\phi}_1^2 \left\{ I_1 + m_1 r_1^2 + m_2 \ell_1^2 + m_p \ell_1^2 \right\}$$

$$+ \frac{1}{2}\left(\dot{\phi}_1 + \dot{\phi}_2\right)\left\{ I_2 + m_2 r_2^2 + m_p \ell_2^2 \right\}$$

$$+ \dot{\phi}_1 \left(\dot{\phi}_1 + \dot{\phi}_2\right)\left(m_2 r_2 + m_p \ell_2\right) \ell_1 \cos \phi_2$$

$$V = \left\{ m_1 r_1 + \left(m_2 + m_p\right) \ell_1 \right\} g \sin \phi_1$$

$$+ \left(m_2 r_2 + m_p \ell_2\right) g \sin \left(\phi_1 + \phi_2\right)$$

Letting

$$A = I_1 + m_1 r_1^2 + m_2 \ell_1^2 + m_p \ell_1^2$$

$$B = \left(m_2 r_2 + m_p \ell_2\right) \ell_1$$

$$C = I_2 + m_2 r_2^2 + m_p \ell_2^2$$

$$D = m_1 r_1 + \left(m_2 + m_p\right) \ell_1$$

and then formulating Lagrange's equations and collecting terms yields:

$$\ddot{\phi}_1(A+C+2B\cos\phi_2) + \ddot{\phi}_2(C+B\cos\phi_2) =$$

$$= \dot{\phi}_2(2\dot{\phi}_1+\dot{\phi}_2)B\sin\phi_2 - Dg\cos\phi_1 - \frac{Bg}{\ell_1}\cos(\phi_1+\phi_2) + Q_1$$

$$\ddot{\phi}_1(C + B\cos\phi_2) + \ddot{\phi}_2 C =$$

$$= -\dot{\phi}_1^2 B\sin\phi_2 - \frac{Bg}{\ell_1}\cos(\phi_1+\phi_2) + Q_2$$

It is easy to show that Q_1 is the applied (control) moment in the shoulder joint, which we will call M_1, while Q_2 is the applied (control) moment in the elbow joint, let it be M_2.

To be able to solve the above system of two second order differential equations by standard numerical methods, we have to solve them for $\ddot{\phi}_1$, and $\ddot{\phi}_2$. Before doing this, we introduce the following:

$$\alpha = A + C + 2B\cos\phi_2$$

$$\beta = C + B\cos\phi_2$$

$$\delta = C$$

$$\gamma = \dot{\phi}_2(2\dot{\phi}_1+\dot{\phi})B\sin\phi_2 - Dg\cos\phi_1 - Bg/\ell_1\cos(\phi_1+\phi_2)$$

$$\varepsilon = -B\sin\phi_2 - \frac{Bg}{\ell_1}\cos(\phi_1+\phi_2)$$

so that

$$\alpha\ddot{\phi}_1 + \beta\ddot{\phi}_2 = \gamma + M_1$$

$$\beta\ddot{\phi}_1 + \delta\ddot{\phi}_2 = \varepsilon + M_2$$

or

$$[A]\,\vec{\ddot{\phi}} = \vec{b}$$

and

$$[A]^{-1} = \frac{1}{\delta\alpha - \beta^2}\begin{bmatrix} \delta & -\beta \\ -\beta & \alpha \end{bmatrix} = \frac{1}{\Delta}\begin{bmatrix} \delta & -\beta \\ -\beta & \alpha \end{bmatrix}$$

and, solved for $\ddot{\phi}_1$ and $\ddot{\phi}_2$

$$\ddot{\phi}_1 = \frac{1}{\Delta}\left\{\delta(\gamma + M_1) - \beta(\varepsilon + M_2)\right\}$$

$$\ddot{\phi}_2 = \frac{1}{\Delta}\left\{-\beta(\gamma + M_1) + \alpha(\varepsilon + M_2)\right\}$$

This concludes the derivation of the equations of motion. It is obvious that any increase in the number of links soon increases the complexity of the equations of motion beyond what one can derive manually.

Design of the Basic Linear Feedback Control System

For a preliminary design of the control system, certain estimates about the geometrical dimensions, the masses and the moments of inertia of the Puma manipulator had to be made. Figure 5 illustrates our assumption about the shape of the lower arm. The distance between elbow joint and the waist joint was taken from Unimation's drawing of the Puma, as shown in Figure 1, all other geometrical dimensions in Figure 5 are estimated values. Note that for a first design, we assumed the arm to be homogeneous; this assumption is quite inaccurate since in reality, the mass of the arm is concentrated around the two joints (where the DC servo-motors are placed) and little mass is around the center of the arm.

If we designate with A the area of the arm, and with V its volume, it follows from Figure 5:

$$A = \frac{.15 + 0.1}{2} * 0.575\ m^2 = 0.071875\ m^2$$

$$V = 0.071875 * 0.03\ m^3 = 2.15625\ 10^{-3}\ m^3$$

Assuming a homogeneous mass distribution of 7860 kg/m^3, the arms total mass will be 16.95 kg.

Under the same assumptions, we may calculate the area's moment of inertia (I_{zz}) about its center of mass, which is approximately

$$I_{22} \cong 0.5 \text{ kg m}^2$$

For simplicity, we assume the same geometry for the upper arm as we have shown in Figure 5 for the lower arm. We can, therefore, summarize the constants appearing in the equations of motion as follows:

$$\ell_1 = \ell_2 = 0.42 \text{ m}$$

$$r_1 = r_2 = 0.17 \text{ m}$$

$$m_1 = m_2 = 16.95 \text{ kg}$$

$$I_{1_{zz}} = I_{2_{zz}} = 0.5 \text{ kg m}^2$$

$$g = 9.81 \text{ m sec}^{-2}$$

$$m_p = 2.5 \text{ kg}$$

A First-Cut Design for the Lower Arm Alone: To get a reasonable structure of the control system and approximate values of gains such that the response of the arm to a commanded step input in angular displacement follows a desired second order response type such that the physically available control torques of the Puma's DC servo motors are not exceeded, we will now proceed to determine values for the two gains K_P (gain of the proportional feedback) and K_R (gain of the rate feedback, called K_V in Dubowsky's paper (7).

Figure 6 shows a control system with proportional and rate feedback for the lower arm, in which $M(m_p)$ indicates the moment about the elbow joint due to the payload. Linearizing the system about $\theta_2 = 0$ yields the following equations:

$$E(s) = R(s) - K_R s\phi_2(s) - K_P\phi_2(s)$$

$$I(s) = K_F E(s) + M(m_p)$$

$$s^2\phi_2(s) = \frac{1}{I_{2_{elbow}}} \cdot I(s)$$

which results in the following transfer function for ϕ_2.

$$\phi_2(s) = \frac{\frac{1}{I_{2_{elbow}}}\left(K_F R(s) + M(m_p)\right)}{s^2 + \frac{K_F K_R}{I_{2_{elbow}}} s + \frac{K_F K_P}{I_{2_{elbow}}}}$$

To obtain reasonable values for K_F, K_P and K_R, we can first neglect the gravity term $M(m_p)$ and, without loss of generality, we can set $K_P = 1$ so that we can write the transfer function in the standard form for linear second order systems.

$$\frac{\phi_2(s)}{R(s)} = \frac{K_F/I_{2_{elbow}}}{s^2 + \frac{K_F K_R}{I_{2_{elbow}}} s + \frac{K_F}{I_{2_{elbow}}}}$$

which we may compare with the normal form:

$$\frac{C(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

such that

$$\frac{K_F}{I_{2_{elbow}}} = \omega_n^2$$

and

$$\frac{K_F \cdot K_R}{I_{2_{elbow}}} = 2\zeta\omega_n .$$

We will determine K_F and K_R so that, for a step input of R, the integral of time multiplied by the absolute value of the error (IATE) will be minimized; in other words:

$$IATE = \int_0^\infty |E(t)| \cdot t \, dt \Rightarrow \text{minimum}$$

It is well known that for a linear second order system IATE is minimized if

$$\zeta = \frac{\sqrt{2}}{2}$$

(See for example Reference 10, page 93).

We may specify a second condition which we want to satisfy, for example the time to the first peak, which is (reference 9, page 30).

$$t_p = \frac{\Pi}{\omega_n \sqrt{1-\zeta^2}}$$

Substituting $\frac{\sqrt{2}}{2}$ for ζ, we obtain

$$\omega_n = \frac{\Pi}{t_p} \sqrt{2}$$

For the Puma manipulator, a time of 0.5 seconds to the first peak appears to be a reasonable, physically relizable choice; thus,

$$\omega_n = \frac{\Pi}{0.5} \sqrt{2} = 8.88 \text{ sec}^{-1}$$

The moment about the elbow is equal to:

$$I_{2_{elbow}} = I_{2_{zz}} + r_2 m_2 = 3.38 \text{ kg m}^2$$

Thus, for the linearized system:

$$K_F = I_{2_{elbow}} * \omega_n^2 = 266\ m^2 kg\ sec^{-2}$$

and

$$K_R = \frac{2\zeta\omega_n I_{2_{elbow}}}{K_F} = 0.1593\ sec$$

At this point, it seems highly desirable to simulate the actual, nonlinear manipulator system, using the feedback system as developed above but dropping the assumption of small angular displacements about $\phi_2 = 0$. By doing so, we will gain insight into how far the results developed for the linearized system are valid for the actual, nonlinear system. We interrupt, therefore, the development of the control system at this point and describe the digital computer simulation of the manipulator.

DEVELOPMENT OF A MANIPULATOR COMPUTER SIMULATION

Overview

The goal of the computer simulation of the manipulator is to have a tool available which is flexible enough to allow the analyst to investigate many different control system designs. It is, therefore, required that the simulation be modular and well structured. One of the difficulties in any digital simulation of a continuous system is the proper choice of the integration step size; this is particularly important in the simulation of a manipulator receiving step inputs for angular positions. It is, therefore, required that an integration method which allows an easy automatic step size adjustment, based both on absolute and relative error criteria, be used. Remember, we are not simulating a linear second order (or higher order) system, but a highly nonlinear system where the pole zero location continuously changes as the manipulator links move relative to each other. A Runge-Kutta type integration method

most appropriate in a situation like this. Fehlberg (12) developed a highly accurate fifth order Runge-Kutta method. In one of the most recent simulation packages, developed by Pritsker, this method is used (13). It seemed appropriate, therefore, to use SLAM as simulation language for the control system development.

The Final Version of the Control System With Fixed Gains

The Single Arm Control Loop: - Figure 7 shows the final version of the control device, using fixed gains. As can be seen by comparing Figure 7 with Figure 6, a term proportional to the error integral has been added to the control system, so that we have more or less a conventional PID regulator. The error-integral term will force, for a step input of R(s), the angular displacement of the arm to assume, at steady state, the desired angular displacement.

It is easy to show that the transfer function for Figure 7 can be expressed as:

$$\phi(s) = \frac{\frac{R(s)K_I}{I_{zz}} + s\left(\frac{R(s)K_F + M}{I_{zz}}\right)}{s^3 + s^2\frac{K_F K_R}{I_{zz}} + s\left(\frac{K_I K_R + K_p K_F}{I_{zz}}\right) + \frac{K_p K_I}{I_{zz}}}$$

The steady state behavior for a step input $R(s) = \frac{R}{s}$ is:

$$\lim_{t \to \infty} \phi(t) = \lim_{s \to 0} s\,\phi(s) = \frac{R}{K_p}$$

Since $K_p = 1$, $\lim_{t \to \infty} \phi(t) = R$

which is the desired steady state response.

The Stability of the Single Arm Control Loop: - Due to the changes in the moments of inertia for large angular deflections, it is difficult to determine stability boundaries. However, one gets a good "feel" for the stability of the system by performing a root locus analysis for the above defined transfer function. This requires the solution of the cubic equation.

$$s^3 + s^2 \frac{K_F K_R}{I_{ZZ}} + s\left(\frac{K_I K_R + K_p K_F}{I_{ZZ}}\right) + \frac{K_p K_I}{I_{ZZ}} = 0$$

or

$$s^3 + a_1 s^2 + a_2 s + a_3 = 0$$

which can be shown to be equivalent to finding the eigenvalues of the matrix A where

$$A = \begin{pmatrix} -a_1 & -a_2 & -a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

(See for example ref. 14, page 233)

A computer program to find eigenvalues was available (EIGEN, California State University, CTS). Figure 8 shows a typical root locus plot for the above transfer function for variable payload masses. It shows the roots to lie in the desirable region for payloads from 1 kg to 5 kg and still be acceptable for zero payload.

