NASA CR-102620

[redacted]

Lockheed

MISSILES & SPACE COMPANY

A GROUP DIVISION OF LOCKHEED AIRCRAFT CORPORATION

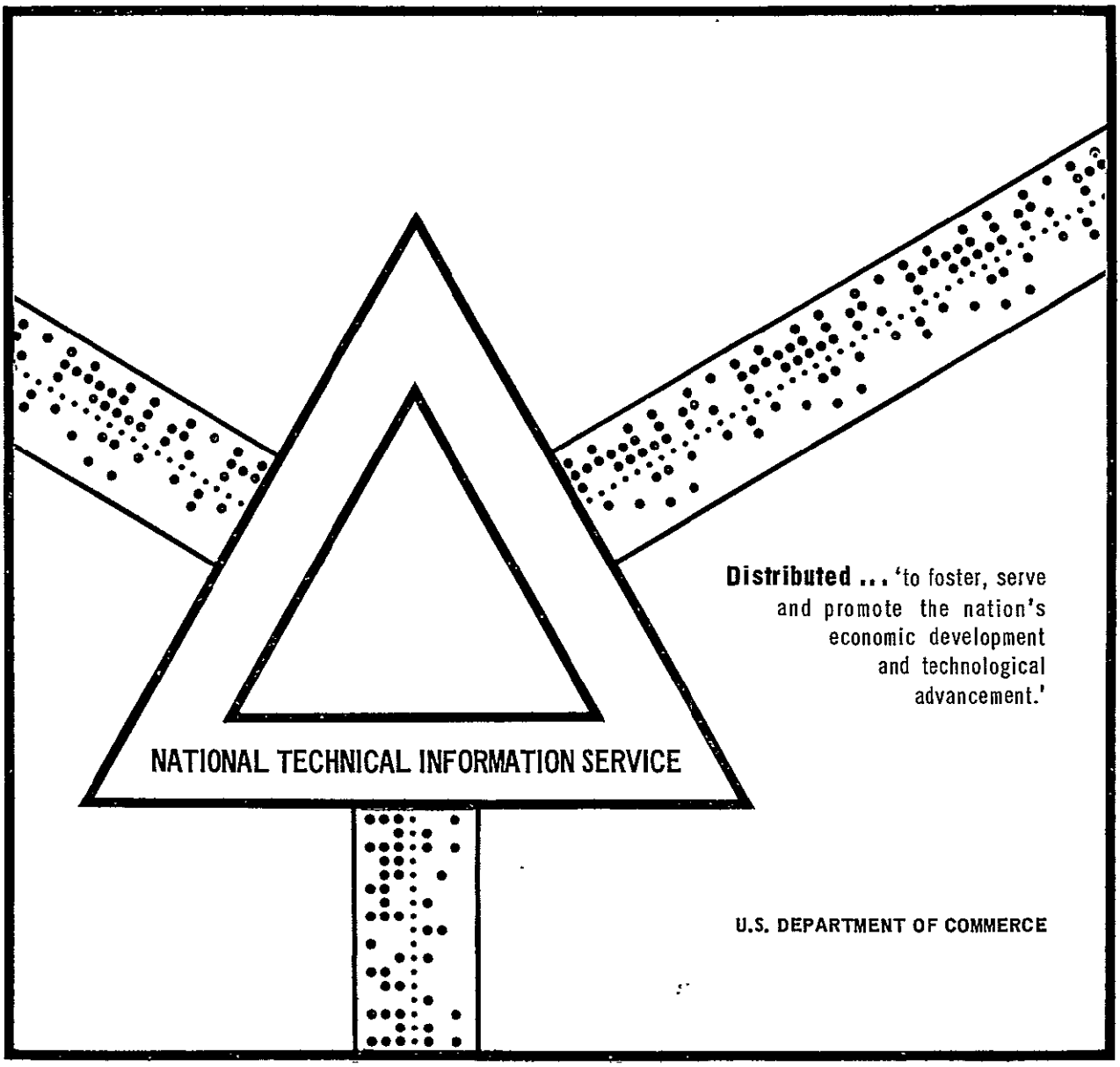
SUNNYVALE, CALIFORNIA

USERS MANUAL, CONTROL SYSTEM OPTIMIZATION HYBRID COMPUTER PROGRAM

C. L. Conner

Lockheed Missiles and Space Company
Sunnyvale, California

February 1970



HREC-0515-3
LMSC/HREC D162122-III

LOCKHEED MISSILES & SPACE COMPANY
HUNTSVILLE RESEARCH & ENGINEERING CENTER
HUNTSVILLE RESEARCH PARK
4800 BRADFORD BLVD., HUNTSVILLE, ALABAMA

USER'S MANUAL
CONTROL SYSTEM OPTIMIZATION HYBRID COMPUTER PROGRAM
FINAL REPORT

February 1970

Contract NAS8-30515

by
C. L. Connor

APPROVED BY: W. Trautwein

W. Trautwein, Supervisor
Flight Dynamics & Control Section

T. R. Beal

T. R. Beal, Manager
Dynamics & Guidance Department

J. S. Farrior

J. S. Farrior
Resident Manager

FOREWORD

This document comprises the control system optimization user's manual prepared by Lockheed Missiles & Space Company (Lockheed/ Huntsville) for the Astrionics Laboratory, Marshall Space Flight Center (MSFC) Huntsville, Alabama, under Contract NAS8-30515, Modification II, Task 1. Together with LMSC/HREC D162122-I and LMSC/HREC D162122-II, these constitute the final reports under this contract.

This report describes a computer program to be used in the design of optimal control systems. The work of refining this optimization program and preparing this manual was performed in the period July 1969 to January 1970. The results of an application study to the Saturn V load relief problem are published in the separate report, LMSC/HREC D162122-III.

Mr. H. H. Hosenthien and Mr. S. N. Carroll were the MSFC technical monitors. Dr. W. Trautwein was the project engineer at Lockheed. Mr. C. L. Connor was the major contributor who conducted the program development. Mr. R. S. Nyhan and Mr. S. Lo of Lockheed/Huntsville performed the hybrid programming and computations.

CONTENTS

Section 1
INTRODUCTION AND SUMMARY

This report describes a computerized tool to be used in the design of optimal control systems. The design method is based on modern control theory, high-speed hybrid simulation and the use of powerful gradient techniques.

The specific purpose of the method is to find for a given control law, time-varying controller adjustments which minimize a selected performance measure J. This performance measure, J, can be freely chosen to best reflect the design goals. Even minimax design criteria can be specified where certain peak values P_{max} of critical state variables must be minimized as indicated in Fig. 1. If P_{max} represents the major term in the performance criterion J, then P_{max} will be minimized in the course of the optimization.

The usual sensitivity of optimal solutions to changes in operating conditions can be markedly reduced by considering two adverse operating conditions, i.e., a Condition A and a Condition B (Fig. 1). The performance index J then is a function of both conditions which are repeatedly simulated on the analog computer in the course of the optimization with iterative adjustments of the critical controller parameters being made by the digital computer.

This concept of considering multiple operating conditions, accounts for the fact that in real life, a control system must be designed to operate successfully under a number of conditions and the system has no way of "knowing" before the fact which of these conditions will actually be encountered. The method is a truly optimal one, which anticipates the occurrence of one of two possible conditions, tests performance against a criterion which is a function of both conditions, and adjusts the control parameters to optimize this performance criterion. Thus the method "optimizes the compromise" of the anticipated conditions.

The program is written for an EAI 8900 Hybrid computer.

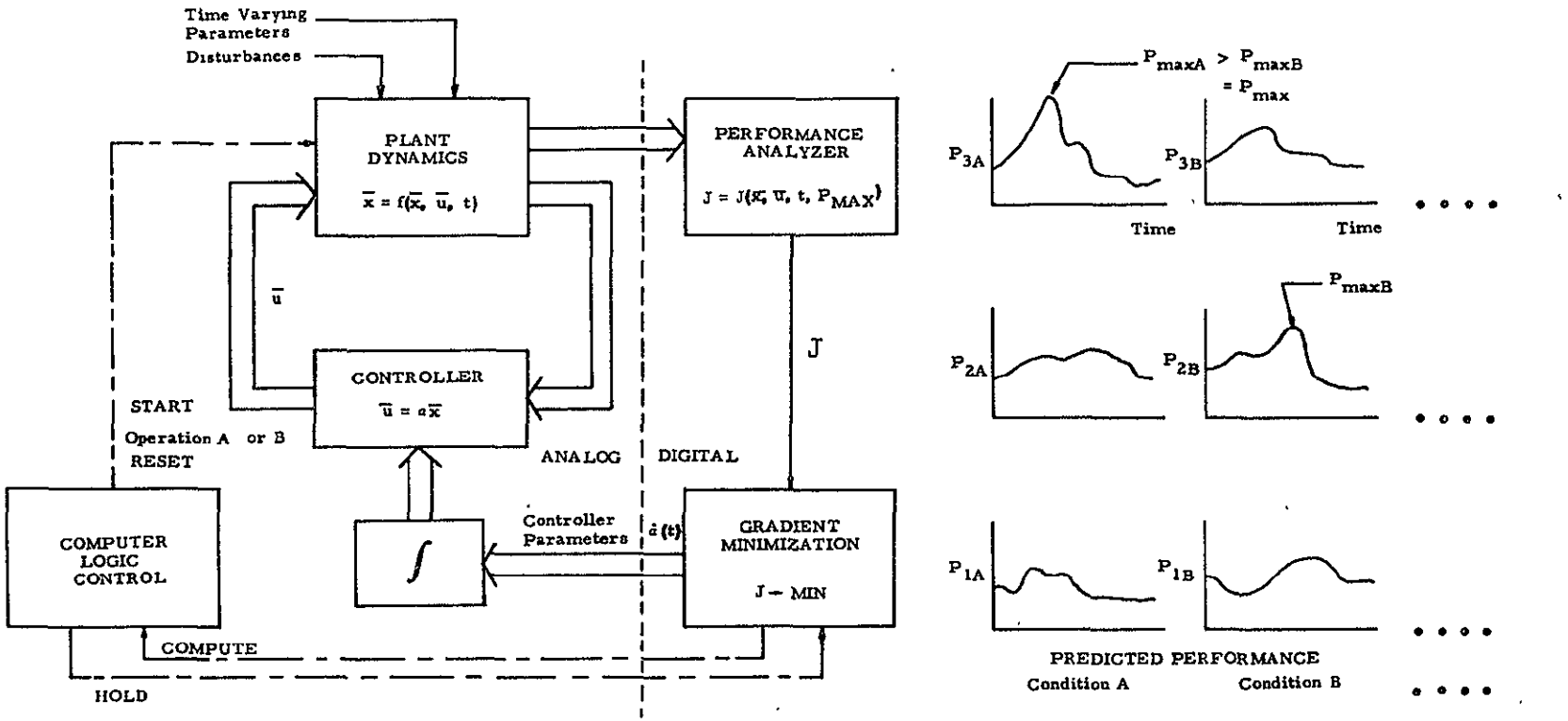


Fig. 1 - Basic Control System Optimization Scheme. Complete plant and control system dynamics are simulated repetitively on analog console of hybrid computer. Performance of one or two operating conditions is analyzed after simulations in digital computer and reduced iteratively. Free-form performance index J allows minimization of critical peak values P_{max} among various conditions (A, B, . . .)

Section 2
OPERATIONAL FEATURES

2.1 FLOATING INTERVAL

In the optimization scheme presented, a floating interval is used to solve for near-optimal polygonal gain schedules with a finite number of break points rather than for the general time functions of the optimal schedules. Using the attitude gain schedule of Fig. 2 as an example (which was obtained by applying linear optimal control theory to a booster control problem), it appears that little optimality is lost if this function is approximated by a ten-segment polygonal curve as shown in Fig. 2. Now the problem of finding optimal time functions for all n gain schedules in each of the optimization intervals is reduced to the determination of a set of n slopes, $\dot{a}_i$ of the gain schedules which are considered constant within each optimization interval of length T. The slopes of each gain curve, as indicated in Fig. 2, can now be iteratively adjusted, and the resulting performance evaluated by a sequence of forward integrations.

Total mission time is divided into a finite number of intervals, as shown in Fig. 3. The interval T represents the optimization interval which may be equal to or greater than the update interval.

2.2 SYSTEMATIC GRID SEARCH

At each update time, t_ν, during the mission, a systematic grid search of parameter space is used to avoid any local minima which might exist. Figure 4 shows an example of a J function where only one parameter ($\dot{a}_1$) would be optimized.

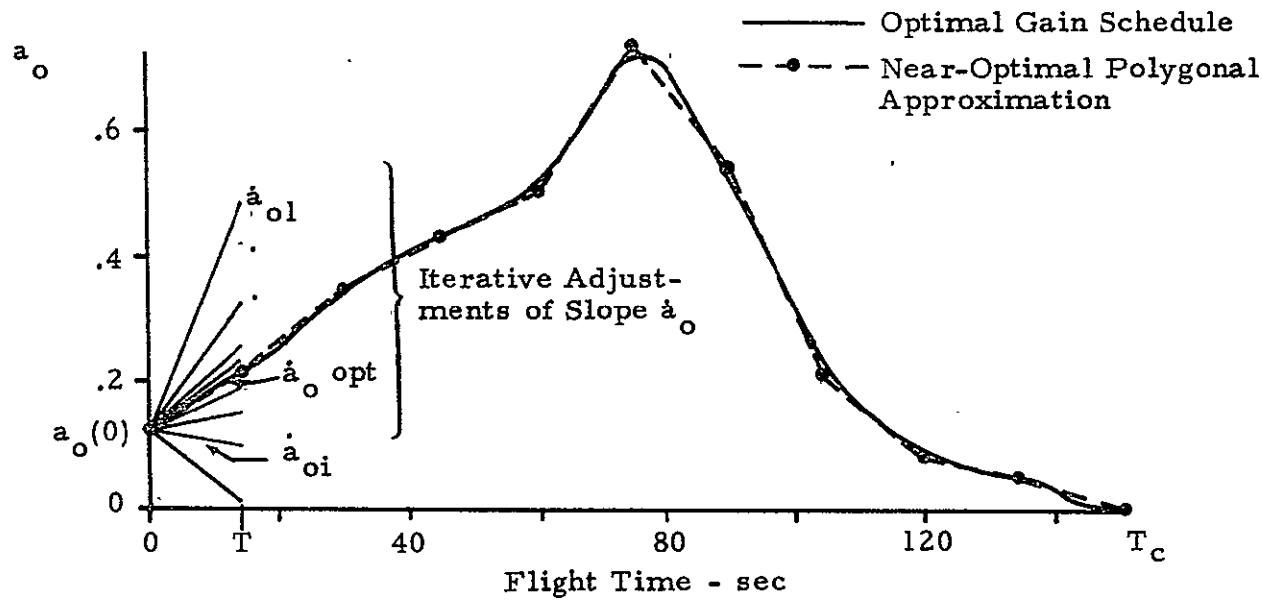


Fig. 2 - Typical Optimal Gain Schedule Approximated by Near-Optimal Polygonal Schedule

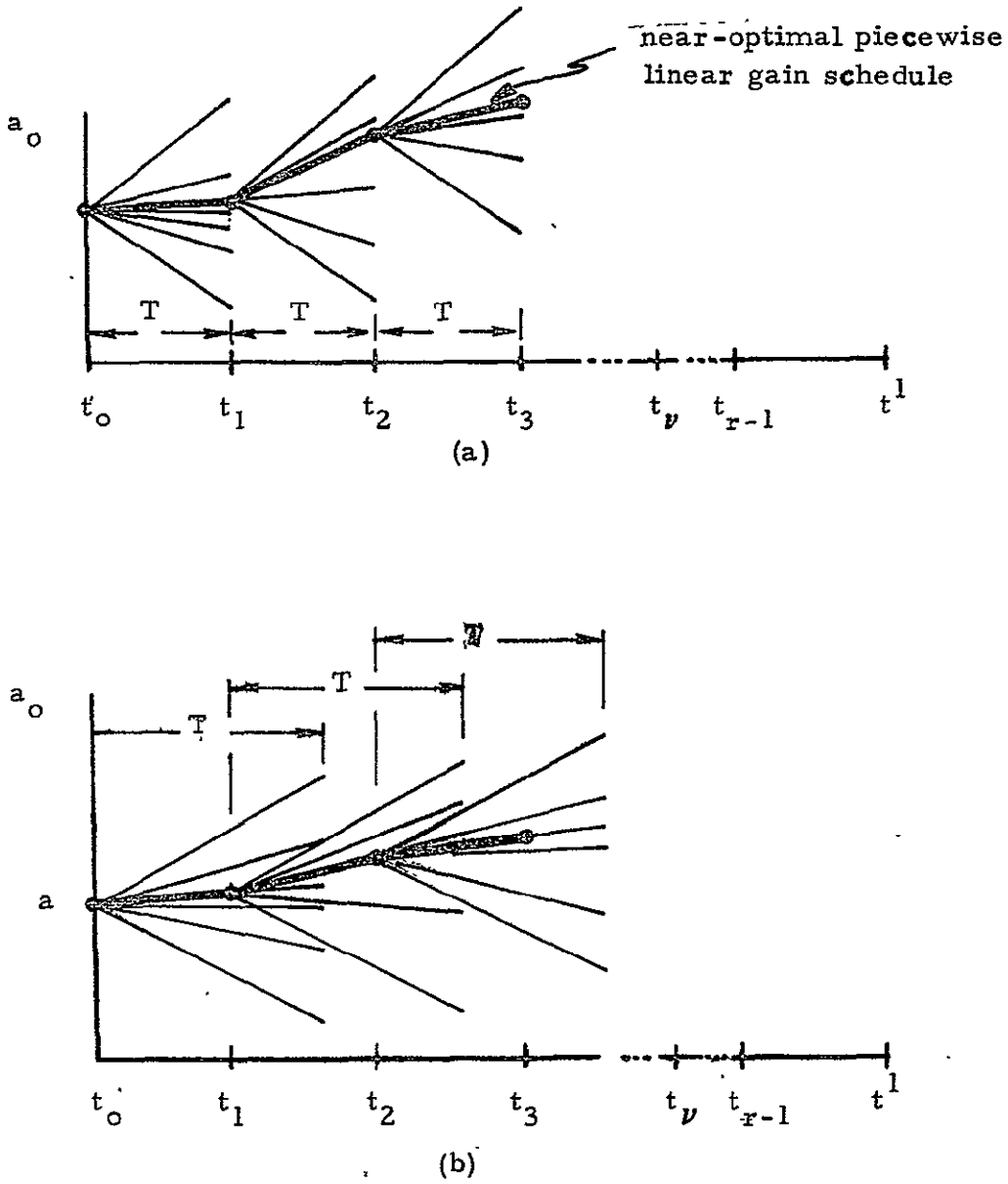


Figure 3 - Total Mission Time (t_o, t^1) Broken Down into a Finite Number r of Updating Intervals (t_o, t_1), (t_1, t_2), . . . (t_ν, $t_{\nu+1}$) (t_{r-1}, t^1). Optimization Intervals (t_ν, t_ν+T) are Identical to Updating Intervals (t_ν, $t_{\nu+1}$) as in Figure (3-a) or are longer than Updating Intervals (Figure (3 - b))

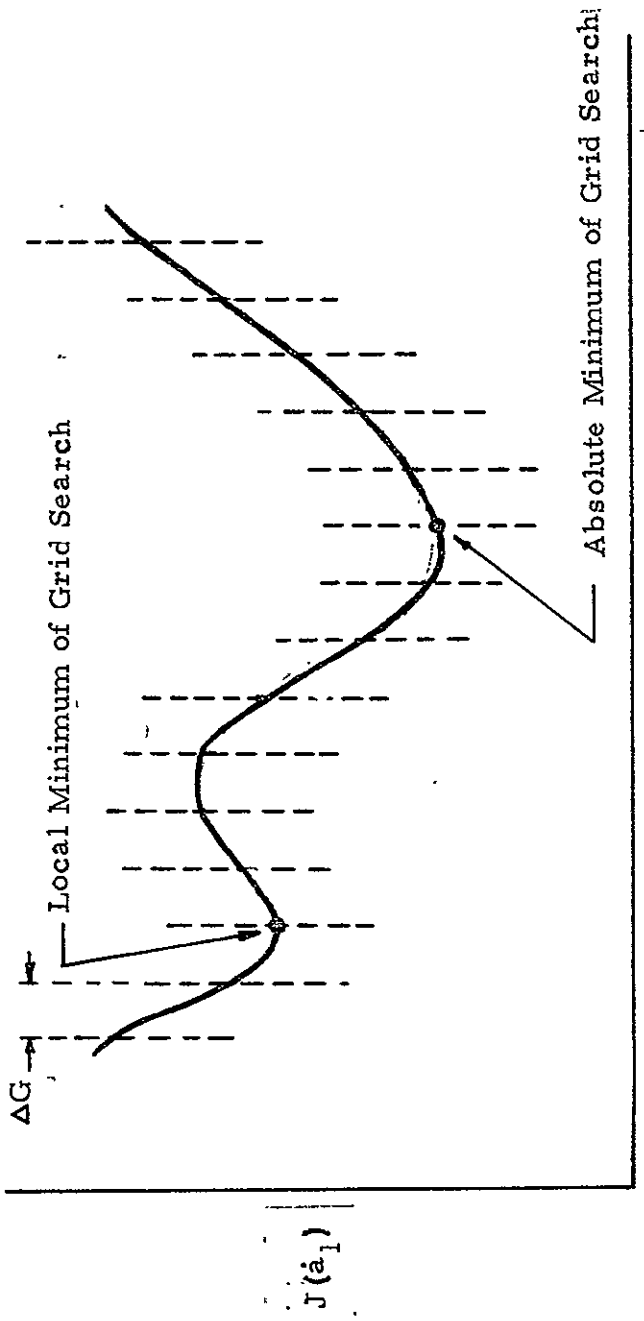


Fig. 4 - Performance Index (J) Versus Controller Gain Slope ($\dot{a}_1$) Evaluated over Time Interval (t_ν, $t_\nu + T$)

The grid is of specified fineness (ΔG) and limits. Figure 5 is an example of a grid search where two parameters are optimally adjusted. It is apparent that the grid search scheme would be difficult to show graphically for three or more dimensions, but the general multi-dimensional case is easily handled by the digital computer.

Besides avoiding local minima, the grid search locates the approximate absolute minimum. The fineness of the grid determines the accuracy of this first coarse minimization. Figure 5 shows how the grid search has located this point. At this time the gradient search takes over and proceeds to find the precise location of the absolute minimum.

2.3 GRADIENT SEARCH

The grid search minimum is used as the starting point of the gradient search. The optimization is then performed in the following steps:

- The plant dynamics are simulated in the first optimization interval (t_o, $t_o + T$), using the gain slopes of the grid search minimum which are lumped into the parameter vector

$$\overline{K}_i = \begin{bmatrix} K_o \\ K_1 \\ K_2 \\ K_3 \\ \vdots \\ K_m \end{bmatrix}_i = \begin{bmatrix} \dot{a}_o \\ \dot{a}_1 \\ \dot{a}_2 \\ \dot{a}_3 \\ \vdots \\ \dot{a}_m \end{bmatrix}_i$$

where the subscript i indicates the number of the current iteration. During this simulation, the performance is measured by computing the performance criterion J, that has been formulated to best reflect the design objectives.

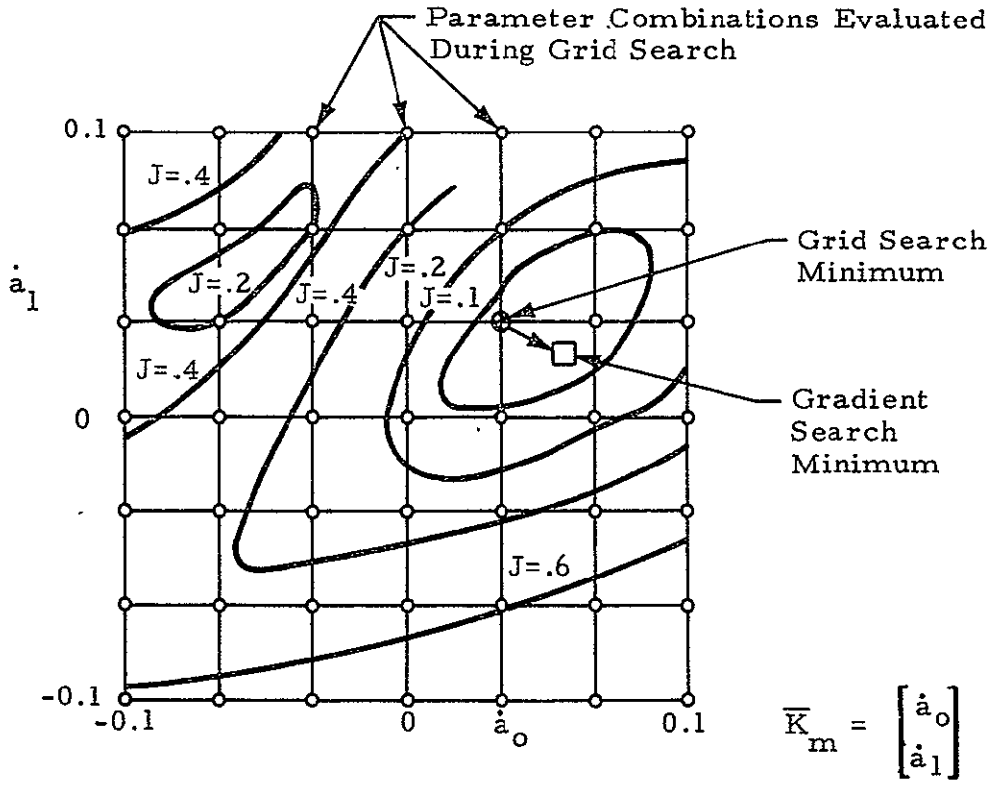


Fig. 5 - Parameter Optimization Performed in Two Phases: (1) Systematic Grid Search (o) for Complete Survey of Parameter Space; Grid Point of Minimum J (⊛) Serves as Starting Point for (2) Gradient Search Which Locates the Minimum More Precisely (□). From Grid Search Contour Plots (Lines of J = Const) can be Drawn for Better Insight into J-Topology.

- At the end of the first simulation, i.e., at time $t_o + T$, J is transferred to the digital computer where the Fletcher-Powell (Ref. 1) gradient minimization scheme is programmed that is capable of minimizing functions of many variables such as $J(\overline{K})$.

- To this end the gradient vector

$$\overline{\nabla J} = \begin{bmatrix} \partial J/\partial \dot{a}_o \\ \partial J/\partial \dot{a}_1 \\ \vdots \\ \end{bmatrix}$$

is required. $\overline{\nabla J}$ is computed using the linearization

$$\overline{\nabla J} \approx \begin{bmatrix} \left[J(\overline{K}_o + \Delta \dot{a}_o) - J(\overline{K}_o) \right] / \Delta \dot{a}_o \\ \left[J(\overline{K}_o + \Delta \dot{a}_1) - J(\overline{K}_o) \right] / \Delta \dot{a}_1 \\ \vdots \end{bmatrix}$$

where

$$\overline{K}_o + \Delta \dot{a}_\nu = \begin{bmatrix} K_o \\ K_1 \\ \vdots \\ K_\nu + \Delta \dot{a}_\nu \\ \vdots \\ K_m \end{bmatrix}$$

Perturbing one parameter at a time by a small amount $\Delta \dot{a}$ and repeating the simulation over the same interval $(t_o, t_o + T)$ yields the PC $J(K_o + \Delta \dot{a})$ required for this gradient computation.

J and $\overline{\nabla J}$ are evaluated for a second guess $\overline{K}_o^*$ over the same interval by subsequent simulations as shown in Fig. 6.

A cubic curve $J_c(\overline{K})$ or, in the case of m adjustable parameters, an m-dimensional cubic surface is fitted through the two points $\overline{K}_o$, $\overline{K}_o^*$ in the m-dimensional parameter space. The minimization scheme then computes the point $\overline{K}_1$ where $J_c(K_1)$ has its minimum.

If $J(\overline{K}_1)$ is smaller than $J(\overline{K})$ and $J(\overline{K}^*)$, then the parameter vector $\overline{K}_1$ is used for the next iteration following the previous steps.

- If after the i^{th} iteration no further reduction of J is possible in the first interval $(t_o, t_o + T)$, the corresponding parameter $\overline{K}_i$ is considered to be optimal.

- A last simulation is made over the first updating interval (t_o, t_1) which may be identical to the first optimization interval, as shown in Fig. 3-a or shorter (Fig. 3-b). The optimal adjustment K_i is used during this simulation. At time t_1 all the state-variables (or integrator outputs) lumped in the state vector $\overline{x}(t_1)$ are stored to be used as initial state for the following optimization in the next time interval $(t_1, t_1 + T)$ which follows the same steps.

Figure 6 illustrates operation of the gradient search.

To solve the whole optimization problem within a short computer time, all simulations are performed in fast time scale (typically 1000 times real time). Only the last simulation using the optimal parameter vector is run at a slower scale (typically real time) in order to obtain a precise recording of desired plant parameters and precise initial conditions for the next optimization cycle.

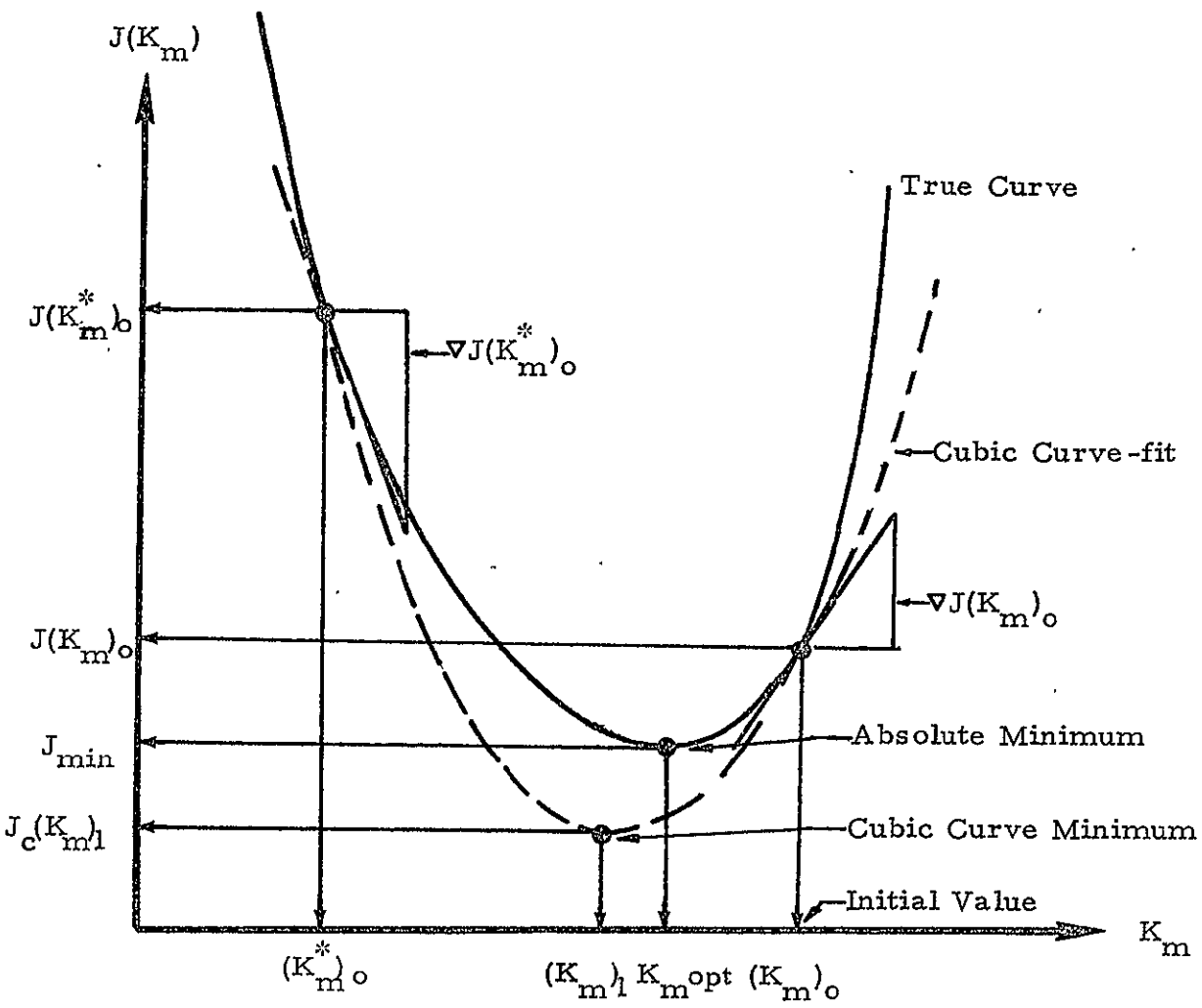


Fig. 6 - Location of Minimum by Gradient Search Techniques

Section 3
CAPABILITIES

3.1 MULTIPLE OPERATING CONDITIONS

The hybrid technique has the capability to consider simultaneously two adverse operating conditions of a control system during optimization. This capability enables the optimization method to generate optimal piece-wise linear gain schedules which are not sensitive to a specific adverse operating condition. An example of two adverse conditions would be the failure of a leeward or the failure of a windward engine of a multiple engine space vehicle booster.