Time Histories of Responses to Step Inputs of Varying Magnitudes and Signs for Single Arm: - Remember that the analyses performed so far were for a system linearized about $\phi_2 = 0$. It is therefore necessary to investigate the dynamics for large step inputs, because the moment of inertia depends on the angle ϕ_2. The response of the system will also be asymmetrical for a commanded step input of the same magnitude but in opposite directions. This is illustrated in Figure 9. The system was initially at rest with the upper arm fixed at $\phi_1 = 0$ and the lower arm supported with $\phi_2 = 0$. At time $t = 0^+$, the support was removed and simultaneously the step input command was applied. Control system gains were as indicated on Figure 8 which guaranteed that control torques did not exceed those physicaly attainable on the PUMA (see Table 1). Commanded step inputs were $\phi_{2c} = 0$ degrees, $\phi_{2c} = \pm 45$ degrees, and $\phi_{2c} = \pm 135$ degrees. The difference in response to a +135 degree and a -135 degree commanded angle is interesting and can easily be explained by noting that in the first case, the moment generated by the payload first opposes the motion, but after the initial overshoot, supports the desired motion; while in the second case, the gravity first acts in the direction of the desired motion, but after the angular displacement exceeds -90 degrees, gravity of the payload opposes the motion. Notice, for example, that in the first case after .2 seconds, the arm reaches an angle of +35 degrees while in the second case, the angle is -57 degrees.

Simultaneous Control of Both Arms: - The extension of the control system from a single arm to both arms simultaneously is straightforward. The same type of control system is applied to both joints, the only difference being the magnitudes of the gains. Figure 10 shows the block diagram for the two-link system in a form suitable for direct translation into the subroutine STATE as required by SLAM. (Note that this would also be suitable for being programmed on an analog computer.) The quantities ϕ_{1_c} and ϕ_{2_c} represent the commanded angles. The other quantities

correspond to the symbols as used in the section "Lagrange's Method for Two-Link System." Figures 11 through 14 show time history responses of the two controlled angles for various combinations of step inputs.

Note that Figure 13 and Figure 14 illustrate the motion of the robot moving the payload from the same initial position

$$(x = \ell_1 + \ell_2 \; ; \quad y = 0)$$

to the same position

$$\left(x = \frac{\ell_1 + \ell_2}{\sqrt{2}} \; ; \quad y = 0\right)$$

in Figure 13 such that at the terminal position $\Phi_1 = +45°$; $\Phi_2 = -90°$; while in Figure 14, the terminal position is $\Phi_1 = -45°$, $\Phi_2 = +90°$. Note the strongly asymmetrical motion, particularly of the angle Φ_1.

Figure 14 shows the motion of the payload in the x-y plane as function of time for the conditions shown in Figure 13.

A Control System with Gains Adjusted by A Finite-State Machine

Description of the Gain Adjuster: - As it was stated in the section "The Different Control Schemes," we proposed a new approach to solve the robot control problem by modifying some of the feedback gains based on the state of a finite-state machine rather than based on the observed difference btween the robots actual response and the response of the reference model.

We will first demonstrate the need for gain adjustment. Consider the case where Φ_1 and Φ_2 are initially zero ... want tc move the payload from ... point on the x-axis ...

both arms are extended to some other point on the x-axis, defined either by:

$$\phi_1 = +45° \qquad = -90° \qquad \text{(case 1)}$$

or

$$\phi_1 = -45° \qquad = +90° \qquad \text{(case 2)}$$

(Compare with Figure 14, showing the motion for the first case).

Figure 15 shows the response (ϕ_1 only) for fixed gains. The difference in the response between the two cases is due to the asymmetry of the moments due to gravity and therefore becomes more pronounced the heavier the payload is.

Only very limited time remained under this contract to develop the finite-state machine gain adjuster. It was therefore decided to adjust only the gains for the controller of the upper arm (ϕ_1). This is the angle which is more difficult to control because the moment of inertia about the shoulder joint depends on the elbow angle. In the following examples, the gains for the controller of ϕ_2 will remain constant.

A state-output finite-state machine was devised to set the gains. The machine consists of eight states, associated with each state are two gain values, one for K_{F_1} and one for K_{R_1} according to the following table:

State Number	K_{F_1}	K_{R_1}
1	600	0.3
2	600	0.2
3	800	0.15
4	800	0.1
5	300	0.4
6	300	0.2
7	500	0.3
8	500	0.2

The input alphabet to the finite-state machine consists of eight symbols, the integers 1 through 8. Th input symbol to the FSM is determined as follows:

$$\text{Let IN3} = 1 \quad \text{if sign } (\phi_{1_d} - \phi_1) \text{ is positive}$$
$$= 0 \quad \text{otherwise}$$

$$\text{Let IN2} = 1 \quad \text{if } | \phi_{1_d} - \phi_1 | \geq \Delta\phi_{1_{Th}}$$
$$= 0 \quad \text{otherwise}$$

$$\text{Let IN1} = 1 \quad \text{if } | \dot{\phi}_1 | \geq \dot{\phi}_{1_{Th}}$$
$$= 0 \quad \text{otherwise}$$

Then, the input symbol to the FSM is defined as

$$\text{INSYMB} = 8 - (4 * \text{IN3} + 2 * \text{IN2} + \text{IN1})$$

The state transition logic is such that no matter in which state the FSM is at the time of receipt of an input symbol, the FSM will transit into the state with the same number as the input symbol.

We realize, of course, that the logic of such a finite-state machine gain adjuster is very simple (it amounts to a table look-up). Two remarks, however, are appropriate. First, the few examples of responses shown in the next section show that the system works significantly better than the one previously described with fixed gains.

Second, the intent was to have a more sophisticated FSM to adjust the gains. A first improvement would expand the input alphabet such that it contains information about the angle ϕ_2 and maybe the angular rate $\dot{\phi}_2$. Since the moment of inertia about the shoulder joint increases with decreasing angle ϕ_2, it seems desirable to increase K_p with decreasing angle ϕ_2.

It is clear that the design of the finite-state machine gain adjuster, as the input alphabet size increases, becomes more and

more difficult and can no longer be achieved by mere intuition of the designer. When the point of complexity which exceeds the intuitive method of designing a FSM gain adjuster is reached, it would be beneficial to incorporate evolutionary programming into the control system design. The evolutionary program could be started with the final "best" machine found by intuition and analysis. This machine would serve as the parent machine for the evolutionary process.

The evolution of the FSM gain adjuster could now be performed on-line, in real-time. Thus, the FSM gain adjuster would not only take care of the time varying dynamic properties (such as changing moments of inertia) of the manipulator, but it would also compersate for unknown physical parameters of the manipulator. With today's available computer resources, such an evolutionary control system design appears to be technically feasible. It might well provide a solution to the problem of designing truly adaptive, multipurpose robots operating in an unknown environment.

Sample Responses: - In Figures 13 and 14 the system responses were shown for the two cases:

1. $\phi_{1C} = +45°$ $\phi_{2C} = -90°$
2. $\phi_{1C} = -45°$ $\phi_{2C} = +90°$

Figures 16 and 17 show the system response for the same two cases, but having the gains adjusted with the FSM gain adjuster. Not only is the response smoother and faster, but much more symmetrical when using the FSM gain adjuster. These two examples clearly show the superiority of the system with adjusted gains over a system with constant gains.

A last example (Figure 18) shows the system response for a ramp input. Even though the system was designed with step inputs in mind, the response to ramp inputs is quite satisfactory.

A complete listing of the simulation source program is presented in Figure 19.

CONCLUSIONS

Robots, whose angles between individual links are controlled by servo motors, can be controlled to yield fast response and remain stable over the entire operating envelope. This may be achieved by control system employing proportional, integral and rate feedback, in which some or all of the gains are adjusted by finite-state machines. These FSM's are "driven" by the systems past response to commanded inputs.

Such control systems might offer advantages over adaptive model-referencing systems because in a digitally controlled robot, finite-state machines are easily integrated with the remainder of the control system.

It would seem worthwhile to implement the control system, as described in this report, in actual hardware and to compare its performance with the one of the simulation.

REFERENCES

1. Proceedings of the 1981 Joint Automatic Control Conference, June 17-19, 1981, Charlottsville, Virginia.

2. Walker, M.W.; and Orin, D.E.: Efficient Dynamic Computer Simulation of Robotic Mechanisms. Proceedings of the 1981 Joint Automatic Control Conference, June 17-19, 1981, Charlottsville, Virginia.

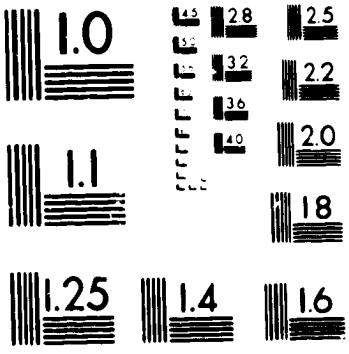
3. Luh, J.Y.S.; Walker, M.W.; and Paul, R.P.C.: Resolved--Acceleration Control of Mechanical Manipulators, IEEE Transactions on Automatic Control, Vol. 25, No. 3, June 30, 1980, pp. 468-474.

4. Markiewicz, B.R.; Analysis of Computed Torque Drive Method and Comparison with Conventional Position Servo for a Computer Controlled Manipulator. Technizal Memorandum 33-501, Jet Propulsion Laboratory, March, 1973.

5. Silver, W.M.; On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators. Proceedings of the 1981 Joint Automatic Control Conference, June 17-19, 1981, Charlottsville, Virginia.

6. Dubowsky, S.; and Des Forges, D.T.: The Application of Model Referenced Adaptive Control to Robotic Manipulators. Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME, Vol. 101, No. 3, pp. 193-200, September, 1979.

7. Dubowsky, S.; On the Adaptive Control of Robotic Manipulator: The Discrete-Time Case. Proceedings of the 1981 Joint Automatic Control Conference, June 17-19, 1981, Charlottsville, Virginia.



8. Burgin, G.H.; and Walsh, M.J.: Finite-State Machines As Elements in Control Systems. Annual Symposium Record of the 1971 IEEE Systems, Man and Cybernetic Group, pp. 241-246.

9. Thaler, G.J.; Design of Feedback Systems. Dowden, Hutchinson & Ross, Inc., Strandsburg, Virginia, 1973.

10. Santee, D.P.; Automatic Control System Technology. Prentice Hall, Englewood Cliffs, New Jersey, 1980.

11. Shinners, S.M.; Control System Design. John Wiley, New York, 1964.

12. Fehlberg, E.; Low-Order Classical Runge-Kutta Formulas with Step-Size Control and Their Application to Some Heat Transfer Problems. NASA Report TR R-315, Huntsville, Alabama, April, 1979.

13. Pritsker, A.; and Pegden, C.: Introduction to Simulation and SLAM. John Wiley, New York, 1979.

14. Ralston, A.; and Wilf, H.: Mathematical Methods for Digital Computers. John Wiley, New York, 1960.

C-2

TABLE I

SELECTED PUMA SPECIFICATIONS

Data Provided by Unimation, Inc.

Degrees of Freedom:		5 rotational no translational
Rotatonal Limits		
(1)	Waist:	±160°
(2)	Shouder:	±165°
(3)	Elbow:	±135°
(4)	Wrist:	±105°
(5)	Joint 5:	±180°
Maximum Static Force at the "hand":		58 N (13 lb.)
Maximum Payload:		223 N (5 lb.)
Maximum Hand Acceleration:		1 g
Maximum Hand Velocity:		1.0 m/s (3.3 feet/s)
Control:		Electric DC Servomotors

Data Obtained from Measurements at LRC

Maximum torque in elbow joint (averaged between up and down motion)	$M_{2_{max}}$	= 163 Nm
Maximum torque in shoulder joint (averaged between up and down motion)	$M_{1_{max}}$	= 244 Nm

TABLE 2

Comparison between Newton/Euler and Lagrange Method.