Referring to Fig. 6 and its detailed explanation in the previous section, this capability is programmed as follows. The performance of the plant is evaluated for each adverse condition using the given parameter vector $(\overline{K}_m)_o$. The adverse condition which causes the worst performance, P_{max}, is used in subsequent simulations to compute the gradients $\overline{\nabla J}(\overline{K}_m)_o$. Plant performance is evaluated for both adverse conditions for the second guess $(\overline{K}^*_m)_o$. As before, the adverse condition which causes the worst performance is used in the simulations to compute the gradients $\overline{\nabla} J\,(K^*_m)_o$. The gradient search technique continues its various computations and whenever it is necessary for the performance of the plant to be evaluated for a given $(\overline{K}_m)_i$, both adverse conditions are simulated and the condition causing the worst performance is used in the simulations for gradient computations. This procedure is justifiable since the perturbations necessary to compute the gradients are small enough to cause only minor changes in the performance of the plant for a given $(\overline{K}_m)_i$.

This gradient technique could easily be extended to consider three or more adverse conditions if such a need existed for a given control system.

3.2 SLOW RUN OPTIONS

A number of slow run options have been built into the program to provide aids in checkout and to provide the capability to run special cases. Figure 7 shows a flow diagram of the slow run section of the program.

Option 1: KPRINT = 1

This option, which instructs the analog digital printer to record all amplifier outputs before and after a slow run, was provided as a check on possible drift of track-and-store units during the optimization cycle and is intended for use when hardware trouble is apparent.

Option 2: TSTART

This variable is defined as the time during the mission at which optimization is desired to start. For example, if mission time starts at 40 sec and ends at 100 sec, it might be desired to begin optimization of controller parameters at 50 sec. If no optimization is desired, TSTART would be set equal to or greater than 100 sec.

Option 3: TFAIL

This variable is defined as the time during the mission when a certain event, such as a failure, occurs. Optimization also ceases at the first update interval after TFAIL. Figures 8 and 9 illustrate the use of TSTART and TFAIL in running constant gain runs and optimizing runs.

Option 4: TSLOPE (ii)

SLOPE(1, ℓ); SLOPE(2, ℓ); . . . ; SLOPE(12, ℓ)

ii = 1, 12

ℓ = 1, m

Assume that after a series of systematic optimization runs have been made, particular time-varying controller schedules are to be run rather than constant controller schedules. This option has been provided by the subroutine SLOPE. The variable TSLOPE (ii), ii = 1, 12, is used to determine

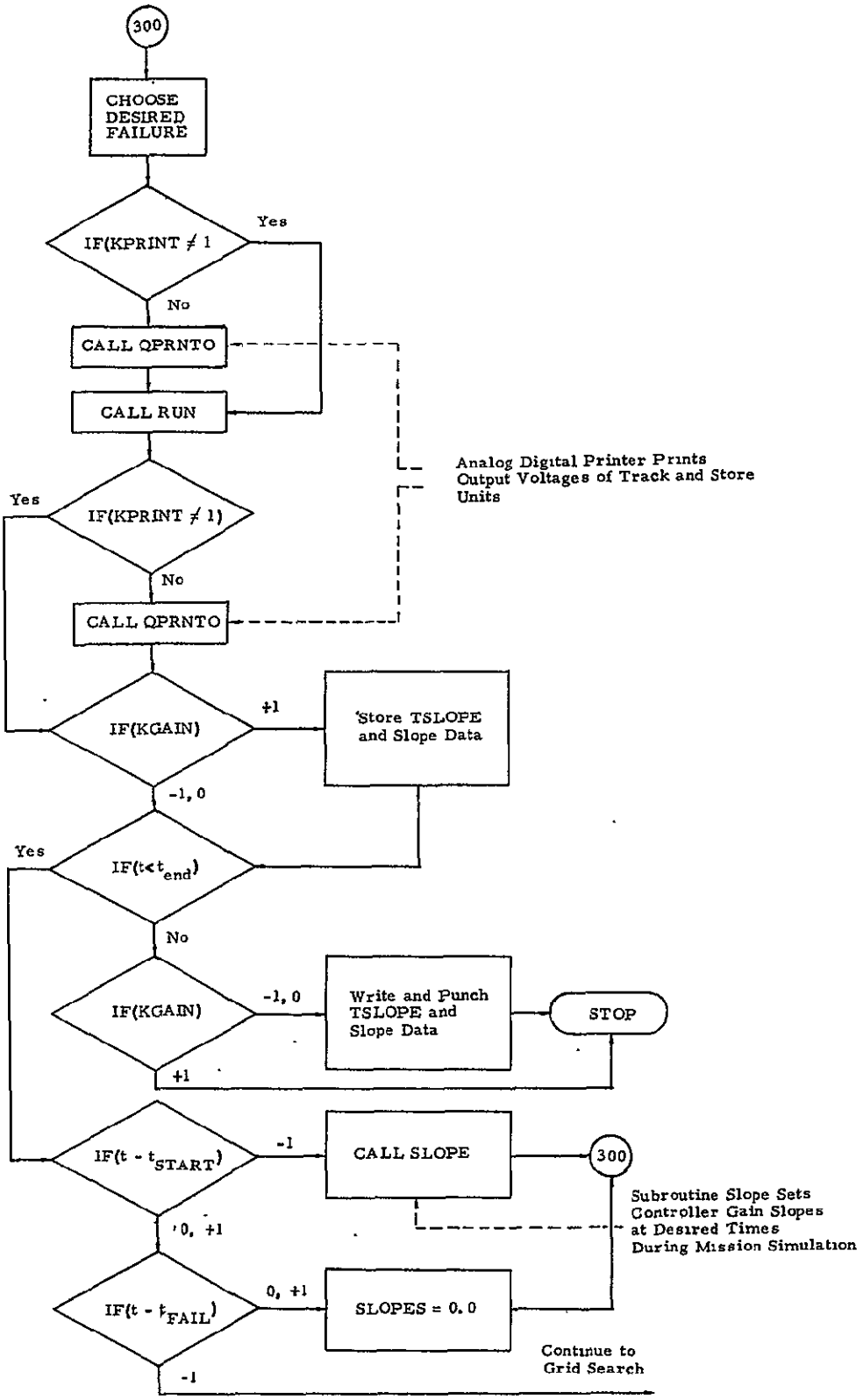
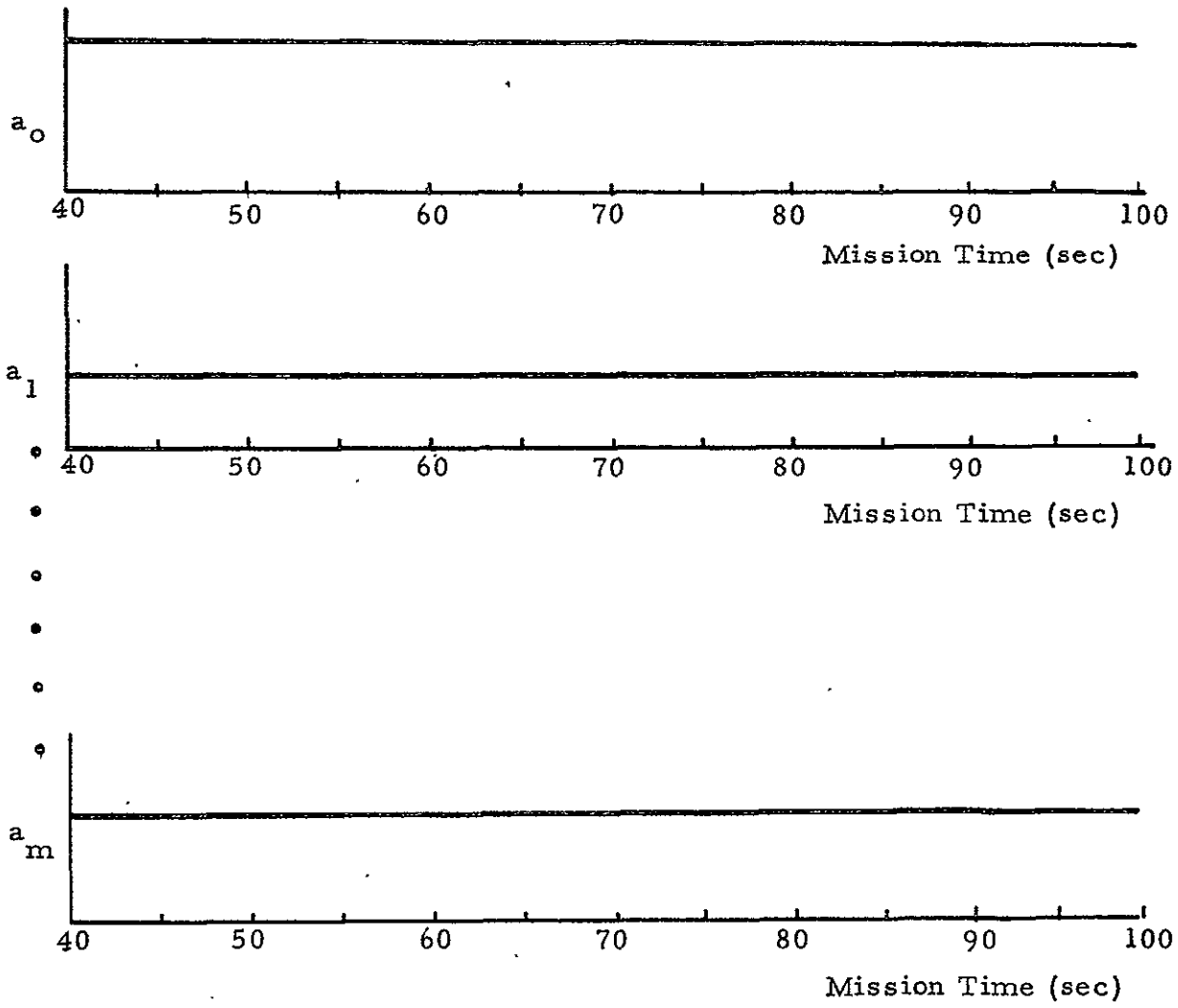
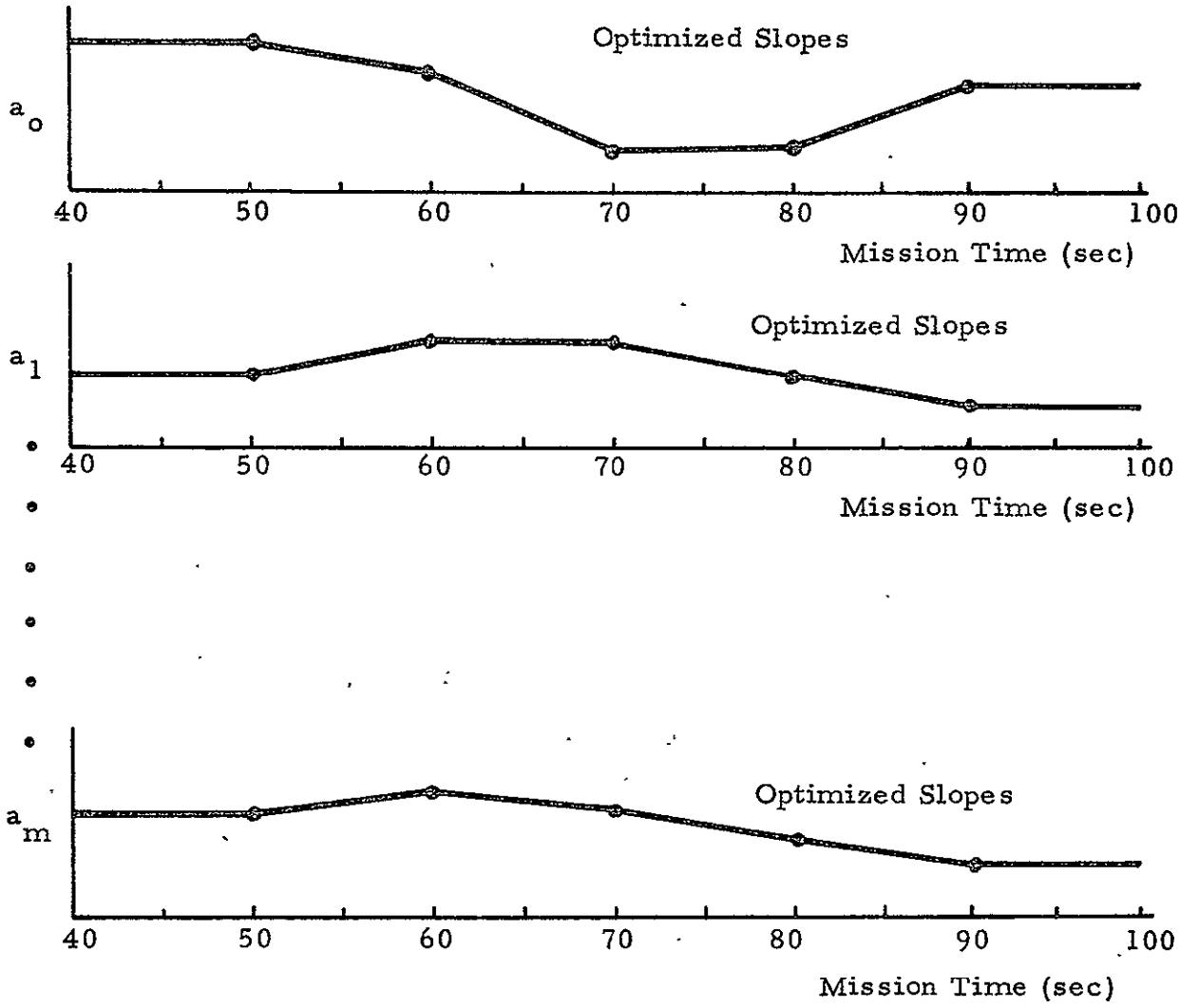


Fig. 7 - Simplified Flow Diagram of Slow Run Section of Optimization Program



Start of Mission Simulation = 40 sec
TSTART = 100 sec
TFAIL = desired failure time
m = number of controller parameters

Fig. 8 - Controller Parameters Maintained Constant for a Mission Simulation by Use of Option TSTART



Start of Mission Simulation = 40 sec
TSTART = 50 sec
TFAIL = 85 sec
m = number of controller parameters optimized

Fig. 9 - Controller Parameters Optimized Starting at 50 sec and Stopping at 90 sec by the use of Options TSTART and TFAIL

the times during the mission simulation when a change in slope of the controller schedules is desired. SLOPE (1, ℓ) corresponds to the desired slopes of each controller parameter at TSLOPE(1). Figure 10 illustrates an example of this option.

Option 5: KGAIN

When desired by the operator, this option can be used to avoid unnecessary writing and punching of TSLOPE and SLOPE data. For example, if an optimization for two adverse Conditions A and B has just been completed and Condition A has been simulated on the slow run recorder, then either the optimization must be duplicated and Condition B simulated, or the controller gain schedules stored and Option 4 used to simulate Condition B. KGAIN is set to zero when an optimization is being run and set to +1 when any other option is being used.

3.3 OSCILLOSCOPE PLOT OPTION

This option is controlled by the variables, KPAUSE and NPOINT and is useful only during the grid search cycle. KPAUSE is set equal to +1 if a plot of the J function is desired during the grid search. NPOINT is set equal to the total number of computations required to complete a grid search. For example, if a two-dimensional grid (n × n) is being searched, then NPOINT would be set equal to n^2. Each time a grid point is simulated, a counter called NRUN is updated by one. When NRUN equals NPOINT, a pause command is automatically given to the hybrid computer. If desired at this point, a photograph of the J function as displayed on the oscilloscope may be taken. A plot of the J function may also be observed during the gradient search, but provision for automatically pausing the computer at the end of this cycle has not been provided.

If an oscilloscope plot is not desired, KPAUSE is set equal to zero.

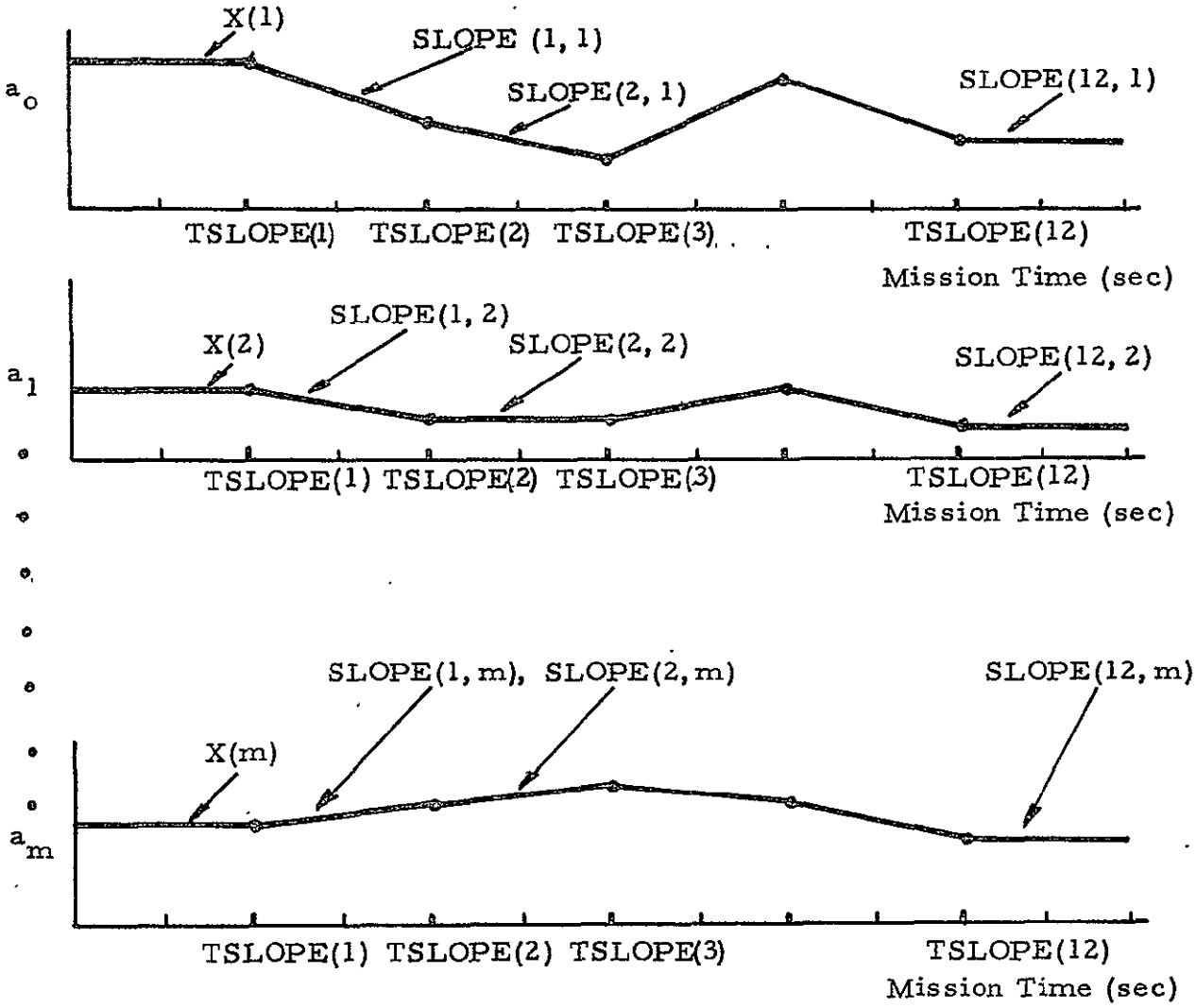


Fig. 10 - Illustration of Option to Use Desired Controller Schedules for Mission Simulation without Optimization

Section 4
IMPLEMENTATION OF HYBRID OPTIMIZATION SCHEME

How the hybrid optimization scheme is implemented to enable the reader to apply this scheme to other problems is explained in this section. The various sections of the digital program and the various control and interface lines are explained. Figure 11 shows the general hybrid scheme and the interface lines being used. Figure 12 shows a flow chart of the minimization scheme. This flow chart is provided as an aid to understanding the operation of the optimizer and for checkout purposes. The digital print statements enable the user to observe the flow of computation of the optimizer by watching the digital printout and comparing it with its location on the flow chart.

4.1 GRID SEARCH

Figure 13 shows a flow chart of the grid search. The variable DELTA is used to control sense line 6, which in turn controls the adverse conditions of the analog simulations. For example, if DELTA = +1, (-1) then sense line 6 is true (false). These two conditions of sense line 6 represent the two adverse Conditions A and (B), respectively, of the dynamic simulation. For each point in the grid, adverse Condition A is simulated first. Necessary data from this simulation such as the J function (DUMMY), the performance measure $P_{max}(P)$, grid point coordinates and any scale changes (ADCV) are stored in memory. DELTA is set to -1 in order for adverse Condition B to be simulated. The same data from this simulation is stored and then compared to the data from Condition A. Digital logic determines the worst condition of the two Conditions A and B, and this worst condition data is held in storage to be compared with the worst condition data at the next grid point. The smallest $P_{max}(P)$ between these two grid points is stored to again be compared with the $P_{max}(P)$ of the worst condition data of the next

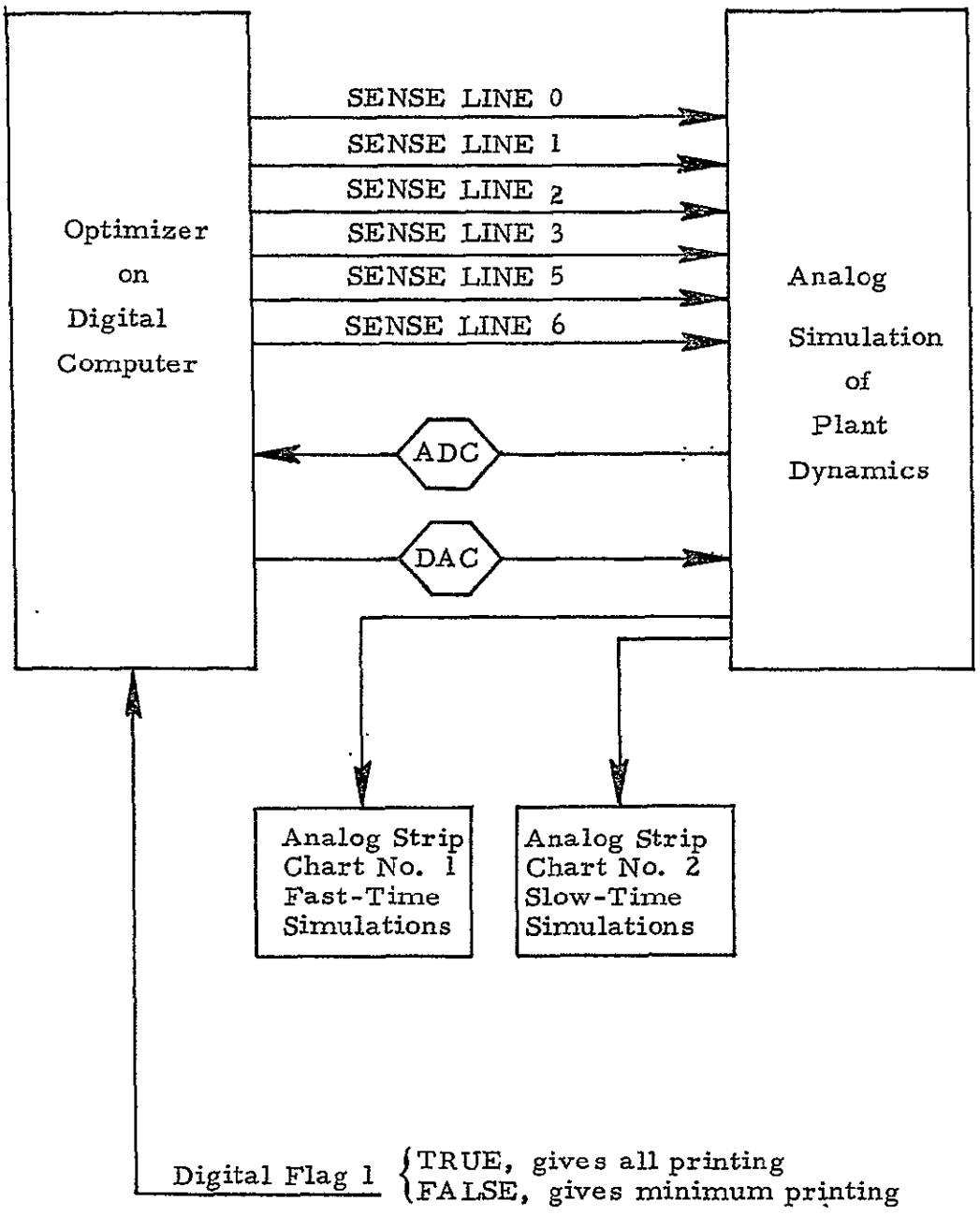


Fig. 11 - Interface Scheme Hybrid Optimization Program

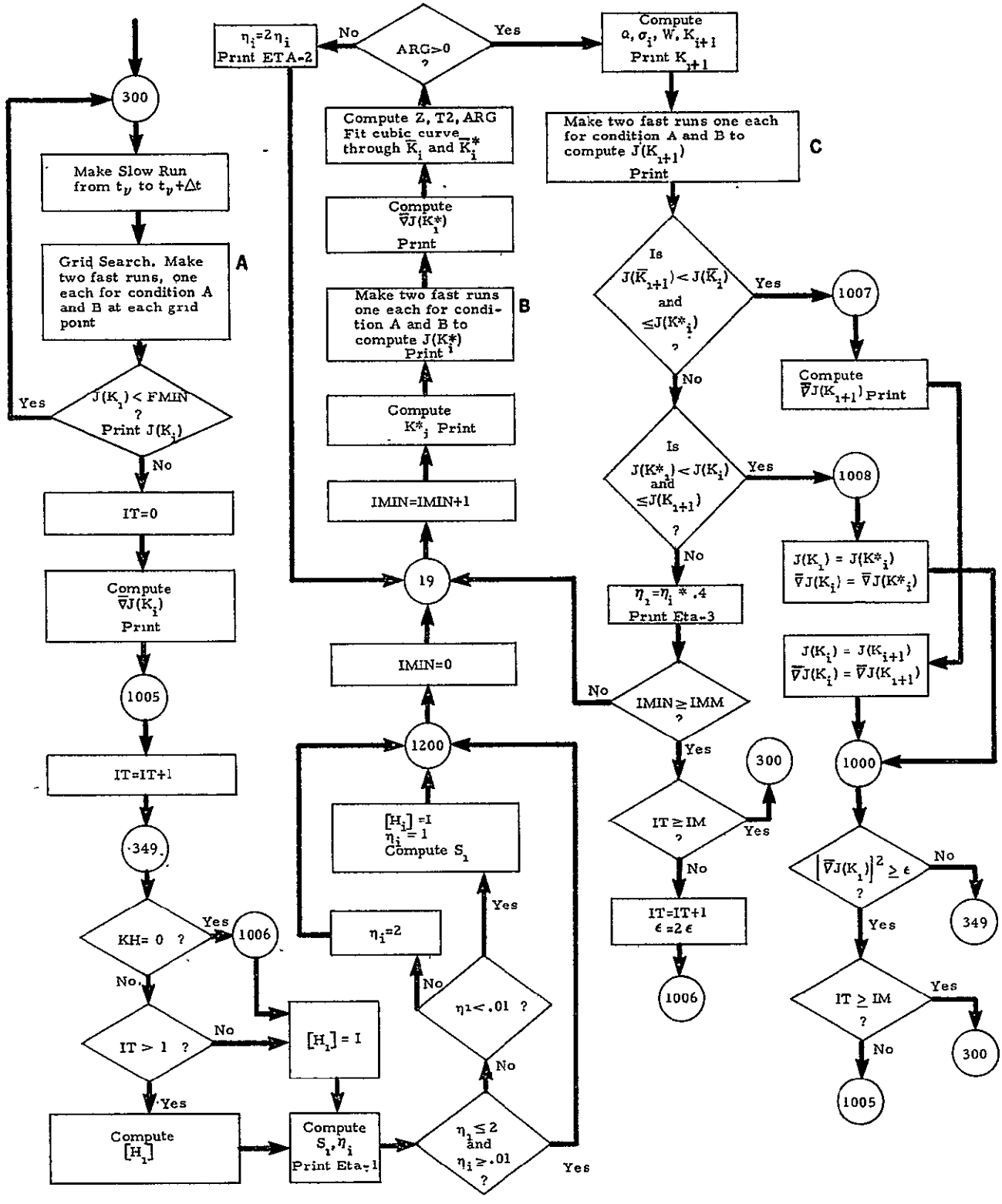


Fig. 12 - Flow Chart of Minimization Scheme Based on Fletcher and Powell (Ref. 1), Extended for Grid Search Preceeding Gradient Search. Performance Index J Depends upon two Dynamic Simulations (Condition A and Condition B).