	NEWTON/EULER	LAGRANGE
ADVANTAGES	• MORE "VISIBLE" CORRELATION BETWEEN EQUATIONS OF MOTION AND PHYSICAL SYSTEM • INTERNAL (REACTION) FORCES AND MOVEMENTS	• SETTING UP EQUATIONS RELATIVELY STRAIGHT FORWARD • ONLY MINIMUM AMOUNT OF EQUATIONS REQUIRED
DISADVANTAGES	• FORMULATION AND SOLUTION OF CONSTRAINT EQUATIONS IS TRICKY • LIKELIHOOD OF SIGN ERRORS HIGH	• SOLVE FOR HIGHEST DERIVATIVES OF EACH STATE VARIABLE • DOES NOT PROVIDE REACTION FORCES AND MOMENTS

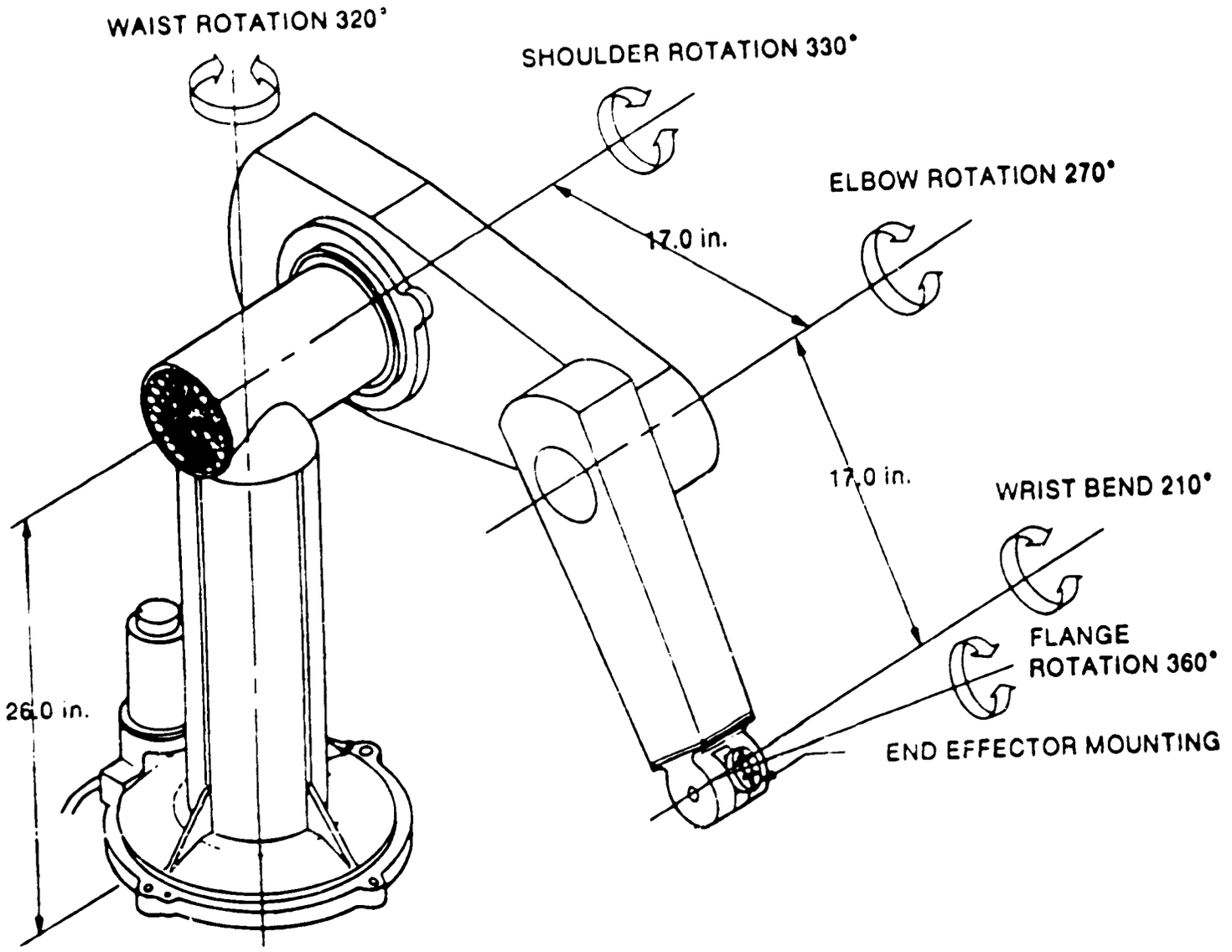


Figure 1: PUMA Manipulator

Unimation Inc.

NASA LANGLEY RESEARCH CENTER

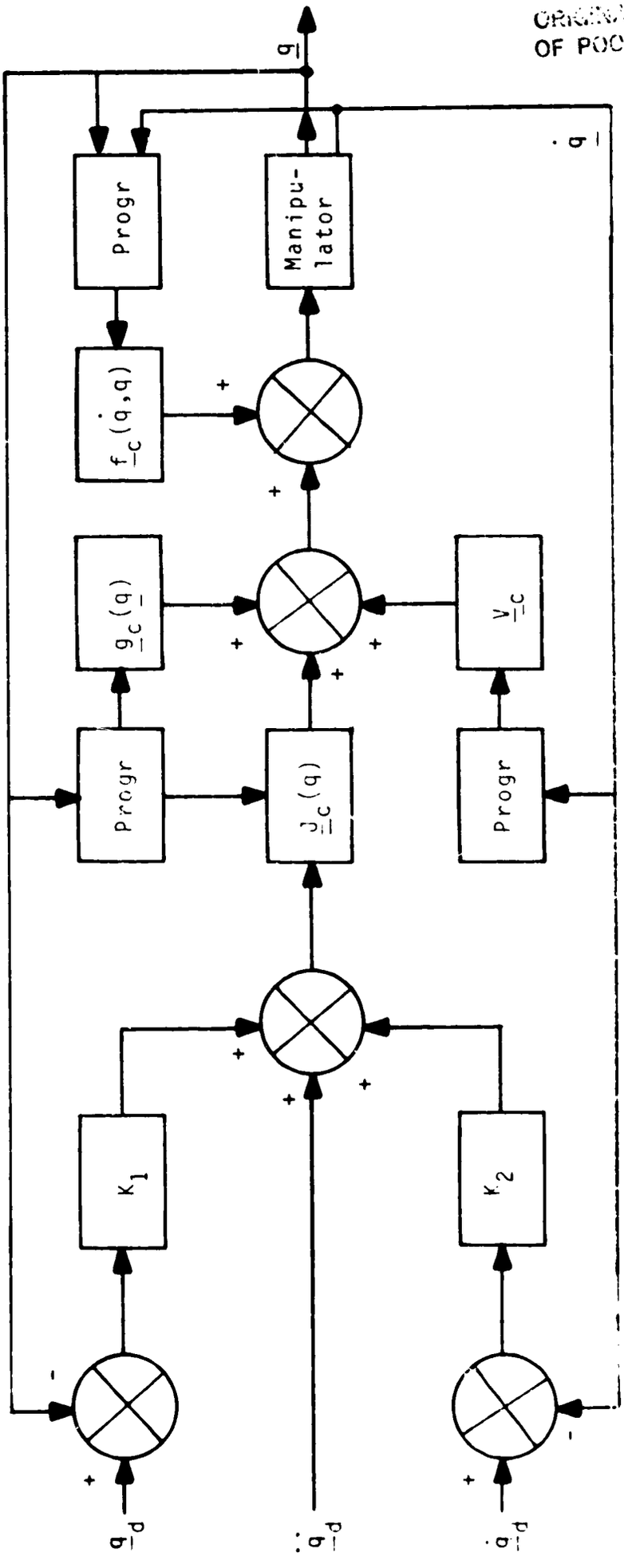


Figure 2: Block Diagram for the "Inverse Problem" Manipulator Control System

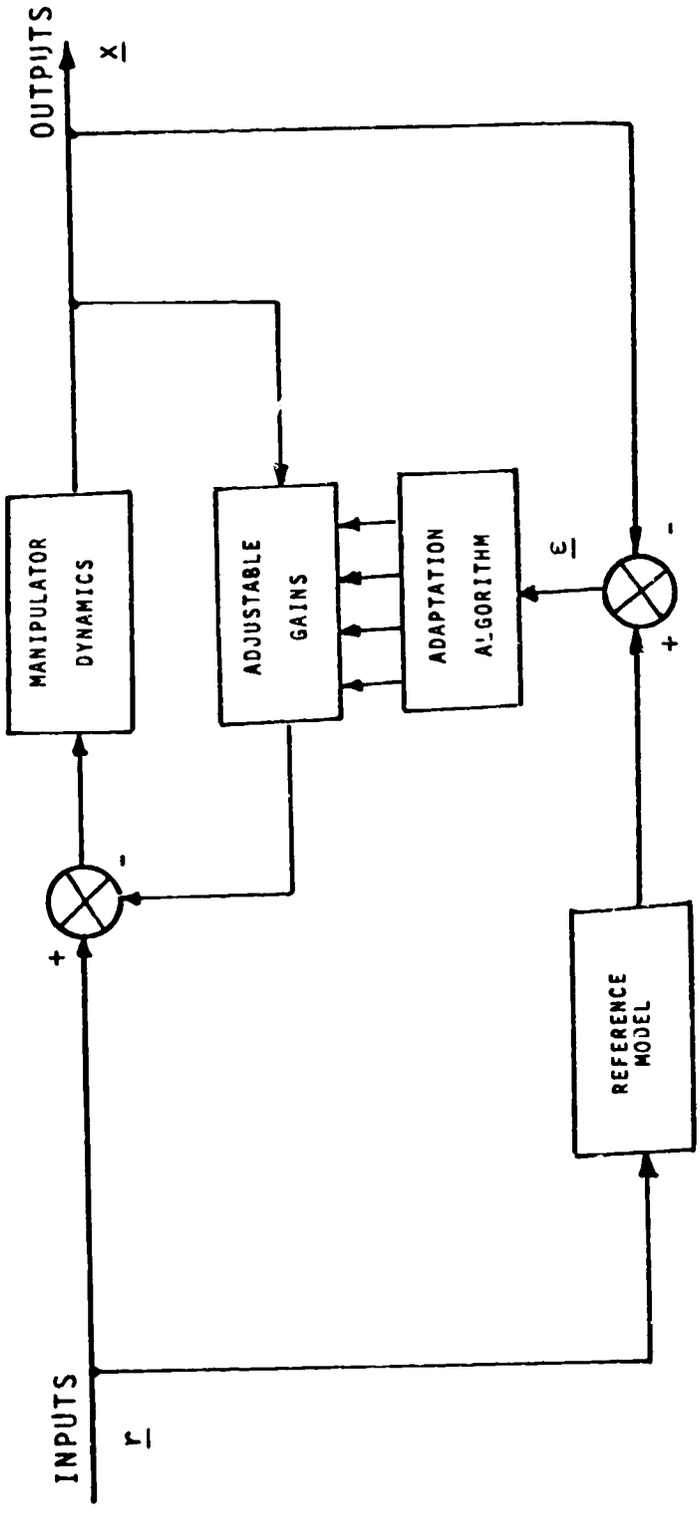


Figure 3: THE GENERAL MODEL REFERENCE ADAPTIVE CONTROL SYSTEM FOR A MANIPULATOR.