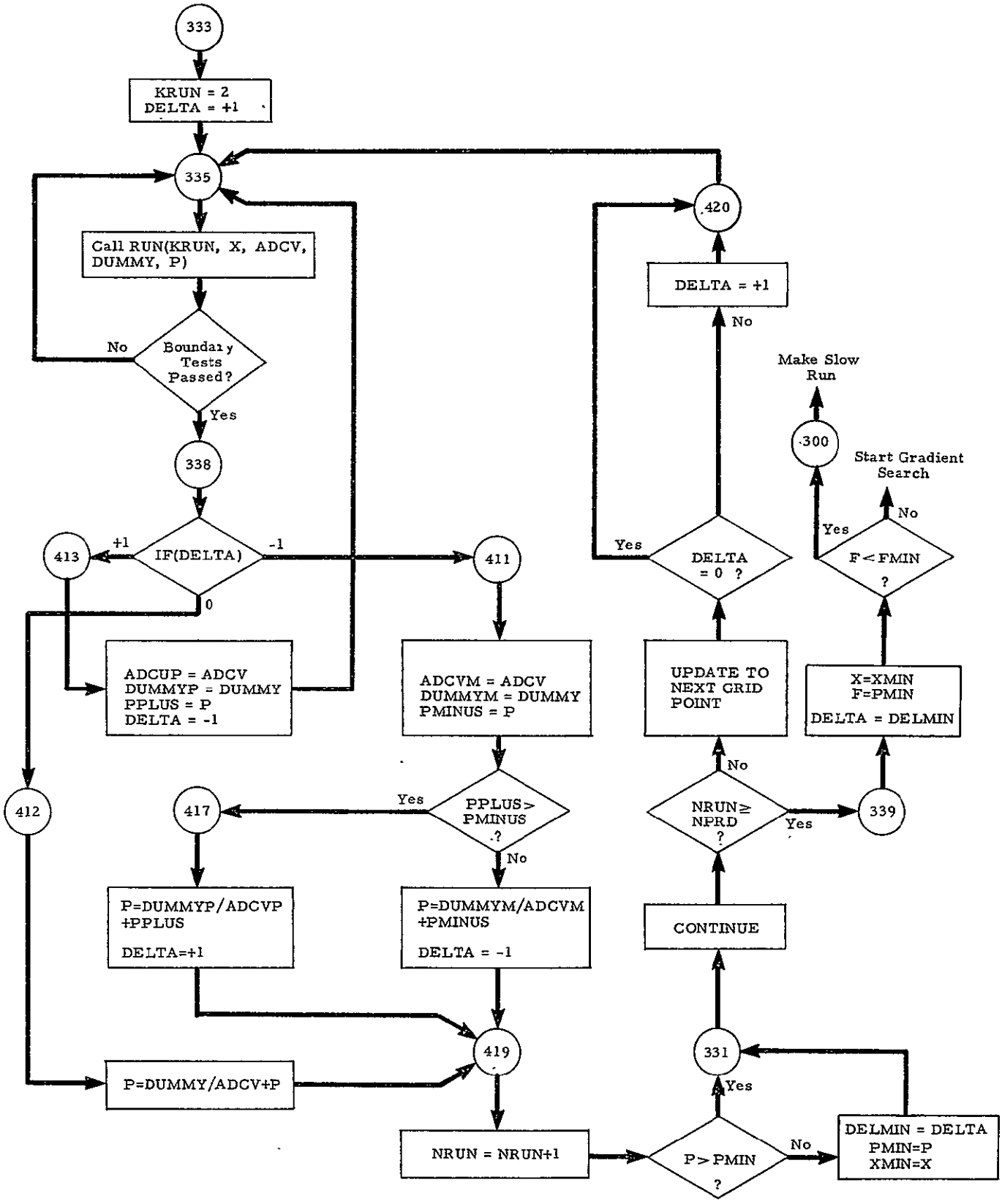


Fig. 13 - Grid Search Flow Chart

grid point. This cycle continues until the smallest $P_{max}(P)$ and its associated controller parameters have been determined for the whole grid. These parameters are then used to initialize the gradient search.

4.2 GRADIENT SEARCH

The gradient search now improves the controller parameters obtained from the grid search. Figures 14 and 15 are included to illustrate the simulation of both adverse Conditions A and B when $\bar{\bar{J}}(\bar{\bar{K}}_i^*)$ and $\bar{\bar{J}}(\bar{\bar{K}}_{i+1})$, respectively, are computed. Condition A is always simulated first, then Condition B, and the worst condition of the two is used to compute gradients. The computation of the gradients $\bar{\bar{\nabla}}J(\bar{\bar{K}}_i)$ and $\bar{\bar{\nabla}}J(\bar{\bar{K}}_i^*)$ considers only one adverse condition, that is, the worst condition of A or B. This is justifiable because the perturbations of $\bar{\bar{K}}_i$ and $\bar{\bar{K}}_i^*$ used to compute $\bar{\bar{\nabla}}J(\bar{\bar{K}}_i)$ and $\bar{\bar{\nabla}}J(\bar{\bar{K}}_i^*)$ are small enough to cause only minor changes in the P_{max} of each case if both cases are considered.

4.3 INTERFACE CONNECTIONS

4.3.1 Sense Line 0

Sense line is identified by logic variable READY and used to hold digital during slow runs.

4.3.2 Control Lines 0, 1, 2, 3

These control lines are used with digital function switches in a digital logic circuit to control the mode of the slow run strip chart and the mode of the track and store units for the various operate modes of the hybrid program (such as pot-set, dynamic check, slow runs and fast runs).

4.3.3 Control Lines 5, 6

Control line 5 is used with digital logic to determine if an adverse event (such as a failure) will occur. Control line 6 determines which event of two adverse events will occur.

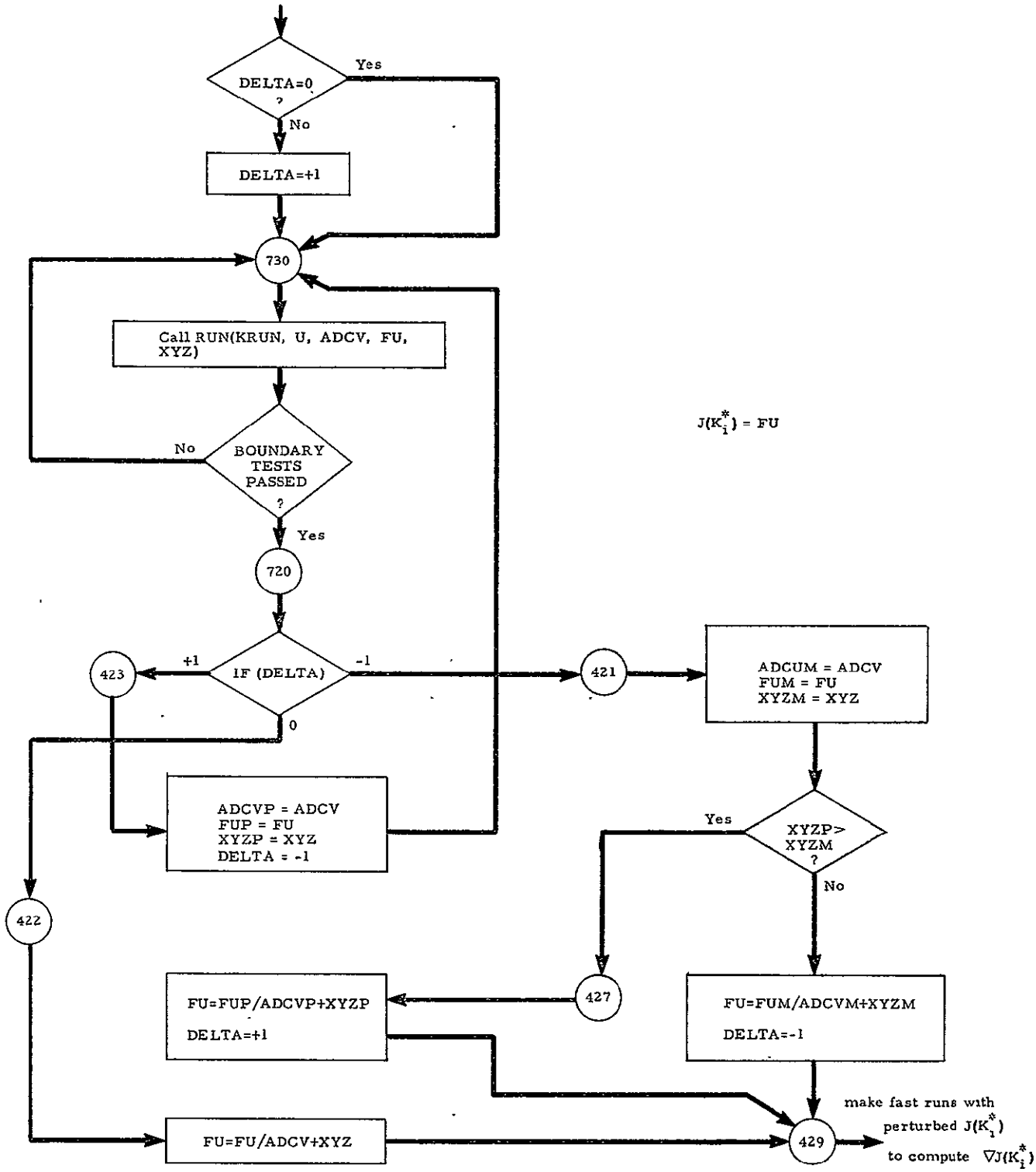


Fig. 14 - Computation of $J(K_i^*)$

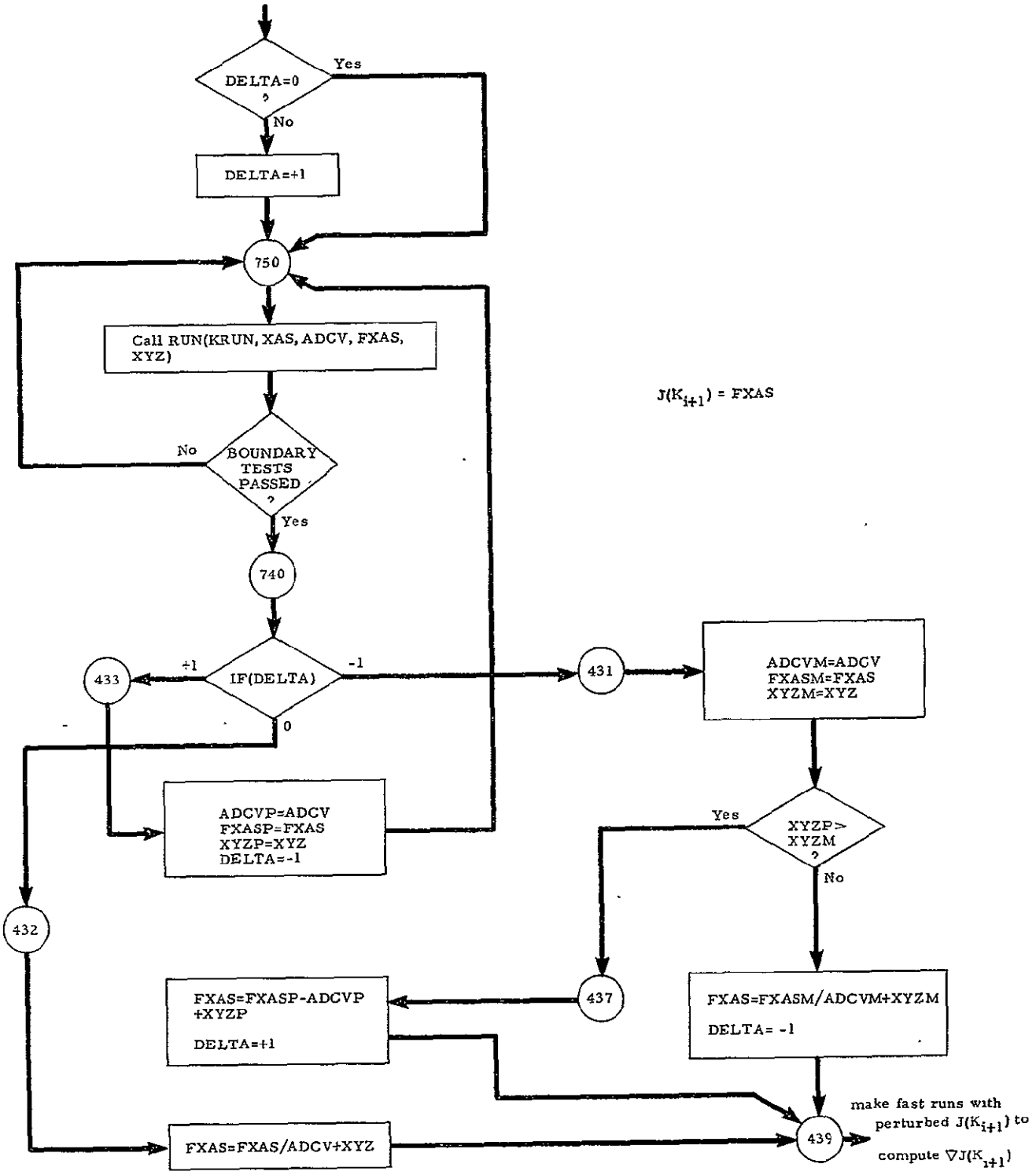


Fig. 15 - Computation of $J(K_{i+1})$

4.4 RECORDING SEQUENCES

To assess the effects of the optimized controller schedules upon both adverse conditions, two simulations should be performed, one assuming that Condition A occurs and a second one assuming that Condition B occurs. Rather than perform the complete optimization cycle twice, it is more efficient to use Option 4. Chart 1 illustrates this sequence of computation. This method reduces total computer time by approximately 50% since the majority of the total computing time is used in optimizing.

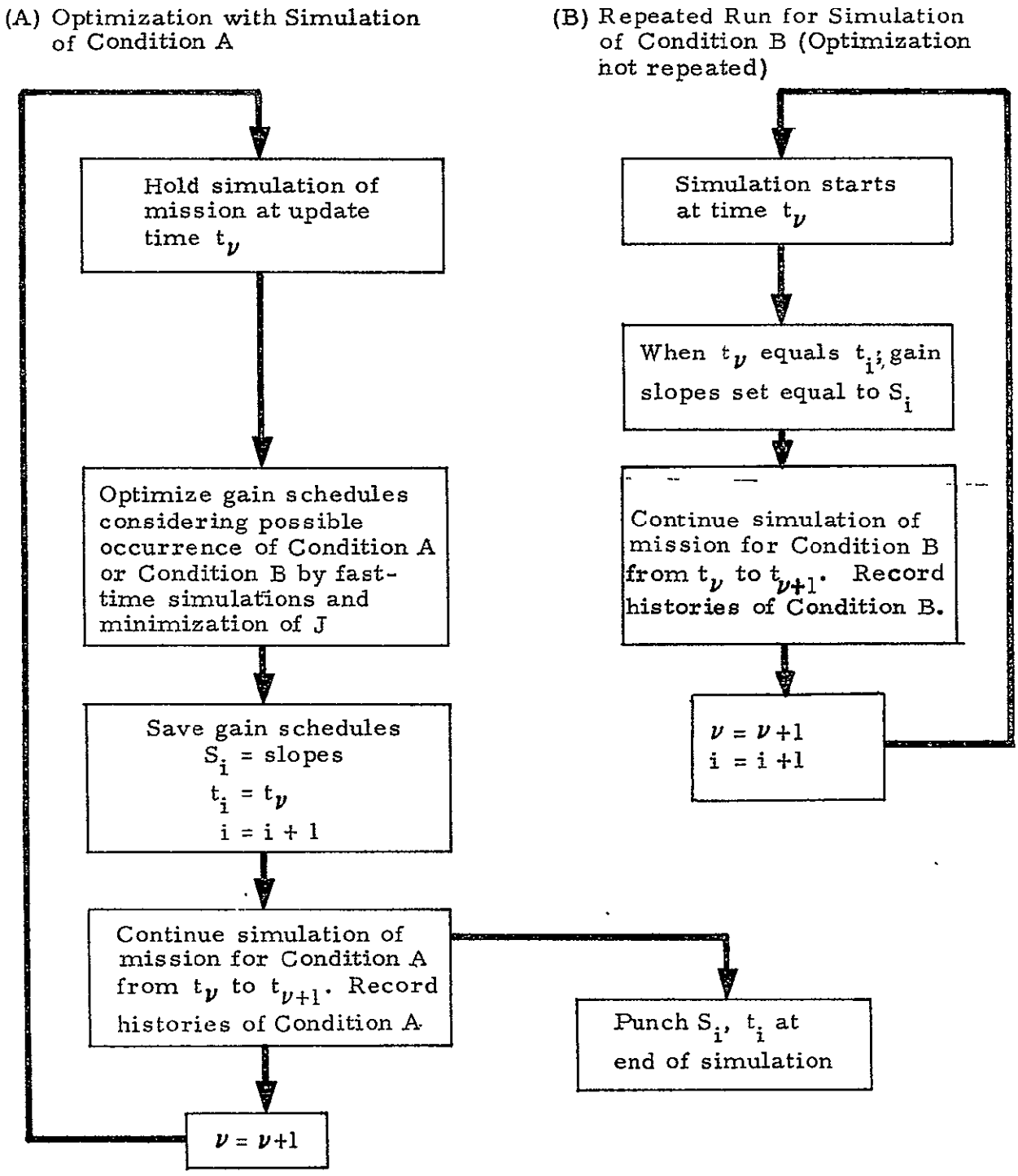


Chart 1 - Sequence of Computations for Simultaneous Optimization of Two Cases

Section 5
EXAMPLE CASES

This section is provided as an aid for data setup for various cases. A thorough study of the digital program and familiarity with the nomenclature is necessary to understand and use the various options in running the following cases.

5.1 CONSTANT GAINS

X = 0.0
KGAIN = 1
DEL = +1
TSTART ≥ TEND
TSLOPE(1) ≥ TEND
SLOPE(I, J) = 0.0
TFAIL = desired time for adverse condition

5.2 OPTIMIZATION OF PIECEWISE LINEAR GAIN SCHEDULES

X = 0.0
TSTART = desired optimization start time
TFAIL = desired time for adverse condition
SLOPE(I, J) = 0.0
TSLOPE(1) = 0.0
DEL = ± 1
KGAIN = 0

5.3 PRESELECTED GAIN SCHEDULE

TSTART = TEND
TFAIL = desired time for special event
DEL = ± 1
TSLOPE(I) = desired times to change slopes (break points)
SLOPE (I, J) = desired slope starting at TSLOPE(I)
KGAIN = +1

Section 6

REFERENCES

1. Fletcher, R., and M.J.D. Powell, "A Rapidly Convergent Descent Method for Minimization," Computer Journal, Vol. 6 (1963).

2. Hewitt, G. R., "Parameter Optimization Study for the ATM-CMG Control System," LMSC/HREC D148796, Lockheed Missiles & Space Company, Huntsville, Alabama, March 1969.

3. Trautwein, W., and J. G. Tuck, "Control System Optimization for Saturn V Launch Vehicles Using Gradient Techniques," LMSC/HREC A791836, Lockheed Missiles & Space Company, Huntsville, Alabama, October 1968.

4. Trautwein, W., and G. W. Coyne, "Control System Optimization for Saturn V Launch Vehicles Using Gradient Techniques," LMSC/HREC D148931, Lockheed Missiles & Space Company, Huntsville, Alabama, April 1969.

Appendix A

NOMENCLATURE USED IN DIGITAL PROGRAM

Appendix A

ADCV	variable representing scale changes during computation of J function
ADCVM	storage location for ADCV when adverse condition B is computed in the grid search, and at various points in the gradient search
ADCVP	storage location for ADCV when adverse condition A is computed in the grid search and at various points in the gradient search
AMULT	analog scale factor for $\left[P_{max}\right]^2$
ANALV	represents P_{max} in subroutine RUN
ANS (I)	storage location for X(I) I = 1, IS
BDLØ	lower bound of J used to trigger scale changes in the computation of J function
BDUP	upper bound of J used to trigger scale changes in the computation of J function
BMULT	analog scale factor for stability term of J function

DEL	constant used to display the desired adverse condition during slow runs DEL = +1 adverse condition A DEL = -1 adverse condition B DEL = 0 no adverse condition
DELMIN	variable used to store the worst condition computed at every grid point during the grid search. It enables the gradient search to know which adverse condition should be used in the initial perturbations of the controller parameters.
DELX(IS)	grid search increment size
DUMMY	variable equivalent to J function
DUMMYM	storage location for DUMMY when adverse condition B is computed in the grid search
DUMMYP	storage location for DUMMY when adverse condition A is computed in the grid search
DX	amount of perturbation of X during gradient search. Used to compute slopes of J function
ETA	optimizer quantity which is the output of the perturbation cycle for the first half of any one iteration and is used to modify the optimized parameters in this part of the iteration. The modification is in the form of a straight-line extrapolation where η acts as the independent variable.
ETAMN	minimum value that ETA can assume

ETAMX	maximum value that ETA can assume
FACT	constant conversion factors for $\dot{a}_1, \dot{a}_2, \ldots, \dot{a}_m$, m=IS
FEPS	tolerance of computed value of minimum of J function
FM	constant used in Fletcher-Powell technique (.5 to .8)
FMIN	minimum value J can have. Optimization procedure is stopped when this value is reached.
FU	variable equivalent to $\overline{J}(\overline{K}_i^*)$
FUM	storage location for computation of $\overline{J}(\overline{K}_i^*)$ when adverse condition B is simulated during the gradient search
FUP	storage location for computation of $\overline{J}(\overline{K}_i^*)$ when adverse condition A is simulated during the gradient search
FXAS	storage location for computation of $\overline{J}(\overline{K}_{i+1})$ during gradient search
FXASM	storage location of FXAS during simulation of adverse condition B in the gradient search
FXASP	storage location of FXAS during simulation of adverse condition A in the gradient search
G(I)	variable representing gradients at various points in the gradient search
GAIN	variable used in Subroutine Scale to make scale changes

GEPSQ	specified admissible $(\text{slope})^2$ of optimum of J function
$[H]$	operational matrix in the optimizer which is used to carry the parameter values along a line of steepest descent such that the performance index is eventually minimized. The initial value of the matrix is the identity matrix. After each iteration, information gained on the preceding pass is used to update the matrix to a new value. The updated matrix in turn insures an evaluation of the optimized parameters that brings the performance index closer to its minimum.
IM	specified number of iterations to locate optimum J function
IMM	number of attempts in one iteration
IS	number of variables to be optimized
JUMP	logic variable used to jump various parts of the digital program Jump = +1 set potentiometers and prepare for dynamic check Jump = +2 change potentiometers and prepare for production run
KFAST	constant used to establish amount of time for a fast hybrid run T
KH	option used to set $[H_i] = [I]$ for first iteration
KPAUSE	option to plot J function on oscilloscope KPAUSE = +1 Plot KPAUSE = 0 Don't Plot

KPRINT	constant used to initiate option for print-out on analog digital printer of amplifier outputs of track-and-store units
KRUN	variable used to determine speed of analog simulation KRUN = 1, real time KRUN = 2, 1000 times real time KRUN = 3, 10 times real time
KSLOW	constant used to establish amount of time for a slow hybrid run
NDX (IS)	grid search dimension
NPOINT	desired number of simulations required before a photograph may be taken of the J function on the oscilloscope
NPRD	total number of grid points
NPROD(NVARY)	total number of grid points
NRUN	counter in grid search loop
NVARY	number of variables to be optimized $\dot{a}_1, \dot{a}_2, \ldots, \dot{a}_m$ m = IS. Used in grid point up-date logic of grid search
P	variable equal to stability term of J function
PMIN	storage location for minimum J function during grid search
PMINUS	storage location for P when adverse condition B is computed in the grid search

PP(M)	temporary variable representing J function during perturbations of X
PPLUS	storage location for P when adverse condition A is computed in the grid search
SUMJ(L)	represents the performance measures of the plant dynamics. They are P_1, P_2, or P_3 where P_{max} is equal to the worst of the three.
SUMX	represents J function computation in Subroutine RUN
TEMP	variable used to store X before X is perturbed, thus having X available after gradients have been computed from the perturbations of X
TEND	time to end mission simulation
TFAIL	desired time for adverse conditions to occur
TOPF	constant used to set the bound of the Y axis in the Subroutine PLOT
TSTART	desired time to start optimization of controller parameters
U(I)	represents the controller parameters when perturbations are being made in the gradient search to compute $\nabla\bar{J}(\bar{K}_i^*)$
UNITY (I, J)	unity matrix used to initialize $\left[H\right]$ in gradient search

VER	variable used to transmit J function computation to Subroutine PLOT in order to be plotted
X(IS)	variables to be optimized where IS is number of variables $\dot{a}_1, \dot{a}_2, \ldots, \dot{a}_m$ $m = IS$
X2	initial values of parameters to be optimized $\dot{a}_1, \dot{a}_2, \ldots, \dot{a}_m$ $m = IS$
XAS(I)	represents the controller parameters when perturbations are being made in the gradient search to compute $\overline{\nabla J}(\bar{K}_{i+1})$
XMAX	maximum absolute value of parameters to be optimized
XMIN (I)	storage location of controller parameters corresponding to the grid search minimum
XYZ	variable equivalent to performance measure P_{max} during computation of $\bar{J}(\bar{K}_i^*)$
SYZM	storage location for XYZ when adverse condition B is simulated during the gradient search
XYZP	storage location for XYZ when adverse condition A is simulated during the gradient search
YMIN	constant used in the computation of $\left[H\right]$