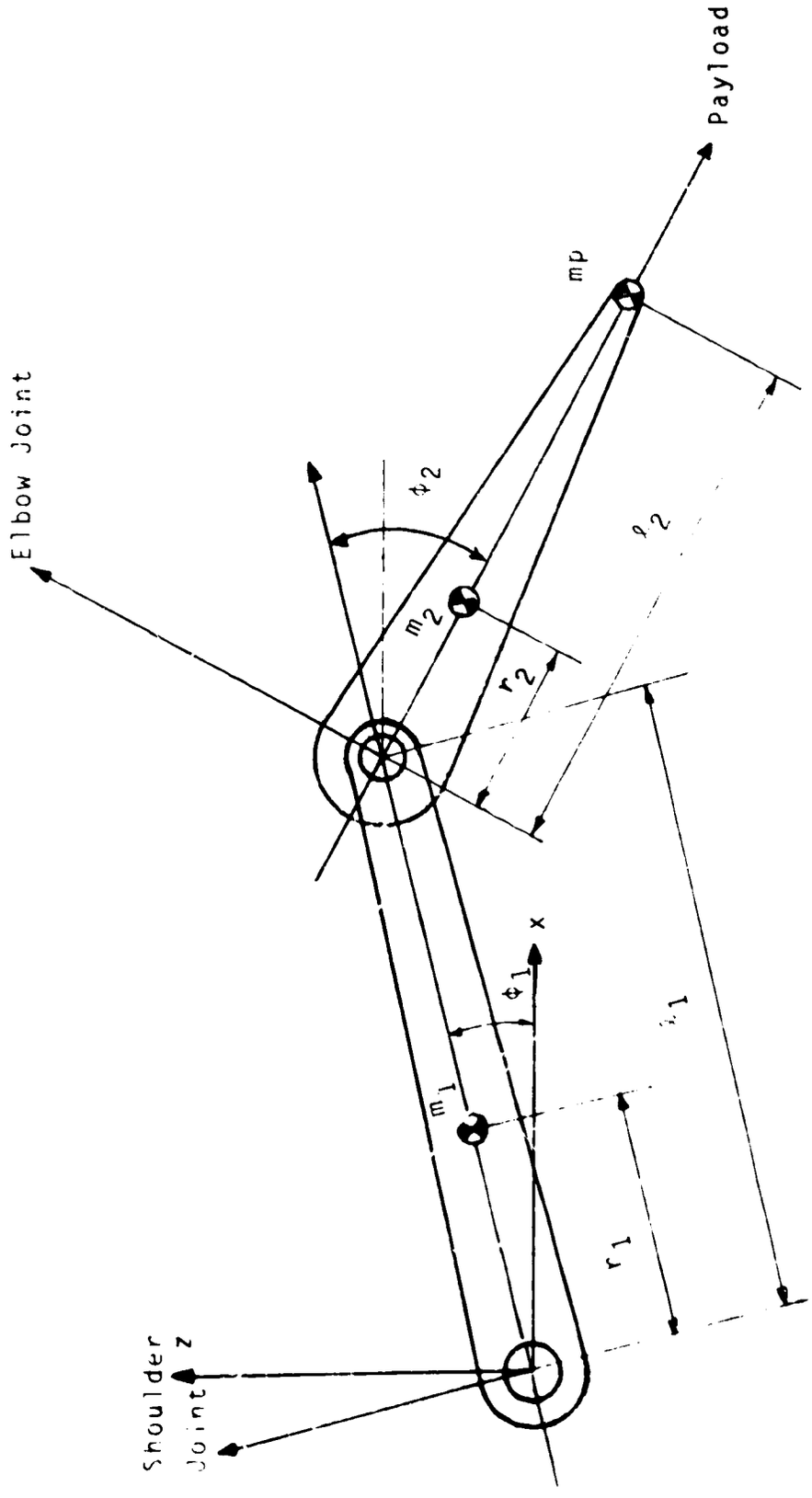


Figure 4: The Two-Link System

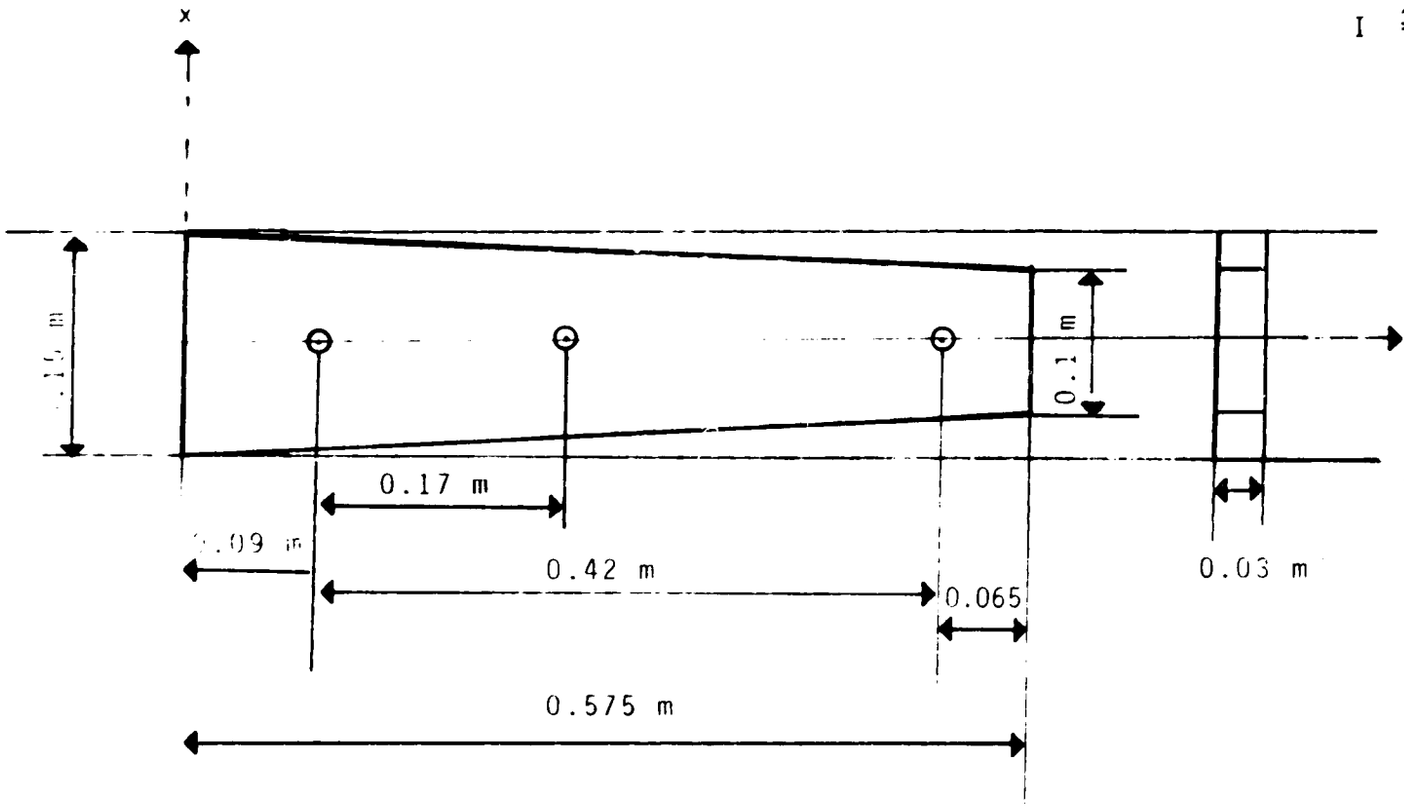
Area: $\frac{0.15 + 0.1}{2} * 0.575\ m = 0.071875\ m^2$

Volume: $0.071875 * 0.03\ m^3 = 2.15625.10^{-3}\ m^3$

Mass = $7.86 * 2.156$ kg.

m = 16.95 kg.

$I \cong 0.52\ m^2$ kg.



8.5 kg @ 0.17 m	=	0.245
8.5 kg @ 0.18 m	=	0.275
		$0.520\ m^2 kg$

Figure 5: Assumed Geometry of Lower Arm

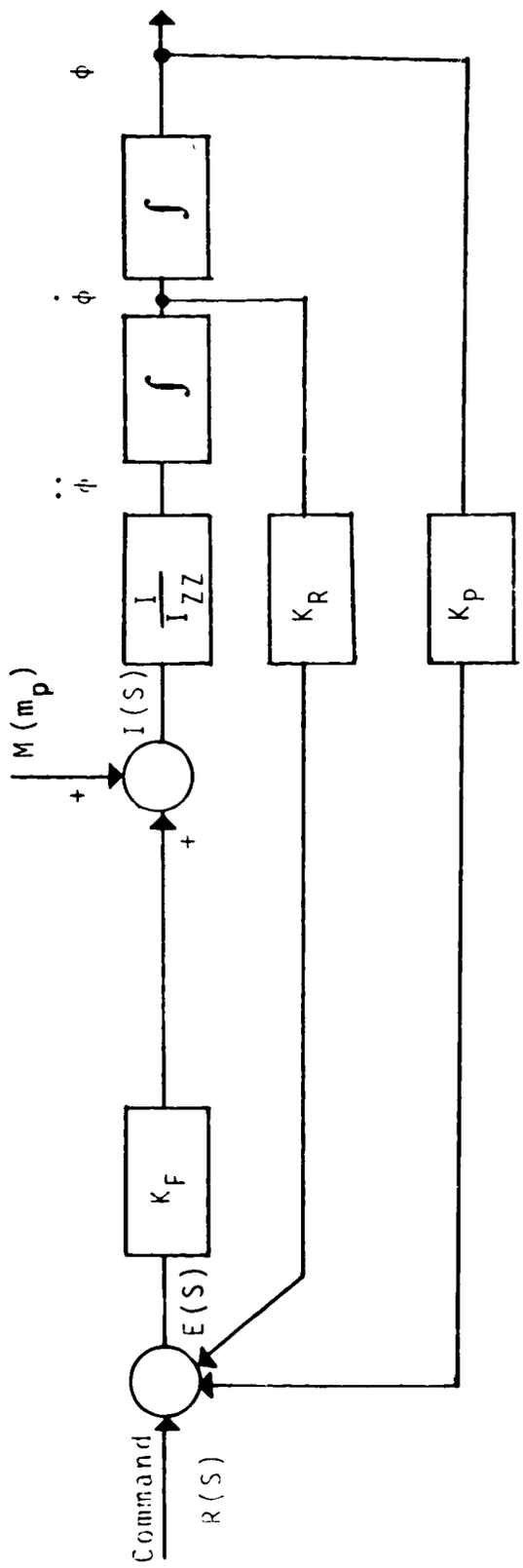


Figure 6: Block Diagram for Single Arm Controller

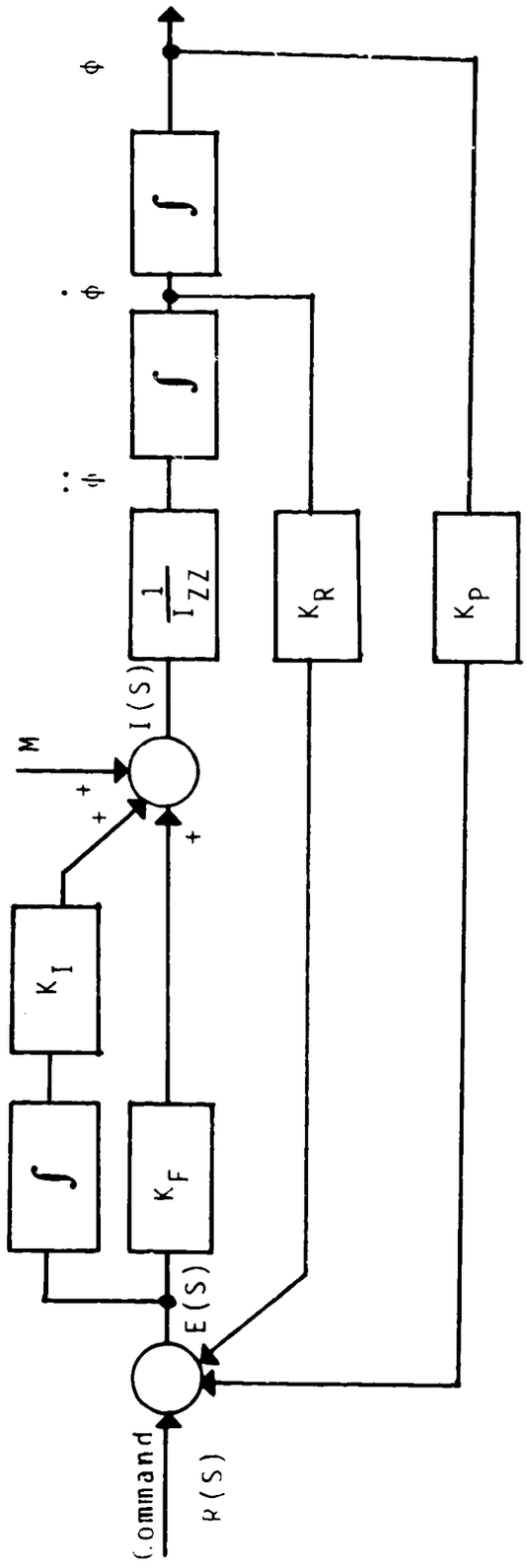


Figure 7: Block Diagram for Single Arm Controller with Error Integral Feedback Added

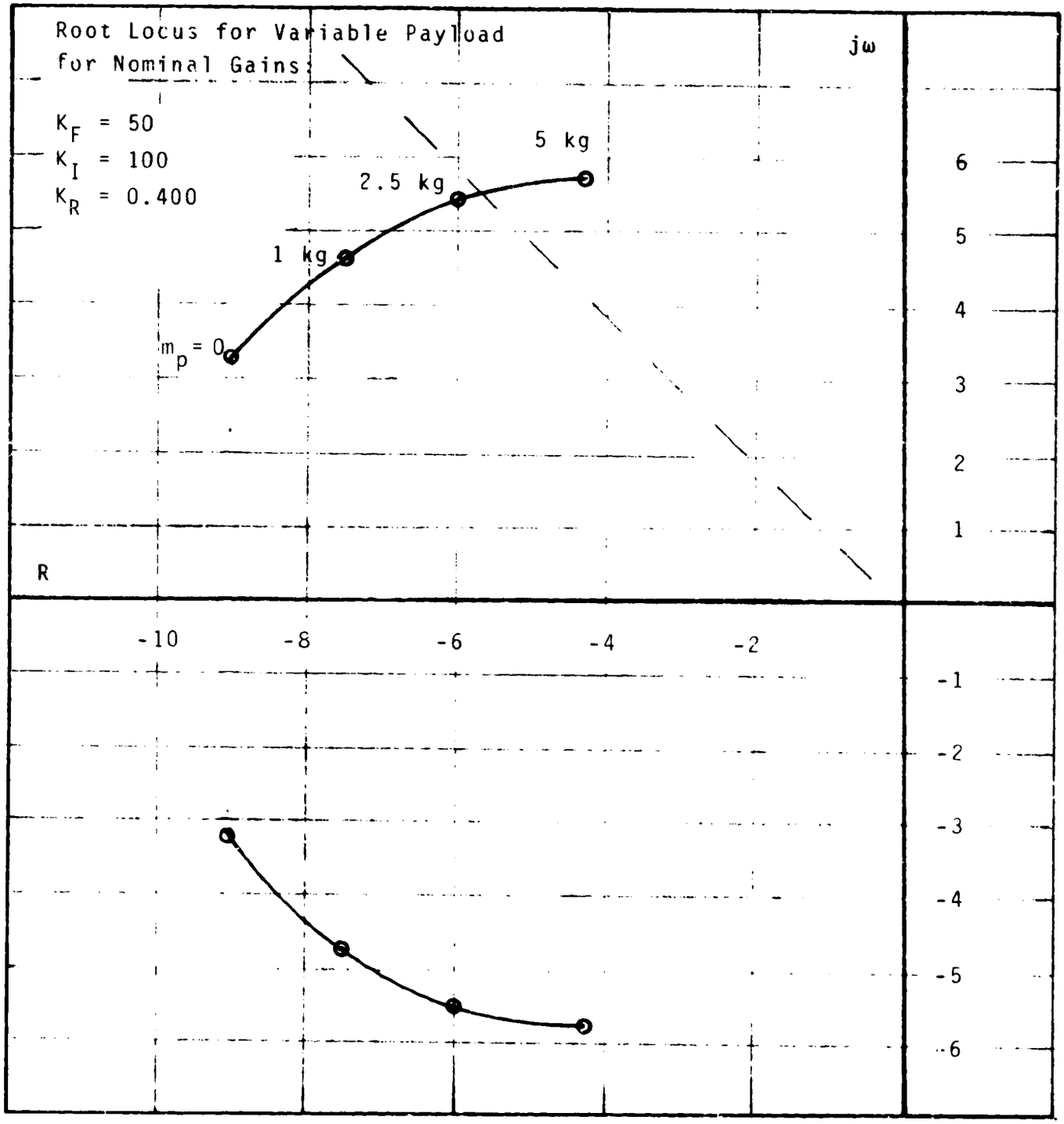


Figure 8: Root Locus for Single Arm Controller for Variable Payload

(K_P = 1; K_F = 50; [illegible]