Appendix B

DIGITAL PROGRAM LISTING

```
$JOB
$FR*,IU1,*4033
      DIMENSION     A ( 5, 5 ) ,     B ( 5, 5 ) ,     E1 ( 5, 5 ) ,
     1             E2 ( 5, 5 ) ,     G (    5 ) ,     GU (    5 ),
     2           GPREV(    5 ) ,     H ( 5, 5 ) ,     S (    5 ) ,
     3            SIG (    5 ) ,     U (    5 ) ,     X (    5 ) ,
     4            XAS (    5 ) ,     Y (    5  ),FACT   ( 5 ) ,
     5            CHEK ( 5 ) , ANS ( 5 ) , UNITY ( 5,5 ) , PP(2)
      DIMENSION NPROD(5),NDEX(5),DELX(5),XO(5),XMIN(5),NDX(5) ,X2(5)
      DIMENSION  XP(5),XM(5),UP(5),UM(5),XASP(5),XASM(5)
      COMMON / PST / NPOT,POT(150),PSET(150),VAL,VALUE
      COMMON / RN / KTIME(3),IS,KMODE,AMULT,KCONT ,BMULT
      COMMON / SCL / BDUP,BDLO,GAIN,ADCV
      COMMON / PLT / VER ( 5 ) , TOPF
      COMMON/DE/DELTA
      COMMON/SLOPES/TSLOPE(12),SLOPE(12,5),MG
      EXTENDED POT , VAL , KTIME
      DATA  CHEK / 5*0,0 /
      DATA VAL/4HA001/
      EXTENDED IFIRST, ILAST, NCOMP
      DATA IFIRST /4HA000/, ILAST/4HA853/, NCOMP /136/
      LOGICAL SET
      KTIME(3) = 14400
      CALL QDSITO (1,IR)
      CALL QREFFO ( 1, 0, IR )
      CALL QREFFO ( 1, 1, IR )
      CALL QREFFO ( 1, 2, IR )
      CALL QSEFFO ( 1,3,IR )
      CALL QSEFFO ( 1,4,IR )
      CALL QPSO ( 1,IR )
      CALL QRUNO ( 1 , IR )
      READ (5,401)JUMP
      GO TO (503,47)JUMP
  503 TYPE 400
  400 FORMAT ( 10X,66HREADY FOR SETTING POTS AND SWITCH INITIAL RELAY TO
     * CENTER POSITION )
      PAUSE 1
C
```

```
C*************************************************************************
C          SET  POTS  ON  ANALOG  COMPUTER
C
      READ ( 5,401 ) NPOT
  401 FORMAT ( I5 )
      READ ( 5,402 ) ( POT(I),PSET(I) , I=1,NPOT )
  402 FORMAT ((8(A4,F6.0)))
      CALL SETPOT
      TYPE 403
  403 FORMAT ( 10X,23HREADY FOR DYNAMIC CHECK )
      PAUSE 2
C*************************************************************************
C          MAKE  DYNAMIC  CHECK
C
      CALL QCLR0 ( 1 , IR )
      KRUN = 3
      IS = 5
      GAIN = 1.0
      KMODE = 0
      CALL QDAJM0 ( 5,1,GAIN,IR,KCH )
      CALL RUN ( KRUN,CHEK,DUMMY )
   47 TYPE 404
  404 FORMAT ( 10X,29HCHANGE POT FOR PRODUCTION RUN )
      KCONT=0
ASSEMBL
       SFL     =^50,,0          RESET ALL INTERRUPTS
       SFL     =^51,,0          PERMIT INTERRUPTS TO BE SET AGAIN
       ECA     PLACE          FETCH INTERRUPT INSTRUCTION
       EST     ^60            STORE AT INTERRUPT LOCATION
       LDE     =^100000       SET BIT 0 IN EXTERNAL MASK FOR INTERRUPT 0
       LDOB    =^40000,,12       CONNECT TO CONSOLE 1
       JSE     49S
       J       49S
PLACE  LRE     601S
FORTRAN
   49 CONTINUE
      PAUSE 3
C*************************************************************************
```

```
C          CHANGE  POTS  FOR  PRODUCTION  RUN
C
      READ ( 5,401 ) NPOT
      IF ( NPOT.EQ.0 ) GO TO 45
      READ ( 5,402 ) ( POT(I),PSET(I),I=1,NPOT )
      CALL SETPOT
   45 TYPE 405
  405 FORMAT(2X,62HREADY FOR PRODUCTION, RESET FLAG 1 OR 2 IF NEEDED, HI
     1T EXEC           )
C
C          MAKE  PRODUCTION  RUN
C
C     FLAGS 1 AND 2 FALSE GIVE PRINTING
C     FLAG 2 ONLY AFFECTS ONE WRITE STATEMENT IN SUBROUTINE RUN
ASSEMBL
ASS     JS1     ASS+1
BASS    JS2     BASS+1
D       JS3     D+1
E       JS4     E+1
F       JS5     F+1
GG      JS6     GG+1
HH      JS7     HH+1
C       HJ      C+1
FORTRAN
  880 CALL QREFF0(1,3,IR)
      CALL QIC0 ( 1,IR )
      CALL QCLR0 ( 1,IR )
      FXAS=0.
      TIME = 0.
      GAIN = 1.0
      KMODE = 0
      CALL QDAJM0 ( 5,1,GAIN,IR,KCH )
C
C        GET INITIAL VALUES  FOR BOTH THE ANALOG  AND DIGITAL  COMPUTERS
C
      READ ( 5,220 ) FEPS,DX,FM,FMIN,BDUP,BDLO
  220 FORMAT  (  6E12.8  )
      READ ( 5,220 ) X
```

```
      READ ( 5,220 ) YMIN,ETAMN,ETAMX,TEND,TOPF,GEPSQ
      READ ( 5,220 ) ADCV,FACT
      READ ( 5,44 ) IS,KSLOW,KFAST,IM,IMM,KH
      READ (5,220)AMULT ,BMULT
   44 FORMAT ( 16I5 )
      READ(5,220)(DELX(I),I=1,IS)
      READ (5,44)(NDX(I),I=1,IS)
      READ(5,220)X2
      READ(5,220) DEL
      READ(5,44)KGAIN
      READ(5,44)KPRINT,KPAUSE,NPOINT
      READ(5,220)TSTART,TFAIL
      READ(5,220)TSLOPE
      READ(5,220) ((SLOPE(I,J),I=1,12),J=1,5)
C     IF KPRINT=1, PRINT AMPLIFIER OUTPUTS
C     IF DEL=0, NO ENGINE FAILURES OCCUR
      IF(DEL.NE.0.0)GO TO 2001
      CALL QREFF0 (1,5,IR)
      GO TO 2002
 2001 CALL QSEFF0 (1,5,IR)
 2002 CONTINUE
      GEPSQ = GEPSQ*FLOAT(IS)
      KTIME(1) = KSLOW * 1000
      KTIME(2) = KFAST
      XKSLOW=KSLOW
      KKK=0
      MG=1
      XMAX = .999 - DX
      CALL QDAJM0 ( 0.5,X,IR,KCH )
      DO 200 I=1,IS
      DO 200 J=1,IS
  200 UNITY(I,J) = 0.0
      DO 201 I=1,IS
  201 UNITY(I,I) = 1.0
      WRITE ( 6,48 ) KSLOW,KFAST,IM,IMM,DX,FM,FMIN,YMIN,FEPS,ETAMN
     1,ETAMX,BDUP,BDLO,TEND,TSTART,TFAIL,AMULT,BMULT
     2,TOPF,DEL,IS,KH,GEPSQ,KPRINT,KPAUSE,NPOINT
   48 FORMAT(1H1,4X,11HSLOW RUN = I2,11X,11HFAST RUN = I2,16X,5HIM = ,
```

```
     *I2,15X,6HIMM = I2//,11X,4HDX =E10.3,9X,4HFM =E10.3,7X,6HFMIN =E10.
     *3,7X,6HYMIN =E10.3//,9X,6HFEPS =E10.3,6X,7HETAMN =E10.3,6X,7HETAMX
     * =E10.3,7X,6HBDUP =E10.3//,9X,6HBDLO =E10.3,7X,6HTEND =E10.3,5X,8H
     *TSTART =E10.3,6X,7HTFAIL =E10.3//,8X,7HAMULT =E10.3,6X,7HBMULT =E1
     *0.3,7X,6HTOPF =E10.3,8X,5HDEL =E10.3//,11X,5HIS = I2,16X,5HKH = I2
     *,13X,8HGEPSQ = E10.3,5X,9HKPRINT = ,I2//,7X,9HKPAUSE = ,I2,14X,
     *9HNPOINT = ,I3//)
      DO  240  I  =  1,  IS
      X (I)  =  X (I)  *  FACT   ( I )
  240 CONTINUE
      CALL QSEFF0 ( 1,1,IR )
  300 CONTINUE
      DELTA = DEL
      IF(DELTA)2004,2004,2005
 2005 CALL QSEFF0 (1,6,IR)
      GO TO 2003
 2004 CALL QREFF0 (1,6,IR)
 2003 CONTINUE
C*************************************************************************
C          MAKE  SLOW  RUN
C
      IF(KPRINT.NE.1)GO TO 2015
      CALL QPRNTO(1,IFIRST,ILAST,NCOMP,IERR)
 2015 CONTINUE
      KRUN = 1
      CALL RUN(KRUN,X,ADCV,DUMMY,TIME)
      IF(KPRINT.NE.1)GO TO 2016
      CALL QPRNTO(1,IFIRST,ILAST,NCOMP,IERR)
 2016 CONTINUE
      IF(KCONT.EQ.1)GO TO 501
      TIME  =  TIME  *  200.0
      IF(KGAIN) 2013,2013,2014
 2013 KKK=KKK+1
      TSLOPE(KKK)=TIME-XKSLOW
      DO 88 I=1,IS
  88  SLOPE(KKK,I)=X(I)
 2014 CONTINUE
      IF(TIME.LT.TEND) GO TO 87
```

```
      IF(KGAIN) 86,86,85
   86 WRITE (6,91) (TSLOPE(I), I=1,12)
      WRITE(6,91)((SLOPE(I,J),I=1,12),J=1,5)
      PUNCH 221, (TSLOPE(I), I=1,12)
      PUNCH 221,((SLOPE(I,J),I=1,12),J=1,5)
  221 FORMAT(6E12.5)
   85 GO TO 501
   87 CONTINUE
      WRITE(6,920)TIME
  920 FORMAT(/15X,6HTIME =F9.4/)
      DO  125  I  =1,  IS
  125  ANS  (I)  =  X (I)  /  FACT  (I)
      WRITE  (  6,  902  )  ( ANS(I), I=1,IS )
  902 FORMAT(/8X,13HGAIN SLOPES =5F9.4)
C*****************************************************************************
      IF((TIME + 0.2) - TSTART)2020,2017,2017
 2020 CALL SLOP (X,TIME,IS)
      GO TO 300
 2017 IF((TIME+0.2)-TFAIL)2019,2021,2021
 2021 DO 2022 I=1,IS
 2022 X(I) = 0.0
      GO TO 300
 2019 CONTINUE
C     CONSTANT STEP GRID SEARCH ROUTINE
C     TWO RUNS, ONE FOR EACH FAILURE CASE A AND B ARE MADE
C     AT EACH GRID POINT.
C
C     GRID SEARCH NOT USED FOR CONSTANT GAIN RUNS
C
      KPOINT = NPOINT
      NVARY = 0
      PMIN = 1.E20
      NPRD = 1
      NRUN = 0
      DO 333 I=1,IS
      X(I)=X2(I)
      XO(I)=X2(I)
      IF(NDX(I).EQ .0) GO TO 333
```

```
      NVARY = NVARY + 1
      NDEX(NVARY) = I
      NPRD = NPRD*(NDX(I)+1)
      NPROD(NVARY) = NPRD
  333 CONTINUE
      DELTA = +1.0
      CALL QSEFF0(1.6,IR)
      KRUN = 2
  335 CALL RUN (KRUN,X,ADCV,DUMMY,P)
      IF(KCONT.EQ.1)GO TO 501
      IF(DUMMY.LT.BDUP.AND.DUMMY.GT.BDLO)GO TO 338
      IF(DUMMY.GE.BDUP.AND.GAIN.LT.0.0015)GO TO 338
      IF(DUMMY.LE.BDLO.AND.GAIN.GT.0.9)GO TO 338
      CALL SCALE(DUMMY)
      GO TO 335
  338 IF(DELTA)411,412,413
  411 ADCVM = ADCV
      DUMMYM = DUMMY
      PMINUS = P
      IF(DEL.NE.-1.0)GO TO 382
      VER(1) = DUMMYM/ADCVM + PMINUS
      CALL PLOT(X,IS)
  382 CONTINUE
      IF(PPLUS.GT.PMINUS) GO TO 417
      P = DUMMYM/ADCVM + PMINUS
      DELTA = -1.0
      GO TO 419
  413 ADCVP = ADCV
      DUMMYP = DUMMY
      PPLUS = P
      IF(DEL.NE.+1.0)GO TO 381
      VER(1) = DUMMYP/ADCVP + PPLUS
      CALL PLOT(X,IS)
  381 CONTINUE
      DELTA = -1.0
      CALL QREFF0(1.6,IR)
      GO TO 335
  417 P = DUMMYP/ADCVP + PPLUS
```

```
      DELTA = +1.0
      GO TO 419
  412 P = DUMMY/ADCV + P
  419 CONTINUE
      IF(KPAUSE.NE.1)GO TO 380
      IF(NRUN.NE.KPOINT)GO TO 380
      PAUSE 4
      KPOINT = KPOINT + NPOINT
  380 CONTINUE
      NRUN = NRUN + 1
      IF(P.GT.PMIN)GO TO 331
      DO 330 I=1,IS
      XMIN(I) = X(I)
  330 CONTINUE
      PMIN = P
      DELMIN=DELTA
  331 CONTINUE
      IF(NRUN.GE.NPRD)GO TO 339
      M1 = NRUN
      DO 334 I=1,NVARY
      L = NDEX(I)
      JK = M1 - (M1/(NDX(L)+1))*(NDX(L)+1)
      XJ = JK
      M1=NRUN/NPROD(I)
      X(L) = X0(L) + XJ*DELX(L)
  334 CONTINUE
      IF(DELTA.EQ.0.0) GO TO 420
      DELTA = +1.0
      CALL QSEFF0(1.6,IR)
  420 CONTINUE
      GO TO 335
  339 DO 340 I=1,IS
  340 X(I)=XMIN(I)
      F = PMIN
      DELTA=DELMIN
      WRITE (6,2100)X
 2100 FORMAT(5X,53HVALUES OF X OBTAINED FROM SEARCH ROUTINE FOR THIS RUN
     1        /10X,6F14.6)
```

```
  332 CONTINUE
      WRITE(6,921) F
  921 FORMAT(/12X,9HJ(K(I)) =F10.5)
      IF ( F .LT. FMIN ) GO TO 300
      IT = 0
C
C          MAKE  FAST  RUNS  WITH  PERTURBED  X
C
      IF(DELTA.EQ.+1.0) GO TO 2010
      CALL QREFF0(1.6,IR)
      GO TO 2011
 2010 CALL QSEFF0(1.6,IR)
 2011 CONTINUE
      KRUN = 2
      DO 350 N=1,IS
  350 VER(N) = F
      CALL PLOT ( X , IS )
      DO  326  I  =1 ,  IS
      TEMP = X(I)
      X(I) = X(I) + 1.5*DX
      DO 327 M = 1, 2
      X(I) = X(I) - DX
      CALL RUN(KRUN,X,ADCV,DUMMY,P)
      IF(KCONT.EQ.1)GO TO 501
      PP(M)=DUMMY/ADCV+P
/      J1       860S
      WRITE ( 6,850 ) I,PP(M)
  850 FORMAT ( 48X,2HP(,I1,4H) = , F10.5 )
  860 CONTINUE
      VER(I) = PP(M)
      CALL PLOT ( X , IS )
  327 CONTINUE
      G(I) = ( PP(1) - PP(2) ) / DX
      X(I) = TEMP
  326 CONTINUE
/      J1       1005S
      WRITE ( 6,901 ) ( G(I) , I=1,IS )
  901 FORMAT(/7X,14HGRAD-J(K(I)) =5F11.5)
```

```
 1005 IT = IT + 1
C
C       COMPUTE  S' S  AND  ETA .
C
  349 IF ( KH .EQ. 0 ) GO TO 1006
      IF ( IT.GT.1 ) GO TO 1004
 1006 DO 1002 I=1,IS
      DO 1002 J=1,IS
 1002 H(I,J) = UNITY (I,J)
      GO TO 1003
 1004 DO 60 I=1,IS
      Y(I) = G(I) - GPREV(I)
      IF ( Y(I).LT.YMIN ) Y(I) = YMIN
   60 CONTINUE
      A1=0.                                                    C69C0710
      A2=0.                                                    C69C0720
      DO 62 I=1,IS                                             C69C0730
   62 A1=A1+SIG(I)*Y(I)                                        C69C0740
      DO 65 I=1,IS                                             C69C0750
      DO 65 J=1,IS                                             C69C0760
      A(I,J)=(SIG(I)*SIG(J))/A1                                C69C0770
   65 A2=Y(J)*H(I,J)*Y(I)+A2                                   C69C0780
      DO 66 I=1,IS                                             C69C0790
      E1(I,1)=0.                                               C69C0800
      DO 66 IJ=1,IS                                            C69C0810
   66 E1(I,1)=E1(I,1)+(-H(I,IJ)*Y(IJ))                         C69C0820
      DO 67 I=1,IS                                             C69C0830
      DO 67 J=1,IS                                             C69C0840
   67 E2(I,J)=E1(I,1)*Y(J)                                     C69C0850
      DO 68 I=1,IS                                             C69C0860
      DO 68 J=1,IS                                             C69C0870
      B(I,J) = 0.                                              C69C0880
      DO 68 IJ=1,IS                                            C69C0890
   68 B(I,J)=B(I,J)+E2(I,IJ)*H(IJ,J)                           C69C0900
      DO 70 I=1,IS                                             C69C0910
      DO 70 J=1,IS                                             C69C0920
      B(I,J)=B(I,J)/A2                                         C69C0930
   70 H(I,J)=H(I,J)+A(I,J)+B(I,J)                              C69C0940
```

```
 1003 DO 10 I=1,IS
      S(I)=0.                                                          C690164
      DO 10 J=1,IS                                                     C690165
   10 S(I)=S(I)-H(I,J)*G(J)                                            C690166
      D1=0.                                                            C690167
      DO 15 I=1,IS                                                     C690168
   15 D1=D1+G(I)*S(I)                                                  C690169
      ETA1=-2.*(F-FM * F )/D1
      ETA=ETA1                                                         C690171
/      J1      861S
   17 WRITE  (  6,  908  )  ETA
  908 FORMAT ( 112X,8HETA-1 = ,E12.5 )
  861 CONTINUE
      IF ( ETA1.LE.ETAMX.AND.ETA1.GE.ETAMN ) GO TO 1200
      IF ( ETA1 .LT. ETAMN ) GO TO 1202
      ETA = ETAMX
/      J1      862S
      WRITE ( 6,908 ) ETA
  862 CONTINUE
      GO TO 1200
 1202 ETA = 1.0
/      J1      863S
      WRITE ( 6,908 ) ETA
  863 CONTINUE
      D1 = 0.
      DO 1201 I=1,IS
      S(I) = - G(I)
      D1 = D1 - G(I) * G(I)
      DO 1201 J=1,IS
 1201 H(I,J) = UNITY (I,J)
 1200 IMIN=0
   19 CONTINUE                                                         C690174
      IMIN=IMIN+1                                                      C690175
C
C      EVALUATE   U ' S .
C
      DO126 I=1, IS
      U (I) =  X  (I) +  ETA * S (I)
```

```
      IF ( ABS(U(I)).GT. XMAX ) U(I) = SIGN( XMAX,U(I))
  126 ANS (I) =U (I) / FACT (I)
/      J1       864S
      WRITE(6,922)(ANS(I),I=1,IS)
  922 FORMAT(/14X,7HK*(I) =5F9.4)
  864 CONTINUE
C
C     MAKE 2 FAST RUNS,ONE FOR EACH FAILURE CASE A AND B FOR COMPUTING F(U)
C
      IF(DELTA.EQ.0.0)GO TO 730
      DELTA = +1.0
      CALL QSEFF0(1,6,IR)
  730 CALL RUN (KRUN,U,ADCV,FU,XYZ)
      IF(KCONT.EQ.1)GO TO 501
      IF ( FU.LT.BDUP.AND.FU.GT.BDLO ) GO TO 720
      IF ( FU.GE.BDUP.AND.GAIN.LT.0.0015 ) GO TO 720
      IF ( FU.LE.BDLO.AND.GAIN.GT.0.9 ) GO TO 720
      CALL  SCALE ( FU )
      GO TO 730
  720 IF(DELTA)421,422,423
  421 ADCVM = ADCV
      FUM = FU
      XYZM = XYZ
      IF(XYZP.GT.XYZM)GO TO 427
      FU = FUM/ADCVM + XYZM
      DELTA = -1.0
      GO TO 429
  423 ADCVP = ADCV
      FUP = FU
      XYZP = XYZ
      DELTA = -1.0
      CALL QREFF0(1,6,IR)
      GO TO 730
  427 FU = FUP/ADCVP + XYZP
      DELTA = +1.0
      CALL QSEFF0(1,6,IR)
      GO TO 429
  422 FU = FU/ADCV + XYZ
```

```
  429 CONTINUE
/      J1       865S
       WRITE (6,303)FU,IT
  303 FORMAT(/11X,10HJ(K*(I)) =F10.5,5X,5HIT = I2)
  865 CONTINUE
       DO 360 N=1,IS
  360 VER(N) = FU
       CALL PLOT ( U , IS )
C
C          MAKE  FAST  RUNS  WITH  PERTURBED  U
C
       DO  356   I = 1 ,  IS
       TEMP = U(I)
       U(I) = U(I) + 1.5*DX
       DO 357 M=1,2
       U(I) = U(I) - DX
       CALL RUN(KRUN,U,ADCV,DUMMY,P)
       IF(KCONT.EQ.1)GO TO 501
       PP(M)=DUMMY/ADCV+P
/      J1       866S
       WRITE ( 6,850 ) I,PP(M)
  866 CONTINUE
       VER(I) = PP(M)
       CALL PLOT ( U , IS )
  357 CONTINUE
       GU(I) = ( PP(1) - PP(2) ) / DX
       U(I) = TEMP
  356 CONTINUE
/      J1       867S
       WRITE(6,923)(GU(I),I=1,IS)
  923 FORMAT(/6X,15HGRAD-J(K*(I)) =5F11.5)
  867 CONTINUE
C
C                              DETERMINE  Z  .
C
       T1=D1                                                    C690179
       T2=0.                                                    C690180
       DO 30 I=1,IS                                             C690181
```

```
   30 T2=T2+GU(I)*S(I)                                                  C690182
      Z=3./ETA*(F-FU)+T1+T2                                             C690183
      ARG=Z**2-T1*T2                                                    C690184
      IF( ARG .GT. 0. ) GO TO 34                                        C690185
      ETA=2.*ETA                                                        C690186
/      J1       868S
      WRITE  (  6,  910  )  ETA
  910 FORMAT ( 112X,8HETA-2 = ,E12.5 )
  868 CONTINUE
      GO TO 19
   34 W=SQRT(ARG)                                                       C690188
C
C   DETERMINE  ALPHA ,  ALPHA * S ,  AND  X +  ALPHA * X .
C
      ALPHA=ETA*(1.-((T2+W-Z)/(T2-T1+2.*W)))                            C690189
      DO 35 I=1,IS                                                      C690190
      SIG(I)=ALPHA*S(I)
      XAS (I) =  X  (I)   +  SIG (I)
      IF ( ABS(XAS(I)).GT.XMAX) XAS(I) = SIGN(XMAX,XAS(I))
   35 ANS(I) = XAS(I) / FACT(I)
/      J1       869S
      WRITE(6,924)(ANS(I),I=1,IS)
  924 FORMAT(/13X,8HK(I+1) =5F9.4)
  869 CONTINUE
C
C     MAKE 2 FAST RUNS, ONE FOR EACH FAILURE CASE A AND B
C     FOR COMPUTING FXAS
C
C
      IF(DELTA.EQ.0.0)GO TO 750
      DELTA = +1.0
      CALL QSEFF0(1,6,IR)
  750 CALL RUN (KRUN,XAS,ADCV,FXAS,XYZ)
      IF(KCONT.EQ.1)GO TO 501
      IF ( FXAS.LT.BDUP.AND.FXAS.GT.BDLO ) GO TO 740
      IF ( FXAS.GE.BDUP.AND.GAIN.LT.0.0015 ) GO TO 740
      IF ( FXAS.LE.BDUP.AND.GAIN.GT.0.9 ) GO TO 740
      CALL  SCALE ( FXAS )
```

```
      GO TO 750
  740 IF(DELTA)431,432,433
  431 ADCVM = ADCV
      FXASM = FXAS
      XYZM = XYZ
      IF(XYZP.GT.XYZM)GO TO 437
      FXAS = FXASM/ADCVM + XYZM
      DELTA = -1.0
      GO TO 439
  433 ADCVP = ADCV
      FXASP = FXAS
      XYZP = XYZ
      DELTA = -1.0
      CALL QREFF0(1.6,IR)
      GO TO 750
  437 FXAS = FXASP/ADCVP + XYZP
      DELTA = +1.0
      CALL QSEFF0(1.6,IR)
      GO TO 439
  432 FXAS = FXAS/ADCV + XYZ
  439 CONTINUE
/      J1       870S
      WRITE ( 6,301 ) FXAS,IT
  301 FORMAT(/11X,11HJ(K(I+1)) =F10.5,5X,5HIT = I2)
  870 CONTINUE
      DO 370 N=1,IS
  370 VER(N) = FXAS
      CALL PLOT ( XAS , IS )
C          TEST  F ( X + ALPHA*S )  VS.  F(X)  AND  F(U)
C
      IF ( FXAS.LT.FEPS*F.AND.FXAS.LE. FU      ) GO TO 1007
C
C          TEST  F(U)  VS.  F(X)  AND  F ( X + ALPHA*S )
C
      IF ( FU.LT.FEPS*F.AND.FU.LE. FXAS      ) GO TO 1008
      ETA = ETA * .4
/      J1       871S
      WRITE ( 6,912 ) ETA
```

```
  912 FORMAT ( 112X,8HETA-3 = ,E12.5 )
  871 CONTINUE
      IF ( IMIN.GE.IMM ) GO TO 520
      GO TO 19
 1007 DO 1100 I=1,IS
      GPREV(I) = G(I)
 1100 X(I) = XAS(I)
      F = FXAS
      DO 336 I=1,IS
      TEMP = XAS(I)
      XAS(I) = XAS(I) + 1.5*DX
      DO 337 M=1,2
      XAS(I) = XAS(I) - DX
      CALL RUN (KRUN,XAS,ADCV,DUMMY,P)
      IF(KCONT.EQ.1)GO TO 501
      PP(M)=DUMMY/ADCV+P
/      J1       872S
      WRITE ( 6,850 ) I,PP(M)
  872 CONTINUE
      VER(I) = PP(M)
      CALL PLOT ( XAS,IS )
  337 CONTINUE
      G(I) = ( PP(1) - PP(2) ) / DX
      XAS(I) = TEMP
  336 CONTINUE
/      J1       873S
      WRITE(6,925)(G(I),I=1,IS)
  925 FORMAT(/5X,16HGRAD-J(K(I+1)) =5F11.5)
  873 CONTINUE
      GO TO 1000
 1008 DO 1101 I=1,IS
      GPREV(I) = G(I)
      G(I) = GU(I)
 1101 X(I) = U(I)
      F = FU
 1000 GSQ = 0.
      DO 41 I=1,IS
   41 GSQ = GSQ + G(I)**2
```

```
      IF ( GSQ.GE.GEPSQ ) GO TO 349
      IF ( IT.GE.IM ) GO TO 300
      GO TO 1005
/      J1       874S
  520 WRITE ( 6,521 ) IMIN
  521 FORMAT ( 42H MINIMIZATION ATTEMPTS IN ONE ITERATION = ,I3 )
  874 CONTINUE
      IF ( IT.GE.IM ) GO TO 300
      GEPSQ = 2.*GEPSQ
/      J1       875S
      WRITE ( 6,1009 ) GEPSQ
 1009 FORMAT ( 112X,8HGEPSQ = ,E12.5 )
  875 CONTINUE
      IT = IT + 1
      GO TO 1006
  501 PRINT 502
  502 FORMAT ( 10X,11HPROGRAM END )
      GO TO 500
/601   PZE      0.,.17
/      JT       600S
  600 CALL QREFF0 ( 1,0,IR )
      CALL QREFF0 ( 1,2,IR )
  500 CALL QREFF0 ( 1,1,IR )
      CALL QSEFF0 ( 1,3,IR )
      CALL QPS0 ( 1,IR )
      CALL QDSIT0 (1,IR)
      GO TO 47
   91 FORMAT(5X,E12.8,5X,E12.8,5X,E12.8,5X,E12.8,5X,E12.8,5X,E12.8)
      STOP
      END
*
      SUBROUTINE RUN (K,ARRAY,ADCV,SUMJA,ANALV)
      DIMENSION ARRAY(5),SUMJ(4)
      COMMON / RN / KTIME(3),IS,KMODE,AMULT,KCONT ,BMULT
      COMMON/DE/DELTA
      EXTENDED  KTIME
      LOGICAL READY
      IF(K-2)1,2,3
```

```
    2 CONTINUE
/       LDOB    =*50.*12        TIME CONSTANT IN MILLISEC.
/       LDOB    =*1451.*12                                MED
        GO TO 10
    1 CONTINUE
/       LDOB    =*450.*12       TIME CONSTANT IN SEC.
/       LDOB    =*1451.*12                              MED
        CALL QSEFF0 ( 1.0.IR )
        GO TO 15
    3 CONTINUE
/       LDOB    =*450.*12       TIME CONSTANT IN SEC.
/       LDOB    =*451.*12                               FAST
   15 CALL QSEFF0 ( 1.2.IR )
   10 CALL QDAJM0 ( 0.IS.ARRAY.IR.KCH )
      IF ( IR.EQ.1 ) GO TO 23
      TYPE 21. IR
   21 FORMAT ( 10X.5HIR = .I1.3X.27HFOR DAC. SKIP ONE SPACE AND /
     1 10X.23HTYPE 0 FOR CONTINUATION/ 10X.22HTYPE 1 FOR ANOTHER RUN )
      ACCEPT 100.KCONT
  100 FORMAT(I1)
   23 IF ( K.EQ.KMODE ) GO TO 20
      KMODE = K
      DELAY  5.E5
   20 CONTINUE
/       LDOB    =*2047.*12      ANALOG TO OPERATE
      CALL DELAY ( KTIME(K) )
/       LDOB    =*2447.*12      ANALOG TO HOLD
      CALL QTEFF0 ( 1.0.READY.IR )
   25 CALL QTEFF0 ( 1.0.READY.IR )
      IF ( .NOT.READY ) GO TO 25
      IF(K-2)4.5.6
    4 CALL QREFF0 ( 1.0.IR )
    6 CALL QREFF0 ( 1.2.IR )
      CALL QADCV0 (K.1.ANALV.IR.KCH)
      GO TO 7
    5 CONTINUE
      CALL QADCV0 (2.4.SUMJ.IR.KCH)
    7 IF ( IR.EQ.1 ) GO TO 30
```

```
      TYPE 22,IR,KCH,SUMJ(KCH)
   22 FORMAT(10X,5HIR = ,I1,3X,15HFOR ADC CHANNEL,I4,F8.4/
     110X,23HTYPE 0 FOR CONTINUATION/ 10X,22HTYPE 1 FOR ANOTHER RUN )
       LDOB     =03047,12      ANALOG TO IC
      ACCEPT 100,KCONT
   30 CONTINUE
       LDOB     =03047,12      ANALOG TO IC
       J3       34S
      DELAY 20.E3
   34 CONTINUE
       J4       35S
      DELAY 30.E3
   35 CONTINUE
       J5       36S
      DELAY 40.E3
   36 CONTINUE
       J6       37S
      DELAY 50.E3
   37 CONTINUE
       J7       38S
      DELAY 100.E3
   38 CONTINUE
      IF(K-2)33,31,33
   31 SUMX=-SUMJ(2)
      DO 32 I=2,4
      SUMJ(I)=-SUMJ(I)
      SUMX=AMAX1(SUMX,SUMJ(I))
   32 CONTINUE
      ANALV=SUMX*AMULT
      SUMJA=-SUMJ(1)*BMULT
      SUMX=SUMJA/ADCV + ANALV
       J2       33S
      WRITE(6,99)(SUMJ(L),L=1,3),ANALV,SUMX,DELTA
   99 FORMAT(5X,3F12.4,5X,6HANALV=,F8.4,5X,5HSUMX=,F8.4,5X,6HDELTA=,F4.1
     1)
   33 RETURN
      END
```

```
      SUBROUTINE SLOP(X,TIME,IS)
      COMMON/SLOPES/TSLOPE(12),SLOPE(12,5),MG
      DIMENSION X(5)
      IF((TIME+0.2)-TSLOPE(MG)) 1,2,2
    2 DO 3 I=1,IS
    3 X(I)=SLOPE(MG,I)
      MG=MG+1
    1 RETURN
      END

      SUBROUTINE  SETPOT
      COMMON / PST / NPOT,POT(150),PSET(150),VAL,VALUE
      EXTENDED  POT,VAL
      CALL QSEFF0 ( 1,3,IR )
      CALL QPS0 ( 1,IR )
      IF ( IR.NE.2 ) GO TO 1
      TYPE 13
      PAUSE
    1 DO 10 I=1,NPOT
      CALL QSTPT0 ( 1,POT(I),PSET(I),IR )
      GO TO  ( 10,2,3,4,5 ), IR
    2 PRINT 11, POT(I),PSET(I)
   11 FORMAT ( 10X,32HINVALID ANALOG ADDRESS SPECIFIED,5X,A4,F10.4 )
      GO TO 10
    3 PRINT 12,POT(I),PSET(I)
   12 FORMAT ( 10X,23HCOEFF OVERFLOW OCCURRED,5X,A4,F10.4 )
      GO TO 10
    4 PRINT 13
   13 FORMAT ( 10X,20HCONSOLE DISCONNECTED )
      GO TO 10
    5 PRINT 14, POT(I),PSET(I)
   14 FORMAT ( 10X,21HNULL FAILURE DETECTED,5X,A4,F10.4 )
   10 CONTINUE
      CALL QRDAL0 ( 1,VAL,VALUE,IR )
      CALL QREFF0 ( 1,3,IR )
      CALL QIC0 ( 1,IR )
      RETURN
      END
```

```
      SUBROUTINE  SCALE ( VOLT )
      COMMON / SCL / BDUP,BDLO,GAIN,ADCV
      IF ( VOLT.LT.BDUP ) GO TO 10
      GAIN = GAIN / 10.
      ADCV = ADCV / 10.
      GO TO 15
   10 GAIN = GAIN * 10.
      ADCV = ADCV * 10.
   15 CALL QDAJM0 ( 5,1,GAIN,IR,KCH )
       J1       20S
      WRITE ( 6,1 ) GAIN,ADCV
    1 FORMAT ( 70X,7HGAIN = ,F6.4,5X,7HADCV = ,F6.4 )
   20 CONTINUE
      RETURN
      END

      SUBROUTINE  PLOT ( HOR , N )
      DIMENSION  HOR( 5),ARY(5)
      COMMON / PLT / VER ( 5 ) , TOPF
      CALL QSEFF0 ( 1,4,IR )
      DO 10 I = 1 , N
      ARY(I) = VER(I) / TOPF
      IF ( ARY(I) .GT. 1.0 ) ARY(I) = 1.0
   10 CONTINUE
      CALL QDAJM0 ( 6,N ,ARY,IR,KCH )
      CALL QDAJM0 ( 11,N ,HOR,IR,KCH )
      CALL QREFF0 ( 1,4,IR )
      RETURN
      END
,
,
$AS,,IU1,,4003,
*   DEVELOPED BY R.M. BOND    11 JUNE 1968
*   THIS SUBROUTINE SETS UP A DELAY IN MILLISECONDS USING THE DIGITAL
*   TIMER.  THE TIMER MUST BE SET UP BY TECHNICIANS TO DECREMENT ONCE
*   EACH MILLISECOND.  THE DELAY VALUE IS DECREMENTED BY ONE AND LOADED
*   INTO THE TIMER.  THE TIMER STARTS AND CAUSES AN INTERRUPT ONE MILLI-
```

```
*   SECOND AFTER THE TIMER REACHES ZERO.  THE FORTRAN CALLING STATEMENT
*   IS - CALL DELAY (NT) WHERE NT IS AN EXTENDED INTEGER VARIABLE EQUAL
*   TO THE NUMBER OF MILLISECONDS DESIRED FOR THE DELAY.
*
DELAY   REL
ITM     EQU     '6
        PZE     **
        ECA     ITLNK           SET UP
        EST     '46                     INTERRUPT LOCATION
        STM     *+2             SET INTERRUPT
        SMI     *+1,,ITM                        MASK
        LDM     =**             LOAD MASK REGISTER
        CA*     /0,6            PLACE DELAY VALUE IN ACCUMULATOR
        SB      =1              SUBTRACT 1 (SEE ABOVE COMMENT)
        ST      /DELAY          STORE VALUE IN MEMORY
        LDT     /DELAY          LOAD TIMER FROM MEMORY
        SFL     ='62            START TIMER
        JSE     *+1             SET ENABLE FLAG
        J       *               LOOP UNTIL INTERRUPT
ITLNK   L       INTO            INTERRUPT INSTRUCTION
INTO    PZE     **
        SFL     ='63            STOP TIMER
        JT      1,6             RETURN
        END
'
'
$LO,,GU1,'0
$XQ,*,0
$ID,RIC2,0,
$SV,,0U2,'12000,'12000,'37777,0,
```