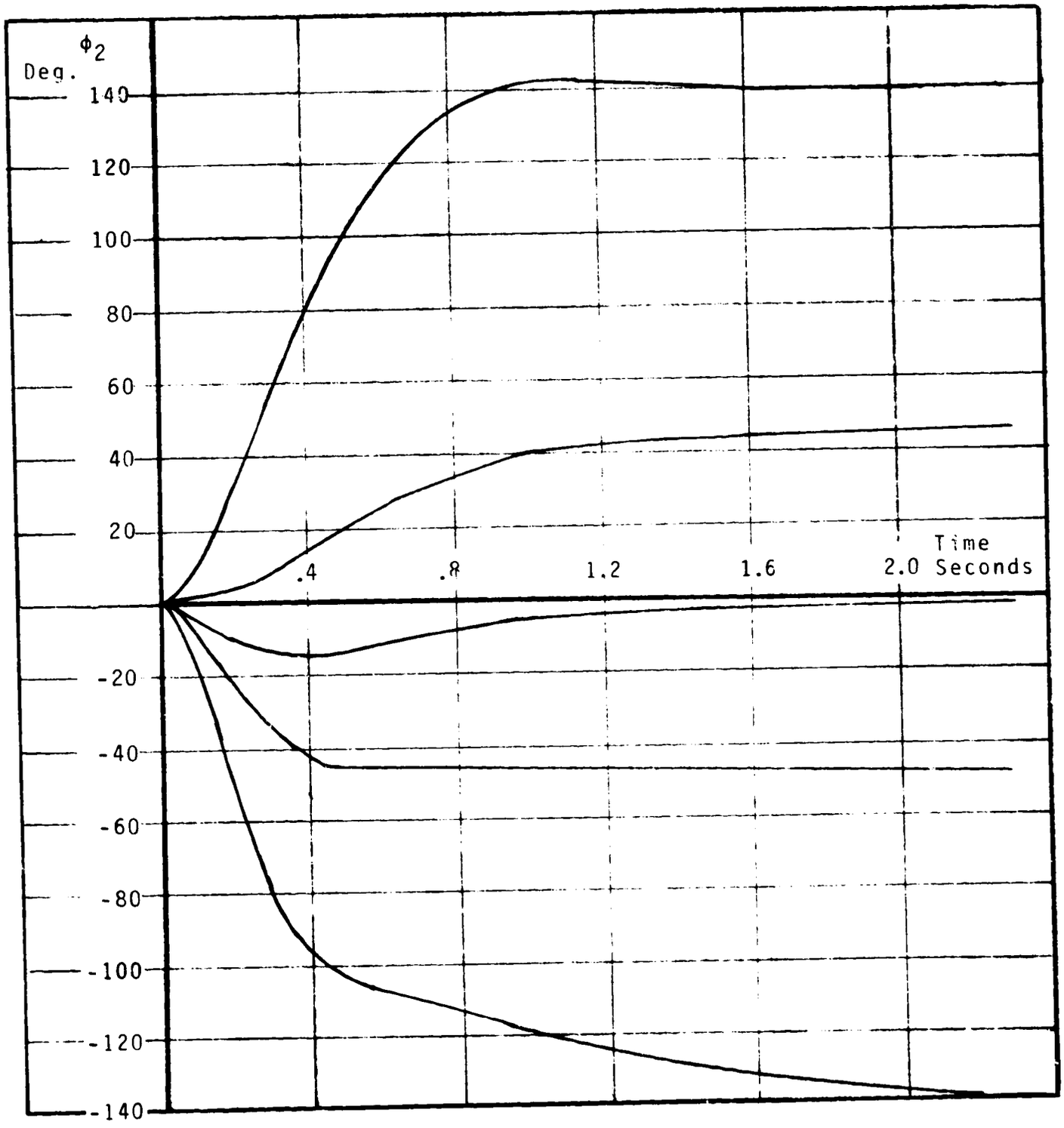


Figure 9: Responses of Lower Arm to Step Inputs of Varying Magnitudes and Signs

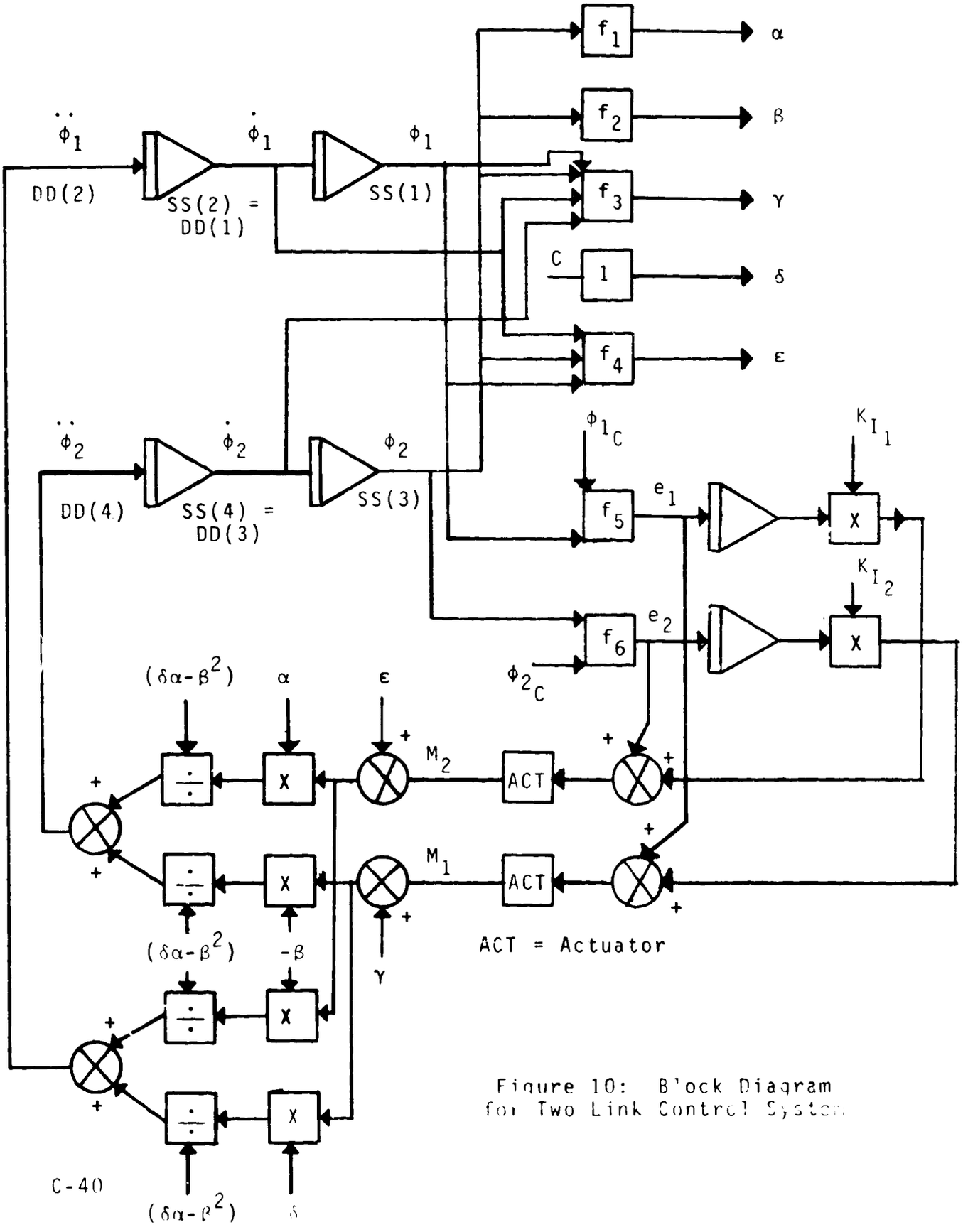


Figure 10: Block Diagram for Two Link Control System

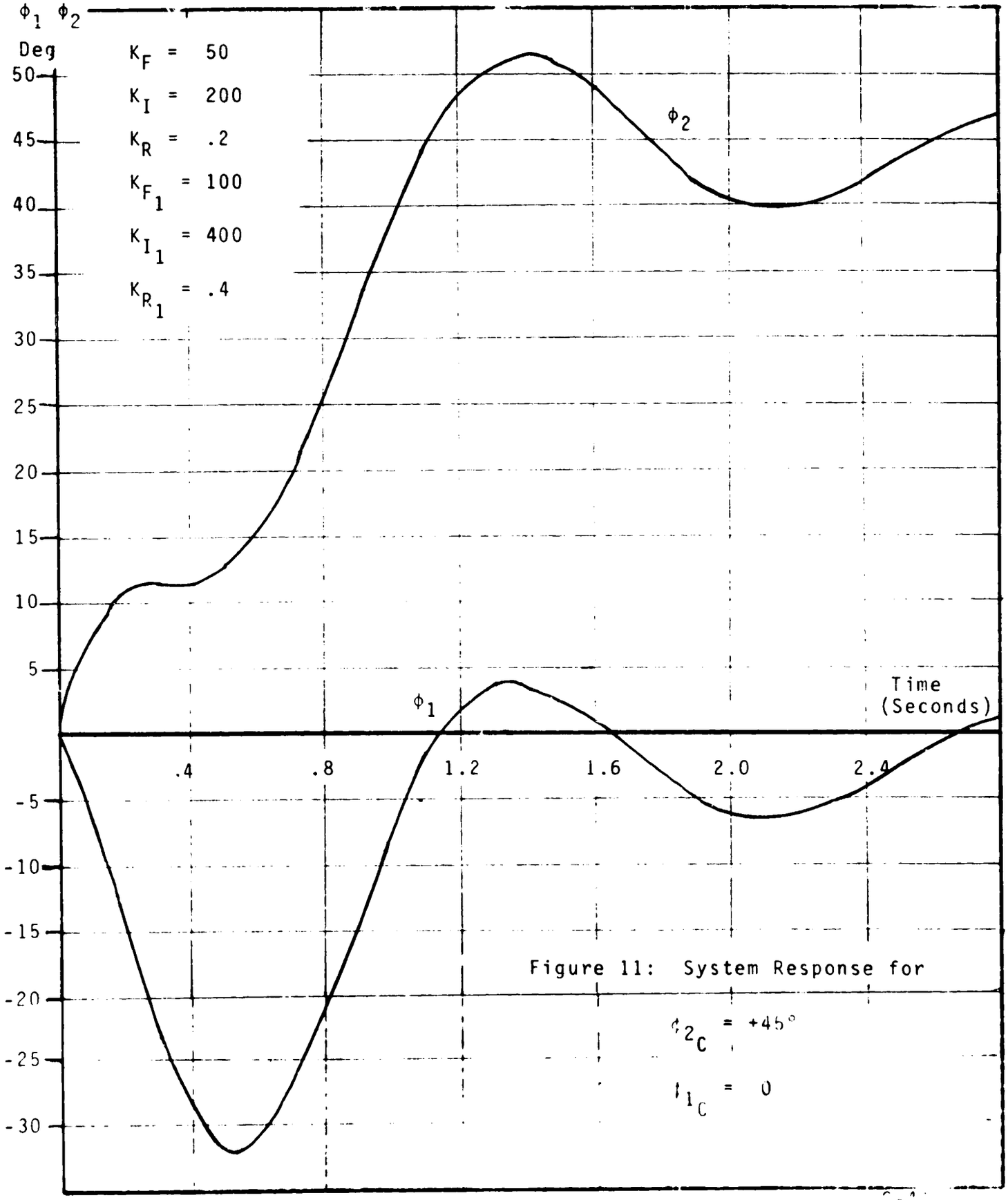


Figure 11: System Response for

ϕ_{2_C} = +45°

ϕ_{1_C} = 0

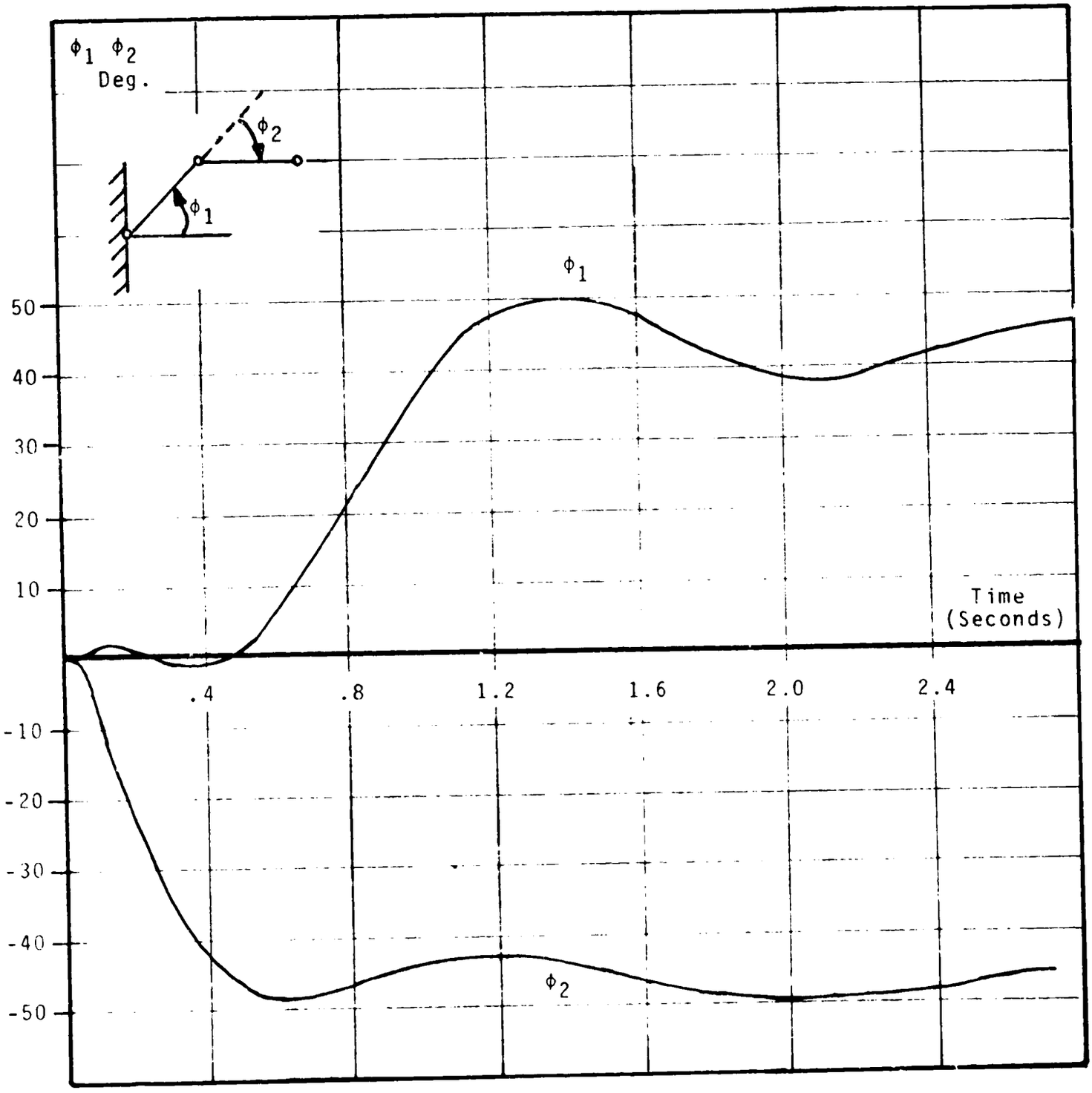


Figure 12: System Response for

ϕ_1 45°

ϕ_2 -45°

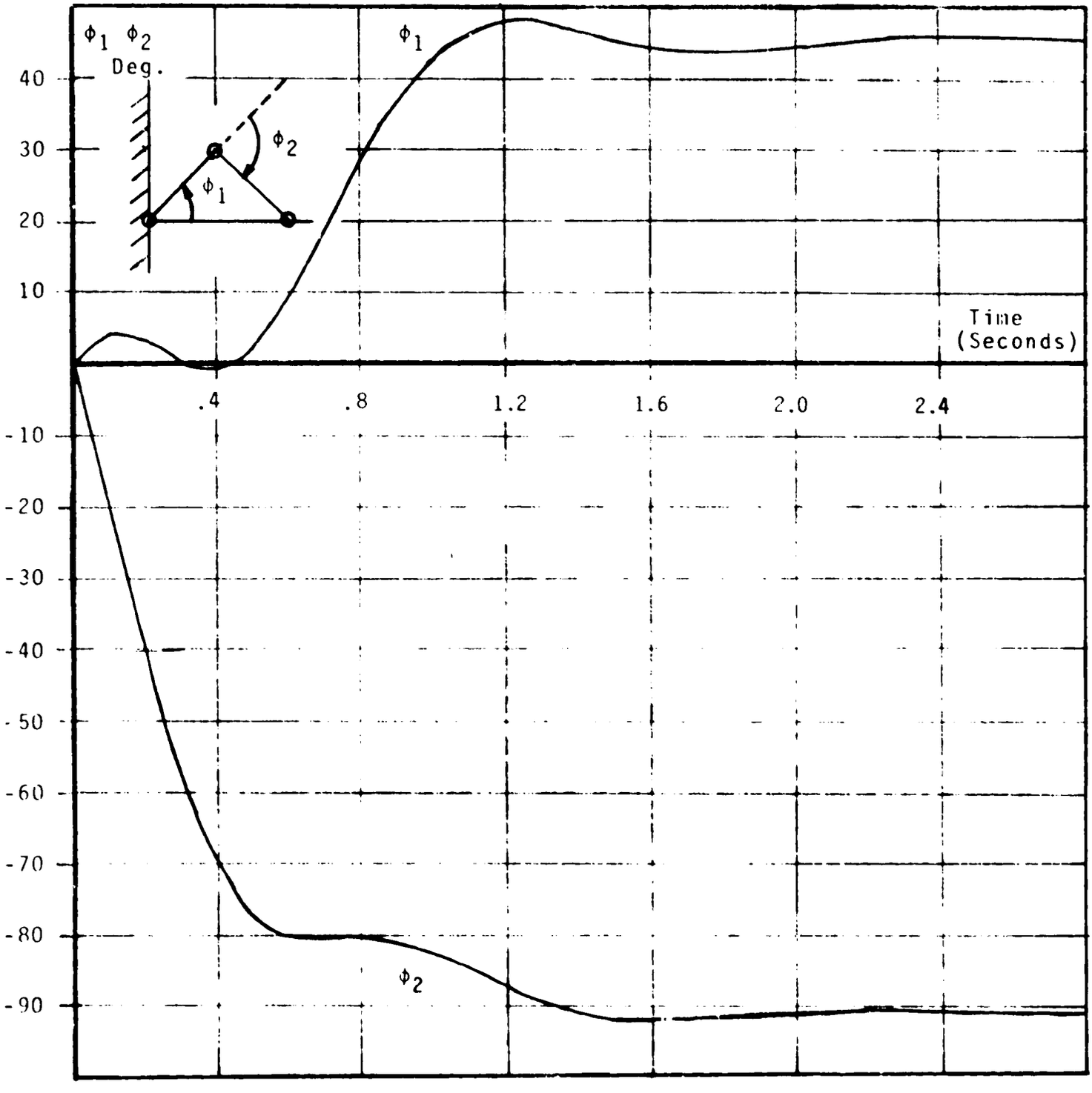


Figure 13: System Response for

$\phi_{1_C} = +45°$

$\phi_{2_C} = -90°$

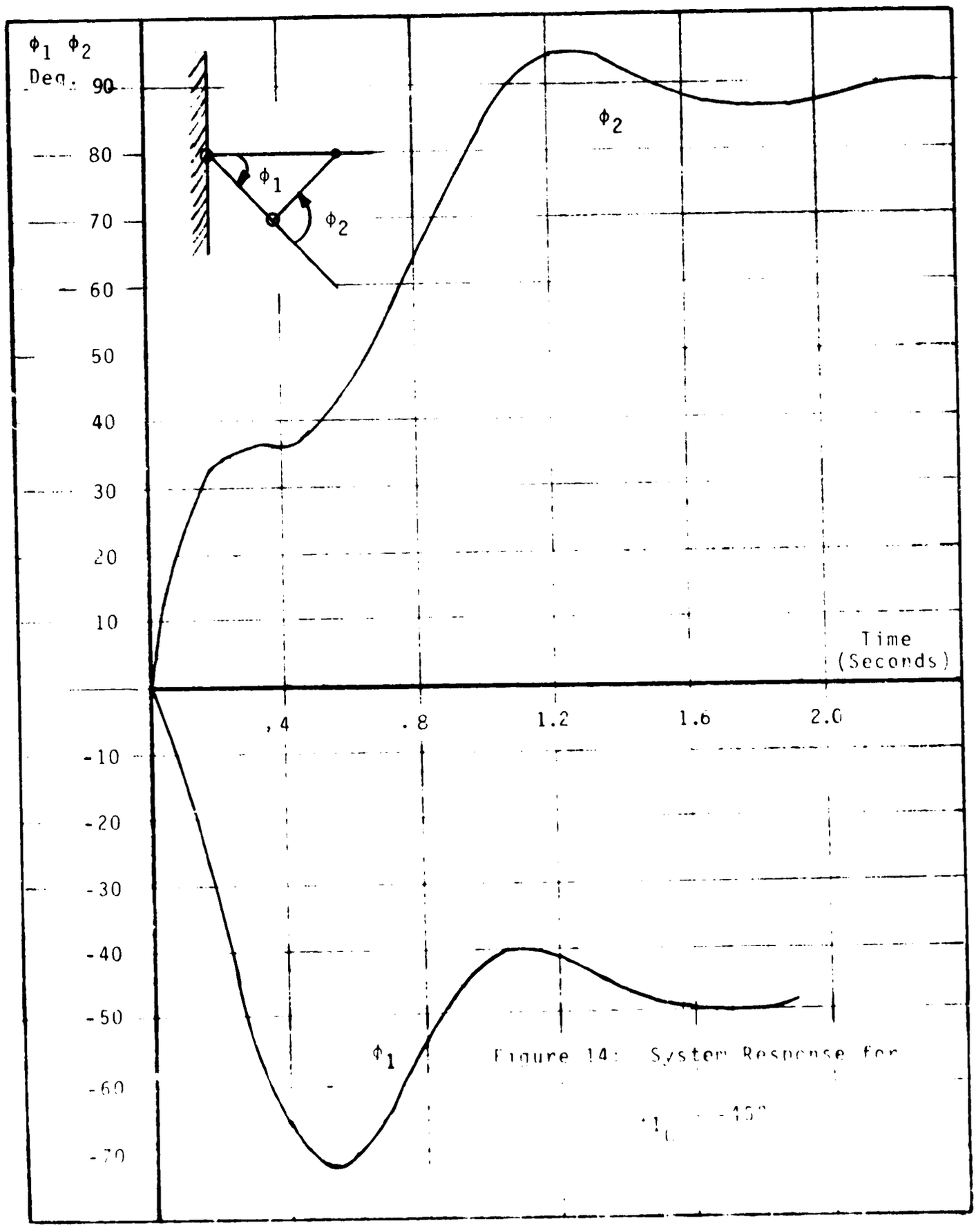


Figure 14: System Response for

ϕ_{1_0} = -45°

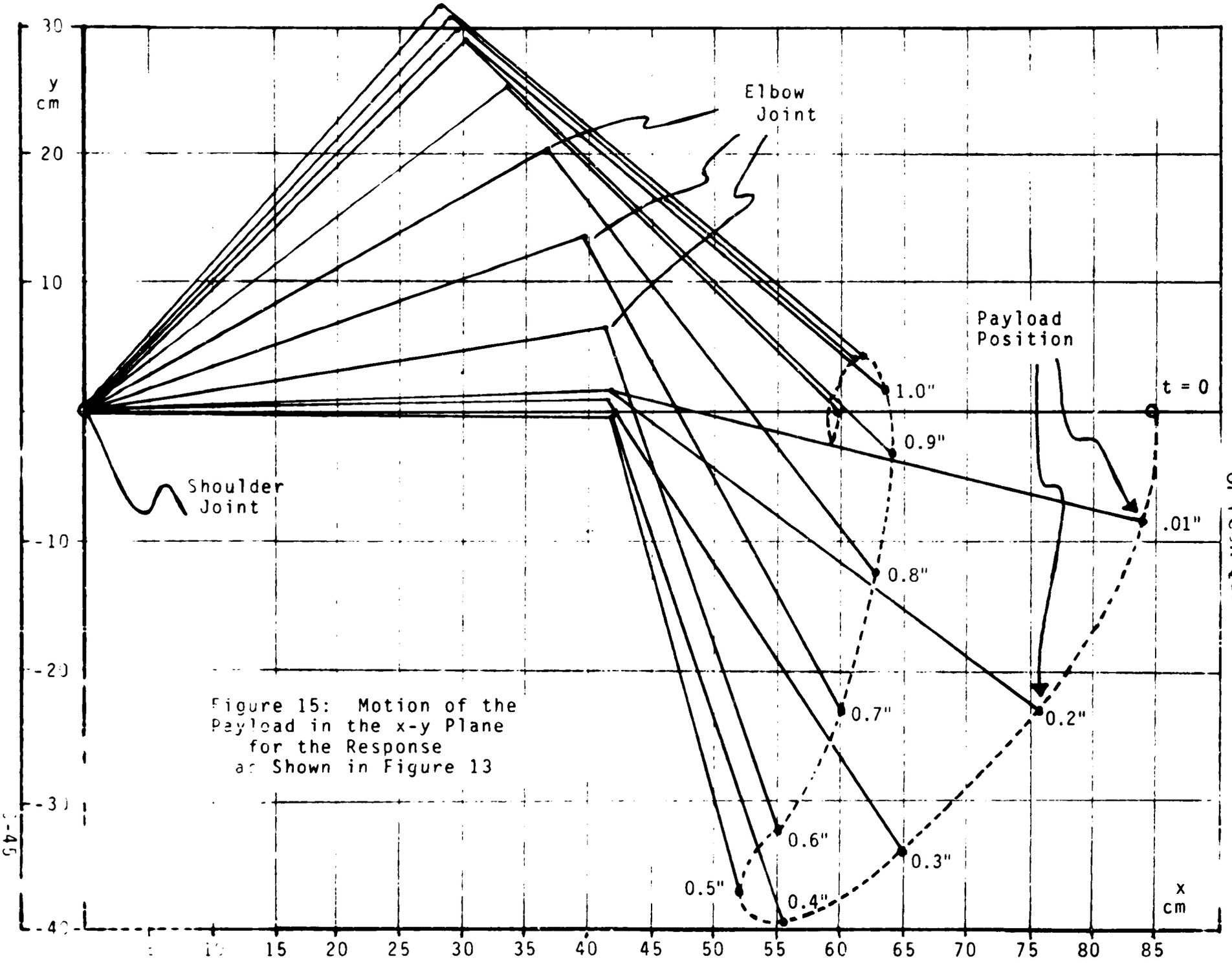


Figure 15: Motion of the Payload in the x-y Plane for the Response as Shown in Figure 13

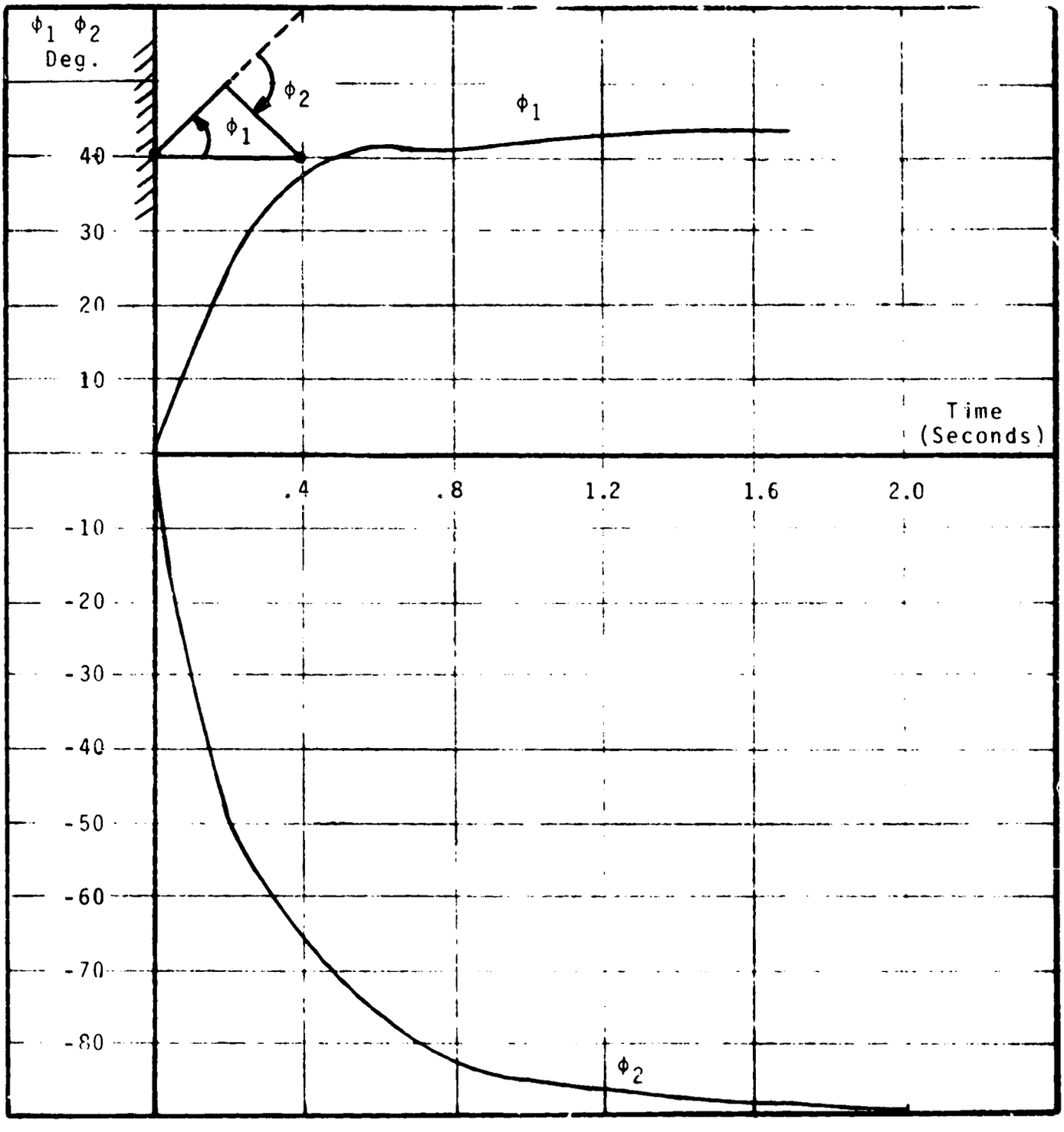


Figure 16: System Response with Finite-State Machine Gain Adjuster for

= +45

= -90

(Compare with Figure 1

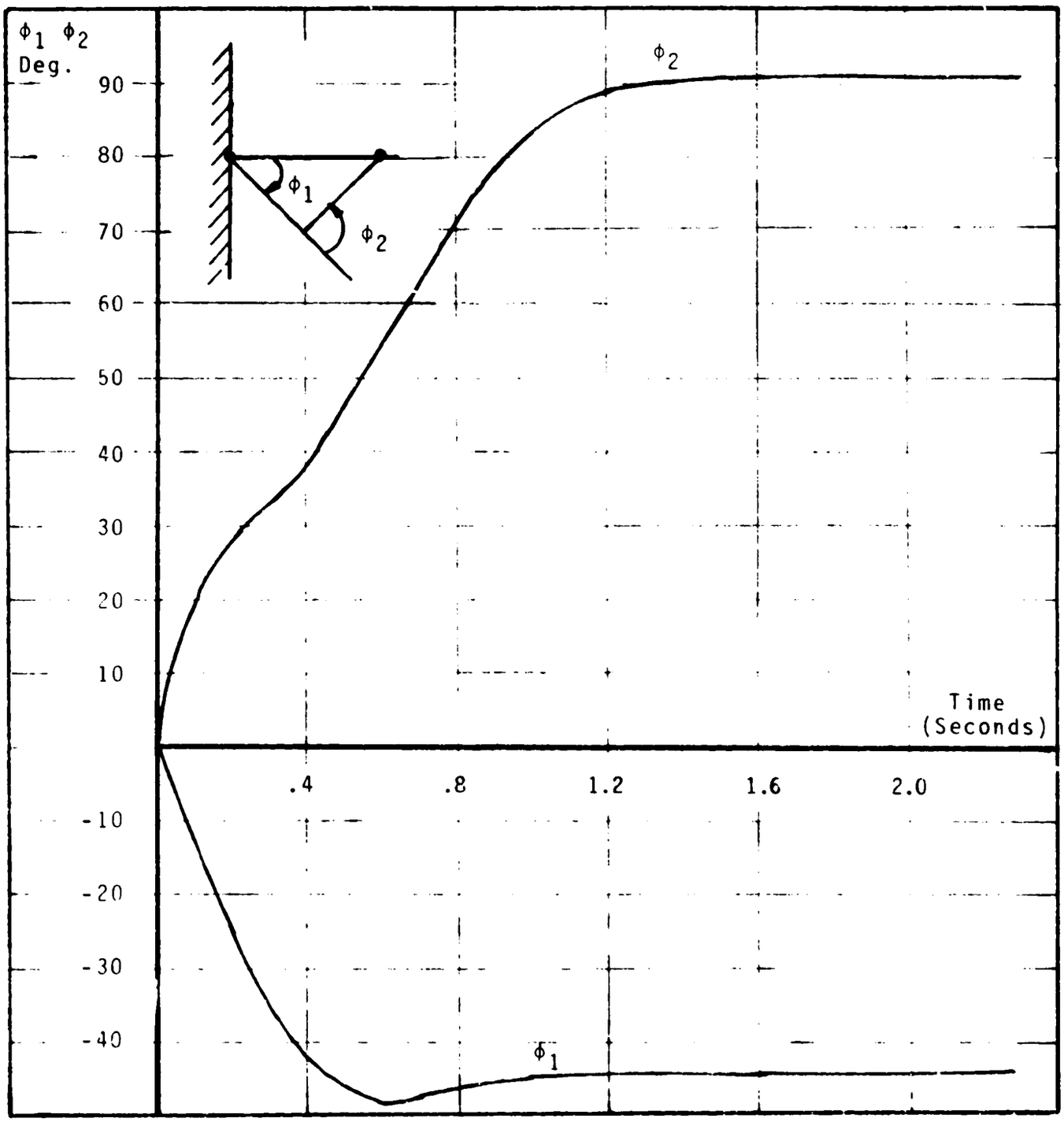


Figure 17: System Response with Finite-State Machine Gain Adjuster for

$\Phi_{1_C} = -45°$

$\Phi_{2_C} = +90°$

(Compare with Figure 14)

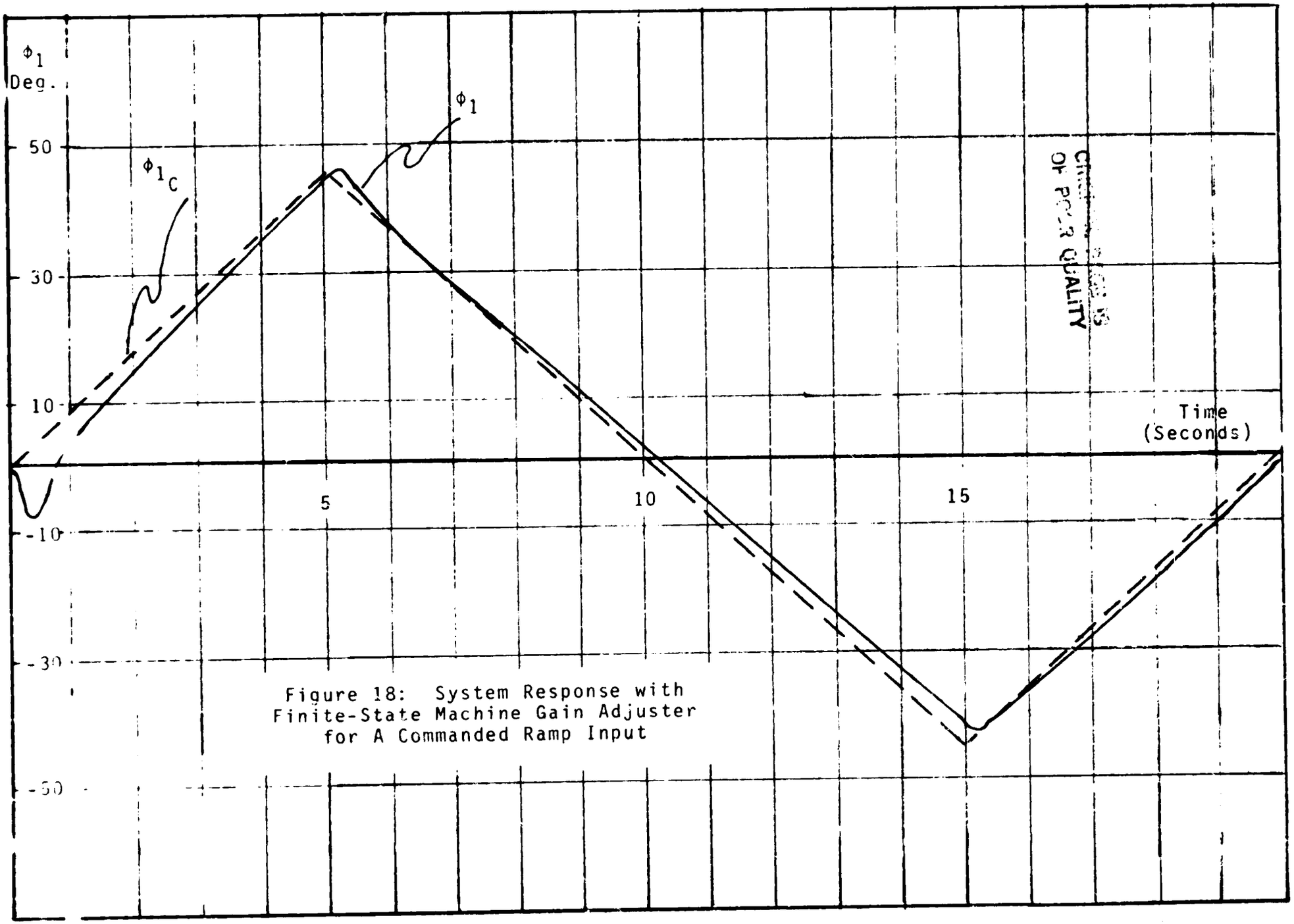


Figure 18: System Response with Finite-State Machine Gain Adjuster for A Commanded Ramp Input

```
GEN,BURGIN,ROBOT,2/12/82;
INITIALIZE,0,25;
CONTINUOUS,6.0,0.0005,0.1,0.1,W;
RECORD,TNOW,TIME,,T,0.1;
VAR,XX(6),PHI1;
VAR,XX(3),PHI2;
VAR,XX(21),MOM1;
VAR,XX(20),MOM2;
VAR,XX(31),STATE;
SIMULATE;
FIN;
END OF FILE
```

```
/get,robot5
```

```
C
      PROGRAM MAIN(TAPE5,TAPE6=80,INPUT,OUTPUT,TAPE17,TAPE18)
C
      DIMENSION NSET(2000)
      COMMON QSET(1500)
      EQUIVALENCE(QSET(1),NSET(1))
C
      COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
C
      COMMON/BURGIN1/AA,BB,CC,DDD,G,PI
C
      COMMON/BURGIN2/L1,L2,M1,M2,MP,I1,I2,R1,R2,MOM1,MOM2,CF1,CF2
      REAL L1,L2,M1,M2,MP,I1,I2,MOM1,MOM2
C
      COMMON/CONTROL/RS,KF,KP,KR,KI
      COMMON/CNTRL1/RS1,KF1,KP1,KR1,KI1
      REAL KF,KP,KR,KI
      REAL KF1,KP1,KR1,KI1
C
      NNSET=1500
      NCRDR=5
      NPRNT=6
      NTAPE=18
      NNSET=1500
C
      REWIND 5
      REWIND 6
      REWIND 17
      REWIND 18
      PI=4.*ATAN(1.)
C
```

Figure 19: Listing of Computer Simulation Source Program

```
    1 CONTINUE
      REWIND 5
      CALL SLAM
      STOP
C
      END
      SUBROUTINE STATE
C-----------------------------------------------------------
C
C
      DIMENSION NSET(1500)
      COMMON QSET(1500)
      EQUIVALENCE(QSET(1),NSET(1))
C
      COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
C
      COMMON/BURGIN1/AA,BB,CC,DDD,G,PI
C
      COMMON/BURGIN2/L1,L2,M1,M2,MP,I1,I2,R1,R2,MOM1,MOM2,CF1,CF2
      REAL L1,L2,M1,M2,MP,I1,I2,MOM1,MOM2
C
      COMMON/CONTROL/RS,KF,KP,KR,KI
      COMMON/CNTRL1/RS1,KF1,KP1,KR1,KI1
      REAL KF,KP,KR,KI
      REAL KF1,KP1,KR1,KI1
C
      EQUIVALENCE(SS(1),PHI1),(SS(2),PHI1DOT),(SS(3),PHI2),
     +           (SS(4),PHI2DOT)
C
      DATA SLOPE1/0.154857/
C
      IF(TNOW.LT.5.)GO TO 710
      IF(TNOW.LT.15.)GO TO 720
      IF(TNOW.LT.20.)GO TO 730
      GO TO 740
  710 RS1=TNOW*SLOPE1
      GO TO 750
  720 RS1=0.77429-(TNOW-5.)*SLOPE1
      GO TO 750
  730 RS1=-0.77428+(TNOW-15.)*SLOPE1
      GO TO 750
  740 RS1=0.
  750 CONTINUE
C
C
      SINPHI2=SIN(PHI2)
      COSPHI2=COS(PHI2)
      COSPHI1=COS(PHI1)
      SINPHI1=SIN(PHI1)
      COS1P2=COS(PHI1+PHI2)
      SIN1P2=SIN(PHI1+PHI2)
C                                                  Figure 19 (Cont'd)
C
      ALFA=AA+CC+2.*BB*COSPHI2
      BETA=CC+BB*COSPHI2
      DELTA=CC
      GAMMA=PHI2DOT*(2.*PHI1DOT+PHI2DOT)*BB*SINPHI2-DDD*G*COSPHI
     +      -BB*G*COS1P2/L1
      EPSILON=-BB*G*COS1P2/L1-PHI1DOT**2*BB*SINPHI2
```

```
C
      C A L L     G A I N S
C-----------------------------
C
      ERROR=RS-KP*PHI2-KR*PHI2DOT
      ERROR1=RS1-KP1*PHI1-KR1*PHI1DOT
      MOM1=KF1*ERROR1+KI1*SS(6)
      IF(MOM1.GT.244.)MOM1=244.
      IF(MOM1.LT.-244.)MOM1=-244.
      MOM2=KF*ERROR+KI*SS(5)
      IF(MOM2.GT.163.)MOM2=163.
      IF(MOM2.LT.-163.)MOM2=-163.
C
      DD(1)=SS(2)
      DENOM=DELTA*ALFA-BETA**2
      DD(2)=(DELTA*(GAMMA+MOM1)-BETA*(EPSILON+MOM2))/DENOM
      DD(3)=SS(4)
      DD(4)=(-BETA*(GAMMA+MOM1)+ALFA*(EPSILON+MOM2))/DENOM
      DD(5)=ERROR
      DD(6)=ERROR1
      XX(3)=SS(3)*XX(57)
      XX(4)=SS(4)*XX(57)
      XX(5)=DD(4)*XX(57)
      XX(6)=SS(1)*XX(57)
      XX(7)=L1*COSPHI1*100.
      XX(8)=L1*SINPHI1*100.
      XX(9)=XX(7)+L2*COS1P2*100.
      XX(10)=XX(8)+L2*SIN1P2*100.
      XX(20)=MOM2
      XX(21)=MOM1
      RETURN
      END
```

Figure 19 (Cont'd)

```
      SUBROUTINE INTLC
C-----------------------------------------
C
C
      DIMENSION NSET(1500)
      COMMON QSET(1500)
      EQUIVALENCE(QSET(1),NSET(1))
C
      COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
C
      COMMON/BURGIN1/AA,BB,CC,DDD,G,PI
C
      COMMON/BURGIN2/L1,L2,M1,M2,MP,I1,I2,R1,R2,MOM1,MOM2,CF1,CF2
      REAL L1,L2,M1,M2,MP,I1,I2,MOM1,MOM2
C
      COMMON/CONTROL/RS,KF,KP,KR,KI
      COMMON/CNTRL1/RS1,KF1,KP1,KR1,KI1
      REAL KF1,KR1,KI1,KP1
      REAL KF,KP,KR,KI
C
      EQUIVALENCE(SS(1),PHI1),(SS(2),PHI1DOT),(SS(3),PHI2),
     +           (SS(4),PHI2DOT)
C
      PI=4.*ATAN(1.)
      XX(57)=180./PI
C
C *** GEOMETRICAL AND PHYSICAL DATA
```

```
C     ---------------------------
C
      L1=0.42
      L2=0.42
      M1=16.95
      M2=16.95
      MP=2.5
      I1=0.5
      I2=0.5
      R1=0.17
      R2=0.17
      MOM1=0.
      MOM2=0.
      G=9.81
      CF1=0.
      KF=50.
      KR=0.4
      KP=1.
      KI=200.
      RS=-00./XX(57)
      KF1=200.
      KR1=0.2
      KP1=1.
      KI1=800.
C
      RS1=+45./XX(57)
      CF2=0.
C
C *** INITIAL CONDITIONS
C
C     ------------------
C
      SS(1)=0.
      SS(2)=0.
      SS(3)=0.
      SS(4)=0.
      SS(5)=0.
      SS(6)=0.
C
      AA=I1+M1*R1**2+M2*L1**2+MP*L1**2
      BB=(M2*R2+MP*L2)*L1
      CC=I2+M2*R2**2+MP*L2**2
      DDD=M1*R1+(M2+MP)*L1
C
      PRINT 91,MP,KF,KI,KP,KR
   91 FORMAT(///" MP KF KI KP KR =",4F10.2,F10.4,///)
      PRINT 92,KF1,KI1,KP1,KR1
   92 FORMAT(/,"      KF1 KI1 KP1 KR1 =",3F10.2,F10.4,//)
C
      RETURN
      END
C-------------------------------------------------------------
      SUBROUTINE GAINS
C-------------------------------------------------------------
C
      DIMENSION NSET(5000)
      COMMON QSET(5000)
      EQUIVALENCE(QSET(1),NSET(1))
C
      COMMON/SCOM1/ ATRIB(100),DD(100),DDL(100),DTNOW,II,MFA,MSTOP,NCLNR
```

Figure 19 (Cont'd)

```
     1,NCRDR,NPRNT,NNRUN,NNSET,NTAPE,SS(100),SSL(100),TNEXT,TNOW,XX(100)
C
      COMMON/BURGIN1/AA BB,CC,DDD,G,PI
C
      COMMON/BURGIN2/L1,L2,M1,M2,MP,I1,I2,R1,R2,MOM1,MOM2,CF1,CF2
      REAL L1,L2,M1,M2,MP,I1,I2,MOM1,MOM2
C
      COMMON/CONTROL/RS,KF,KP,KR,KI
      COMMON/CNTRL1/RS1,KF1,KP1,KR1,KI1
      REAL KF1,KR1,KI1
      REAL KF,KP,KR,KI
C
      EQUIVALENCE(SS(1),PHI1),(SS(2),PHI1DOT),(SS(3),PHI2),
     +           (SS(4),PHI2DOT)
C
      DATA PHI1DMX/0.7853/,ILAST/0/
      ERR1=RS1-PHI1
      IN3=1
      IF(ERR1.LE.0.)IN3=0
      IN2=1
      IF(ABS(ERR1).LT.PI/4.)IN2=0
      IN1=1
      IF(ABS(PHI1DOT).LT.0.7853)IN1=0
C
      INSYMB=8-(4*IN3+2*IN2+IN1)
      GO TO(1100,1200,1300,1400,1500,1600,1700,1800)INSYMB
C
 1100 INEW=1
      IF(INEW.NE.ILAST)CALL PRTI(TNOW,INEW,ILAST)
      KF1=600.
      KR1=0.3
      GO TO 999
 1200 INEW=2
      IF(INEW.NE.ILAST)CALL PRTI(TNOW,INEW,ILAST)
      KF1=600.
      KR1=0.1
      GO TO 999
 1300 INEW=3
      IF(INEW.NE.ILAST)CALL PRTI(TNOW,INEW,ILAST)
      KF1=800.
      KR1=0.15
      GO TO 999
 1400 INEW=4
      IF(INEW.NE.ILAST)CALL PRTI(TNOW,INEW,ILAST)
      KF1=800.
      KR1=0.1
      GO TO 999
 1500 INEW=5
      IF(INEW.NE.ILAST)CALL PRTI(TNOW,INEW,ILAST)
      KF1=300.
      KR1=0.4
      GO TO 999
 1600 INEW=6
      IF(INEW.NE.ILAST)CALL PRTI(TNOW,INEW,ILAST)
      KF1=300.
      KR1=0.2
      GO TO 999
 1700 INEW=7
      IF(INEW.NE.ILAST)CALL PRTI(TNOW,INEW,ILAST)
      KF1=500.
```


Figure 19 (Cont'd)

```
      KR1=0.3
      GO TO 999
 1800 INEW=8
      IF(INEW.NE.ILAST)CALL PRTI(TNOW,INEW,ILAST)
      KF1=500.
      KR1=0.2
  999 CONTINUE
      XX(31)=INEW
      END
C-----------------------------------------------------------------
      SUBROUTINE PRTI(TNOW,INEW,ILAST)
C-----------------------------------------------------------------
C     PRINT 91,TNOW,ILAST,INEW
   91 FORMAT(" STATE CHANGED AT TIME "F12.3" FROM"I4" TO"I4)
      ILAST=INEW
      RETURN
      END
C-----------------------------------------------------------------
END OF FILE
```

Figure 19 (Cont'd)